A series of case studies to enhance the social utility of RSS

A series of case studies to enhance the social utility of RSS

Martin O'Shea

December 2016

A Thesis Submitted to
Birkbeck, University of London
in Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Department of Computer Science & Information Systems
Birkbeck
University of London

Declaration

This thesis is the result of my own work, except where explicitly acknowledged in the text.

Martin O'Shea _____

December 22, 2016

Abstract

RSS (really simple syndication, rich site summary or RDF site summary) is a *dialect* of XML that provides a method of syndicating on-line content, where postings consist of frequently updated news items, blog entries and multimedia. RSS feeds, produced by organisations or individuals, are often aggregated, and delivered to users for *consumption* via *readers*. The semi-structured format of RSS also allows the delivery/exchange of machine-readable content between different platforms and systems.

Articles on web pages frequently include icons that represent social media services which facilitate social data. Amongst these, RSS feeds deliver data which is typically presented in the journalistic style of *headline*, *story* and *snapshot(s)*. Consequently, applications and academic research have employed RSS on this basis. Therefore, within the context of social media, the question arises: *can the social function, i.e. utility, of RSS be enhanced by producing from it data which is actionable and effective?*

This thesis is based upon the hypothesis that the fluctuations in the keyword frequencies present in RSS can be mined to produce *actionable and effective* data, to enhance the technology's social utility. To this end, we present a series of laboratory-based case studies which demonstrate two novel and logically consistent RSS-mining paradigms. Our first paradigm allows users to define *mining rules* to mine data from feeds. The second paradigm employs a semi-automated classification of feeds and correlates this with sentiment. We visualise the outputs produced by the case studies for these paradigms, where they can benefit users in real-world scenarios, varying from statistics and trend analysis to mining financial and sporting data.

The contributions of this thesis to web engineering and text mining are the demonstration of the *proof of concept* of our paradigms, through the integration of an array of open-source, third-party products into a coherent and innovative, *alpha*-version prototype software implemented in a Java JSP/servlet-based web application architecture.

Publications

The following publications by the author are related to this thesis:

- M. O'Shea and M. Levene. Mining and visualising information from RSS feeds: a case study, *International Journal of Web Information Systems*, 7(2):105–129, 2011.
- M. O'Shea and M. Levene. visualRSS: a Platform to Mine and Visualise Social Data from RSS Feeds, *In Proceedings of the 12th International Conference on Web Engineering (ICWE 2012) 4th International Workshop on Lightweight Integration on the Web (Composable Web)*, pages 121–133, held in Berlin, Germany, Jul 2012.

For Father.

Acknowledgements

Supervisors

A special thank you must be extended to Professor M. Levene, the author's long-suffering principal supervisor for encouragement, motivation and enduring what must have seemed like an endless series of emails, delays and doubts.

Thanks are also extended to Professor G. Loizou for all the guidance concerning the structure, content and proof reading of this thesis, and for tolerating the author's frequent changing of meeting arrangements.

For insight, support and playing the role of *devil's advocate*, the author is also grateful to Professor D. Zhang.

Students

To former Masters students M. Danani, D. Harriott, Y. R. Shema and M. Wilce, thank you for your contributions.

Family and friends

The author also thanks family and friends for support during an arduous and protracted process.

And finally...

"We must use the Box! because the Box! must be used.
The Box! must be used because we must use the Box!."

The Kōan of Clog?, quoted from the Book of Morgan Entities¡

Let us make sure that history never forgets: *Morgan likes potatoes!*

Contents

List of Abbreviations

1:1 One-to-one.
1:M One-to-many.

ACID Atomicity, consistency, isolation, durability.
AJAX Asynchronous JavaScript and XML.
ANSI American National Standards Institute.
AOL America Online.
AOP Aspect–oriented programming.
API Application programming interface.
ARFF Attribute relationship file format.
ARPA Advanced Research Projects Agency.

BARF Barf Archives RSS Feeds.
BBS Bulletin board system.
BFE Business, finance and economics.
BNF Backus–Naur form.
BoW Bag-of-words.
BST British summer time.

CF–IDF Concept frequency - inverse document frequency.
CRESDUP Content Recommendation System based on Private Dynamic User Profile.
CRUD Create, read, update and delete.
CSS Cascading style sheet.

DAO Database access object.
DBCP Database connection pooling.
DFD Data flow diagram.
DMOZ Directory Mozilla.
DOM Document object model.
DT Decision tree.

EA Entertainment and arts.
EBNF Extended Backus-Naur form.
EL Expression language.
EMM Europe Media Monitor.
ERDM Entity relationship diagram.
ETL Extract, transform and load.
EUA End user automation.

FCL Fashion, celebrity and lifestyle.
FICUS Filtering and clustering non-redundant RSS news articles.
FK Foreign key.
FN False negative.
FORTRAN Formula translating system.
FOSS Free and open software.
FP False positive.

GMT Greenwich mean time.
GNSS Global navigation satellite system.
GPS Global positioning system.

HTML Hypertext markup language.
HTTP Hypertext transfer protocol.
HTTPS Secure HTTP.

IBM International Business Machines.
IDE Integrated development environment.
IETF Internet Engineering Task Force.
IG Information gain.
IM Instant messaging.
IR Information retrieval.
ISO International Organization for Standardisation.
IT Information technology.
IWT Intelligent Web Teacher.

JDBC Java database connectivity.
JEE Java Enterprise Edition.
JSON JavaScript object notation.
JSP JavaServer page.
JSTL Java standard tag library.

KDD Knowledge discovery in databases.
KDT Knowledge discovery in text.
KNN k-nearest neighbour.

M:N Many-to-many.
MCC Matthew's correlation coefficient.
MI Mutual information.
MNB Multinomial naïve Bayes.
MOOC Massive open on-line course.
MTT Multiple Topic Tracking.
MVC Model-view-controller.
MWCC Media Watch on Climate Change.
myDS myDataSharer.

NATO North Atlantic Treaty Organisation.
NB Naïve Bayes.
NCA News and current affairs.
NDS News Directory System.
NER Named entity recognition.
NISO National Information Standards Organization.
NLP Natural language processing.
NoSQL No SQL, not only SQL *or* not relational.

OED Oxford English Dictionary.
Ofcom Office of Communications.
OM Occurrence mining.
OMG Object Management Group.
OOP Object–oriented programming.
ORM Object relational modelling.
OS Operating system.

PK Primary key.
POC Proof of concept.
POJO Plain old Java object.
POS Part-of-speech.
PRC Precision recall curve.

QBE Query-by-Example.

RDBMS Relational database management system.
RDF Resource description framework.
REST Representational state transfer.
RFC Request for comment.
ROC Receiver operating characteristic.
RoSeS Really Open Simple and Efficient Syndication.
RSS Really simple syndication, rich site summary *or* RDF site summary.
RSSAB RSS Advisory Board.

SD Standard deviation.

SEO Search engine optimisation.

SInfoNS Secure Information Notifying System with RSS Technology for Mobile Users.

SMS Short message service.

SNT Science, nature and technology.

SOAP Simple object access protocol.

SQL Structured query language.

SSADM Structured systems analysis and design methodology.

SVG Scalable vector graphic.

SVM Support vector machine.

TCP/IP Transmission control protocol/internet protocol.

TF-ICF Term frequency - inverse corpus frequency.

TF-IDF Term frequency - inverse document frequency.

TN True negative.

TP True positive.

uBioRSS Universal Biological Indexer and Organiser RSS.

UI User interface.

UML Unified modelling language.

UNIX Uniplexed information and computing service.

URI Uniform resource identifier.

URL Uniform resource locator.

URN Uniform resource name.

VISA Visual Sentiment Analysis System.

VLE Virtual learning environment.

VM Value mining.

VML Vector mark-up language.

vRSS visualRSS.

VSM Vector space model.

W3C World Wide Web Consortium.

WEKA Waikato Environment for Knowledge Analysis.

WNSS Web News Search System.

WSDL Web service definition language.

WWW World wide web.

WYSIWYG *What you see is what you get.*

XML Extensible mark-up language.

List of Algorithms, Code and Pseudocode

List of Figures

List of Tables

Part I

Introduction and opening comments

Chapter 1

Introduction

1.1 The phenomenon of feeds and RSS

The OED at `http://www.oxforddictionaries.com/definition/english/digest` defines the term *digest* as a verb to "Arrange in a systematic or convenient order, especially by reduction." When applied to the need to present summaries of detailed or otherwise lengthy information, common contemporary examples of a digest include a document abstract or synopsis, an executive or legal summary, a headline or a *soundbite*. Internet- or web- based digests extend this principle to *cyberspace*. Since their inception, i.e. *circa* late 1990s - early 2000s, as part of *Web 2.0* (Appendix B.2) social media (Section 2.3), digests, or *feeds* as they are more correctly known, have been broadcast, i.e. *syndicated*, by media or commercial organisations, or by individuals. A consequence of this has been the diversity of feeds produced and their applications. Social networking web sites such as Twitter [408] and Facebook [106] permit micro-blogging by individuals or organisations, exchanges between *friends* to do with matters of interest or current affairs, whilst other services provide facilities for sharing graphics or particular media, news and opinion promulgation/dissemination. Each of these constitutes "a compilation or summary of material or information", to employ an OED definition of *digest* as a noun.

Two popular formats of feeds are RSS (Section 2.2) and its rival Atom (Appendix B.2), both of which deliver frequently updated content such as blog entries, news headlines and multimedia. RSS, the focus of this thesis, is a *dialect* of XML although, as described by Pilgrim [299], the term *RSS* refers to an "umbrella format that spans several different versions of at least two different (but parallel) formats." These alternative versions date from the late 1990s when a syndication format known as *Scripting News* was developed by Winer at UserLand Software (`http://www.userland.com/`). In 1999, RSS 0.90 was

created, and after a series of revisions, the RSS 2.0 specification [332] was released in 2003. RSS 2.0 has since been maintained by the RSSAB [334], and the latest version of the specification is 2009's RSS 2.0.11.

A principal application of RSS (Section 3.2) is aggregation, i.e. where large numbers of RSS feeds are collected into a one or more feeds for delivery to users. This facility is often provided either via open-source, third-party products or search engines, e.g. Google News [153] and Yahoo News [457]. RSS-based management services allow customers to publish their RSS feeds and provide traffic analysis, advertising and emailing services. Search engines for RSS also permit users to discover feeds of interest or perform real-time searches of feed content. Moreover, given the XML-based origins of RSS, the technology consists of a semi-structured, machine-readable format (Section 2.2.3) which allows the on-line delivery/exchange of information between different platforms and systems.

Despite these applications and academic research employing RSS (Chapter 3), feeds are typically delivered via a browser, or by *readers* (Section 3.2). Content is then presented to users in the journalistic style of *headline*, *story* and *snapshot(s)* (Section 2.2.4), and *consumed*. In this thesis, we are concerned with the function, i.e. *utility*, of RSS within the context of social media, and how the social utility of the technology can be enhanced by producing from RSS feeds data that is of a more *actionable and effective* nature.

1.2 Motivation and application

The motivation for this thesis, i.e. to enhance the social utility of RSS within the context of social media, originated from a combination of personal and professional interests. The author has been a user of social media (Section 2.3) covering various topics for many years, and was formerly employed as an analyst/programmer in the financial sector with a particular focus on web applications and software engineering. A PhD provided the opportunity to merge these subjects within the framework of research at a *proof of concept* level to employ datasets (Appendix B.2) or types of social media, e.g. feeds, *to produce new or enhanced outputs to represent the data inherent in the content of these media in order to benefit users in real-world scenarios.*

This concept supplied the initial basis for our work, and in the software we originally developed to realise it, we sought to employ data from a wide variety of formats (Section 5.3) within a social *data-sharing* environment where we could model user behaviour, although we subsequently focused solely upon the use of RSS. For this reason, it is necessary that we answer the *inevitable* question: *why did we choose RSS and not another type of*

feed? Other types of feeds were available to us, e.g. Facebook [106] and Twitter [408], or through the use of Atom. Stated simply, our answer to this question is threefold: (1) the *ubiquity* of RSS given the range of applications and academic research employing the technology: we review these in Chapter 3, (2) the *longevity* of RSS as a *mature* technology: the earliest version of RSS dates from 1999 (Section 2.2.2), and therefore predates the other feed types, i.e. Atom (2005), Facebook (2004) and Twitter (2006), and (3) the *diversity* of subject matter found in the content of RSS feeds which consists of frequently updated news items, blog entries and multimedia produced by individuals, media or commercial organisations: this diversity is apparent in the RSS feed and category corpora we have employed (Appendix A).

By adopting RSS as a form of social media (Section 2.3), we have been able to concentrate on a single, popular technology: this allowed us to develop the aforementioned initial basis for this thesis into a specific and *mature* form to ask: *can the social function, i.e. utility, of RSS be enhanced by producing from it data which is actionable and effective?* It is from here that our hypothesis and objectives (Section 1.3) were developed. It follows from this that we can define the application, i.e. *importance*, of our work according to the reasons of *ubiquity*, *longevity* and *diversity* we have cited for our choice of the technology given: (1) the *landscape* of this thesis (Chapter 2), and (2) the contributions made by this thesis to several of the subject areas of this landscape, which are referred to in Section 1.3.

1.3 Hypothesis, objectives and contributions

Predicated upon the context described in Section 1.2 we present our hypothesis to state that:

> *Data of an actionable and effective nature can be produced from the fluctuations in the keyword frequencies present in the text of RSS feeds to enhance the social utility of the technology, where this data can benefit users in real-world scenarios, varying from statistics to marketing and trend analysis, correlating or tracking topical issues, or for mining financial and sporting data.*

In order to validate this hypothesis, we define the three inter-related objectives for this thesis, as follows:

1. **Definition:** The need to review the current function of RSS within the context of social media, and the utility gained from this.

2. **Production:** To develop the appropriate methods, i.e. paradigms, to mine the text of RSS in order to produce data of an actionable and effective nature.

3. **Demonstration:** To formulate and carry out a series of specifically designed and implemented case studies to demonstrate the paradigms and to use an appropriate medium to represent their outputs.

The following sections of this chapter define the *proof of concept* of these objectives which is demonstrated by the paradigms we employ to mine RSS, the series of case studies presenting the paradigms, and the software implementing the case studies. Together, these elements form the contributions made by this thesis to the subject areas of web engineering (Section 4.2) and text mining (Section 2.6). We exclude any contribution to information visualisation from this list because we employ the medium of visualisation (Section 2.8) only as a means of representing the actionable and effective data (Section 2.9) produced by the case studies for our RSS-mining paradigms.

To the best of our knowledge, the paradigms and case studies (Sections 1.4 and 1.5 respectively) that we present in this thesis are unique in their use of RSS as a technology, and in the use of the web engineering/application principles and content we document herein, to integrate an array of open-source, third-party products into a coherent and innovative, *alpha*-version prototype software (Section 1.6).

We evaluate the realisation of our objectives and the contributions made by this thesis to the subject areas of web engineering and text mining in Chapter 13. In doing this, with the exception of the student corpora in the case studies for our first paradigm, we do not address the social or demographic issues inherent in the work we present or in its *landscape* (Chapter 2). Similarly, we do not consider issues of privacy or security.

1.4 RSS-mining paradigms

1.4.1 Definition

Our RSS-mining paradigms are formally defined as:

1. **Paradigm one: *Defining mining rules upon RSS to determine and visualise trends from textual and numeric data*:** The definition of mining rules (Section 4.7.2) upon RSS is intended to provide a straightforward means for users to specify how textual and numeric data is to be mined from feeds during polling to update and visualise the objects the rules become part of. This permits the visual

analysis of trends (Appendix B.2) based upon the fluctuations in the frequencies of popular keywords (Section 1.7) present in the text of RSS feeds, and the use of modern forms of ticker-tape (Appendix B.2) data, such as financial movements, sports or lottery results

The premise of this paradigm is to demonstrate that we can use mining rules to produce from RSS data that is more actionable and effective than we currently see in the use of the technology (Chapter 3).

It is necessary to state unequivocally here that in this thesis the term *mining rules* is of our own devising and applies only to the software implementing the case studies for our first paradigm. No further meaning of this term is intended or should be inferred.

2. **Paradigm two: *Classifying RSS according to the fluctuations in the frequencies of popular keywords and correlating this with sentiment*:** This paradigm concerns a semi-automated application of well-known classification techniques to RSS in order to classify feeds into categories, and to determine a correlation between this and sentiment. We present a set of time-series (Section 2.8.3) plots visualising this correlation for potential use in business intelligence, statistics, politics, market research and related subject areas. This paradigm also employs changes in keyword frequencies in the text of RSS feeds as the basis for classification and sentiment analysis.

   Our second paradigm forms a logical extension to the first, i.e. if our first paradigm can produce basic actionable and effective data from RSS, can we not apply data mining techniques to RSS in order to further enhance the technology's social utility?

1.4.2 Correspondence

Our paradigms are logically consistent with our hypothesis because they are both concerned with the *proof of concept* demonstration of enhancing RSS's utility within the social media context. This consistency of our paradigms is further based upon several common elements in their case studies: (1) the text of RSS feeds is used as input, (2) the fluctuations in the frequencies of popular keywords present in the text are used as data, (3) that these fluctuations are measured between pre-defined starting and ending dates/times, (4) that a minimal ETL (Appendix B.2) is applied to the text for data cleansing, and (5) that visualisation is used as the medium for the output (Section 2.8) because of its ability to coherently represent actionable and effective data to users.

At the same time, both paradigms are conceptually distinct: as befits their nature, the case studies for our first paradigm are primarily user-driven despite automated elements in case study two, and our second paradigm does not strictly require user-interaction because it is based upon semi-automated batch processing.

1.5 Case studies

We employ a series of three laboratory-based case studies to demonstrate our paradigms. The first two case studies concern our first paradigm, whereas case study three concerns our second paradigm. The following list summarises each case study:

1. Our first case study took place in late 2009 and employed a corpus of thirty-five part- and full-time Masters-level students. This case study sought to answer a series of research questions concerning the feasibility of our first paradigm's mining rules and the use of its mining types.

2. Case study two made use of a second corpus of thirty-six part-time Masters-level students during Dec 2011. This allowed us to research preferences of the mining types employing textual mining rules refined from case study one (Section 6.4.2), visualisations, distribution of categories of feeds visualised, and the common use of these amongst the mining types.

3. Our third case study deals with our second paradigm. The first component of this case study describes the semi-automated classification of RSS feeds into categories according to the fluctuations in the frequencies of popular keywords present in their text, and the second component concerns the determination of a correlation between the changes in the keyword frequencies and sentiment.

1.6 Software

For our paradigms and their case studies, two *alpha*-version applications, i.e. myDataSharer (myDS) and visualRSS (vRSS), were designed and written. Both applications employ the n-tiered Java JSP/servlet-based web application architecture described in Section 4.5, and share several common components and related terminology. In addition, the two applications make use of the relational database model (Section 4.5.7), and include many open-source, third-party products (Appendix B.1) from the Java *ecosystem* on a black-box, mash-up basis.

1.7 Keywords

In this thesis, the term *keyword* conforms to the definition provided by the OED at `http://www.oxforddictionaries.com/definition/english/keyword`, i.e. "a word used in an information retrieval system to indicate the content of a document." In our paradigms and their case studies, with one exception, our keywords are naïve n-grams (Appendix B.2), and in all cases are based upon the fluctuations in the frequencies of popular keywords in the text of RSS feeds measured between pre-defined starting and ending dates/times. The conventions and characteristics of our keywords are discussed in Section 4.6.

1.8 The relationship of RSS-mining paradigms, case studies and software

The relationship between our RSS-mining paradigms, case studies and the software demonstrating them is formally illustrated in Figure 1.1.

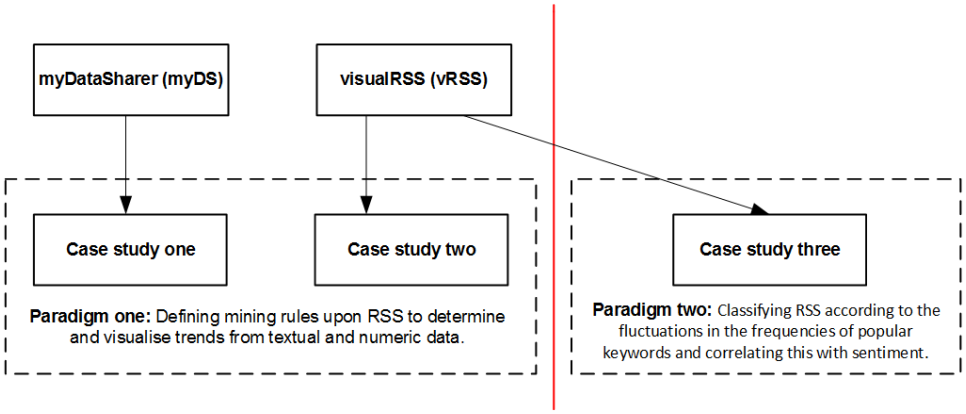


Figure 1.1: The relationship of our RSS-mining paradigms, case studies and software.

1.9 Thesis structure

This thesis is organised into a *series* of five parts according to the relationship of our RSS-mining paradigms, case studies and software (Figure 1.1). In connection with this, Table 1.1 provides a correspondence of the parts and chapters vis-à-vis the paradigms and case studies, after which we briefly describe the parts and chapters.

Part	Chap/ App	Para-digm	Case study	Title
I	1			Introduction
I	2			Background
I	3			A review of RSS
I	4			Web engineering and software architecture
II	5	One	1	The myDataSharer software
II	6	One	1	Mining and visualising textual and numeric data from RSS
II	7	One	2	The visualRSS application
II	8	One	2	Mining and visualising data trends in RSS feeds
II	9	One	1 and 2	Paradigm one and related work
III	10	Two	3	Category-based classification of RSS feeds
III	11	Two	3	Correlating keyword frequencies with sentiment in RSS feeds
III	12	Two	3	Paradigm two and related work
IV	13			Conclusion
V	A			Case study reference materials
V	B			Glossaries
V	C			The Android OS client *app* for visualRSS
V	D			Miscellaneous

Table 1.1: Parts and chapters according to RSS-mining paradigms and case studies.

I. **Introduction and opening remarks**: The first part of this thesis introduces our work and the background to it. There are four chapters:

- Chapter 1 discusses the phenomenon of WWW-based feeds and RSS to provide the motivation, hypothesis and objectives of this thesis. We describe our paradigms, case studies, and software, and also define the function of these items in connection with our objectives and the contribution(s) to web engineering and text mining made. Finally, we describe the *vocabulary* and formatting conventions used in this thesis.
- Chapter 2 provides the *landscape* of this thesis. We begin by defining RSS and continue by describing social media and utility, and other subject areas relevant to our work including data and text mining, sentiment analysis and the use of visualisation to represent data. We also define the concept of *actionable and effective* data in this chapter, but defer web engineering and web application architecture until Chapter 4.

- Chapter 3 reviews the applications and academic research employing RSS in the last decade. Moreover, *much of the related work cited in this chapter is provided for reference purposes only.* We do not compare our paradigms to appropriate related work in this chapter: this comparison is made in the final chapters in Parts II and III respectively, only after the case studies for our paradigms have been presented.
- Before we can present our software and its use in the case studies for our paradigms, it is necessary to consider web engineering and web application architecture because these subject areas define the context for the development of our myDS and vRSS software. This is one of the two purposes of Chapter 4. In addition to this, this chapter concerns the technical nature of common components in our myDS and vRSS applications which we describe and illustrate.

II. **Paradigm 1: Defining mining rules upon RSS:** The five chapters in Part II are concerned with the two case studies which demonstrate our first paradigm (Figure 1.1):

- The first four chapters form two pairs, where each pair of chapters deals with a single case study: Chapters 5 and 6 present case study one, whilst Chapters 7 and 8 focus upon our second case study.
- In Chapter 9 we compare case studies one and two with appropriate examples of related work.

III. **Paradigm 2: Classifying RSS according to the fluctuations in the frequencies of popular keywords and correlating this with sentiment:** Our third case study deals with our second paradigm, and is presented in the three chapters in Part III below:

- Chapters 10 and 11 respectively describe the classification and sentiment analysis components of case study three.
- Chapter 12 compares our third case study with appropriate examples of related work.

IV. **Conclusion and closing remarks:** Chapter 13 presents a summary of this thesis in terms of our motivation, hypothesis and objectives, paradigms and case studies. We also reflect upon the PhD programme and our work, and we discuss the future directions this may take.

V. **Appendices:** This thesis includes four appendices. Appendix A provides reference material for our case studies, whilst Appendix B is made up of two glossaries: B.1 lists the open-source, third-party products used in our software, and B.2 concerns related IT/industry terminology used in presenting our paradigms, case studies and software. Appendix C describes the Android OS client *app* for vRSS developed by Shema [354] for the mobile platform, and Appendix D lists additional resources used in writing our software and information related to this.

1.10 Vocabulary

The terminology below defines the *vocabulary* of our paradigms, case studies, and software:

- **Actionable and effective data:** This term refers to the nature of the outputs produced by the case studies for our paradigms (Section 2.9).
- **Automatic mining:** A mining type in vRSS for our second case study which allows users to select popular keywords in the *rssosphere* and track these from system-indexed categories of RSS feeds.
- **Dataset:** A dataset is a collection of related information made up of distinct elements concerning a given subject (Appendix B.2).
- **Keyword:** Although we briefly refer to them in Section 1.7, the conventions and characteristics of our keywords are discussed in Section 4.6.
- **Manual mining:** Manual mining is a mining type used in vRSS in case study two. It allows users to enter RSS feeds and keywords of their own choice, to perform a *granular* tracking of subjects.
- **Mining:** Described in Section 4.7.5, mining is part of the polling process. For an object which mining rules have become part of, when the RSS feeds defined in the rules are polled for new postings, the text of the postings is mined according to the rules, and data is persisted to database storage to update the object.
- **Mining rules:** Mining rules, the basis of our first paradigm for case studies one and two, are defined upon RSS by users to specify how data is to be mined from feeds during polling to update the objects that the rules become part of, i.e. datasets for case study one and visualisations in our second case study. Mining rules are more extensively described in Section 4.7.2.

- **Mining type:** A mining rule, the term mining type identifies how RSS feeds are mined during polling. Occurrence and value mining were used in our first case study one with myDS (Section 5.4.1). vRSS's three mining types in case study two are automatic, semi-automatic and manual where each was used to select RSS feeds, or feeds and categories thereof, and keywords (Section 7.4.1).
- **Occurrence mining:** OM is a mining type used in myDS in our first case study. It counts the occurrences of strings present in the text of RSS feeds to explore trends or track issues.
- **Paradigm:** We make use of the definition of *paradigm* provided by the OED at `http://www.oxforddictionaries.com/definition/english/paradigm`, i.e. "A world view underlying the theories and methodology of a particular scientific subject". We use this term to describe the two methods we present in this thesis to mine RSS feeds (Section 1.4).
- **Persistence:** We define *persistence* as the writing of the current *state* of an object or data structure to a platform's file system or to database storage. Further information is provided in Appendix B.2.
- **Polling:** The process carried out by myDS and vRSS to: (1) control how frequently RSS feeds are polled for new postings, and (2) the mining of any new postings that have been made to the feeds since they were last polled (Section 4.7.3).
- **Postings:** The addition of new content to an RSS feed by the feed's publisher. Each posting represents a *story*.
- **RSS feeds elements:** Due to the XML-origins of RSS (Section 2.2.3), we define an element of an RSS feed to be either the `<channel>`, an `<item>`, i.e. a posting, or any of their respective constituents, e.g. the `<title>` of an `<item>`. Therefore, when discussing RSS's format, we use the term *element* interchangeably.
- **Semi-automatic mining:** A mining type in vRSS in our second case study allowing users to track their own keywords from system-indexed categories of RSS feeds.
- **Value mining:** VM is a mining type in myDS used in case study one. It analyses RSS feeds which provide structured content. Such feeds are produced by dedicated web sites which report modern forms of ticker-tape (Appendix B.2) data, such as financial movements, sports or lottery results.
- **Visualisation:** A graphical representation of data (Section 2.8).

1.11 Conventions

We *substantially* employ the following conventions in this thesis:

- **Bibliography:** The items in the Bibliography are ordered by author(s), title, publication details and year. URL-based items are listed in name, URL and year order. The use of case in the titles of items listed in the Bibliography corresponds to the titles of actual versions of the items we consulted. Furthermore, where an item is sourced from a conference, we have used the prefix In *Proceedings of the* with regard to the conference's name.

  Several URL-based items refer to teaching materials: where it has proved possible, we have included the subject and details of the relevant course and its venue. In addition to this, the year listed for URL-based items in the Bibliography refers to the year in which the item was published or last modified. If this is not available, the year given refers to when that item was accessed.

  Citations in the body text of paragraphs to items in the Bibliography are denoted by the surname of at least one of the item's authors and a reference number. We have also frequently employed the generic term *authors* in conjunction with an adjacent citation. In such cases, the term *authors* is distinct from our own self-referential use of the alias *author*, and no other meaning is intended or should be inferred.

- **Case study references in caption text:** Captions of algorithms, figures and tables are typically post-fixed with text identifying them to one or more case studies, e.g. Figure *x.y*: *Title*: *cf. case study one.*

- **Code contributions:** We refer to work contributed to our myDS and vRSS software by former Masters students Danani [86], Harriott [167], Shema [354] and Wilce [442], at DCSIS, Birkbeck, University of London. Each contribution took the form of the project required of each student's respective course. In each case, the author was the student's supervisor being responsible for the specification of requirements, project management and communication.

- **Cross references:** Cross references refer solely to items appearing in the contents or lists of algorithms, code and pseudocode, figures and tables, e.g. Section *x.y.* Similar rules are used for lists of algorithms, code and pseudocode, figures and tables. Cross references to appendices refer to the appropriate section of the appendix concerned, except for those to Appendices B.1 and B.2.

If a cross reference does not refer directly to a URL an item in the Bibliography, the reference is to a chapter or section describing the use of an open-source, third-party product: a further reference to the Bibliography or Appendix B.1 is made from there. Sundry cross references are also made to Appendix B.2.

- **Pseudocode:** Pseudocode listed in the algorithms presented is highly abstracted from actual low-level program source code, where the intention is to convey the outline of a particular process. To this end, excluding typical programming control structures, e.g. **for each**/**endfor** or **if**/**endif**, we have used a minimal set of operators, e.g. **add**, **get**, **in**, **of**, **set**, and **write**. Similarly, sub-processes have been *condensed* into simple expressions, e.g.:

  *sort(globalNgrams **in** descending order);*

- **Quotations:** Direct quotations from items listed in the Bibliography and glossaries are frequently used herein. In doing this, we have preserved the punctuation and syntax of the original text including the use of italics and alternative spelling of terms, e.g. we have used *tree-map* (Appendix B.2) whilst *treemap* has been used by Hearst [172] to refer to the visualisation type. In some quotations: (1) words delimited by square brackets have been added for context and clarity, (2) the expression [*sic*] is used to refer to the syntax of the original text, and (3) ellipses, i.e. . . . , have been used to reduce the length of certain quotations.

- **Software written:** We refer to our myDS and vRSS software by using the full names of the applications in titles and captions, but in the body text in chapters, contractions are used: thus, titles and captions include either *myDataSharer* or *visualRSS*, whilst *myDS* or *vRSS* are used in body text.

- **Use of abbreviation RSS:** Our use of the abbreviation RSS is twofold: (1) when we refer to *RSS*, we mean the technology of the medium (Section 2.2), or (2) if a reference is made to a specific RSS feed, we are referring to that feed as an example or to its use in one of our case studies. This referral could be according to a specific feed's URL, content, numbering or other criteria.

Chapter 2

Background

2.1 Foreword

This chapter provides the *landscape* of this thesis. Section 2.2 concerns the technology of RSS which we define together with its history, format and utility. Given the focus in this thesis of RSS within the context of social media, we describe the latter, its applications and the utility derived from it in Section 2.3. Subsequent sections of this chapter address other subject areas relevant to our work: Section 2.4 describes data mining, Section 2.5 considers the classification task of this, and Sections 2.6 and 2.7, respectively, define text mining and sentiment analysis. The use of the medium of visualisation to represent data is the theme of Section 2.8. With these subjects areas described, we then focus upon the nature of the *actionable and effective* data (Section 2.9) produced by the case studies for our RSS-mining paradigms (Section 1.4.1).

Our descriptions of the subject areas listed above are necessarily brief given the mass of material available which is beyond the scope of this thesis to examine.[1] We do not discuss web engineering in this chapter despite its application to our work: this subject is more appropriately described in Chapter 4 together with the architecture and common components of our software.

[1] To demonstrate this, we carried out a Google [148] search for keywords *SVM*, *text* and *classification* on 29 Jan 2016 @ 22 07 GMT which returned 850,000 results: an identical search on Google Scholar [155] returned 141,000 items.

2.2 RSS

2.2.1 A definition

RSS provides an open method of syndicating and aggregating on-line content. Pilgrim [299] has described RSS as an "umbrella term for a format that spans several different versions of at least two different (but parallel) formats." We describe these versions and the history of RSS in Section 2.2.2.

Administrators and other staff of web sites belonging to media or commercial organisations, or simple individuals, create RSS *feeds*, i.e. semi-structured XML documents, which most commonly consist of frequently updated news items, blog entries and multimedia. These XML documents, often having `.rss` or `.xml` file extensions, are published on the servers hosting the web sites and made available by HTTP links as social media. This is because, as described by RSS Specifications [437], "RSS is a free and easy way to promote a site and its content without the need to advertise or create complicated content sharing partnerships." Furthermore, according to RSS Specifications, on the client side "consumers use RSS readers and news aggregators to collect and monitor their favorite feeds in one centralized program or location." Aggregators and readers (Section 3.2) typically present RSS content to users in the journalistic style of *headline*, *story* and *snapshot(s)* (Section 2.2.4).

RSS is a *pull* technology (Ayers and Watt [22]), where the client requests data hosted by a server: the data is then *pulled* down from the server to the client as a *stream of text* (Section 2.8.2). In contrast to this, *push* services see new content *pushed* out to subscribers by the server: an example of this latter type is provided by Twitter [408]. As a type of social media (Section 2.3), the use of RSS is ubiquitous, and its *radio wave* icon displayed in Figure 2.1, or a variation thereof, is a frequent annotation on web sites and browsers. In connection with this, in Chapter 3 we review the applications and academic research employing RSS.

Figure 2.1: The RSS *radio wave* icon.

2.2.2 History and versions

The following chronological history of RSS is inextricably connected with the numerous versions of the RSS format published to date, and is based upon the succinct account

provided by W3Schools [423]. In order to provide additional information, annotations from the more detailed history at Cover Pages [318] and the *three* alternative versions from RSS Specifications [175] are included.

- **1997:** Winer at UserLand Software (`http://www.userland.com/`) developed an XML-based syndication format known as *Scripting News*.
- **1999:** RSS 0.90 was created by Libby and Guha at Netscape: it was known as *RDF site summary* because it included a *header* which used RDF to store metadata. RSS 0.90, which also supported Scripting News, was released in Mar 1999 for the `My.Netscape.Com` portal.
- **1999:** Winer developed Scripting News 2.0b1 which included RSS 0.90 features.
- **1999:** Libby at Netscape developed RSS 0.91 which included most features from Scripting News 2.0b1.
- **1999:** UserLand Software adopted RSS 0.91 and deprecated Scripting News.
- **1999:** Netscape was acquired by mass media corporation AOL.
- **1999:** Netscape discontinued their development of RSS because AOL stopped including external RSS feeds in their services. AOL later drops Netscape in 2007.
- **2000:** UserLand Software released the official RSS 0.91 specification.
- **2000:** Dornfest at O'Reilly [287] developed RSS 1.0, a brand new format using RDF and namespaces, which was unrelated to previous versions.
- **2000:** Winer at UserLand Software developed RSS 0.92 which included RSS 0.91 and optional elements.
- **2002:** After leaving UserLand Software, Winer developed RSS 2.0 which included RSS 0.92 with optional elements.
- **2003:** The copyright of the RSS 2.0 specification [332] was transferred by Winer and UserLand Software to the Berkman Center for Internet & Society at Harvard Law School (`http://cyber.law.harvard.edu/`).
- **2003:** The RSSAB [334] was founded by Winer to maintain the RSS 2.0 specification in cooperation with the Berkman Center at Harvard.

- **2009:** Since 2003 the RSSAB has published several new versions of the RSS 2.0 specification. The current version is 2009's RSS 2.0.11.

The result of this history is that there are a number of versions of RSS which can be grouped into: (1) the RDF (Appendix B.2) or RSS 1.* branch originated by Netscape, and (2) the RSS 2.* branch which descends from Userland Software and now Harvard. W3Schools [423] estimates that about 50% of all RSS feeds use RSS 0.91, and that another 25% use RSS 1.0, with the remaining 25% split between RSS 0.9x versions and RSS 2.0.[2] For convenience in this thesis, we identify RSS to mean any and all of these versions.

2.2.3 Format

RSS is a dialect of XML [52] which, as defined by Walsh [425], provides "a markup language for documents containing structured information." The textual data format of XML allows rules to be specified to encode documents and information using tags, i.e. *elements* and *attributes*, in a semi-structured, tree-like form, which hierarchically represents individual items of data, and the relationships between them. The widespread use of XML is revealed by the numerous *dialects* which have been developed for different applications and initiatives [453]. These vary from a storage medium for files in application suites such as Microsoft Office [260] and Apache's *OpenOffice* [284], to those for feeds such as RSS and Atom (Appendix B.2). Algorithm 2.1 lists XML defining two late 20th century novels to illustrate elements, e.g. `<title>` (3, 9), and attributes, e.g. `publisher` (5, 11).

```
 1: <?xml version="1.0" encoding="UTF-8"?>
 2:     <book>
 3:         <title>Dr Haggard's Disease<title>
 4:         <author>Patrick McGrath</author>
 5:         <pubDetails publisher="Penguin" year="1993"/>
 6:         <description>Doomed love for a married woman becomes a religion:
                        what is wrong with Dr Haggard?</description>
 7:     </book>
 8:     <book>
 9:         <title>In the Eye of the Sun<title>
10:         <author>Ahdaf Soueif</author>
11:         <pubDetails publisher="Bloomsbury" year="1992"/>
12:         <description>It is the great English novel about Egypt, and also
                        the great Egyptian novel about England.</description>
13:     </book>
14: </xml>
```

Algorithm 2.1: XML elements and attributes representing two late 20th century novels.

[2] Several reported cases of incompatibilities between the different versions of RSS are documented in Section 3.5.1.

Version 2.0.11 of the RSS 2.0 specification [332] states that an "RSS document is a `<rss>` element, with a mandatory attribute called version, that specifies the version of RSS that the document conforms to." The specification further states that "Subordinate to the `<rss>` element is a single `<channel>` element": an RSS feed's `<channel>` is made up of three further mandatory elements, i.e. the `<title>`, `<link>` and `<description>`, and a series of other optional elements which contain descriptive metadata (Appendix B.2) about the feed and its contents. The edited text in Algorithm 2.2 represents the principal elements of a typical `<channel>` published by the BBC in their web site's principal RSS news feed at `http://feeds.bbci.co.uk/news/rss.xml`. The `<item>` elements which make up the content, i.e. *postings*, of the feed are denoted in line (9) by the XML comment symbols `<!--` and `-->`.[3]

```
 1: <?xml version="1.0" encoding="UTF-8" ?>
 2: <rss version="2.0">
 3:     <channel>
 4:         <title>BBC News - Home</title>
 5:         <link>http://www.bbc.co.uk/news/#sa-ns_mchannel=rss&amp;ns_source=
                PublicRSS20-sa</link>
 6:         <description>The latest stories from the Home section of the BBC
                    News web site.</description>
 7:         <language>en-gb</language>
 8:         <copyright>Copyright: (C) British Broadcasting Corporation...
            </copyright>

 9:         <!-- Location of <item> elements. -->

10:     </channel>
11: </rss>
```

Algorithm 2.2: Principal elements of an RSS feed's `<channel>` (edited for clarity).

The following list describes several other `<channel>` elements not mentioned in Algorithm 2.2, together with our reasons for not using them in the case studies for our paradigms.

1. **`<lastBuildDate>` and `<pubDate>`:** According to the RSS 2.0 specification [332], the `<lastBuildDate>` element provides the "last time the content of the channel changed." The `<channel>` also includes a `<pubDate>` element which lists the "publication date for the content in the channel."

   We made no use of these temporal elements in our work because of their optionality. For the currency of RSS content, we chose instead to rely upon the `<pubDate>` (Table 2.1) element of each `<item>` in the `<channel>`, even where this may have resulted in

[3] As per Section 1.10, when discussing the format of RSS, we use the term *element* interchangeably to refer to the `<channel>`, `<item>` or any of their respective constituents.

a marginal, but unavoidable duplication based upon the inclusion, by the publisher, of repeating content in a feed: an example of this during the classification component of our third case study is described in Section 10.7.2.

2. `<category>`**:** The `<category>` element in an RSS feed's `<channel>` is optional: therefore, when a feed is published, this element will either be populated with the names of one or more categories that the publisher believes the content of the feed belongs to, or alternatively, the element will be unpopulated.

   In the author's opinion, either option means that the `<category>` element in the `<channel>` is potentially ambiguous given an RSS feed with either a single or multi-category content. This opinion also applies to the `<category>` element found in each `<item>` in the feed's content. Therefore, we made no use of `<category>` elements.

An RSS feed's `<channel>` also contains numerous postings, i.e. *stories*, each of which forms an `<item>` element in the feed: the location of the `<item>` elements in the `<channel>` is given by line (9) in Algorithm 2.2. Each `<item>` is made up of the elements listed in Table 2.1 which are, with the exception of the `<title>` or `<description>`, optional. An `<item>` element usually consists of plain text but HTML, XML and multimedia content may also be present: thus, RSS's format is semi-structured.

Element	**Description**
`<title>`	The title of the `<item>`.
`<link>`	The URL of the `<item>`.
`<description>`	A description of the `<item>`, which can be text, HTML or XML, and may include multimedia.
`<author>`	A URL or email address of the author of the `<item>`.
`<category>`	One or more categories, given to the `<item>` by the publisher.
`<comments>`	The URL of a page for comments relating to the `<item>`.
`<enclosure>`	Describes a media object which is attached to the `<item>`.
`<guid>`	A string that uniquely identifies the `<item>`.
`<pubDate>`	The publishing date/time of the `<item>`, commonly in BST/GMT format as originally defined in the RFC 822 standard for ARPA internet text messages [82].
`<source>`	The RSS `<channel>` that the `<item>` belongs to.

Table 2.1: RSS feed `<item>` elements according to the RSS 2.0 specification [332].

Algorithm 2.3 represents a typical populated RSS `<item>` element. This `<item>`, again originally published by the BBC in their web site's principal RSS news feed on 02 Jan 2015, has been edited so that it lists only those elements we employ in our paradigms and their case studies, i.e. the `<title>`, `<description>` and `<pubDate>` elements within each `<item>` in a feed. Moreover, we regard these elements to be structural metadata (Appendix B.2) because of the organisational role they perform in RSS's format.

```
1: <item>
2:     <title>US sanctions North Korea over Sony</title>
3:     <description>The US imposes new sanctions on North Korea in response to
                    a major cyber-attack against Sony Pictures.</description>
4:     <pubDate>Fri, 02 Jan 2015 20:07:38 GMT</pubDate>
5: </item>
```

Algorithm 2.3: A typical populated RSS feed `<item>` element (edited for clarity).

2.2.4 RSS use and utility

> "The original, and still the most common, use for RSS and Atom is to provide a *content syndication feed*: a consistent, machine readable file that allows web sites to share their content with other applications in a standard way. Originally... this was used to share data among web sites, but now it's most commonly used between a site and a desktop application called a *reader*."

This quotation, taken from the introduction to Hammersley's *Developing Feeds with RSS and Atom* [165], which was published in 2005, identified the fundamental role of RSS in providing a common format to share, i.e. *syndicate*, on-line content, and this remains the contemporary case. Statistics concerning the early use of RSS provided by Gill [140] refer to: (1) the tracking by Technorati [387] of "7.7 million blogs in March 2005, compared with 2 million in March 2004", and (2) (the now demised) Syndic8 [379] reported that the number of feeds they tracked grew from "2,500 in mid-2001 to 286,000 in January 2005." Since that time, RSS applications and tools have become more diversified and widespread because of social media (Section 2.3), as well as applications and academic work employing the technology: we review the latter two of these subject areas in Chapter 3. Contemporary use of RSS is estimated by BuiltWith [55], who in measuring WWW technology trends for 2015, wrote that their records included some "20,518,155 live web sites using RSS" (Figure 2.2).

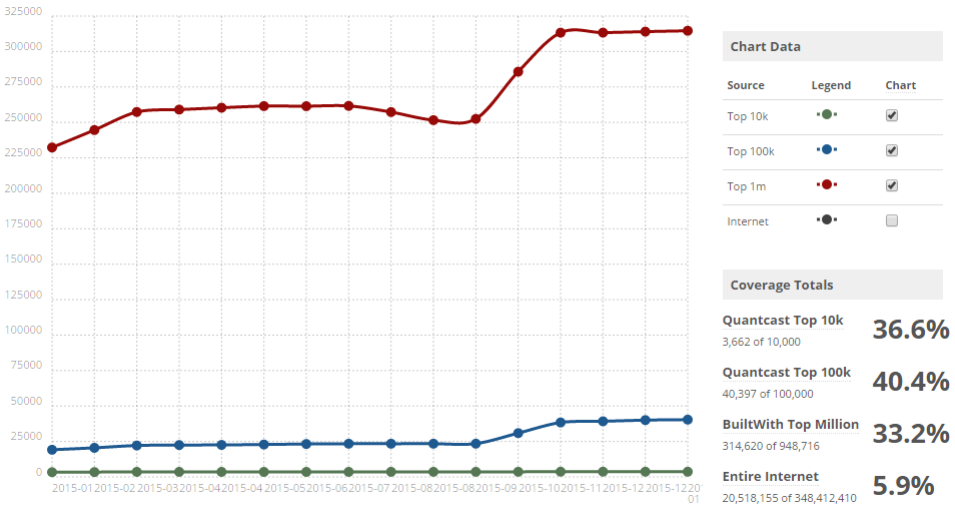


Figure 2.2: Current RSS usage (reproduced from BuiltWith [55]).

Figure 2.3 illustrates the journalistic presentation of RSS content in the style of *headline*, *story* and *snapshot(s)*. On the left, the content is rendered as an `<item>` in a feed in a browser: on the right, the same item is displayed by aggregator/reader Flipboard [127] on an Android OS (Appendix B.1) mobile device. In Algorithm 2.4, the edited text of the `<item>` in the original RSS feed from TechCrunch [386], syndicated by FeedBurner [111] at `https://techcrunch.com/2015/12/15/spotify-poaches-bbc-radio-exec-in-push-for-localised-curated-and-undiscovered-content/`, is displayed.

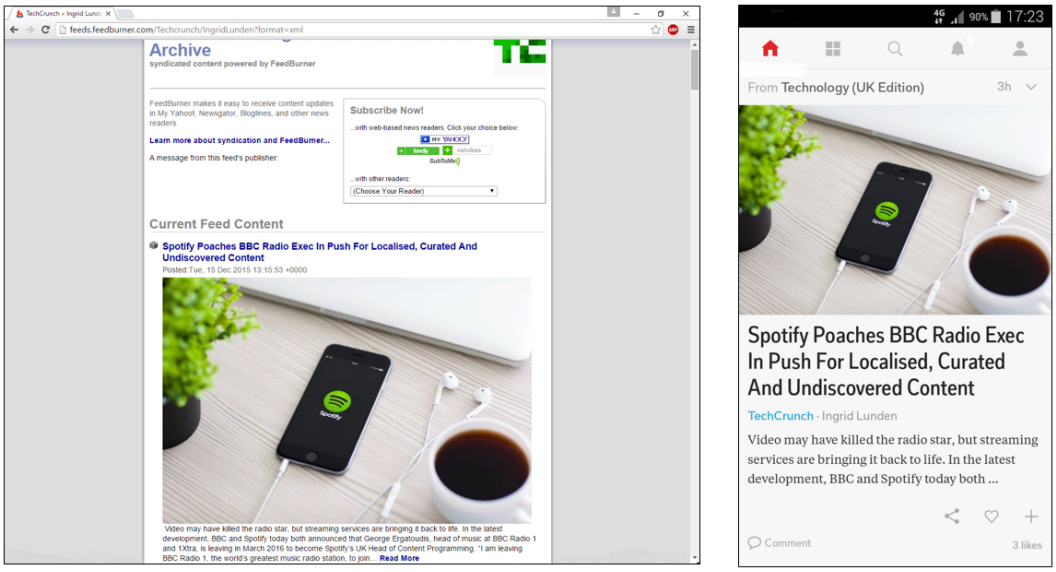


Figure 2.3: The journalistic presentation style of *headline*, *story* and *snapshot(s)*: RSS feed in browser (left), and in a mobile device-based aggregator/reader (right).

```
1: <item>
2:     <title>Spotify Poaches BBC Radio Exec In Push For
              Localised, Curated And Undiscovered Content</title>
3:     <description>Video may have killed the radio star, but streaming
                    services are bringing it back to life...</description>
4:     <pubDate>Tue, 15 Dec 2015 13:15:53 +0000</pubDate>
5: </item>
```

Algorithm 2.4: The RSS feed `<item>` element of the content displayed in Figure 2.3 (edited for clarity).

At `http://www.oxforddictionaries.com/definition/english/utility` the OED defines *utility* to mean "The state of being useful, profitable, or beneficial". Based upon this, RSS must permit a "useful", "profitable, or beneficial" utility to both publishers and consumers. We define RSS's utility by summarising the "social, spiritual, and mercenary" reasons given by Hammersley [165] to explain why organisations and individuals syndicate content in feeds: (1) to increase web site traffic, and to (2) to build up a web site's "brand awareness". Both of these elements also assist in placement in search engine listings and allow the building of a (3) "community of sites". The addition of new technologies allows (4) "others to give additional features to your service", where this (5) "makes the Internet an altogether richer place, pushing semantic technology along and encouraging reuse."

In addition to Hammersley [165], Singh and Sahu [360] have also identified benefits for both publishers and consumers. We summarise these benefits where RSS: (1) keeps users up to date with news or other media, (2) provides subscriptions for individual users, (3) presents article summaries, (4) lacks spam received via email, (5) has a simple subscription/unsubscription model, (6) increases web site traffic because feeds help with SEO, and (7) for advertising and marketing. Many of these reasons for the utility of RSS concern social media, and it is for that reason that we look at this subject in the next section (Section 2.3).

2.3 Social media

2.3.1 Defining social media

Although, a massive amount of information is available to do with social media, in this thesis we are required only to provide an overview of the subject area: we do this by asking four questions: (1) *what is it?* (2) *what services does it provide?* (3) *what are these services used for?* and (4) *why are these services used?*

Articles on web pages, or web sites are frequently annotated with plug-ins that display icons representing social media services. An example of this can be seen in Figure 2.4 for IT developers site DZone at `https://dzone.com/` in late 2015. In this example, from left to right, the icons represent DZone's RSS feeds, Twitter [408], Facebook [106], Google+ [149] and LinkedIn [227].

Figure 2.4: The social media icon strip from the home-page of IT development web site DZone at `https://dzone.com/` in late 2015.

We refer to the concise definition of social media provided by WhatIs at `http://whatis.techtarget.com/definition/social-media` to answer our first question, i.e. that social media is "the collective of online communications channels dedicated to community-based input, interaction, content-sharing and collaboration." These "online communications channels" are enabled by *Web 2.0* (Appendix B.2) technologies and accessed using a browser or mobile device.

2.3.2 Applications

Our second and third questions concern the applications and services provided by social media. These include:

- **Blogging:** Where web sites permit blogging or micro-blogging. These include Blogger [41], Drupal [99], Joomla [208], LiveJournal [234], Tumblr [407], Twitter [408] and WordPress [450].
- **Content aggregation:** As described in Section 3.2.
- **E-commerce:** Facilitated by on-line banking and retail service providers such as Amazon [16] and Ebay [101].
- **Education:** In the use of VLEs or on-line course/MOOC providers, e.g. Blackboard [39], Coursera [79], Moodle [269], the Open University [285], Udacity [409] and Udemy [410]. Another branch of educational use is the *sharing* of academic content by Academia [2], Mendeley [256] and ResearchGate [325].
- **Information:** Where postings are made concerning particular reference subjects, e.g. Encyclopædia Britannica [103], Scoop.it [348], Techopedia [388], Webopedia [432] and Wikipedia [440].

- **Instant messaging:** IM text or video messaging facilities include Facebook Messenger [107], Skype [361], SnapChat [363] and WhatsApp [438].

- **Personalisation and recommendation:** Defined by Levene [224], this concerns the customisation of web sites, social media or mobile devices, "by users explicitly specifying their preferences, or implicitly through collection of data about the users from the log of their searches." This collection of data is frequently based upon recommender systems[4] employing user preferences and records of past activities, e.g. Amazon [16] or Ebay [101], or where a social media service recommends content to users. Two well-known techniques used to generate recommendations are: (1) collaborative filtering, i.e. recommendations based upon known preferences of a group of users, and (2) content-based filtering which recommends content based upon user profiles.

- **Sharing:** Media sharing allows users to upload photographic, text, multimedia or other content. This content is then *shared* with other users. Examples include BBS software (Appendix B.2) as well as [24], [98], [126], [128], [193], [195], [196], [197], [199], [229], [272], [261], [302], [368], [375], [417] and [461]. A variation on sharing is the posting of comments to articles on web sites of newspapers and other periodicals using Disqus [95] or equivalent services.

- **Social networking:** Defined by Boyd and Ellison [50] these are "web-based services" allowing users to: (1) "construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system. The nature and nomenclature of these connections may vary from site to site." On this basis, social networking could extend to include all of the social media services listed in this section, although for convenience we restrict the term to refer to popular services such as Bebo [29], Facebook [106], Google+ [149] and Twitter [408]. More dedicated web sites include LinkedIn [227], questions and answers web site Quora [314], and the family of web sites at StackExchange [370].

  Two subsets of social networking concern: (1) services for meeting people such as Match [250], Lovestruck [237] or Tinder [400], and (2) *social bookmarking* where users *bookmark* web pages or other content, and share it with other members. Examples include BibSonomy [37], CiteSeer [73], Delicious [91], Reddit [319] and StumbleUpon [376].

[4] We consider the use of RSS by recommender systems in Section 3.4.

- **Streaming:** Despite an overlap with sharing above, streaming video services offering films or television series are provided by Amazon Prime [17], NetFlix [276], TalkTalk TV Store [383] (formerly Blinkbox), and (inter)national television networks. Similarly, *podcasts* provide audio streaming (Section B.2).

2.3.3 Utility and use

This subject concerns our last question, i.e. *why are these services used?* This question addresses the utility of social media, and builds upon our definition of RSS's utility in Section 2.2.4.

The *uses and gratification* theory (Blumler and Katz [43]), as explained by Lane [219], "suggests that media users play an active role in choosing and using the media. Users take an active part in the communication process and are goal oriented in their media use. The theorist say that a media user seeks out a media source that best fulfills the needs of the user. Uses and gratifications assume that the user has alternate choices to satisfy their need."

In a review of uses and gratification theory applied to social networking web sites, Gallion [133] listed *socialising*, *entertainment*, *self-status seeking* and *information* as the reasons why people use social media. Williams and Whiting [439] further referred to "the application of uses and gratification theory to social media research" as a means to "explain the many and varied reasons why consumers use and like social media." We summarise the "varied reasons" given by Williams and Whiting as follows: (1) *social interaction* to maintain contact or interact with others, (2) *information seeking* to learn or for awareness, (3) to *pass time* where users have "idle time or when they are bored", (4) *entertainment* is concerned with enjoyment, (5) *relaxation* where this focuses on stress relief, (6) for the *expression of opinions*, (7) as a *communicatory utility* where social media provides material for communication between related groups of people, (8) a *convenience utility* which is always available, and (9) *information sharing* about individuals or groups of people sharing information about themselves with others.[5] In connection with this, Tamir and Mitchell [384] cited Naaman et al. [273] with reference to surveys of internet use indicating "that upwards of 80% of posts to social media sites (such as Twitter) consist simply of announcements about one's own immediate experiences." Williams and Whiting [439] also listed one further reason for the use of social media, i.e. *surveillance/knowledge about others* in watching or finding out about other people and their activities on-line.

[5] Examples of this include the posting on-line of *selfie* photographs, and *sharenting* where parents post pictures of their children on social media.

Froget et al. [131] also listed similar reasons for the use of Facebook [106]. Commercial uses of social media cited by Baker [23], include building up a business and promoting it upon establishment, and learning about other employment/employer availability.

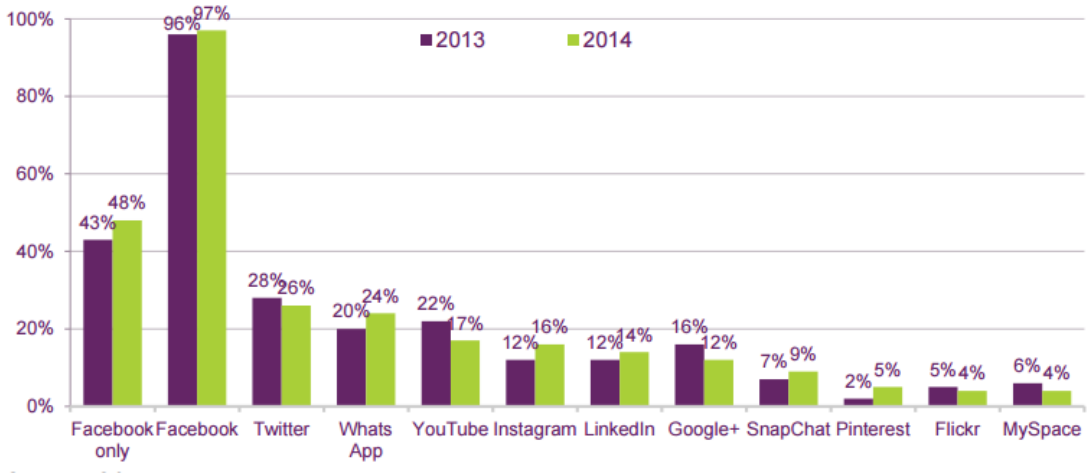


Figure 2.5: UK use of social media in 2013 - 2014 (reproduced from Ofcom [283]).

The popularity of social media, and the utility derived from it, can be measured by the use of the applications providing it and their types of services (Section 2.3.2), where these are browser- or mobile device- based. Figure 2.5 displays use of appropriate services by UK adults, i.e. sixteen years or older, with a social media profile in 2013 - 2014 (Ofcom [283]).

2.4 Data mining

Stated simply, data mining is the analysis of existing data to discover patterns. These patterns, according to Witten and Eibe [447] must be "meaningful in that they lead to some advantage, usually an economic advantage." Data in data mining is also typically quantitative especially when we consider the exponential growth in data produced by social media (Section 2.3) in recent years, i.e. *big-data* (Appendix B.2). With reference to Martin [248], applications of data mining may vary between domains but include fraud detection and e-commerce, gaming and financial services, as well as scientific applications such as analysing X-ray images and modelling gene behaviour. Whatever the application, according to Witten and Eibe [447], the search for patterns in data "is automated—or at least augmented—by computer."

The "unifying goal" of this *knowledge discovery in databases* (KDD) process is defined by Fayyad et al. [108] as "extracting high-level knowledge from low-level data in the context of large data sets." We summarise the iterative stages of this process, with reference to Figure 2.6, as follows: (1) identifying the end user's goals by understanding the application's domain and prior knowledge, (2) the creation of a target dataset (Appendix B.2), or samples, upon which discovery is to be performed. This requires pre-processing (3) where the data is transformed by data cleansing or ETL in order to remove "noise" and resolve any missing or temporally-based data. (4) concerns data reduction and projection to determine useful features to represent the data depending on the goal of the task, and to reduce the number of variables. (5) is when the data mining method is chosen, e.g. clustering or classification, whereupon (6) selects the actual data mining algorithm to be used. (7) performs the actual data mining where the search for patterns in data is carried out. The outputs of (7) are evaluated and interpreted in (8) which may require the repeating of any or all of the previous stages to actually discover knowledge. Finally, stage (9) sees action taken on the discovered knowledge based upon the domain's requirements.

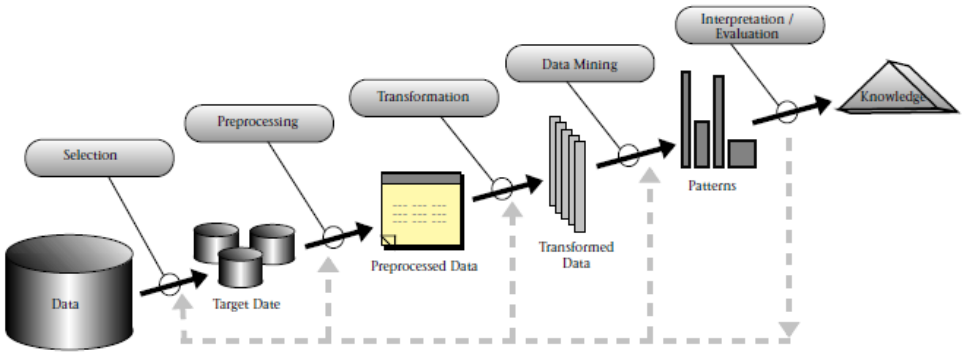


Figure 2.6: The *knowledge discovery in databases* process (reproduced from Fayyad et al. [108]).

Fayyad et al. [108] further wrote that "the two high-level primary goals of data mining in practice tend to be prediction and description." The authors described the boundaries between these goals as being "not sharp", where *prediction* predicts "unknown or future values of other variables of interest", and *description* which focuses on "finding human-interpretable patterns describing the data." Both goals can be achieved using "a variety" of data mining methods which include: (1) *classification* where, given a set of classes, we

need to determine which class a new sample will belong to, (2) *regression* which models the data with the least error, (3) *clustering* where members of a set are grouped according to similarity measures (Appendix B.2), (4) *summarisation* represents the data or subsets of it, (5) *dependency modelling* seeks to model dependencies between variables, and (6) *change and deviation detection* concerns discovering changes in data since it was previously measured.

2.5 Classification

There are two principal variants of classifying data:[6]

1. ***Supervised* learning:** Where classification tasks make use of training data instances which are labelled with one of a finite set of classes. This data is then used to *train* a classifier, i.e. a technique, to produce a *model*. The model is subsequently evaluated on testing data to determine which class the individual members of the test data belong to. Well-known supervised techniques include DT, MNB and SVM, and feature selection methods such as chi-square (Appendix B.2). According to Aggarwal [9], applications of classification techniques include multimedia, text and time-series data.

2. ***Unsupervised* learning:** Applications of unsupervised learning include pattern recognition, recommender systems and neural networks. One common technique employed is known as *clustering* where unlabelled data is grouped, i.e. clustered, during the classification process based upon similarity measures. In k-means clustering, this process is iterative until a termination condition, i.e. a pre-defined convergence criterion or a fixed number of iterations, is met.

In the classification work for the first component of our third case study (Chapter 10), we apply the aforementioned supervised DT, MNB and SVM techniques to RSS in order to classify feeds into categories based upon the fluctuations in the frequencies of popular keywords present in their text. We do not describe these classifiers or the reasons for our selection of them here because these subject areas are more appropriately discussed in Section 10.5. Nevertheless, it is necessary to comment upon the nature of text mining in connection with case study three: we discuss this subject in the next section.

[6] cf. Phan et al. [298], Witten and Eibe [447] and Zhu et al. [467], who refer to *semi-supervised* learning as a third variant where both unlabelled and labelled data are used during classification.

2.6 Text mining

Hotho et al. [181], with reference to the Feldman and Dagan's [118] "Knowledge Discovery in Text (KDT)", defined text mining as dealing:

> "with the machine supported analysis of text. It uses techniques from information retrieval, information extraction as well as natural language processing (NLP) and connects them with the algorithms and methods of KDD, data mining, machine learning and statistics. Thus, one selects a similar procedure as with the KDD process, whereby not data in general, but text documents are in focus of the analysis."

As Hearst wrote [171], "large text collections as a resource to be tapped" may be structured, e.g. in an RDBMS or XML documents, or unstructured multimedia content discussed by Kappel et al. [209], with reference to Anastopoulos and Romberg [18] and Ceri et al. [62], which may be produced by social media (Section 2.3).

Gupta and Lehal [163] identified nine methods of text mining. Briefly described, these methods are: (1) *information extraction* which identifies key phrases and relationships within text by using pattern matching to infer relationships between people, places and time according to pre-defined sequences in text (Section 2.8.2), (2) *topic tracking* which informs users of new topics based upon their previous viewing of a document(s), (3) *text summarisation*, using sentence extraction or topic tracking, to provide document summaries, (4) *categorisation* or classification where a document may be classified as a bag-of-words (BoW) (Appendix B.2), (5) *clustering* to group documents, (6) *concept linkage* connects related documents by identifying common themes or subjects, (7) *information visualisation* (Section 2.8), (8) *question answering* where NLP (Appendix B.2) is used to find the best answer to a question, and (9) *association rule mining* (Appendix B.2).

Applications of text mining include sentiment analysis (Section 2.7), news filtering and organisation, document retrieval and classification, as well as email filtering (Aggarwal and Zhai [11]), in the domains of publishing, telecommunications, IT, finance, politics and pharmaceuticals. Moreover, these applications can be grouped into the "four main typologies" discussed by Bolasco et al. [44], i.e.: (1) knowledge management and human resources, (2) customer care and customer relationship management, (3) technology, and (4) NLP.

2.7 Sentiment analysis

Sentiment analysis, also called *opinion mining*, has been defined by Liu [230] as:

> "the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes."

As described by Medhat et al. [253], sentiment analysis is an "ongoing field of research in text mining". In the second component of our third case study (Section 2.7), we make use of sentiment analysis to determine a correlation between the sentiment in the text of RSS feeds and the fluctuations in the frequencies of keywords representing named entities (Appendix B.2), where the results can be applied to trend analysis. It is necessary to state that the black-box nature of our sentiment analysis work did not require the development by the author of a sentiment analyser. Nevertheless, Figure 2.7 represents a generic sentiment analysis architecture, and the following paragraphs provide a précis of sentiment analysis methods and applications. These methods and applications are discussed in more detail in many of the surveys carried out in recent years which include [78], [85], [204], [230], [253], [274] and [292]. We refer to RSS-based related work employing sentiment analysis in Section 3.3.3.

Collomb et al. [78], Liu [230] and Medhat et al. [253] have referred to three levels of sentiment analysis:

1. **Document level:** Whether an entire document can be positively or negatively classified, where the document necessarily focuses on a single subject. The weakness of this approach is that there is no fundamental difference between document- and sentence- level sentiment analysis because, as Liu [230] described them, "sentences are just short documents."

2. **Sentence level:** This approach determines if each sentence in a document has a positive, neutral, i.e. no, or negative, sentiment. Point of view is an important consideration here because, although a sentence can be either subjectively positive or negative, sentiment is not necessarily subjective (Wilson et al. [445]).

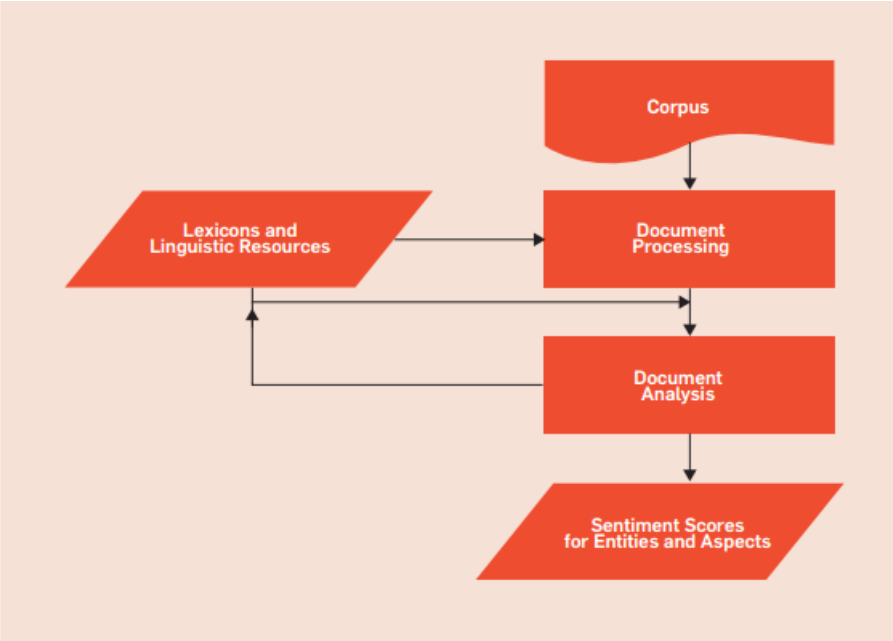


Figure 2.7: A generic sentiment analysis architecture (reproduced from Feldman [117]).

3. **Aspect level:** This is also known as *word* or *feature* level, and looks at the opinion, which is either positive or negative, and a *target*, i.e. the subject, of the opinion. Rohrdantz et al. [328] wrote that "Most approaches for feature-based sentiment analysis involve three or four consecutive steps." These steps involve: (1) features for different sentiment "targets" which are generated automatically from the text corpus, or are based upon other methods including pre-defined keywords or part-of-speech (POS) (Appendix B.2), (2) sentiment words which "evoke positive or negative associations" are searched for in the documents, (3) a "mapping strategy" relates sentiment words to features to allow sentiment to be rated, and (4) a visual representation of the feature-based results permitting interactive exploration of the results.

   An example of aspect-level sentiment analysis is provided by the sentence *I love Star Trek but I really hate Star Wars*. Two sentiments are produced, i.e. *love* and *hate*, and two aspects, i.e. *Star Trek* and *Star Wars*. Therefore, we can summarise aspect/opinion sentiment for analytical purposes despite issues of emotion and disambiguation.

A number of methods used for sentiment analysis have been documented by [78], [230] and [253]. The *machine learning* method uses supervised learning (Section 2.5) techniques to determine sentiment by training a known dataset. Application of DT, NB, SVM and

other classification techniques to sentiment analysis have also been documented by [54], [58], [147], [159], [270], [365] and [418], and in several of the surveys listed in Section 2.6.[7]

Lexicon-based sentiment analysis involves calculating the sentiment *polarity* of a piece of text, where positive and negative sentiment are produced by the use of *lexicons* of positively and negatively weighted "opinion words and phrases" (Medhat et al. [253]). SentiStrength, used in our sentiment analysis component of our third case study and described in Chapter 11, is a lexicon-based analyser using weighted opinion words, e.g. *I love you* scores 3, but when qualified by a *booster* word, *I really love you*, scores 4. Medhat et al. also identified three ways of compiling the lexicon, i.e. (1) manually, (2) using a limited manual lexicon augmented by on-line services, or by (3) employing a corpus. Augustyniak et al. [21] wrote that "simplistic Bag of Words (BoW) lexicon methods for sentiment polarity assignment with ensemble classifiers are much faster than a supervised approach to sentiment classification while yielding similar accuracy."

Collomb et al. [78] described the rule-based method as looking in text for "opinion words" and then classifying the text "based on the number of positive and negative words. It considers different rules for classification such as dictionary polarity, negation words, booster words, idioms, emoticons, mixed opinions etc."

Statistical methods of sentiment analysis were also described by Collomb et al. [78], where these methods represent each review as a combination of aspects and ratings: it is "assumed" that these items "can be represented by multinomial distributions and try to cluster head terms into aspects and sentiments into ratings." Medhat et al. [253] also refer to the use of statistical methods to find "co-occurrence patterns or seed opinion words", or by studying the frequency occurrence of words. Hybrid approaches have also been documented, e.g. the use by Weichselbraun et al. [435] of machine learning and lexical analysis to identify ambiguous terms and storing them in "contextualized sentiment lexicons", where in "conjunction with semantic knowledge bases, these lexicons help ground ambiguous sentiment terms to concepts that correspond to their polarity."

Applications of sentiment analysis are of course *legion*, varying from sentiment gathered from reviews, e.g. relating to film on IMDb [196] or products on Amazon [16], postings to Facebook [106], micro-blogging on Twitter [408], RSS feeds, blogs or other social media (Section 2.3). With reference to Feldman [116], these applications include: (1) businesses and organisations which require consumer opinions to do with products they market and services they produce, (2) individuals who make decisions to purchase products or services

[7] cf. Hotho et al. [181] and Aggarwal and Zhai [11], where both refer to the application of DT, NB and SVM classifiers in text mining: we employ these classifiers in the classification component of our third case study (Chapter 10).

based upon *word of mouth* or on-line reviews, or to find public opinion, e.g. concerning politics or local issues, (3) on-line advertising where in social media, an organisation may place an advertisement in response to a favourable review of a product, or a rival product could be advertised upon receipt of a bad review, and (4) opinion retrieval for general searches of opinions.

Lastly, we must mention the problems of sentiment analysis. As Liu [230] wrote, "sentiment analysis is a NLP problem. It touches every aspect of NLP, e.g. coreference resolution, negation handling and word sense disambiguation". Medhat et al. [253] have also cited the lack of resources for Middle Eastern languages including Arabic.

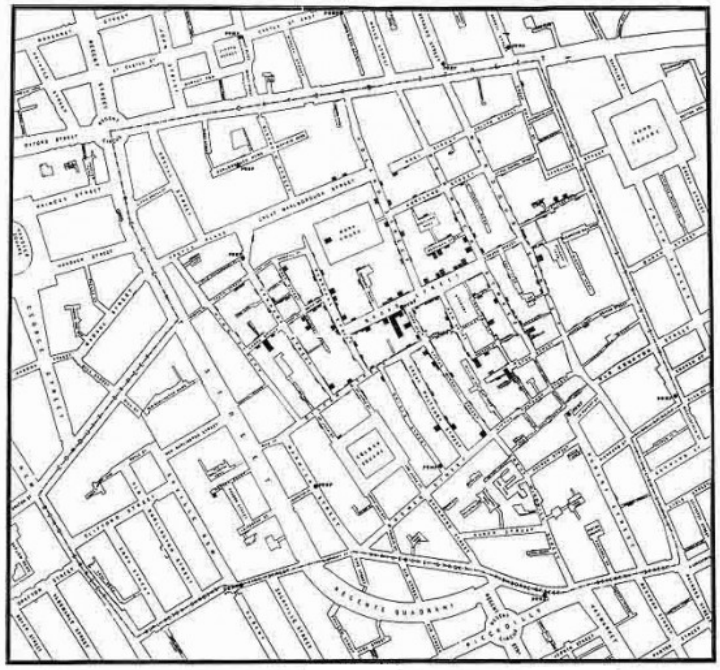
2.8 The visual representation of data

2.8.1 Principles

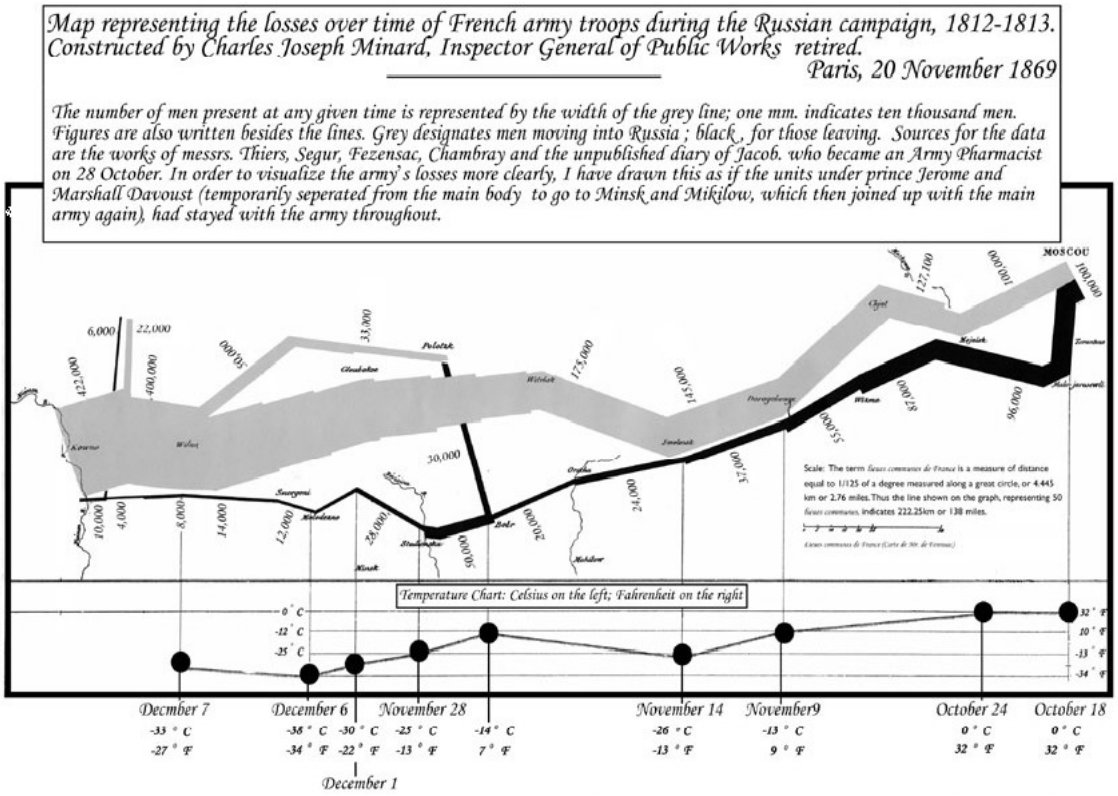
According to Simoff et al. [359] "the visual data mining process relies on visualisation and the interaction with it". In order to achieve this, an effective visual representation of data must, as Tufte [406] wrote, "visually display measured quantities of social data by means of the combined use of points, lines, a coordinate system, numbers, symbols, words, shading and color." Furthermore, Tufte wrote that:

> "It was not until 1750-1800 that statistical graphics-length and area to show quantity, time-series, scatterplots, and multivariate displays-were invented, long after logarithms, Cartesian coordinates, the calculus and the basics of probability theory. The remarkable William Playfair (1759-1823) developed or improved upon nearly all the fundamental graphical designs, seeking to replace conventional tables of numbers with the systematic visual representations of his "linear arithmetic." "

Two classical examples of these principles are: (1) the mapping by Dr John Snow (1813 - 1858) of the victims of the 1854 Broad (now Broadwick) Street cholera outbreak in London's Soho (indicated by the dots on Figure 2.8a), which permits "graphical analysis" despite lacking any reference to population density, and (2) the 1869 map by French engineer Charles Joseph Minard (1780 - 1871), displayed in Figure 2.8b, which combines data, map and time-series (Section 2.8.3) to portray the losses suffered by Napoleon's Grand Army in 1812. It is through this "graphical excellence" that visualisations reveal interesting data to truthfully communicate complex ideas with, as Tufte [406] wrote, "clarity, precision and efficiency."



(a) The Soho cholera outbreak in London in 1854 by Snow.



(b) Minard's representation of Napoleon's retreat from Moscow in 1812 (translated).

Figure 2.8: Two classical examples of the visual representation of data.

Ware [429] defined the process of contemporary data visualisation as consisting of four iterative stages, which we summarise as follows: (1) data collection and storage, (2) pre-processing of data to transform it into a comprehensible format, (3) the hardware and software to render an image on screen, and (4) human perception and cognition. This cognition is primarily visual and it is frequently conveyed by the use of information graphics, or *infographics* (Appendix B.2). Common contemporary examples of infographics include: (1) Harry Beck's map of the London Underground in 1931 [405], originally based upon circuit diagrams and which has since been duplicated globally, (2) the use of traffic lights and street signs which give instructions to road users and pedestrians alike, and (3) in the displaying of weather, satellite navigation or other dedicated on-line mapping tools such as Google Maps [152] and Bing Maps [38]. In IT itself, infographics are used to represent database designs, application architecture and software processing by employing diagram types such as DFDs and ERDMs, and those used in SSADM, UML and other design methodologies.

A frequent use of infographics is to provide a visual representation of a corpora of text documents, which may be temporal in nature. We discuss two alternative approaches to this in the following sections of this chapter.

2.8.2 Text streams

In discussing information visualisation for text mining, Hearst [172] referred to three analytical issues of concern: (1) "visualizing connections among entities within and across documents", (2) methods for visualising "occurrences of words or phrases within documents", and (3) visualising "relationships between words in their usage in language and in lexical ontologies." These issues apply to text streams as a subset of text documents.

In their review of the visualisation of text streams in 2010, Silić and Bašić [358] defined three types of text data: (1) a "collection of texts", (2) a "single text", and (3) a "short interval of a text stream" arriving in real time, where the last of these is "used to visualize trends in texts" in real-time. Streaming is defined by Luo et al. [239]:

> "By "streaming" we mean that all text documents are divided into a sequence of batches based on their time stamps and the intake and processing of the documents are in a batch-by-batch manner. By "incremental" we mean that processing the current batch of data does not involve reprocessing of data in previous batches, and its processing results are seamlessly merged into the final outputs."

Unstructured text in text streams is not considered by Silić and Bašić [358] to be "suitable for visualization, so a text is usually represented in [the] *vector space model*." Although they refer to alternative approaches, Silić and Bašić cite BoW as "an instance of VSM", which produces a word frequency "vector in the space of features, which correspond to the words found in the text." The authors listed five methods of feature selection for use when visualising text streams:

1. **Bag-of-words:** BoW can be improved by removing stop-words, disambiguation or n-gram extraction.
2. **Entity recognition:** Using NER (Appendix B.2) to identify entities, and relationships between them.
3. **Summarisation:** To present the most relevant information, techniques include keyword extraction, keyword assignment, thematic categorisation, and fact extraction.
4. **Document structure parsing:** To visualise *structural* data such as the title, author, and publication date.
5. **Sentiment and affect analysis:** To emotionally characterise the content of texts (Section 2.7).

Silić and Bašić further divided text visualisation methods into two categories:

1. **Semantic space:** Where "the vectors representing texts are of high dimensions because textual features are numerous, so dimensionality reduction techniques are employed in order to map these vectors to 2D or 3D space." The authors wrote that "For now, most methods that use the semantic space approach enable trend discovery" by "time slicing" to create a series of time-based views which can be analysed to determine "changes in the text stream."
2. **Term trend:** For trend analysis (Appendix B.2) in text streams using the "plot frequencies of important terms found in texts at a given window of time." Feature selection can be employed to reduce the amount of data to be displayed, and this can vary from simple keyword frequencies "to more complex statistical measures of feature importance such as information gain or χ^2". A frequently cited example of the term trend approach is *ThemeRiver* by Havre et al. [170] which displayed a

representation of a *river* in order to visualise the changes in frequencies of keywords over a period of time: this is illustrated in Figure 2.9.[8]

The "term trend" approach is often employed in conjunction with sentiment analysis (Section 2.7), e.g. to determine sentiment relating to a particular trend, topic or event, and it is to this subject that we return in Section 3.3.3 when we consider academic research involving the application of sentiment analysis to RSS. Applications of text stream visualisation are described briefly in Section 2.6.

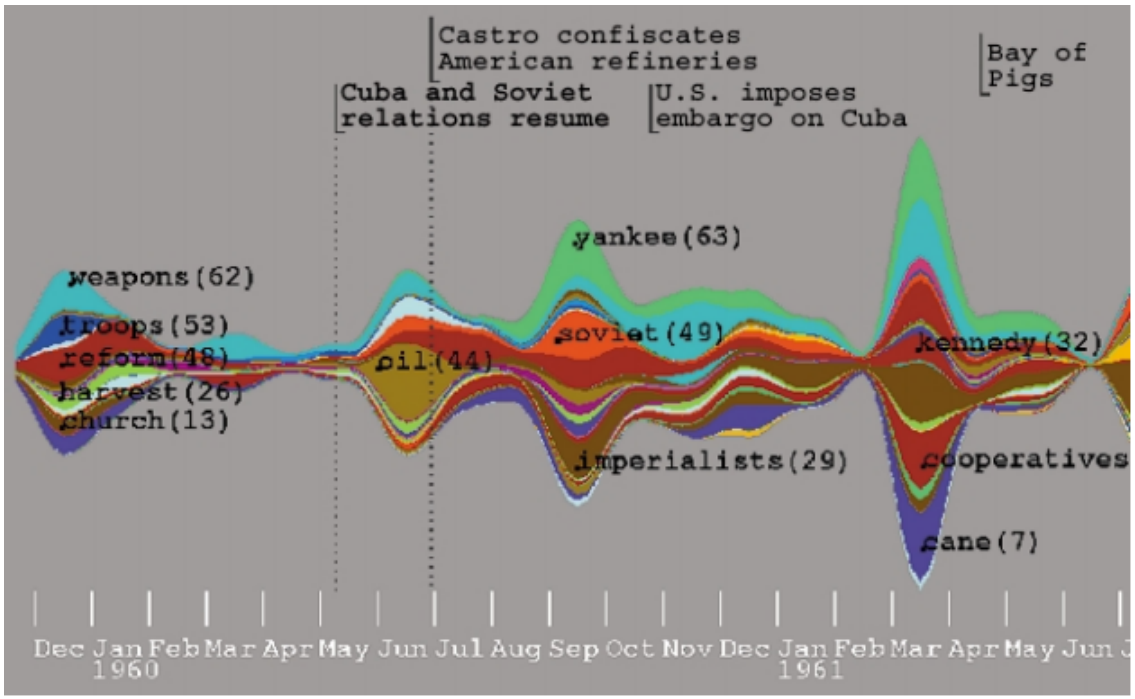


Figure 2.9: *ThemeRiver* displaying data concerning Cuban leader Fidel Castro between Nov 1959 - Jun 1961 (reproduced from Havre et al. [170]).

[8] cf. Wanner et al. [428], who in a survey of visual analytics for event detection in 2014 (described in Section 3.3.3), identified that fifteen of the fifty-one works surveyed made use of the *river* motif.

2.8.3 Time-series plotting

A time-series plot is a set of data points typically measured over a period of time and arranged chronologically. Time-series plots, used since the late 1700s according to Tufte [406], can be *univariate*, i.e. single variable, or *multivariate*, i.e. multi-variable, and the data points represented may be measured at fixed or varying lengths of time. The description provided by Investopedia at `http://www.investopedia.com/terms/t/timeseries.asp` states that:

> "Time series analysis can be useful to see how a given asset, security or economic variable changes over time. It can also be used to examine how the changes associated with the chosen data point compare to shifts in other variables over the same time period.
>
> For example, suppose you wanted to analyze a time series of daily closing stock prices for a given stock over a period of one year. You would obtain a list of all the closing prices for the stock from each day for the past year and list them in chronological order. This would be a one-year daily closing price time series for the stock."

The four main components of a time-series plot are, according to Adhikari and Agrawal [6]: (1) *trend* where the observable movement of the data measured over time will increase, decrease or simply stagnate, (2) *seasonal variations* which vary according to climate, or cultural traditions, e.g. pre-Christmas preparations in *Christian* countries, (3) *cycle* which relates to circumstances that repeat over a term of years such as the boom/slump economic cycle, and (4) *irregular components* which concern the effects of natural catastrophes or human actions including wars, strikes and political change.

In considering "the effects of these four components", Adhikari and Agrawal referred to two models of time-series plots: (1) the *multiplicative* model which is "based on the assumption that the four components of a time series are not necessarily independent and they can affect one another", and (2) the *additive* model wherein "it is assumed that the four components are independent of each other."

In Section 1.4 we identified our paradigms with the exploration of trends. Moreover, several of the visualisation types employed in our case studies serve as time-series plots. The x - y charts used in case studies one and two (Section 4.7.7) are a combination of uni- and multi- variate plots. Figure 2.10 displays a sample multivariate plot from the results of our first case study (Section 6.3.3) which illustrates keyword frequency data for a twelve hour period. Multivariate plots are also provided by the keyword frequency/sentiment

plots in our sentiment analysis work in Section 11.8.4 and Appendix A.3.4. Examples of related work in Section 3.3.3 also employ time-series plots to detect trends, topics or events, whilst others employ the *river* motif illustrated in Figure 2.9. A detailed study of the use of visualisation and visual analysis for visualising "time-oriented data" is provided by Aigner et al. [12].

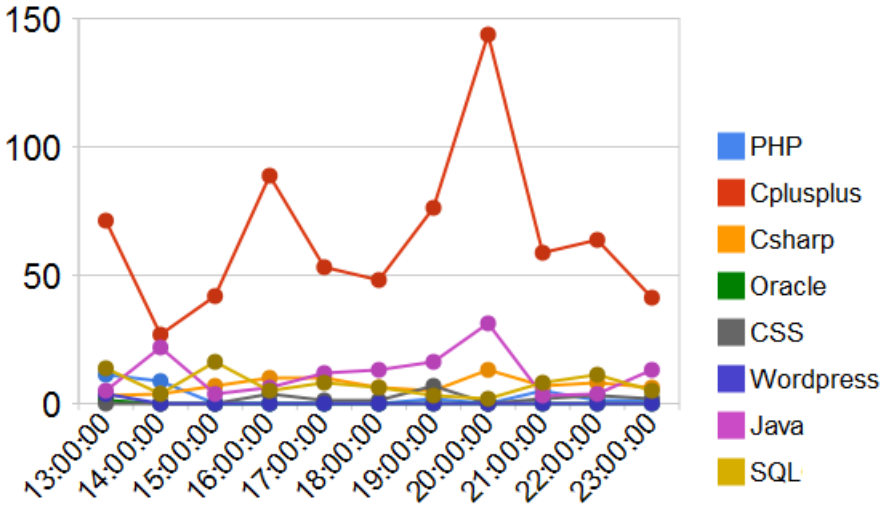


Figure 2.10: Time-series plot of frequencies of questions asked for various programming languages at `http://www.stackoverflow.com/feeds` between 13 00 - 23 00 on 30 Nov 2009: cf. *case study one*.

2.8.4 Software

Many web sites provide tools for data visualisation. Examples of these include CanvasJS [59], FusionCharts [132], Google Charts,[9] GraphViz [160] and VisualComplexity [420]. Several of these products are included in recent surveys by Kuzniewicz [216], Machlis [241] and Suda [377] of tools for data visualisation and analysis which concern business intelligence: Microsoft Excel [259] and Tableau [380] extend this into data mining (Section 2.4). Related to these products are the visualisation facilities provided by the data mining tools described in Section 10.5. It is also necessary to mention several former web sites that tended towards *data-sharing*, e.g. DabbleDB [83], Many-Eyes [245] and Swivel [378], where users were able to upload data, visualise it and allow comments to be added to it.

[9] We refer to Google Charts (Appendix B.1) in Section 4.7.7 in the review of products for use in the case studies for our first paradigm.

2.9 Actionable and effective data: a definition

The major theme of the aforementioned subject areas is the mining and sentiment/visual analysis of data within the context of social media. Social data is the sum of the information produced daily by millions of people globally as they participate in the use of various types of social media services (Section 2.3.2). A more *granular* definition is provided by Vis [419], with reference to Ellis [102], where:

> "social media is essentially about communication and users expressing themselves, where their content is 'delivered to other users'. Social data on the other hand 'expresses social media in a computer-readable format (e.g., JSON) and shares metadata about the content to help provide not only content, but context. Metadata often includes information about location, engagement and links shared. Unlike social media, social data is focused strictly on publicly shared experiences.' "

Therefore, data mining and analysis techniques can be applied to social data where, with reference to Professor I. Witten [446], the creator of Weka (Section 10.5), "the information extracted should be "potentially useful." In one sense, this means *actionable*–capable of providing a basis for actions to be taken automatically." We derive the term *actionable and effective* from this quotation, and use it to describe the outputs of the case studies for our paradigms (Section 1.4) whether the resulting actions are automatic or manual. This is because: (1) RSS is an example of social media, and (2) the applications and academic research employing RSS (Chapter 3).

This explanation of actionable and effective data provides the basis within social media, data and text mining, sentiment analysis and the use of visualisation to represent data, for our paradigms and their case studies. In our opinion, our case studies are not using RSS as a source of data for a particular classification of other type of operation. Instead, we believe that our case studies enhance RSS's social utility by employing the technology itself to produce a comprehensible and explainable representation of data which is inherent within the fluctuations in the keyword frequencies present in the text of RSS feeds: where according to Witten [446], the "information extracted must be *comprehensible* in that it helps to explain the data." Consequently, the data provided by our case studies is more actionable and effective than the journalistic style of presentation (Section 2.2.4) that we currently see in the use of the technology (Chapter 3).

We employ the medium of visualisation (Section 2.8) for the outputs of our case studies. This is because of the medium's ability to coherently represent actionable and effective data in a manner which can benefit users in several real-world domains by correlating or tracking topical issues for trend analysis in business intelligence, statistics, politics, market research and related subject areas, or for mining modern forms of ticker-tape data, such as financial movements, sports or lottery results. Alternatively, users in these domains could benefit from a raw form of actionable and effective data.

2.10 Afterword

We began this chapter by defining RSS, together with its history and format. The technology's utility with respect to social media and data, data mining and related disciplines are then described. The rationale for this is that these subject areas, and web engineering (Section 4.2), coalesce to define the *landscape* of this thesis. In addition, Chapter 3 provides a related review of the applications and academic research employing RSS.

Chapter 3

A review of RSS

3.1 Foreword

In connection with Chapter 2, the purpose of this chapter is to provide a *sampling* of the use of RSS within the context of social media during the last decade. We partition this review into several sections: Section 3.2 outlines different types of RSS-based applications, Section 3.3 discusses academic research using RSS feeds in classification and sentiment analysis, whilst Section 3.4 concerns other related work. Lastly, in Section 3.5, we focus upon related work which has been affected by the format of RSS, or factors to do with the technology's format and use.

In presenting this review, it is necessary for us to state that *much of the related work cited in this chapter is provided for reference purposes only.* Moreover, we only compare appropriate examples of related work with our RSS-mining paradigms (Section 1.4.1) in the final chapters in Parts II and III, i.e. Chapters 9 and 12 respectively, following the presentation of the case studies for each paradigm.

3.2 Applications

According to an article in 2014 titled *How the Feed Changed the Way We Consume Content* by Hiscott [174], the "RSS feed - or as we now know it, "the feed" - organizes the world into a series of neat, clickable, constantly updating bits of information." The article continued: "The feed now dominates online content consumption, from the news we read on our mobile devices to the social networks we check constantly throughout the day, as well as the ads [*sic*] that integrate onto those platforms."

The inter-related categories in the list below group *indicative but by no means exhaustive* examples of this "content consumption" as they, the categories, document applications employing RSS within the context of social media (Section 2.3) and other services:

- **Advertising:** Using RSS for advertising includes services such as: (1) BrandRepublic [51] and equivalent web sites providing RSS feeds to advertise particular services, e.g. jobs or parts of industry, (2) allowing advertisers to buy space in RSS or other feeds for their advertisements, e.g. BuySellAds [56], and where (3) advertisements are displayed in RSS based upon the content of the feeds, e.g. Google AdSense [150] or FeedBlitz [110].

- **Aggregation and readers:** RSS aggregation is a process by which software known as *aggregators* collect large numbers of RSS feeds, blogs or other types of syndicated content, and *aggregate* them into one or more feeds for user *consumption* via *readers*. Aggregation may be based upon keyword filtering, user preferences (either user-defined or gathered based upon usage), or other classification/text mining techniques, some of which are described in Chapter 2. Readers, which may be free of charge, fee-paying or membership-based, allow users to register, or *subscribe* to, individual RSS feeds or aggregations thereof, or to other on-line content. This process removes from the user the inconvenience of manually *trawling* numerous web sites for material of interest. In 2008 the published results of the survey by Lee et al. [223] of RSS usage in Klang Valley, Kuala Lumpur in Malaysia, identified two types of aggregator, i.e. those "browser-based (online) or client-based (desktop) application." Since that time, given the growth of social media (Section 2.3) and the mobile platform (Appendix B.2), we do not make use of this distinction in this thesis and group aggregators with readers together: in some of the examples cited below, the terms *aggregator* and *reader* are *substantially* synonymous because both functions are provided by the same web site, mobile device *app* (Appendix B.2) or web application.

  Content provided by aggregators varies: dedicated news aggregators include Google News [153], Huffington Post [189] and Yahoo News [457], whilst other aggregators such as BuzzFeed [57], Feedly [114], Flipboard [127], Pulse [228] and Quartz News [312], and portals such as iGoogle [194] and NetVibes [277], are more broadly based in the content they supply. The *Europe Media Monitor* (EMM) [105] is a multilingual news aggregator system, which monitors some 10,000 RSS feeds using some 500 predefined news categories. EMM provides information about entities, such as people,

organisations and geographical locations mentioned in the news: the related *Medical Information System*, i.e. MediSys [255], is concerned with public health.

Other aggregator services, e.g. Flipboard [127] and Pulse [228], provide dedicated client readers, some of which take the form of *apps*. Equivalent software is also available for *digests*, which may include all or partial content, from other social media services including questions and answers web site Quora [314], the StackExchange [370] family of web sites, Twitter [408] and photo-sharing web sites such as Flickr [126], Instagram [199] and Pinterest [302]. Sage [341] provides an example of a reader dedicated to the FireFox [121] browser. Popular email clients such as Microsoft Outlook [262] and Mozilla's Thunderbird [399] also provide RSS reader services.

Despite the fact that Google Reader [154] was discontinued in 2013, many third-party *hosted* aggregator/reader services exist on-line including BazQux [28], Feedbin [109], InoReader [198], NewsBlur [278], NewzCrawler [279], Reedah [321], RSSOwl [340] and The Old Reader [392]. Several of these services are included in an extensive list of RSS aggregators and their characteristics maintained by Wikipedia [441] which covers different platforms, operating systems and business models. RSS Reader [335] lists aggregators and readers for dedicated platforms including several for the mobile platform. In contrast to the above, Fever [119], FreshRSS [130] and Selfoss [350] are applications which can be downloaded to act as *self-hosted* RSS aggregators.

- **Bulletin board system (BBS):** Where users can discuss RSS, Atom (Appendix B.2) and related technologies. Examples include Tiny Tiny RSS [401] and WebMasterWorld [430].

- **Delivery mechanism:** The existence of RSS feed on a web site or a browser, is typically revealed by the presence of the *radio wave* icon (Section 2.2.1), or a variation thereof. With reference to the quotations from Hiscott [174] at the beginning of this section concerning feeds dominating "online content consumption", this is because RSS, as a dialect of XML, is machine-readable and can therefore exchange content between systems as a web service, especially where multimedia is concerned. As described by Møller and Schwartzbach [268], "Together, the XML data representation format and the HTTP communication protocol provide a powerful foundation for building *Web services*." We see this in blogging and social media (Section 2.3), and in the mobile platform where RSS (or XML) facilitates the delivery of content in the *background*. Examples of this include newer services such as Feedly [114] and Flipboard [127]: both ultimately are readers relying upon aggregated content from

RSS and other sources of data, and presenting it in the journalistic style of *headline*, *story* and *snapshot(s)* (Section 2.2.4).

- **Management services:** FeedBurner [111] and FeedBlitz [110] allow customers to publish their own RSS feeds and provide traffic analysis and emailing services. These web sites are equivalent to email marketing organisations such as AWeber (`http://www.aweber.com/`) and MailChimp (`http://mailchimp.com/`). Other sites, e.g. NetVibes [277], provide brand monitoring and reputation management services to track trending topics. Sentiment analysis (Section 2.7) of RSS (and other data formats) is also offered as a fee-paying service by organisations including Semantria [351], Trackur [404], WebLyzard [431] and Zapier [464] who also provide a service to convert email to RSS.

- **Search engines:** Like conventional search engines, these allow users to search RSS feeds using their own or topical keywords. Additional features may include labelling news stories, notifications and storing favourite stories. Examples of RSS search engines include Instant RSS Search [200], RSSMicro [339], and the Ukora News Search Service [411]. Other RSS search engines provide a dedicated service, such as RSS4Medics [337].

- **Tools:** These products can be approximately divided between client and server. The former, which include the now demised Yahoo *Pipes* [456] and deprecated Google Feed API [151], allow RSS feeds to be combined with other content and displayed on web pages using JavaScript or AJAX. Alternative tools include FeedsAPI [115], Huginn [190] and the now discontinued Kimono [210].

  By contrast, server-side tools include Rome (Section 4.7.5) and rss4j [71], both of which are Java-based tools which can parse RSS feed content programmatically. Additional tools include Microsoft's Windows RSS Platform as part of their Developer Network [264], and examples for other programming languages include Groovy's *XmlSlurper* [454], Python's *PythonRSS* library [310] and *Feedjira* for Ruby [113].

  Other software allows RSS feeds to be created, e.g. FeedForAll [112], or for HTML to be converted into RSS (RSS Wizard [336]). In a reversal of this, Page2RSS [290] can check web pages for updates and generate RSS feeds with the updated sections of the web pages. Full-Text RSS [125] and WizardRSS [448] transform partial web feeds into full content feeds. RSS4Twitter [338] allows the delivery of Twitter [408] *tweets* in RSS feeds. Voice RSS [421] converts the raw text of RSS into speech in

twenty-six languages and language variations.

- **Visualisation:** The format of RSS (Section 2.2.3) includes no native facilities for visualisation but RSS data is frequently visualised: we discuss this subject area in Sections 2.8 and 3.3.3.

3.3 Academic research

3.3.1 The structure of this review

The *massive* corpus of pre-existing RSS-based related work in the subject areas of data and text mining, sentiment analysis and visualisation of data is beyond the scope of this review. Nevertheless, the following sections of this chapter are *substantially* organised to provide indicative examples of RSS-based related work in each area. In some cases, the sections overlap because of the combination of subjects in particular examples of the related work presented. In addition, in Chapters 9 and 12, we look in greater detail at specific examples of the related work described below that we consider appropriate to our paradigms.

3.3.2 Aggregation and classification

In this section, we are concerned with related work which applies well-known classification techniques to RSS feeds: this work frequently employs aggregation and filtering of content (Section 3.2).

We find several instances of association rules (Appendix B.2) being applied to text mining in academic work. Kittiphattanabawon and Theeramunkong [211] used an association rule-based approach to mining relations within Thai news articles about politics, economics and crime. The WNSS, i.e. *World News Search System*, by Hsu [184] analysed news collected and published on-line via association rules and clustering to discover "useful" news by extracting phrases from "large corpora of web news stories." In 2010 Rahman et al. [315] used association rules to derive feature sets from 115 document abstracts and applied NB to the features for final classification.

In 2006 Garcia and Ng [135] demonstrated the filtering of RSS content, based upon a similarity measure (Appendix B.2) between words, and cited the accurate removal of redundant and less-informative RSS news articles. The *PersoNews* reader by Banos et al.

[26] in 2006 employed an NB classifier to filter "uninteresting" RSS feed content based upon user feedback.

During 2008 Phan et al. [298] focused on large-scale "universal datasets" including Wikipedia [440], feeds and other sources, to present "a general framework to build classifiers for short and sparse text & Web data [*sic*] by making the most of hidden topics discovered from huge text & Web collections." RSS feeds were used by Han et al. [166] in 2009 to "calculate the relevance between the news title and each sentence" in order to detect, and thereby acquire, the "the news article contents from the news pages without the analysis of news sites before extraction." Furthermore, "the relevance between the news title and each sentence in the news page" was calculated in order to "detect the news article contents." Han et al. documented results of up to 98.00% precision, using feeds from major international news sources during the May 2005 - Sep 2008 period.

The news collection method employed by Han et al. [166], referred to in the previous paragraph, was used by Liu et al. [231] in 2009. Citing the absence in RSS of "uniform standards for categorization" and variations in the categories used by publishers to classify content, Liu et al. developed their NDS, i.e. *News Directory System*, application to allow users to create "customizable personal news RSS feeds using existing ones." NDS categorised news stories using "one-level flat" and "multi-level tree" directory structures: the former approach made use of WordNet [449] or Wikipedia [440] to build collections of names: the latter approach used classification trees. This categorisation of content worked on the basis of defining a "news article A to be contained in category B" where "the article A has an occurrence of the word B." Categories provided by NDS concerned named entities such as countries, people, organisations or events, and from these categories, personal news feeds could be configured for individual demands. Users were also able to re-categorise content as they required. Liu et al. [231] documented tests involving news extraction and automated classification "which confirmed the availabilities of our approach."

In 2007 Fiumara et al. [124] employed an XML database to retrieve and query RSS and Atom feeds using XQuery. During the same year *uBioRSS* was developed by Leary et al. [220] to aggregate syndicated content from academic publishers and science news feeds, which were subjected to a taxonomic NER (Appendix B.2). The resulting "name index" was "cross-referenced to current global taxonomic datasets, to provide context for browsing the publications" by relevant parties. uBioRSS is available at `http://ubio.org/rss/`.

AtomsMasher by Van Kleek et al. was the subject of a series of papers and articles

between 2007 - 2009 which included [414], [413] and [412]. AtomsMasher allowed users to write simple scripts to express rules for querying, filtering and specifying behaviours on RSS feeds, email and weather forecasts, which were then carried out by the application.

The prototype of the RoSeS, i.e. *Really Open Simple and Efficient Syndication*, framework for crawling, filtering and aggregating RSS feeds was documented by Creus et al. [80] in 2011. The purpose of RoSeS was to permit users to create "personalized feeds by defining and composing content-based filtering and aggregation queries on collections of RSS feeds." These queries were then "translated into continuous multi-query execution plans". RoSeS was also employed, or referred to, in work by: (1) Hochard et al. [179] to develop a "Semantic Map" for RSS feeds to "present a method to annotate RSS feeds with a domain ontology to support discovery queries expressed as linear expressions in terms of the ontology", (2) Horincar et al. [180] in respect of a best-effort refreshing of RSS under limited bandwidth to prevent information loss, and (3) Shaikh and Rajawat [353] who in 2012 described work concerning RoSeS to enable large-scale aggregation of science-based Atom feeds.

Saha et al. [342] employed TF-IDF (Appendix B.2) to create keyword vectors which were classified using SVM. In 2010 the authors wrote that their prototype web service was restricted to RSS feeds concerning sports and business news. Data was collected from various news websites and some twenty newsgroups: the training dataset contained "50 articles for business news and 50 articles for sports news." A further 1,200 files were collected "from news group dataset, BBC, News and CNN websites" for use as testing data. Citing random precision results produced by their prototype, the authors wrote that "Experimental results suggest that the proposed method is effective and saves a significant amount of user processing time."

Using data collected from Chinese language RSS feeds, in 2010 Teng et al. [391] sought to build a "self-sufficient news collection system in [*sic*] disaster domain." The authors presented a system of three components: (1) *article acquisition* employed the URLs in `<item>` elements (Section 2.2.3) in the feeds to download the HTML data of the referenced web pages in order to extract news information from them, (2) *article summarisation* used clustering and sentence similarity to identify local topics, and (3) *article classification* where for the natural disaster data, an SVM classifier with a linear kernel determined which "articles are in the predefined domain and which are not." Teng et al. reported that evaluation of their summarisation techniques, using valid word and high-frequency word coverage ratios, returned higher results than alternative approaches,

and their classification techniques produced average F-measures (Appendix B.2) of 90.30%.

In 2012 Cingiz and Diri [72] combined data from 2,105 RSS feeds into four categories of training data, and data from "tweets of 26 normal microbloggers and tweets of 27 bots", i.e. Twitterbots (Appendix B.2), for testing. After pre-processing, involving TF-IDF and chi-square feature selection, data was classified using SVM and MNB. In their results, Cingiz and Diri found that "bots provide more categorical content than normal users."

Longe and Salami [236] integrated an aggregator with a multi-class SVM classifier to automatically categorise feed content, and called their application *NBlogs*. An experiment was presented in 2014 which used: (1) 1,020 "manually categorized posts" retrieved from Nigerian blogs and web sites, and (2) a series of ten categories including *Business*, *Current Affairs*, *Education*, *Entertainment* and *Science & Technology*. The authors documented results showing that their "text classifier performs well in categorizing" RSS feed content, but in conclusion Longe and Salami wrote that "it is not reliable to categorize contents consumed from a feed using the pre-defined category of the Feed". This was because their SVM classifier placed some "68% of the feed content retrieved" in a different category compared to the category specified by the feed's publisher.

During 2015 Hurtado [191] proposed a method to improve the quality of RSS feeds. This method involved the web page where a feed is located. For each `<item>`, the main textual content and the most prominent image were retrieved: each `<item>` was then classified via a one-versus-rest algorithm. Given the 39.98% to 95.62% improvement in item quality, Hurtado proposed this method's use in RSS aggregators and readers (Section 3.2).

Clustering of RSS content has also been performed. In 2007 the *RSS Clusgator* by Li et al. [226] helped users to read blogs by clustering stories into hierarchies. If a user subscribed to a group of feeds, the clusters containing the stories of the feeds were returned to the user for faster browsing. The authors compared their results favourably with equivalent times taken to find stories of interest on (the now demised) BlogLines [42]. Qincheng et al. [311] proposed extracting content from web pages based upon a template index of pages. *NewsStand* by Teitler et al. [390] extracted geographic information from RSS feeds and grouped articles into story clusters: users were able to use a map to view results based upon significance and location.

Banerjee et al. [25] sought to improve the clustering of small pieces of text by supplementing their descriptions with text from Wikipedia [440], and claimed improvements in overall clustering accuracy (Appendix B.2). According to Hu and Chou [185], their *RSS Watchdog* produced online "news clustering and event monitoring over multiple RSS

news streams", and grouped them temporally per channel, or group of channels, to find cross-channel temporal relationships.

In 2009, Pera and Ng [295] clustered non-redundant and informative RSS news articles to filter and classify feed data using a fuzzy logic equivalence relation. This work was extended by the authors in 2012 [296] with a filtering and clustering approach to RSS, i.e. FICUS, which started with identifying and removing redundant news articles using a "fuzzy set information retrieval approach" which then clustered "the remaining non-redundant RSS news articles according to their degrees of resemblance."

Caveats to clustering RSS are discussed by Roesler [327] as a subset of text clustering generally, i.e. the number of documents or RSS feeds/items to be clustered, semantic and linguistic issues, and the time taken to cluster content, especially in a real-time application.

Other instances of classification/filtering of RSS content include: (1) Wegrzyn-Wolska and Szczepaniak [434] who classified RSS documents using a fuzzy full-text similarity. (2) Reed et al. [320] proposed a new term weighting measure known as TF-ICF, i.e. *Term Frequency - Inverse Corpus Frequency*, which unlike TF-IDF "does not need term frequency information from other documents within the document collection", and which permits the generation of "document vectors of N streaming documents in linear time." Reed et al. documented "above average performance in most cases" when TF-ICF was compared with five other commonly used term weighting methods in the context of document clustering tasks: 127,742 news feed documents from the *Los Angeles Times* newspaper and other sources were used for data. (3) *RSS_PROYECT* by de la Torre-Díez et al. [89] concerned the use of RSS for syndicating content using filters installed in content management system *Joomla* [208]. (4) Bouras et al. [49] published *PeRSSonal* in 2008 which was based upon work to do with personalisation and classification by Antonellis et al. [20]. PeRSSonal used cosine similarity (Appendix B.2) to summarise and categorise RSS data before delivering personalised outputs, and was later supported with the *advaRSS* crawler by Adam et al. [3]. (5) Darabi et al. [87], who in 2012 proposed a ranking model called *LabelledNews* which used query-expansion (Appendix B.2) to categorise streaming news items directly from news feeds without retrieving original content. (6) Adeniyi et al. [5] employed KNN (Section B.2) clustering to identify click stream data belonging to clients/visitors, and sought to match it to user groups in order to recommend tailored browsing.

3.3.3 Sentiment analysis

Scope

In this section, we consider RSS and sentiment analysis. This work frequently extends into the subject areas of visualisation and trend, topic or event detection, or a combination thereof. For convenience of presentation, we describe RSS-based related work in these subject areas together in a single section below, which is organised in approximate chronological order by year of publication. In connection with this, we refer to Wanner et al. [428] who in 2014 published a *state-of-the-art* survey of the use of visual analysis for event detection in text streams. In this study, the authors identified twelve distinctive sources of text data including email, micro-blogging, RSS and others, used in a total of fifty-one papers between 2007 - 2013. Wanner et al. wrote that:

> "One thing we noticed was that data sources have dramatically changed from news to social media since 2010. Mainly due to the burst of social media, many research studies used text data streams generated out of Facebook or Twitter."

We document the six examples of RSS-based work identified by Wanner et al. because of the aforementioned consideration by this section of RSS and the subject areas of sentiment analysis and visualisation. Accordingly, we do not document related work concerning "text streams generated out of Facebook or Twitter" or other sources cited by Wanner et al. Nevertheless, several examples of related work *compatible* with RSS are briefly considered at the end of this section. Moreover, Section 2.7 refers to a series of surveys of products for, and the application of well-known classification techniques to, sentiment analysis of RSS. We do not repeat that material here.

Chronology of related work

Gruhl et al. [161] tracked 11,804 RSS blog feeds over a month in autumn 2003 to amass a total of 401,021 RSS feed postings.[1] Based upon the topics discussed in the feeds, postings were placed into one of the following three categories: (1) *Just Spike* where topics spiked and then became inactive, (2) *Spiky Chatter* topics with a significant chatter level and which are very sensitive to external world events, and (3) *Mostly Chatter*, i.e. a consistent volume of chatter. Gruhl et al. also investigated methods of propagation of news between individuals by drawing on the "theory of infectious diseases to model the flow."

[1] Gruhl et al. [161] also referred to hourly tracking of "fourteen RSS channels" from `rss.news.yahoo.com` to "identify when topics were being driven by major media or real-world events".

Glance et al. [143] used clustering and cosine similarity to discover trends in approximately 100,000 blogs in 2004. In their work, the authors discussed a Lucene-based searchable index of blog data, which was used to graph "the normalized trend line over time for a search query" as a means "to estimate the relative buzz of word of mouth for given topics over time." Daily lists of key persons, events and paragraphs were published by the authors on-line at BlogPulse (`http://www.blogpulse.com/`) until its demise in 2012. Also in 2004, Albrecht-Buehler et al. [13] published work for their *TextPool* application which summarised recent content in live text streams (Section 2.8.2) including RSS, by extracting keywords to display related terms in a dynamic "text collage" wherein related terms were grouped together.

In Thelwall et al. [398] the authors introduced the phrase *broad issue scanning* in connection with the use of RSS for "the task of identifying and tracking important public debates arising within a given broad issue, such as public science concerns." With reference to:

> "two relevant research traditions for web data analysis, which will be described here as *purist* and *pragmatic*. Either could potentially be suited to broad issue scanning. The purist approach is to analyse an Internet phenomenon as it was found, seeking to describe it as accurately as possible. The pragmatic approach is to analyse a phenomenon from the perspective of attempting to gain information about an underlying phenomenon, rather than the Internet data (i.e. for indirect research). The key difference between the two approaches is that the former typically does not use any data cleansing whereas the latter tends to use extensive data cleansing."

The authors sought "to assess whether a *purist* approach to RSS feeds (i.e. using the raw feeds without data cleansing) is suitable for broad issue scanning, using a co-word frequency time series approach." A corpus of 19,587 RSS feeds was tracked concerning public fears over science where the authors reported a "low success rate" and wrote that data cleansing of RSS "is necessary for efficient broad issue scanning. Raw RSS feeds are unsuitable because some feeds carry extensive and repetitive content."

Ali et al. [14] applied their *DescribeX* tool for exploring and visualising the structural patterns present in XML documents, to a collection of RSS feeds in 2007. The purpose of this work was to assist in the formulation of XPath queries to find items which could be aggregated by focusing on metadata (Appendix B.2), e.g. articles with known creation dates or authorship.

During 2006 Prabowo and Thelwall [305] proposed the use of feature selection techniques such as chi-square, mutual information (MI) and information gain (IG) (Appendix B.2) as alternative approaches for ranking term significance in an evolving RSS feed corpus in order to identify significant topics. Prabowo and Thelwall concluded that chi-square was the best of the three techniques when they were evaluated to determine the significance of a term on a certain date, but that it was "far from perfect" because of high scores given to "relatively insignificant terms." On similar ground, Prabowo et al. [306] described a feature-based, clustering approach to generate "overview timelines for major events" from a "general-purpose corpus of RSS feeds." The clusters were later assessed by ten people who found that 68.60% of them "apparently" represented significant events, and were used to document three then-current (2007) events.

Benson et al. [32] presented a tool in 2008 to provide users with information about "topics of interest" in RSS and other text streams by using *agents*. An agent "represents a significant word, visualizing it by displaying the word itself, centered in a circle sized by the frequency of word occurrence." Dynamic visualisation was enabled by changes in the colour and position of agents and in the ways they responded to one another.

Using data gathered from fifty different RSS feeds between 09 Oct - 10 Nov 2008, Wanner et al. [427] used sentiment analysis and visualisation to reveal the sentiment in news concerning the presidential candidates during 2008's US election campaign. Data covering the period before this election was also used by Fisher et al. [122] in one of the two case studies documented for their *Narratives* application. Narratives combined "keywords from news articles with reactions from social media" to display time-series plots (Section 2.8.3) showing how stories evolve over time in order to determine their "historical and social context". To this end, Narratives included a series of tools to investigate correlations between keywords.

In 2009 Hristidis et al. [182] employed "keyword search over a time span on multiple textual streams" which included RSS feeds, blogs and emails from the Enron Corpus[2] to "solve the problem of answering a keyword query on a collection of text streams, where a result is defined as a combination of events from a set of correlated streams such that these events collectively contain all the query keywords." Hristidis et al. defined a result as a tree of events based mainly upon the commonality of two streams, and presented an

[2]The Enron Corpus [104] is a collection of 500,000+ emails generated by employees of the Enron Corporation during that organisation's collapse in the USA in 2001.

incremental algorithm for computing the answer set of a continuous keyword query performing a minimal amount of operations for each event. *Media Watch on Climate Change* (MWCC) was also documented in 2009 by Hubmann-Haidvogel et al. [188]: the application "aggregates, filters and visualizes environmental Web content from several sources including 150 Anglo-American news media sites." Two-dimensional geographic maps, semantic maps, domain ontologies, and word-clouds were integrated together to allow visual and textual content to be displayed and navigated. Sentiment was automatically calculated and averaged for keywords within a selected period. Further work on MWCC was documented by Scharl et al. [347] in 2013 concerning the application's evolution into an on-line "domain-specific portal" for "environmental stakeholders." MWCC can be found at `http://www.ecoresearch.net/climate/`.

Krstajić et al. [213] documented the collection of 1,736,246 articles of news data from the multilingual Europe Media Monitor (EMM) [105] aggregator during 2010. The authors referred to this corpus as "semantically enriched metadata" of "great interest for our analysis", and consisting of "entities, categories, geo-tags, URL of the article, source, publishing time, date and language." Following the transformation of the data into the authors' own XML format, it was analysed for entities and their co-occurrence in news articles. Two case studies were carried out: (1) where "temporal analysis of entity occurrences over a time period of two months" included "cross-language comparison of entity occurrences" and their visualisation using "stacked time series graphs", and (2) the "analysis of relationships among entities, which we realized using a radial graph layout." This work was extended by Krstajić et al. [214] in 2012 when *Story Collector* clustered "new articles in 24-hour time intervals" from EMM to: (1) show "temporal characteristics of stories in different time frames with different levels of detail", (2) allow displays to be updated incrementally, and (3) to sort "the stories by minimizing clutter and overlap from edge crossings." Story Collector was demonstrated using events from the "the Arabic uprising in 2010 and 2011."

Snowsill et al. [364] sought to detect events in text data streams by "identifying statistically significant increases in the frequency of n-grams within the stream." The resulting n-gram frequencies of were stored in a suffix tree (Appendix B.2), and weighted averages of frequencies were used to overcome the issue of topic drift. The authors documented successful tests identifying events in 2008.

Eventscapes was presented by Adams et al. [4] in 2011 to provide a "visual depiction of event history, mood and controversy." Although "agnostic to the document source",

Adams et al. referred to RSS feeds as input where topic modelling (Appendix B.2) and clustering were used to determine topics and events which were then arranged chronologically: document mood and event controversy were also calculated. Eventscapes displayed images retrieved from RSS to "encapsulate the most salient elements of the textual content", and different colours were used to identify the mood in the "timeline and documents" displayed. Adams et al. described two evaluative case studies which "reflected positively" on their approach, and discussed future work.

Work by Steed et al. [374] in 2012 focused upon the "interactive exploration of high-throughput, unstructured text streams." Data was taken from RSS feeds and other social media, and animated graphs were used to depict the "term frequency patterns in a text stream." Steed et al. used "geospatial metadata" to "reveal temporal trends for specific areas of the world." Heat-maps (Appendix B.2) were used to represent trends and the authors documented these using London's 2012 Olympic Games and *tweets* from Twitter [408] at that time. Steed et al. concluded by referring to future plans which included the use of sentiment analysis and semi-supervised machine learning (Section 2.5).

During 2012 Wills [444] documented a "road map to generating quick, efficient, and accurate analytics of RSS and Twitter feeds." NB and DT classifiers were used to predict sentiment. Based upon F-measure results, NB outperformed DT with respect to both Twitter and RSS data. Wills also proposed that production grade classifiers should pay attention to accuracy of assessment in addition to speed.

Modha et al. [267] described a planned method of analysing Indian political news articles for sentiment in 2013. We summarise the stages: (1) classifying sentences into opinionated and non-opinionated groups, (2) dividing opinionated sentences into subjective sentences, (3) analysing subjective sentences for positive, negative or neutral sentiment, and (4) carrying out the same for objective sentences. The authors referred to SVM, NB, BoW and other data mining techniques for (3) and (4), but no results were presented. Also in 2013, according to the English language abstract of their paper, Gomes et al. [146] sought to detect the positive, neutral and negative polarity of sentiment in economic news headlines in RSS feeds: unfortunately, the Portuguese language body of their paper precludes any further consideration of it in this review. Trabelsi and Yahia [403] employed NB to identify events in RSS feeds produced by Flickr [126] using their *RssE-Miner* application. Trabelsi and Yahia wrote that the "main originality of RssE-Miner stands in achieving a meaningful tradeoff between runtime performance and event identification accuracy from

social media RSS feeds."

In 2014, Hennig et al. [173] applied sentiment analysis to the "detection of trends" determined by the mining and analysis of blogs performed by BlogIntelligence (`http://www.blog-intelligence.de/`). Using three criteria: (1) term relevance calculated by TF-IDF, (2) the count of the number of times the term is used as a tag for a specific posting, and (3) the number of incoming links for a specific posting containing the term, the "*degree* and *intercept*" was calculated to determine if a term's trend was "ascending, descending or popular." A term's trend was subsequently "enriched" with "sentiment information" which included the "computation of the term sentiment using sentiment keywords and the boost of the term trend." Hennig et al. wrote that "the implementation with an in-memory database on the BlogIntelligence data set shows promising results in running time and quality." More recently in 2015, Bharathi et al. [36] discussed the use of cosine similarity to classify RSS feeds into one of several news categories and then analysing them for sentiment using categories *happy*, *sad*, *fear*, *excited*: unfortunately, few specifics were provided concerning their method of sentiment analysis.

A *divertimento* for related work *compatible* with RSS

Examples of related work that are not specific to RSS, but which are claimed by the authors to be compatible with the technology, include: (1) VISA, i.e. *Visual Sentiment Analysis System*, by Duan et al. [100] in 2012 which was described by its authors as a "generic sentiment analysis paradigm" which used a *mash-up* (Appendix B.2) visualisation to give "analysts and users" "coordinated multiple views" of the sentiment in a document collection, (2) Wang et al. [426] presented *SentiView* in 2013 which analysed trends in sentiment found in forums and blogs in order to consider the "time-varying direction of the sentiments", the "number of participants engaged in the discussion", the "relationships between public sentiments and participants", and the "relationships of relevant topics", and (3) Steed et al. [373] whose *Matisse* software employed Twitter [408] despite the authors writing that it is "capable of consuming any textual information (e.g., RSS news feeds, blog streams)".

3.4 Other RSS-related work

- **Archiving:** The format of RSS lacks any provision for archiving published content, although private sources such as the Internet Archive [201] may provide some facilities. Alternatively, in 2005 Chmielewski and Hu [69] proposed their BARF server,

i.e. *Barf Archives RSS Feeds*, as a distributed platform to archive, retrieve, and synchronize RSS feeds allowing users to search for updates made when they were not on-line. Romsaiyud [330] further proposed an algorithm for retrieving data from BARF through a network.

- **Education:** (1) Glotzbach et al. [144] employed RSS as a means of distributing course announcements to new students, and for students to obtain information and course-related materials from on-line web sites without visiting them, (2) Cold [77] described the use of RSS by students to share project work, (3) De Maio et al. [90] proposed the integration of the *Intelligent Web Teacher* (IWT) "e-learning Web-based platform" and RSS "for supporting and improving personalized e-learning processes", and (4) Lan and Sie [218] sought to determine the most suitable medium among SMS, email and RSS to improve learning activities: this work concerned *media richness theory* which describes how and why particular media are selected to deliver a message [84].

- **Media:** Lee et al. [221] extracted television schedule data from RSS, and Messina and Montagnuolo [257] presented a system "for aggregating and retrieving RSS items and broadcast news streams."

- **Mobile devices:** In 2006, Blekas et al. [40] proposed RSS feeds for the adaptation of web content for use in mobile phones. More recently in 2012, Sajjanhar and Zhao [343] proposed the use of SVM to classify RSS content for delivery to an Android OS (Appendix B.1) client with "Geoparsing and geocoding web services" for location-based access to news.

- **Ontology:** (1) Villoria et al. [416] who sought to make the RSSOwl [340] aggregator "ontology powered" by adding three semantic functions to it, (2) in 2012 Agarwal et al. [8] used CF-IDF[3] with an ontology based upon "news industry standards" as a text classifier to classify news terms, (3) Hsu [183] integrated ontology technologies into feeds and user profiles to provide customised feeds for personal use, and (4) Yuan et al. [463] proposed a "fuzzy method for matchmaking between a subscriber's interest and RSS items", where ontology was used to link "heterogeneous publishers with subscribers in semantics rather than in words."

- **Recommender and ranking systems:** We note several instances of RSS-related work involving recommender systems (Section 2.3.2).

[3]cf. Goossen et al. [157], who in 2011 proposed CF-IDF as a variation of TF-IDF (Appendix B.2).

Collaborative filtering was used in *FeedMe* by Sen et al. [352]. FeedMe made use of traditional rule-based alert filtering and a collaborative NB filtering algorithm, based upon user preferences from feedback, to reduce the noise for unwanted interruptions for knowledge workers. Related to this, the *Personal Information Manager* by Sia et al. [356] helped users to monitor their subscription lists and recommended relevant articles based on their browsing history.

Typically, data processing by recommender systems is performed on the server, e.g. processing logs and user profiles. In a reversal of this by Chen et al. [67], *Content REcommendation System based on private Dynamic User Profile* (*CRESDUP*) discovered "preferred messages" on the internet according to private user data which resided, and was processed, on the client side. In 2007, Pon et al. [303] proposed MTT, or *Multiple Topic Tracking*, with *iScore* to "better recommend news articles for users with multiple interests and to address changes in user interests over time." The authors documented several test cases to determine whether RSS stories were interesting or not to specific users.

Recommendations to users of new content based upon their past was the focus of RSS reader *NectaRSS*, by Samper et al. [346], which employed VSM. Wu [452] used DT, NB, SVM and random forest (Appendix B.2) classifiers to rank user preferences in RSS feeds: Wu cited the performance of NB, which was slightly behind random forest, but ahead of SVM in tests applied to a corpus of 112,828 RSS entries from a total of 163 unique RSS feeds, of which 2,607 had "been clicked".

More recently in 2010, Ji and Zhou [205] developed an RSS reader and collected user data to study the effectiveness of recommendation for different features and feature combinations: their experimental results showed that "*favorite fraction*", i.e. past preference, was the single most important feature and that a combination of it and text similarity "performs the best." Citing an increase in blog traffic, Paik and Hiroshi [291] documented their use of TF-IDF calculations which focused upon the words in the `<title>` elements in RSS feeds. The authors proposed a new recommendation method employing an NB classifier and referred to this work outperforming previous research of theirs when recommendations were based upon term frequencies in RSS feed content and the browsing history of users.

In 2014 Hassan et al. [169] used VSM to rank *mash-ups* (Appendix B.2) built from RSS feeds: precision and recall results were reported as being on average 30.00%

better than binary rankings. Also, in 2014 Tang and Ma [385] put forward their *RSSCube* as an improved content syndication and recommendation architecture to address issues they found with searching, synchronisation performance, and the user experience in then-current products.

- **RSS algebra:** Two papers by Getahun et al. [138] and Taddesse et al. [382] concerned the semantic relatedness of RSS news stories and the use of a VSM classifier (Appendix B.2) to merge news items by comparing relationships between text, elements and items. This work was later extended in 2013 by Getahun and Chbeir [137] who proposed a "dedicated RSS algebra based on semantic-aware operators which are capable of considering RSS characteristics", as a means to address issues they cited concerning the temporal nature, relatedness/similarity and relationships between items of RSS content.

- **Search engines:** To the RSS search engines referred to in Section 3.2, we must add Zhou et al. [465] who proposed *soSpace* in 2006 as a self-organising search engine for RSS, built upon a scalable peer-to-peer technology: the tool enabled content to be indexed and searched. The authors documented "satisfactory" precision and recall (Appendix B.2) results.

- **Security and notification systems:** The open format of RSS (Section 2.2.1) lacks any provision for security. Preechaveerakul and Kaewnopparat [307] developed *Secure Information Notifying System with RSS Technology for Mobile Users* (SInfoNS) as a response to this by applying a cryptography algorithm to an RSS feed before it is disseminated to relevant users. Work by Gioachin et al. [142] concerned secure RSS based emergency notifications. Makpangou et al. [242] proposed *Friticores*, an "RSS Feed Monitoring and Dissemination System" which advised subscribers when new content matching their subscription details arrived, and was designed to "help researchers from underdeveloped Countries." *Post@*, i.e. *PostAt*, an RSS-based web service that automatically delivered announcements, posted by a publisher to subscribers, was proposed by Alomari et al. [15].

3.5 Discussion: the format of RSS

The preceding sections of this chapter have concerned the applications and academic research employing RSS. Here, we change *perspective* and discuss related work concerning the format of RSS (Section 2.2.3), factors relating to it and its use.

3.5.1 Versions

Despite backwards-compatibility in the different versions of RSS (Section 2.2.2), several cases to the contrary have been reported: (1) by Pilgrim in *The myth of RSS compatibility* [300] in 2004, and (2) where "syndication confusion" (Lee et al. [222]) with the different formats of RSS caused "uncategorized and irrelevancy of aggregated result[s]", and led the authors to develop their *PheRSS* analyser/aggregator. Changes in the format of RSS also appear to be unlikely according to the RSS *roadmap* [333]:

> "RSS is by no means a perfect format, but it is very popular and widely supported. Having a settled spec is something RSS has needed for a long time. The purpose of this work is to help it become a [*sic*] unchanging thing, to foster growth in the market that is developing around it, and to clear the path for innovation in new syndication formats. Therefore, the RSS spec is, for all practical purposes, frozen at version 2.0.1. We anticipate possible 2.0.2 or 2.0.3 versions, etc. only for the purpose of clarifying the specification, not for adding new features to the format. Subsequent work should happen in modules, using namespaces, and in completely new syndication formats, with new names."

Liu et al. [231] and Longe and Salami [236], described in Section 3.3.2, have also documented issues with the published content of the `<category>` element types in RSS's format.

3.5.2 Characteristics of data

In their 2005 paper Liu et al. [233] described hourly polling of 100,000 RSS feeds over a forty-five day period in order to look at feed size and format, and to analyse updates. The authors found that a "majority of feeds (55%) update every hour, while many feeds (25%) do not change for days together." In other findings, the authors wrote that only "small portions of RSS content typically change during an update; 64% of updates involve less than three lines of the RSS content." As a result of this, the authors proposed that content providers could indicate "when and at what rate to poll a particular feed." In 2007 Lambiotte et al. [217] focused on the statistics of word occurrences and of the waiting times between such occurrences in RSS feeds and blogs.

In a similar vein, as part of the RoSeS project (Section 3.3.2), Hmedeh et al. [177] surveyed a *testbed* of 10,794,285 items collected from 8,155 "productive" RSS feeds (from an overall total of 12,611) during an eight month period in 2010. The authors sought to

analyse "three complementary features of real-scale RSS/Atom feeds, *namely, publication activity, items structure and length*, as well as, *vocabulary of the textual content*." The principal findings were: (1) "17% of RSS/Atom feeds produce 97% of the items of the testbed", (2) the most popular RSS/Atom textual elements found were the `<title>` and `<description>`, while the "average length of items is 52 terms", and (3) that "the total number of extracted terms from items written in English is 1,537,730 out of which only a small fraction (around 4%) is found in the WordNet [449] dictionary." This was attributed to the "heavy use in RSS/Atom textual elements of named entities (person and place names), URLs and email addresses as well as numerous typos or special-purpose jargon."

The need to disambiguate keywords in RSS feeds in order to identify context can be achieved using WordNet [449] or DMOZ, i.e. *(D)irectory (Moz)illa*, a "human-edited directory of the Web" at `https://www.dmoz.org/about.html`, or the "graphical interface" proposed by Webster et al. [433] to reside between the user and DMOZ. Sia et al. [355] proposed that "RSS aggregation services should monitor the data sources to retrieve new content quickly using minimal resources and to provide its subscribers with fast news alerts." Pinheiro et al. [301] discussed the removal of "irrelevant bits of information" from RSS with *Data Killing Operators*.

Scalability was the focus of work by Hmedeh et al. [176] in 2012. The authors cited the success of syndication technologies and the need to look at "*real-time filtering methods across feeds* which allow users to effectively follow personally interesting information." The authors considered a series of "three index structures implementing different counting techniques for pruning as early as possible non matching subscriptions to an incoming item." Moreover, Hmedeh et al. advocated a "*content-based Publish/Subscribe paradigm*" for syndication where information providers and consumers are "decoupled".

3.5.3 Extensions

The following extensions of RSS have been proposed/implemented:

- **GeoRSS:** GeoRSS [136] tags RSS feeds with geographically encoded objects so that applications can request, aggregate, share and map geographically tagged feeds. GeoRSS has been employed in work by Anjomshoaa et al. [19] and Tok et al. [402].
- **GeoTracker:** In 2007 GeoTracker by Chen et al. [68] displayed RSS data in a "geographic presentation layer" which allowed users to "navigate (zoom, pan) the RSS view on a world map" using Google Maps [152] to "render locations on the map automatically." Photo-sharing web site Flickr [126] also provides a similar service.

- **GPS/GNSS:** The delivery of GPS/GNSS satellite data by RSS has been proposed by Hu et al. [186].
- **jQRSS:** A plug-in for jQuery (Appendix B.1) by Petrova-Antonova and Rosen[297].
- **Media RSS:** Media RSS was a project co-developed by Yahoo [455] in 2004 to syndicate media types "such as short films or TV, as well as provide additional metadata with the media" [254] in RSS feeds by employing the `<enclosure>` element in each `<item>` in a feed (Section 2.2.3). In 2009 Media RSS was transferred to the RSSAB.
- **Other:** In 2006 Bossa et al. [48], created RSS feeds by extracting information from HTML pages via their own set of XML-like annotations.

3.5.4 *Push* or *pull*?

As described in Section 2.2.1, RSS is a *pull* technology, i.e. where a request made by a client *pulls* data from a server as a *stream of text*. In contrast to this, servers in other technologies *push* services out to subscribers. Two examples of related work have used RSS as a push service: (1) Ma [240] constructed a "theoretical game model to study the profitability of an RSS-PUSH delivery mechanism." Ma concluded that although RSS "always helps a website to attract more users, it may also reduce the website's profit. This happens because newly attracted users are not profitable enough to offset the website's increase in maintenance costs and decrease in advertising revenue." (2) Silberstein et al. [357] also considered the pull/push issue. Their perspective was the managing of events where consumers *pull* events from a "per-producer event store" at query, e.g. log in, time, as opposed to producers who *push* events to materialised feeds. The authors concluded that it is best to "decide whether to push or pull events on a per producer/consumer basis."

3.6 Afterword

This chapter has reviewed the current state of the applications and academic research employing RSS. We have also cited other work concerning the format of RSS and extensions to the technology. In Chapter 4, we define and illustrate the architectural and technical foundations of the software which implements the case studies for our paradigms, before we focus upon the case studies proper in Parts II and III. We also compare appropriate examples of related work with our paradigms in Chapters 9 and 12.

Chapter 4

Web engineering and software architecture

4.1 Foreword

Before the case studies for our RSS-mining paradigms (Section 1.4.1) can be presented, it is necessary to consider web engineering and the architecture of web applications, i.e. these subject areas define the context for the development of our myDS and vRSS software.

This is the purpose of the first half of this chapter. To this end, Section 4.2 defines web engineering and the development cycle(s) this entails. Section 4.3 describes web applications: Section 4.4 extends this theme by focusing upon the MVC design pattern and the layered architecture typical of n-tiered web applications.

The second half of this chapter concerns our myDS and vRSS software written for the two case studies for our first paradigm. Architectural and organisational issues of these applications are described in Section 4.5, before Section 4.6 explains the general keyword conventions and characteristics used in our paradigms and their case studies. Section 4.7 defines the architecture and implementation of common components, and related terminology in myDS and vRSS for our first paradigm. Section 4.8 serves the same function for our second paradigm with regards to the extension of vRSS and software characteristics of this in case study three.

Section 4.9 ends this chapter with reference to the tools used to develop our software.

4.2 Web engineering

Web engineering has been defined by Deshpande et al. [94] as:

> "the application of systematic, disciplined and quantifiable approaches to development, operation, and maintenance of Web-based applications. It is both a pro-active approach and a growing collection of theoretical and empirical research in Web application development."

This definition concerns "the platform" (Wilde and Gaedke [443]), i.e. the rise to prominence of the web/browser paradigm since its inception in 1989 by Berners-Lee [33], and the evolution of web applications (Section 4.3) from static HTML pages to highly complex, often inter-connected web services within the social media (Section 2.3) context, and the newer mobile platform (Appendix B.2). Although many of the aspects of traditional software engineering are used in the development of web applications, recent years have seen the web engineering process emerge as a "specialization" (Mayr [251]) of application development, which can be differentiated from classical software engineering. Deshpande et al. [94] documented sixteen "Major Differences between Web Applications and Conventional Software": we list several of these differences below:

- The use of "small teams" and "compressed development schedules".
- Where " "content is king", i.e. it is integrated inextricably with procedural processing."
- An "understanding of additional disciplines required for Web applications, such as hypertext, graphic design, information architecture".
- The "evolving standards to which Web applications should or must comply, depending on the specific circumstances".
- The use of a "Rapidly evolving implementation environment, encompassing various hardware platforms".

Deshpande et al. [94] also referred to a lack of "accepted testing processes", "criticality of performance" and risks of competition. Other differences cited by the authors conform to the two "key attributes" of web application development defined by Ginige and Murugesan [141] which distinguish it from traditional software development, i.e. the "rapid growth of the requirements of Web-based systems and the continual change of their information content." Ginige and Murugesan further wrote that "scalability and maintainability" must be present from the beginning. Web applications must also accommodate "different stakeholders", e.g. the range of the system's users involved in planning and management, maintenance, funding the development and the organisation which requires the system.

Ginige and Murugesan also discussed the "evolutionary" nature of web application development because "it's not possible to specify fully what they should or will contain at the start of their development, because their structure and functionality will evolve over time." Therefore, an iterative development cycle consists of "many phases, steps and activities" (Ginige and Murugesan [141]), which are not carried out in a fixed, consecutive sequence like those of the classical *Waterfall* model (Appendix B.2) of software development. Pressman and Lowe [308] have instead referred to the "process framework" of web engineering which "incorporates rapid development cycles." Each "cycle results in the deployment of a WebApp increment." Thus, with reference to Figure 4.1, during a web application's lifespan, each increment will iterate from *Web Site Maintenance* back through the earlier phases illustrated, i.e. this explains the permanent *beta*-version status of much *Web 2.0* (Appendix B.2) web development. *Web Site Maintenance* and the other phases of the development process are summarised from Ginige and Murugesan [141] in Table 4.1.

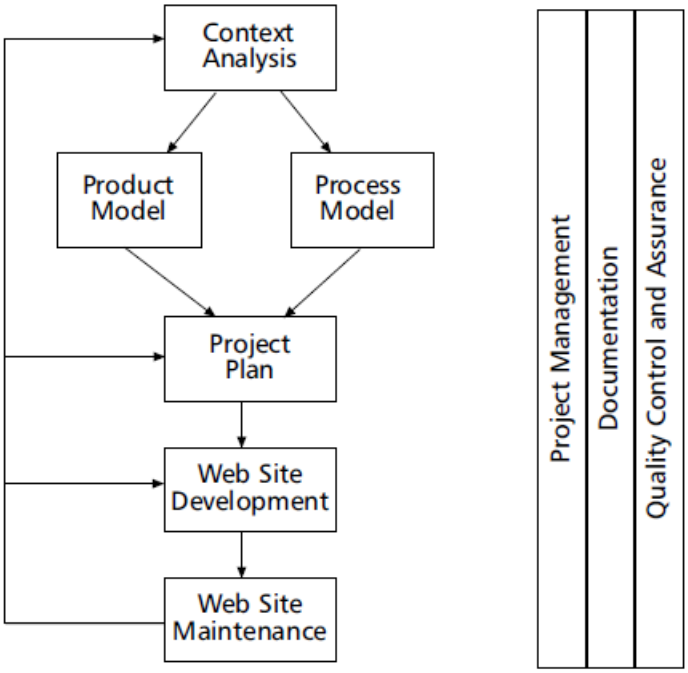


Figure 4.1: The development process for web applications (reproduced from Ginige and Murugesan [141]): cf. *all case studies*.

Name	Description
Context analysis	Where we determine and understand the system's major objectives, functional and other requirements. Information is gathered about the application's environment and operation, and stakeholders are identified. The result is a set of requirements which will influence the web application's design.
Product model	Where the relationships between the various system components are defined. System architecture will focus on hardware, application architecture will concern functionality required whilst software architecture identifies alternative ways to implement the application architecture. From this, a model of the web application will be built, based upon requirements determined from *context analysis*.
Process model	Parallels the *product model* and defines the activities necessary to implement the system. These activities, which will involve frequent liaison with stakeholders, include requirements analysis, design, testing and deployment, and may be decomposed into sub-activities.
Project plan	Project managing and scheduling the development.
Web site development	The development of the required application software to deliver the content and functionality. In tandem with this, hardware testing and integration must be carried out to meet performance and security requirements.
Web site maintenance	Concerns the maintenance of application content, software and hardware required after the application's deployment.

Table 4.1: The development process for web applications (summarised from Ginige and Murugesan [141]): cf. *all case studies*.

In addition to the phases listed in Table 4.1, Ginige and Murugesan [141] also wrote that project management ensures "that all the key processes and activities work in harmony." Furthermore, project management, quality assurance and documentation are "spread throughout the Web development cycle."

4.3 Web applications

4.3.1 What is a web application?

In the definition of web engineering by Deshpande et al. [94] in Section 4.2, reference is made to "Web-based applications." In order to define this term, we defer to the description provided by Mashable [249] of a web-based, i.e. web, application as:

> "an application that is accessed by users over a network such as the Internet or an intranet. The term may also mean a computer software application that is coded in a browser-supported programming language (such as JavaScript, combined with a browser-rendered markup language like HTML) and reliant on a common web browser to render the application executable.
>
> Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis and many other functions."

4.3.2 Types of web applications

In order to identify the different types of web applications, accessed either through a browser or via *apps* (Appendix B.2) using mobile devices, we summarise the taxonomy of domains provided by Deshpande et al. [94], i.e.: (1) *informational* where examples include newspapers, catalogues and other periodicals, (2) *interactive* involving user-provided information for customised access, e.g. registration, presentation or games, (3) *transactional* for on-line shopping, ordering goods and banking, (4) *workflow* for planning and scheduling systems, inventory management and status monitoring, (5) *collaborative work environments* such as distributed authoring systems and tools for collaborative design, (6) *on-line communities* or *marketplaces*, e.g. chat groups, recommender systems, marketplaces and auctions, (7) *portals* for electronic shopping malls and intermediaries, and (8) *web services* provided by enterprise applications, information and business intermediaries.

4.4 Web application architecture

The use of a particular software architecture for a project will be determined by differing business and technical factors, as well as the data required of the application. It is not within the scope of this thesis to document business considerations, and current language choices are described in a recent on-line study by Luenendonk [238]. Within the Java *ecosystem* itself, where our software resides (Section 4.5), Maple [246] has described a small number of contemporary frameworks and tools.

The nature of data and data format(s), discussed by Kappel et al. [209], with reference to Anastopoulos and Romberg [18] and Ceri et al. [62], takes "the layering aspect of architectures, or the support of different data and data formats - the data aspect of architecture - into account". The basis of this distinction is the format of the data that the web application will process,[1] i.e. whether the format conforms to table-based data in an RDBMS or XML documents, or if it is unstructured "multimedia contents, e.g., images, audio, and video" that may be produced by social media (Section 2.3) and which "typically do not follow an explicit scheme."

We do not consider the *peer-to-peer* or multimedia *streaming* architectures in this thesis because they do not apply to our paradigms (Section 1.4.1). Alternatively, architectures for structured data are made up by a series of *layers*. Several alternative implementations of this *layered* architecture exist, extending from the two-tier client/server model to *n*-tiered variants, where the intention is to *decouple* data access and business logic from the user interface (UI) to allow *separation of concerns*. This is achieved through the incorporation into the architecture of the model-view-controller (MVC) design pattern by Reenskaug [322]. Table 4.2 describes the various components of MVC, and their interaction is illustrated in Figure 4.2. Moreover, Table 4.2 lists the approximate correspondence of MVC with the *layers* of a typical *n*-tiered web application architecture: these layers are described in Table 4.3 and their placement in the architecture is illustrated in Figure 4.3. Layered web application architectures typically include the following components:

- **Client:** The *user agent*, i.e. a browser or browser plug-in, mobile device, or where a server delivers machine-readable data, e.g. RSS, to a client.
- **Firewall:** To protect a web application from unauthorised or hostile access.
- **Proxy:** To provide a cache of pages frequently accessed or for tracking purposes.

[1] We consider the format of RSS to be semi-structured (Section 2.2.3).

- **Servers:** These will typically include a *web server* to manage user requests and responses, a *database server* where the database resides, and an *application server* where system functionality is based.[2] Media or content management servers may also be present in particular systems.
- **Legacy applications:** Older systems which may form part of a web application.

Architectural/developmental characteristics of layered web applications include:

- **An adaptive interface:** Where a web site's interface must be flexible enough to allow for the rapid introduction of new features or services, e.g. the API of a *fashionable* third-party product, or because of the use of a *mash-up* (Appendix B.2).
- **Aspect-oriented programming:** AOP is described in Appendix B.2.
- **Patterns:** The use of creational, structural and behavioural patterns in application development, e.g. Gamma et al. [134] or Reenskaug's MVC pattern [322].
- **Frameworks:** Software frameworks include support programs, compilers, code libraries, an API and tools that bring together the different components to enable software development. Frameworks for web applications, e.g. Spring (`http://spring.io/`) and Hibernate (`http://hibernate.org/`), extend this to support the development of web applications and services by providing libraries for database access, interface design and session management. Web application frameworks also alleviate the overhead associated with common activities performed in web application development, e.g. writing repetitive boilerplate (Appendix B.2) code for DAO classes or HTML.

[2] For convenience, when discussing the architecture of our myDS and vRSS software in the following sections of this chapter, we employ the generic term *web server* to *substantially* include both application and web servers. This principle is illustrated in Figure 4.3.

Model-view-controller	***n*-tiered web application architecture layer**
The *model* handles the web application's domain data via business objects.	Data
The *view* is seen by users in a browser via HTML, JSPs or equivalent technologies.	Presentation
The *controller* manages the flow of the web application by receiving requests from users, interacting with the *model*, and returning responses via the *view*.	Application

Table 4.2: Approximate MVC and *n*-tiered web application architecture correspondence: cf. *all case studies*.

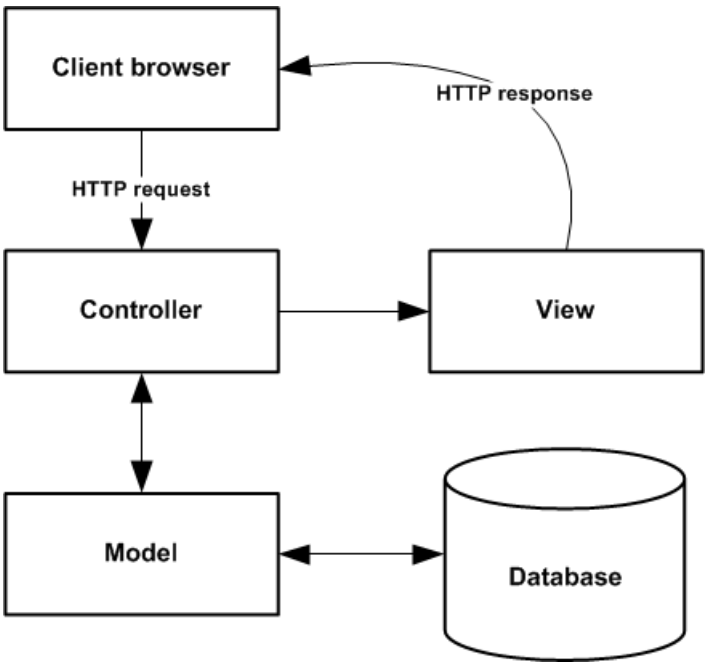


Figure 4.2: The MVC design pattern (adapted from Murach and Steelman [271]): cf. *all case studies*.

Layer	Description
Presentation	HTML pages and JSPs, or other equivalent technologies, which interact with web application via requests and responses rendered in a browser.
Application	Implements business logic and requires a run-time environment to do so. The application layer processes the requests received from users and serves responses back to them.
Data	Provides database access and retrieval services.

Table 4.3: Typical layers of the n-tiered web application architecture: cf. *all case studies*.

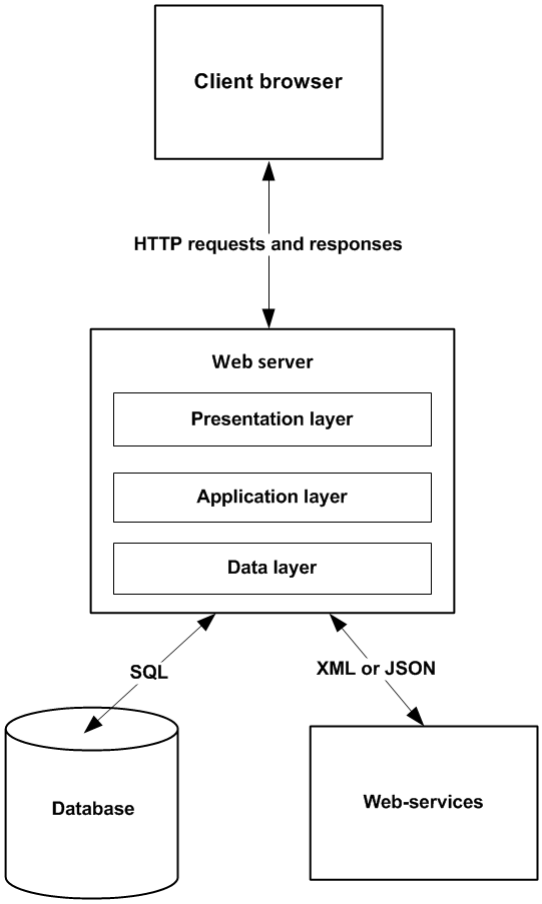


Figure 4.3: Typical n-tiered web application architecture: cf. *all case studies*.

4.5 The architecture of myDataSharer and visualRSS

4.5.1 Software overview

For the two case studies of our first paradigm, two *alpha*-version applications, i.e. myDataSharer (myDS) for case study one and visualRSS (vRSS) for our second case study,[3] were developed. myDS and vRSS conform *substantially* to MVC and the n-tiered web application architecture described in Section 4.4: we discuss the Java JSP/servlet-based implementation of these elements in Sections 4.5.5 and 4.5.6. The use of the relational database model by each application is described in Section 4.5.7.

Both applications share several common components and related terminology. The generics of these items are described in Section 4.7.1, and their specialisations in case studies one and two for our first paradigm are considered in Chapters 5 and 7 respectively. The extension of vRSS for the classification (Chapter 10) and sentiment analysis (Chapter 11) components of case study three for our second paradigm, and software characteristics of this, are the subject of Section 4.8. For our case studies, myDS and vRSS also make use of a series of open-source, third-party products[4] available within the Java *ecosystem* (Appendix B.1). The use of the principal products in each paradigm is described in Sections 4.7.1 and 4.8.1 respectively.

4.5.2 Choice of architecture

Within their n-tiered web application architecture, myDS and vRSS both employ Apache Tomcat (Appendix B.1) as a *container*, i.e. web server (Figure 4.3), to implement Java JSPs and servlets to support a request and response programming model (Section 4.5.5). These technologies are defined in the JEE specification [92] which documents Java APIs and how to coordinate them. This choice was based upon: (1) the popularity of the Java language, estimated at approximately 3.00% by W^{3}Techs in their "Usage of server side programming languages for websites" [451], (2) authorial familiarity with Java, and (3) the availability of the Java-based open-source, third-party products listed in Appendix B.1. We discuss our use of Java and OOP in Section 4.5.4.

[3] In Section 1.11 the naming convention we have used to refer to our software is described, i.e. full application names are employed in titles and captions, and contractions are used in the body text in chapters.

[4] Approximate counts of the number of lines of code in the principal open-source, third-party products used in our software, together with equivalent counts of the number of the *indigenous* lines of code in myDS and vRSS, are listed in Appendix D.2.

4.5.3 Operating system

myDS and vRSS are hosted on two virtual servers running 64-bit versions of Microsoft's Windows 7 Professional OS [263].

4.5.4 Programming model

Nørmark [281] defined four programming models: (1) *functional* in which all "computations are done by applying (calling) functions", (2) *imperative* or *procedural* programming, is a *top-down* paradigm involving the "execution of computational steps in an order governed by *control structures*". This can be traced to Backus, BNF and *Fortran* in the 1950s [192]. (3) *logic* based upon "axioms, inference rules and queries", and (4) the use of *object-oriented* programming concerns the "theory of concepts, and models of human interaction with real world phenomena" and their behaviour as classes and methods. The brief history of OOP given by Capretz [60] describes the origin of the term "object-oriented" in connection with the language *Smalltalk* by Goldberg and Robson [145]. Capretz [60] further wrote that Smalltalk was "first developed in 1972 in the Learning Research Group at Xerox Palo Alto Research Center", and was "greatly influenced by Simula as well as by Lisp." In addition, Coad and Yourdon [74] have succinctly defined object-oriented programming (OOP) as:

```
object-oriented = objects + classification + inheritance + communication
```

The object-oriented language Java, originally developed by Gosling in the mid-1990s [158], is the core language of our myDS and vRSS software given the n-tiered Java JSP/servlet-based web application architecture (Section 4.5.5) that they employ. Our use of Java in myDS and vRSS shares elements of both the object-oriented and procedural programming models. Examples of conventional OOP can be found in the servlets of the application layer in each application, and in the POJO classes which map the *model* to the tables of the database via DAOs and ORM (Section 4.5.6). In addition to this, in case study three a hybrid OOP/procedural model is used in the semi-automated batch processing of RSS feeds for classification and sentiment analysis (Section 4.8.2).

4.5.5 Managing requests and responses

With reference to the n-tiered architecture defined in Section 4.4, in Java-based web applications the presentation layer is frequently implemented by HTML pages and JSPs. A JSP, i.e. a JavaServer page, is defined by version 2.3 of the JSP specification [70] as "a text-based document that describes how to process a *request* to create a *response*." JSPs makes

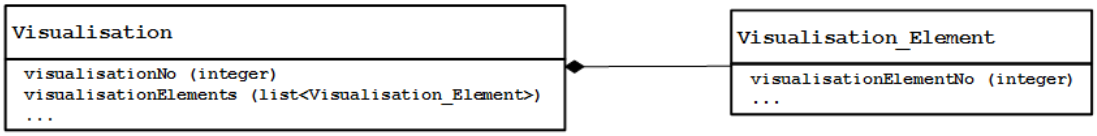
use of EL and JSTL *tag* libraries to display *dynamic* data. When an HTML *form* on a JSP is submitted via a browser, the HTTP request made interacts with the application layer via Java servlets running in a *container*, i.e. web server, implemented in myDS and vRSS by Apache Tomcat (Appendix B.1). Although we use version 2.5 of the servlet specification in our software, the current version 3.1 [64] describes a servlet as a "Java technology-based Web component, managed by a container, that generates dynamic content." The JSP/servlet interaction is enabled in Apache Tomcat by XML- or Java- based *deployment descriptors* which map the URLs of the HTML forms in JSPs to dedicated servlets: the descriptors also store authentication information for the URLs. Thus, given a particular request, the descriptor mappings are used by the web server to authenticate the user, and if this succeeds, the request is forwarded to the appropriate servlet. The method of the servlet corresponding to the HTTP GET or POST contained in the request is invoked, necessary processing is performed, and an HTTP response is returned back to the user, often as a second JSP. Appropriate to a request, servlets communicate with the database via the data layer (Section 4.5.6) to persist (Appendix B.2) data for storage and retrieval.

The HTTP request and response cycle is *stateless*, i.e. once a request has been made by a client via a browser, its connection to the web server is lost because neither device retains *state*, i.e. data or instructional information, between different requests and responses. To maintain state, web applications can use session tracking, where a *session* can identify a user across more than one page request or visit to a web site, and store information about that user. The servlet container, creates a *session ID* and *session object* per session, to maintain contact between client and web server. The session ID, often stored in a small text file known as a *cookie* is attached to each request from the client allowing the web server to relate it to the session object. Cookies, defined by Barth [27] as "name/value pairs and associated metadata", reside in the cache of a client's browser.

4.5.6 Application and data layer correspondence

Figure 4.4 illustrates the correspondence between the application and data layers of the n-tiered Java JSP/servlet-based web application architecture used by myDS and vRSS. In this example, two Java classes in vRSS for visualising RSS-mined data are displayed in an edited UML (Appendix B.2) class diagram, and the equivalent database tables are represented in an ERDM. *Object relational mapping* (ORM) maps the Java classes of the application layer to the data layer. In this example, vRSS's Java class `Visualisation` represents visualisation objects with a name, data, created date/time and other attributes. One of these attributes is the list `visualisationElements` of type `Visualisation_Element`

which is used to represent descriptive metadata (Appendix B.2) for the visualisation, e.g. axis labels. This arrangement allows an instance of class `Visualisation` to include *one-or-many* objects of class `Visualisation_Element`. In the ERDM, this resolves to a classical 1:*M* relationship between the two database tables persisting the data at run-time.



(a) UML class diagram.

(b) ERDM of *1:M* database relationship.

Figure 4.4: ORM in visualRSS using Java classes (top) and corresponding database tables (bottom) for visualisations (attributes and methods have been edited for clarity): cf. *all case studies*.

Access by the data layers in myDS and vRSS to the relational databases forming their data models (Section 4.5.7) is enabled by DAO classes. These objects, typically containing boilerplate code written to enable CRUD methods for operations upon database tables, contain code adapted by the author from Murach and Steelman [271]. In order for the DAOs to connect to their respective databases, each application uses JDBC, i.e. *Java database connectivity* (Appendix B.2), implemented through a database connection pool. In DBCP a collection of connection *objects* is maintained in a *pool* shared between all application users. Therefore, when a request from a user requires database access, the servlet concerned *spawns* a thread which gets a *connection* object from the pool. The database is accessed using the connection object through the appropriate method in a DAO class, where the necessary SQL for the action is embedded in the code of the method. Following this, the thread returns the connection object to the pool for subsequent use, and the server serves the response to the user.

4.5.7 Data model

Both myDS and vRSS employ the popular open-source RDBMS MySQL (Appendix B.1). MySQL is an implementation of the classical relational database model proposed by Codd in 1970 [75] and uses *relations*, i.e. tables, to store data. Each table maintains one or more

primary key attributes (columns) to uniquely identify records (rows): these attributes may be present on other tables as *foreign keys* to establish relationships which describe the ways in which the tables are related. Each relationship has a degree, i.e. one-to-one (*1:1*), one-to-many (*1:M*) or many-to-many (*M:N*), to determine its cardinality. Principal constraints on these relationships, defined by Codd in 1979 [76], concern *entity integrity* to guarantee the uniqueness of the primary keys in each table, and *referential integrity*[5] to prevent mismatched foreign keys between tables. Both constraint types are used by an RDBMS to ensure the ACID properties (Appendix B.2) of database transactions for CRUD operations. The *schema* or data model of a database is frequently represented visually by an ERDM.

SQL, originally developed by Chamberlin and Boyce [63] in the early 1970s, is typically used to define tables and to query and update data in relational databases. SQL has since been extended several times and it became an ANSI standard in 1986, and an ISO standard in 1987. The latest version of the language is SQL 2011 [203]. MySQL and its compliance with ANSI/ISO standards are described in Appendix B.1.

4.6 Keyword conventions and characteristics

Our use of the term *keyword* is according to the definition provided by the OED at `http://www.oxforddictionaries.com/definition/english/keyword`, i.e. that a *keyword* is a "word used in an information retrieval system to indicate the content of a document." Other definitions refer to a "word or concept of great significance" or a "significant word mentioned in an index". A contemporary example of the use of keywords is provided by a search engine: keywords are entered into the home-page of the search engine in a browser by a user with the expectation of a list of links to web sites, i.e. *documents*, related to the keywords, being provided as the response. SEO techniques applied to web pages, e.g. the use of `<meta>` tagging in HTML, are frequently used to improve a web site's placing in search engine listings. Similarly, in the case studies which demonstrate our paradigms, a "document" is an RSS feed and the keywords are used to "indicate the content" in the text of the feed.

In our paradigms and their case studies, user-entered or system-generated keywords are, with the exception of the sentiment analysis component of case study three (Chapter 11), naïve, i.e. simple *n-grams* (Appendix B.2), which are not disambiguated, subject to NER or significant cleansing. Furthermore, excluding the user-entered keywords in the case

[5] cf. Abiteboul et al. [1], where referential integrity is defined as *inclusion dependency*.

studies for our first paradigm, our keywords are *independent* of each other, i.e. we pay no attention to *dependent correlation*[6] or to any other type of keyword *relatedness*. In addition to this, we base the *popularity* of our keywords upon the fluctuations in their frequencies in the text of RSS feeds measured between pre-defined starting and ending dates/times (Section 1.4.2). Within this context, we do not account for the *temporality* of keywords, i.e. the duration for which a given keyword is present in, or absent from, RSS feeds between the aforementioned dates/times. Therefore, our keywords and their frequencies conform to the vector space model (VSM) (Appendix B.2) representation of documents and contents, i.e. the RSS feeds comprise *documents* and the *content* of each vector corresponds to a bag-of-words (BoW) (Appendix B.2) containing the frequencies of keywords therein. We employ no other keyword-based metrics. Although variations exist in our paradigms and their case studies, our general keyword conventions and characteristics are listed below:

- **Paradigm 1: case studies one and two:** Keywords are lowercase and without limitation. Case study two further excludes stop words and restricts keywords to unigrams.
- **Paradigm 2: case study three:** For our classification work, we allow a maximum length of trigram[7] but use stop words and *stemming* (Appendix B.2). For our sentiment analysis work, a basic NER identifies *proper* nouns as keywords (Section 11.4.5), and we allow stop words, but not stemming.

4.7 Paradigm one: software fundamentals

4.7.1 Common software components

myDS and vRSS both make use of several common software components and related terminology, e.g. *mining rules*, and *polling and mining*. These components are conceptual rather than actual, i.e. where our code implementing each component in one application is distinct from the equivalent code in the other.

[6] cf. Fisher et al. [122], described in Sections 3.3.3 and 12.5.4, who defined *dependent correlation* as a means to "discover when co-occurrences between a pair of words are coincidental, and when they are part of the story."

[7] In our opinion, as the length of a keyword increases, the semantics of the term *keyword* are *diluted*.

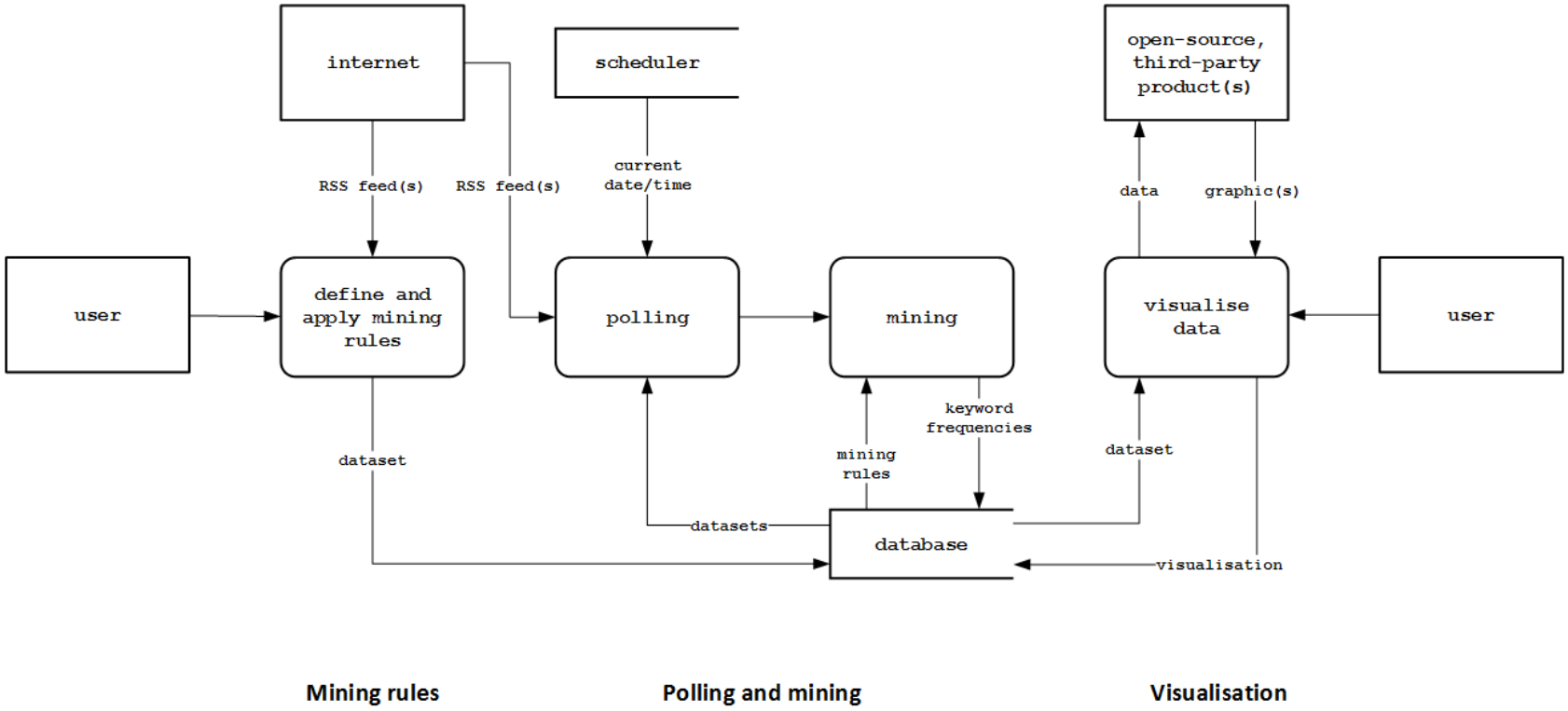


Figure 4.5: DFD representing the common software components and related terminology of myDataSharer and visualRSS: cf. *case studies one and two.*

Case study	Application/ component	Products Mining rules	Polling and mining	Visualisation
One	myDataSharer	Rome	Quartz Scheduler Rome	Google Charts
Two	visualRSS	Lucene Rome	Lucene Quartz Scheduler Rome	Google Charts
One and two	Database	MySQL		
	Web server	Apache Tomcat		

Table 4.4: Distribution of principal open-source, third-party products by common conceptual components: cf. *case studies one and two.*

The common components and concepts of myDS and vRSS are represented in Figure 4.5 as generic processes in an abstract DFD.[8] This DFD (Appendix B.2) is *illustrative*, i.e. it does not display all of the data flows between the processes within either application. For this reason, the *generic* nature and common use of these processes are described in the following sections of this chapter. Where the specifics of the actual implementation of these components in our myDS or vRSS software extends or differs to these descriptions, Chapters 5 and 7, respectively, provide more information.

In addition to the DFD in Figure 4.5, Table 4.4 lists the distribution of the principal open-source, third-party products (Appendix B.1) from the Java *ecosystem* that are used in the common components of our myDS and vRSS software on a black-box, mash-up basis. These products are referred to in the following sections of this chapter for both of our paradigms, and in Parts II and III for their respective case studies.

4.7.2 Mining rules

Mining rules[9] form the basis of our first paradigm. They are defined upon RSS and are intended to provide a straightforward means for users to specify how textual and numeric data is to be mined (Section 4.7.5) from feeds during polling (Section 4.7.3) and persisted to database storage. This is in order to update and visualise the objects that the rules become part of, i.e. datasets for case study one (Section 5.3), and visualisations in our second case study (Section 7.3). Table 4.5 lists generic mining rules which apply to the common components of myDS and vRSS referred to in Section 4.7.1.

[8] A sibling DFD for case study three for our second paradigm is displayed in Figure 4.6.

[9] As described in Section 1.4.1, the term *mining rules* is our own and applies only to the software implementing the case studies for our first paradigm: no further meaning of this term is intended or should be inferred.

Mining rule	Description
RSS feeds	The RSS feeds to be mined for data textual and numeric data.
RSS feed elements	RSS feed elements, defaulted to the `<title>`, `<description>` and `<pubDate>`, from which mining will occur during polling.
Keywords	As described in Section 4.6.
Mining type	Identifies how RSS feeds are mined. In case study one, myDS employs occurrence (OM) and value mining (VM). In vRSS for case study two, automatic, semi-automatic and manual mining are used.
Duration	Dates/times between which polling will occur.
Visualisation	The visualisation type to represent data mined from RSS.
Polling frequency	The number of hours between polling cycles, defaulted to one.
Name and description	Descriptive representation of the objects that the mining rules become part. In case study one mining rules become part of datasets (Section 5.4) and in our second case study, the rules become part of visualisations (Section 7.4).

Table 4.5: Generic mining rules: cf. *case studies one and two.*

4.7.3 Polling

Polling in myDS and vRSS is scheduled (Section 4.7.4) on an hourly basis and serves two purposes: (1) to control how frequently RSS feeds are polled for new postings, and (2) the mining of any new postings that have been made to the feeds since they were last polled. This comparison is based upon the BST/GMT date/time recorded in the `<pubDate>` element of each `<item>` in an RSS feed (Section 2.2.3), and the polling date/time. If they are found, new postings are mined (Section 4.7.5) and data is persisted to database storage (Section 4.7.6).

4.7.4 Scheduling

myDS and vRSS both employ the open-source, Java-based Quartz Scheduler (Appendix B.1) to control the frequency of polling in case studies one and two. Despite differences in each application's polling arrangements, the basic configuration of Quartz Scheduler remains the same. This is illustrated in the Java code in Algorithm 4.1 which initialises the hourly Quartz Scheduler *job* in vRSS to poll RSS feeds.

```
1: int schedulerJobNo = 1;
2: SchedulerJob sj = Scheduler_Job_DB.get(schedulerJobNo);
3: jd = new JobDetail(sj.getJobName(), Scheduler.DEFAULT_GROUP,
                      Poll_RSS_Feeds_ScheduleJob.class);
4: ct = new CronTrigger(sj.getJobQuartzTrigger(), Scheduler.DEFAULT_GROUP, "" +
                        sj.getJobCronExpression().trim() + "");
5: scheduler.scheduleJob(jd, ct);
```

Algorithm 4.1: visualRSS's Java code for hourly polling using Quartz Scheduler: cf. *all case studies*.

The integer variable `schedulerJobNo` (1) is *initialised* with the value of the primary key (Section 4.5.7) of the row in the `Scheduler_Job` table in vRSS's database which stores the scheduling information needed for polling. The polling information is retrieved from the database (2) using the *get* method of the `Scheduler_Job` DAO class (Section 4.5.6). Implicit in (2) is the *instantiation* of the `Scheduler_Job` object that provides the scheduling information to the Quartz Scheduler-based `JobDetail` and `CronTrigger` (3, 4) objects. Finally, Quartz Scheduler submits an instruction to the OS (5) to run the job every hour according to the `CRON`-based (Appendix B.2) timer expression retrieved in (2).

4.7.5 Mining data from RSS

At the heart of the polling process (Section 4.7.3) is Rome, i.e. *(R)ss and at(OM) utiliti(E)s*, which provides a set of utilities to parse RSS and Atom (Appendix B.2) feeds. During polling to update an object which mining rules have become part of, new postings made to the RSS feeds defined in the object's rules are parsed by Rome. The text of the postings is then mined according to the mining rules and data is persisted to database storage as part of the object. Rome is also used during the definition of mining rules (Sections 5.4 and 7.4 respectively) allowing users to sort and restrict the current RSS feed content, before the objects the rules become part of (Section 4.7.2) are similarly persisted to the database. Algorithm 4.2 lists Java code for Rome to open a connection to a web site (1, 2) in order to retrieve the *hosted* RSS feed. This exchange is made in lines (3) and (4), after which the `<SyndFeed>` object (5) is populated with the feed's content.

```
1: url = new URL(rssFeedURL);
2: URLConnection urlConn = url.openConnection();
3: XmlReader reader = new XmlReader(urlConn);
4: SyndFeedInput input = new SyndFeedInput();
5: SyndFeed rssFeed = input.build(reader);
```

Algorithm 4.2: Java code to allow Rome to connect to a web site and retrieve the *hosted* RSS feed: cf. *case studies one and two*.

Rome parses RSS and Atom feeds by using its own generic `<SyndFeed>` objects. Each `<SyndFeed>` contains a variable number of `<SyndEntry>`, i.e. *syndicated entry*, objects. Each `<SyndEntry>` represents an `<item>` of the content parsed from an RSS feed, i.e. a posting (Section 2.2.3). Algorithm 4.3 lists Java code populating a `String` object named `title` with text parsed by Rome from the `<title>` element of an RSS feed `<item>`.

```
1: List<SyndEntry> rssFeedItems = rssFeed.getEntries();
2: for (SyndEntry rssFeedItem : rssFeedItems) {
3:      String title = rssFeedItem.getTitle();
4:      ...
5: }
```

Algorithm 4.3: Use of Rome objects to populate a Java `String` object: cf. *case studies one and two.*

Rome (Appendix B.1) was selected for use in our software because it is written in Java and open-source, well-documented and, at the time of selection, was sponsored by Sun Microsystems at `http://www.java.sun.com`. The product's generic handling of feed types also precluded any incompatibilities between the different versions of RSS: several reported cases of this nature are documented in Section 3.5.1. Moreover, Rome was also used in several of the appropriate examples of related work we compare with our first paradigm and its case studies in Chapter 9.

4.7.6 Persisting RSS-mined data to database storage

Previous sections of this chapter have described the general arrangements that myDS and vRSS share for mining data from RSS during polling. The specifics of each application's persistence (Appendix B.2) of mined data to database storage are described in Sections 5.6 and 7.6 respectively.

4.7.7 Visualising data

Section 2.8.4 lists a series of third-party products which can be used to visualise data. In connection with this, for the two case studies for our first paradigm, the products listed in Table 4.6 were considered according to the following criteria which were based upon Harriott [167], i.e.: (1) the appropriateness of the product given the Java-based implementation (Section 4.5.4) of myDS and vRSS, (2) the ability to create a variety of qualitative and interactive visualisations, (3) to be produced by an established organisation, (4) a *shallow* learning curve, (5) to be open-source, and (6) the availability of support.

The evaluation of the products listed in Table 4.6, according to these criteria, resulted in the selection of Google Charts (Appendix B.1) for the case studies for our first paradigm, because it is free to use, web-based and support is provided via a mailing list.

The Google Charts visualisation process consists of five stages: (1) data to be visualised is retrieved from the database according to the mining rules specified, (2) the data is formatted in JavaScript and then (3) transmitted on-line to Google Charts. An SVG or VML graphic is then returned from Google Charts (4) for rendering in the browser (5). The Google Charts visualisation types, which provide the majority of the available types in our myDS and vRSS software, consist of pie and x - y charts: a Google Charts tree-map (Appendix B.2) was also used in case study two. Other types include a *word-cloud* written by the author and a Weka decision tree (Section 10.5) in myDS.

Product	**Description**
JFreeChart	An open-source Java chart library available at `http://www.jfree.org/jfreechart/`.
GnuPlot	A "portable command-line driven graphing utility for Linux, OS/2, MS Windows, OSX, VMS, and many other platforms." GnuPlot is available from (`http://www.gnuplot.info/`).
Google Charts	A JavaScript-based set of libraries that allows interactive visualisations to be embedded in web pages, available at `https://developers.google.com/chart/`.
Processing	Available at `https://processing.org/`, Processing is a "flexible software sketchbook and a language for learning how to code within the context of the visual arts."
prefuse flare	Prefuse at `http://prefuse.org/` provides "provides visualization and animation tools for ActionScript and the Adobe Flash Player".
Simile Exhibit	Allows creation of web pages "with advanced text search and filtering functionalities, with interactive maps, timelines and other visualizations", at `http://www.simile-widgets.org/exhibit/`.

Table 4.6: Third-party products considered for visualising data mined from RSS (the descriptions quote text from the web sites of the respective software publishers): cf. *case studies one and two.*

4.8 Paradigm two: software miscellany

4.8.1 Extending visualRSS

The common components of our myDS and vRSS software (Section 4.7.1) for the case studies for our first paradigm, had no significant application in case study three for our second paradigm (Section 1.4.1). Nevertheless, the n-tiered Java JSP/servlet-based web application architecture (Section 4.5.5) used in myDS and vRSS does apply to our second paradigm. This is because the software written for case study three comprises an extension of vRSS from case study two: therefore that application's architecture and database were retained.

Figure 4.6 provides an abstract DFD (Appendix B.2) which displays the basic processes of the classification and sentiment analysis components for our third case study. As with the sibling DFD for our first and second case studies (Figure 4.5), this DFD is *illustrative* and does not include all of the data flows and processes. Chapters 10 and 11 describe the processes shown in the DFD for: (1) classifying RSS feeds into categories according to the fluctuations in the frequencies of popular keywords present in their text, and (2) to determine a correlation between the changes in the keyword frequencies and sentiment. The DFD also refers to manual facilities originally developed for our sentiment analysis work (Figure 4.7): these were superseded by the batch processing elements described in Section 4.8.2. Table 4.7 complements Figure 4.6 by listing the distribution of the principal open-source, third-party products employed in these processes. Moreover, the following sections of this chapter discuss software characteristics of vRSS in case study three.

4.8.2 Batch processing

In Section 4.5.4 Java is referred to as the OOP-based programming language used in our software. We also employ a hybrid OOP/procedural model in the semi-automated batch processing of RSS feeds, written for the classification component of our third case study to validate its use at RSS feed category-level for our sentiment analysis work. The batch processes manipulate POJOs (Section 4.8.3), e.g. when processing data returned by a call to a method in a DAO class (Section 4.5.6), but the Java classes implementing the batch processes are never actually *instantiated* as objects in the typical object-oriented manner. The batch processing classes lack *constructors* and do not implement accessor and mutator, i.e. *get* and *set*, methods, based upon individual attributes of objects: these classes also include class-based, not instance-based, *static* methods. Quartz Scheduler (Section 4.7.4) was used to run the batch processing during case study three.

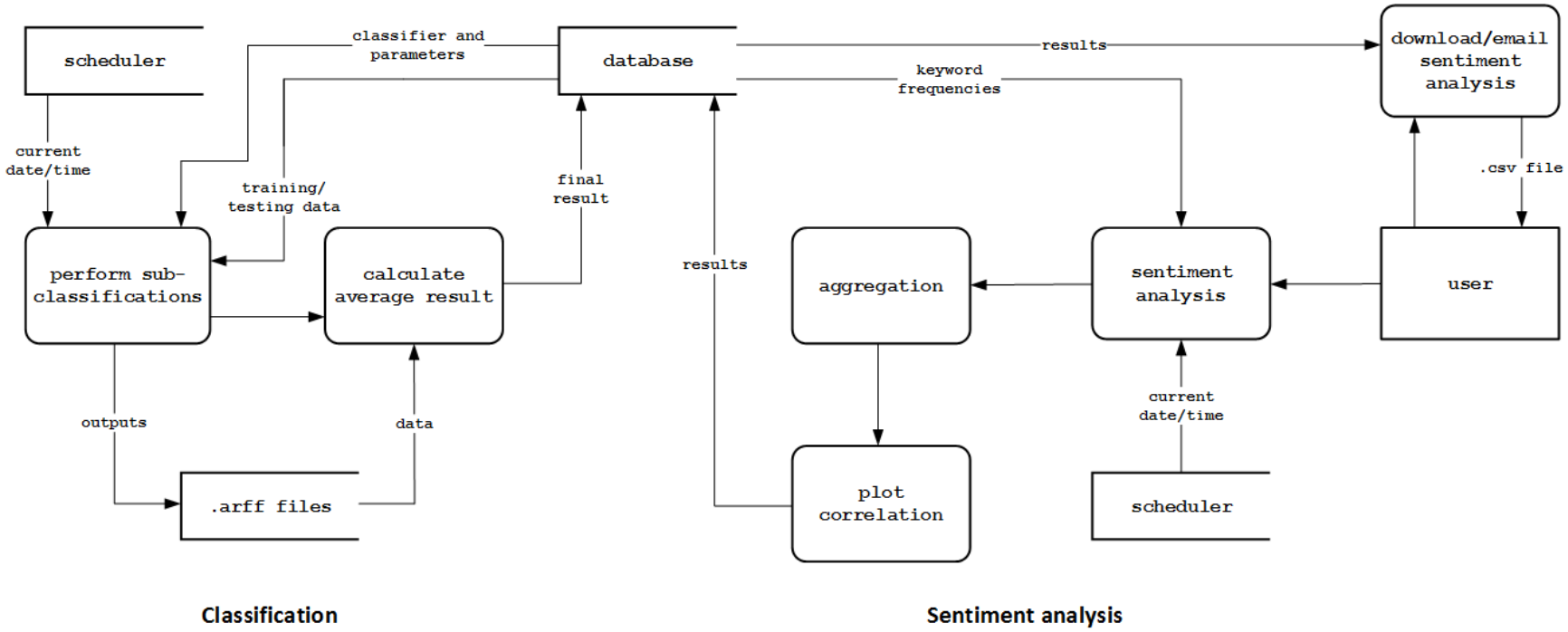


Figure 4.6: DFD representing the processes of classification and sentiment analysis in visualRSS: cf. *case study three*.

Case study	Application/ component	Products	
		Classification	Sentiment analysis
Three	visualRSS	Lucene	Lucene
		Quartz Scheduler	Quartz Scheduler
		Weka	SentiStrength
	Database	MySQL	
	Web server	Apache Tomcat	

Table 4.7: Distribution of principal open-source, third-party products: cf. *case study three*.

4.8.3 Class hierarchy

The processes illustrated in Figure 4.6 for classification and sentiment analysis employ a simple hierarchy of Java classes to maintain the RSS feeds, keyword frequencies and other data required of their various functions. These classes are illustrated in the edited UML (Appendix B.2) class diagram in Figure 4.7.

The superclass of this hierarchy, i.e. `RSS_Feed_Miner`, has dedicated naming attributes and an `RSS_Feed_Polling` object which includes start and stop dates/times. The RSS feeds are stored in a series of parallel lists along with the elements and categories to be used. Keywords are stored in a key/value *map* in class `RSS_Feed_Occurrence_Miner`. These classes were originally written as part of our vRSS software to store mining rules defined in case study two (Section 7.5), but were readily extended for our third case study, i.e. where the attributes of classes `RSS_Feed_OccurrenceData_Miner` and `RSS_Feed_OccurrenceSentiment_Miner` address classification and sentiment analysis respectively.

4.8.4 Interface design

In Section 6.4.2, we explain why myDS's textual mining rules in our first case study were refined for vRSS in case study two. A consequence of this allowed the development in vRSS of a *lightweight*, uncluttered interface, with common controls, e.g. search and customised *blurb* at the top of each page, to identify a page's operation and navigation within the application. We provide an example of this in Figure 4.8 using one of the pages of the manual facilities, originally developed but subsequently discontinued, for the sentiment analysis component of case study three (Chapter 11).

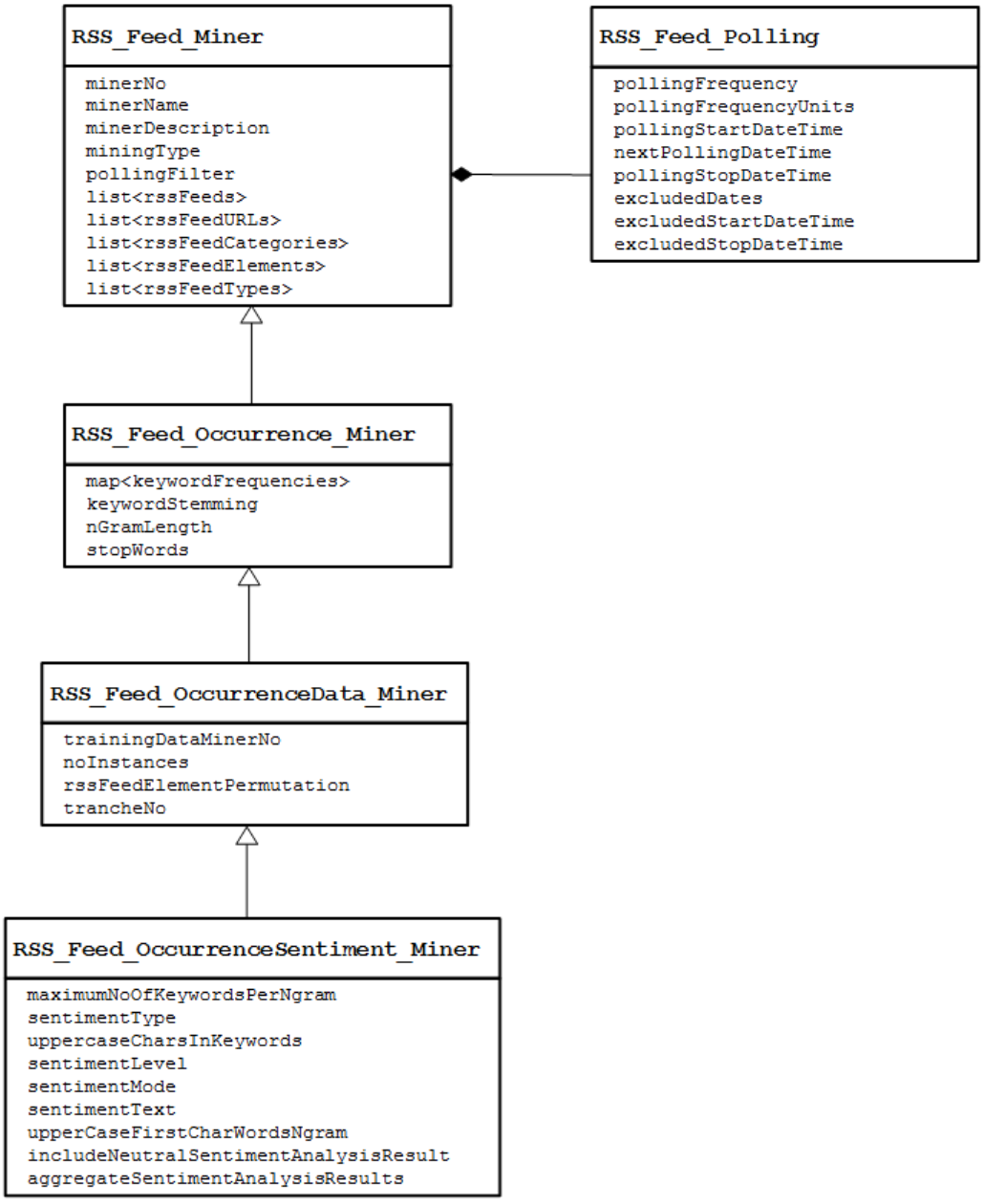


Figure 4.7: UML class diagram of visualRSS's class hierarchy for case study two, and its extension for our classification and sentiment analysis work (methods have been edited for clarity): cf. *case study three*.

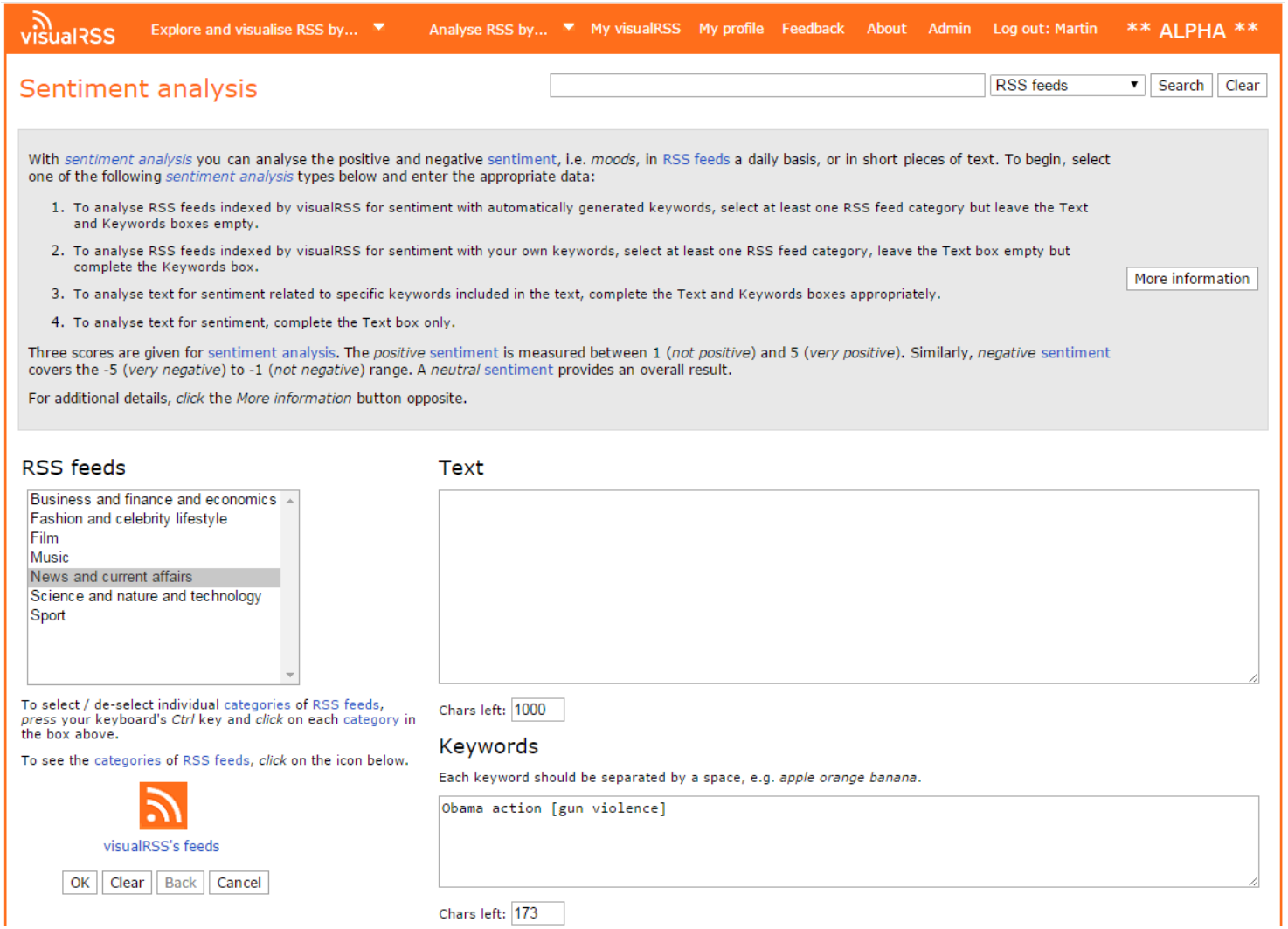


Figure 4.8: Screen-dump of first page for manual sentiment analysis, developed but discontinued, in visualRSS: cf. *case study three*.

4.9 Development tools

myDS and vRSS were developed using NetBeans IDE which is described in Appendix B.2.

4.10 Afterword

This chapter has described web engineering and web application architecture as they apply to our myDS and vRSS software. This material provides the basis for our description of the architectural and technical foundations of both applications, the common components they share, and the distinct implementation of these components according to the case studies of our first paradigm. Case study three's extension of our vRSS software, and several characteristics therein for our second paradigm, have also been briefly commented upon.

The following chapters present the case studies for our paradigms: Part II documents our first paradigm and its two case studies, whilst Part III focuses upon case study three for our second paradigm.

Part II

Paradigm 1: Defining mining rules upon RSS to determine and visualise trends from textual and numeric data

Chapter 5

Case study one: The myDataSharer software

5.1 Foreword

This foreword serves two purposes. The first of these concerns the organisation of Part II of this thesis wherein five chapters present our first RSS-mining paradigm (Section 1.4.1) and its two case studies. The first four chapters, which include this one, form two pairs where each pair addresses one of the two case studies. Together, this chapter and Chapter 6 address case study one, whilst Chapters 7 and 8 focus upon our second case study. The first chapter of each pair concentrates upon software, and the second chapter concerns actual work and results. Finally, in Chapter 9, we compare our first paradigm with appropriate examples of related work.

The second purpose of this foreword is to introduce this chapter specifically as part of our presentation of case study one's use of mining rules for our first paradigm. Therefore, we begin with a brief reference to the case study in Section 5.2. The specifics of our myDS software (Section 5.3) are then described in terms of the common components and terminology shared by the two applications written for our first paradigm (Section 4.7.1). This description extends to the two mining types (Section 5.4) implemented in myDS to define mining rules, and the polling of RSS feeds for new data (Section 5.5). We continue with database persistence in Section 5.6, the use of myDS's *diary* to record user actions in Section 5.7, and Section 5.8 which concerns visualising data in myDS.

5.2 Case study one

myDS was used during our first case study which took place in late 2009. This case study, described in Chapter 6, employed a corpus of thirty-five part- and full- time Masters-level students. We sought through our first case study to answer a series of research questions concerning the feasibility of our first paradigm in terms of the mining types (Section 5.4.1) developed, and the efficiency of their use.

5.3 The myDataSharer platform

myDS formed the platform for our first case study. The application was originally based upon the premise of modelling user behaviour within a social *data-sharing* environment, i.e. bringing community and data together into a *Web 2.0* (Appendix B.2) *entity*. myDS allows members to freely form communities of like-minded or interested individuals and parties. In these communities, *textual* or *numeric* data, uploaded from a wide variety of formats, or mined from RSS, forms datasets that are unrestricted in range and visible to community members and non-members alike. Various facilities are included to allow community members to define content upon the datasets, i.e. queries, analyses and visualisations, and each community has an integrated BBS (Appendix B.2) supplied by jforum (Appendix B.1) providing for discussion and debate.

In order to model user behaviour, myDS records information about their activities, data uploaded or mined and facilities used, via a logging tool called the *diary* (Section 5.7). To facilitate this, each myDS user has a personalised home-page listing the communities they are members of, datasets and other content they have created. This content can be accessed via a *tree*-control appearing on every page. Browse pages are also present and a search facility uses keywords produced from tags entered and content names. Feedback, profile maintenance and a series of descriptive *About* pages explain the application.

Despite myDS's original objectives, in this chapter we focus specifically upon the application's RSS-mining component.

5.4 Defining mining rules

5.4.1 Mining types

We define generic mining rules as they apply to the case studies of our first paradigm in Section 4.7.2. One of these mining rules is the *mining type* which identifies how RSS feeds are mined (Section 4.7.5): two distinct mining types were implemented in myDS and used by the student corpus in our first case study:

1. **Occurrence mining:** OM counts the occurrences of specified strings present in the text of RSS feeds to explore trends or track issues. Examples of these may include recording the number of times a politician, sportsman or media personality is mentioned in feeds relating to a particular event. The three variants of OM are listed in Table 5.1.

2. **Value mining:** VM analyses RSS feeds which provide structured content. Such feeds are produced by dedicated web sites which report modern forms of ticker-tape (Appendix B.2) data, such as financial movements, sports or lottery results. Therefore VM allows users to define and mine values of interest. An example of VM is where a user mines values of several currency exchange rates from an RSS feed for a few days to compare their fluctuations.

Variant	**Description**
Count all items	To count all of the `<item>` elements in an RSS feed.
Count items	This will mine the count of `<item>` elements containing a specified string in any of the displayed columns, i.e. elements, of an RSS feed. An `<item>` is only counted once even if it contains more than one instance of the string.
Count occurrences	To mine the count of occurrences of a specified string in a particular column of an RSS feed. If the column contains multiple occurrences of the string, each occurrence is counted separately.

Table 5.1: Occurrence mining variants: cf. *case study one*.

5.4.2 The relationship of mining rules to columns of datasets

The definition of mining rules for a dataset in myDS works on two levels:

1. **Dataset-level:** Includes a name and description for the resulting dataset, created once column-level mining rules have been defined.

2. **Column-level:** A set of mining rules defined upon one or more `<item>` elements (Section 2.2.3) of one or more RSS feeds. A dataset can have many sets of column-level rules.

Once mining rules have been defined for a dataset (Appendix B.2), they are persisted to database storage. This involves the dynamic creation of a dedicated table for the dataset in the database (Section 5.6), which is subsequently populated with data mined during polling of the RSS feeds defined in the column-level mining rules. Each column in the table corresponds to a set of column-level mining rules. The definition of sets of column-level mining rules for a dataset works on an iterative, *columnar* basis, i.e. a set of column-level rules is produced per iteration of the definition process and requires a mining type (Section 5.4.1), the RSS feeds to be mined, XML filters which identify elements in the feeds for the mining type (Section 5.4.3) to mine, and a defining name. Therefore, a set of OM column-level mining rules can include one or many RSS feeds from which data will be mined, but because of the nature of VM, only a single RSS feed is permitted in a column-level set of mining rules.

This connection, between the sets of column-level mining rules and the columns of the dataset's table, is the *relationship of mining rules to columns of datasets*. It follows from this that a dataset-level set of mining rules can include column-level sets of rules for OM and VM, but a single column-level set of rules cannot make use of both mining types.

The process to define dataset- and column- level mining rules for a dataset is illustrated in the edited DFD in Figure 5.1. In Figure 5.1, the terms *dataset* and *column* are *substantially* equivalent to their respective level of mining rules.

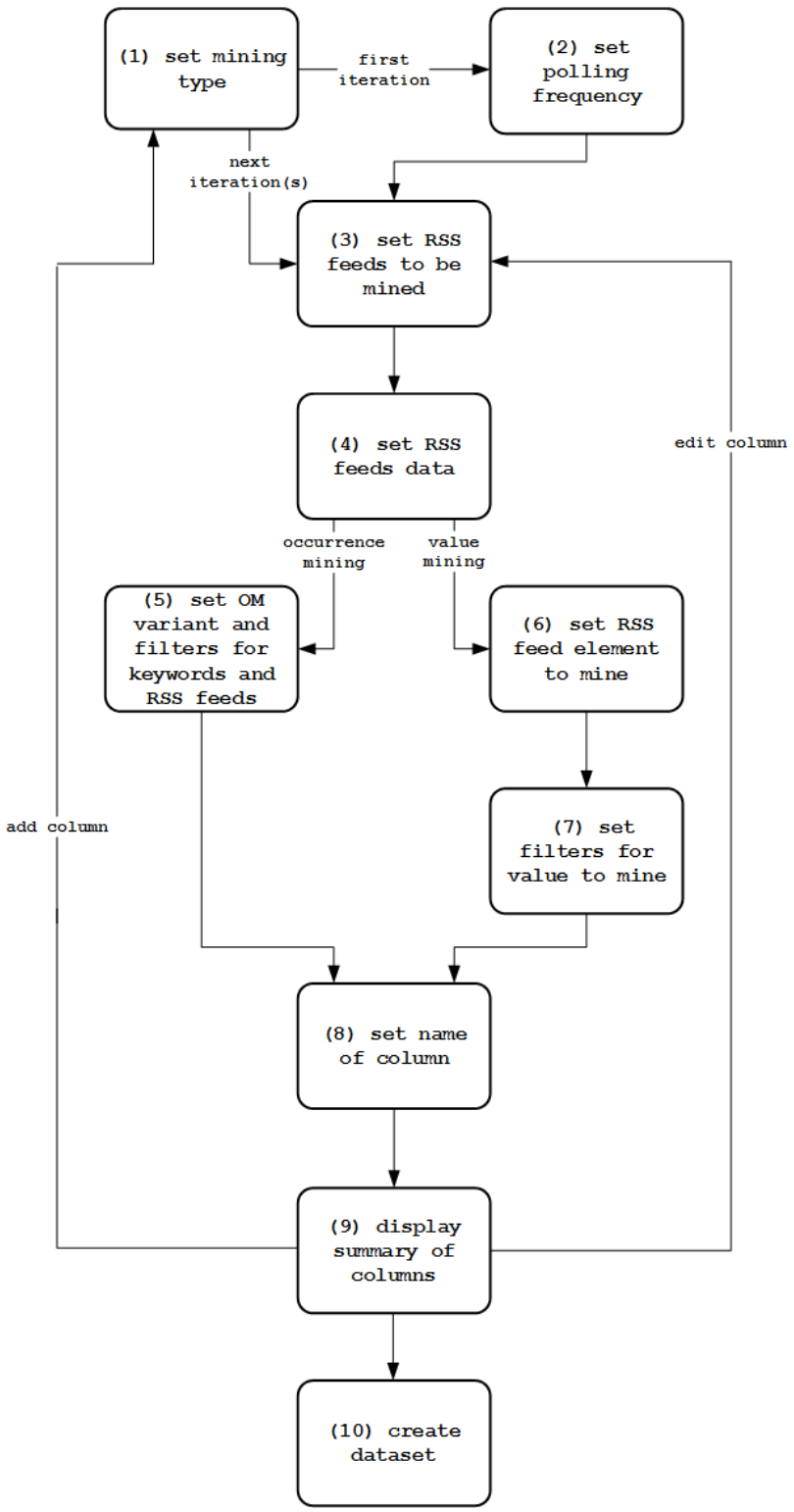


Figure 5.1: DFD illustrating the process to define mining rules for a dataset in myDataSharer (data stores and external entities have been edited for clarity): cf. *case study one*.

The process illustrated in Figure 5.1 includes a total of ten stages, each of which has a dedicated page in myDS. To explain this process: each column-level set of mining rules requires: (1) the choice of the column's mining type (Section 5.4.1), (2) the polling frequency (Section 5.5), which is only specified during the first iteration for the first set of column-level rules, and (3) the RSS feeds to be mined for the column. (4) displays the data from the RSS feeds already entered and provides various controls to allow users to sort and restrict it, e.g. to show only RSS feed `<item>` elements where the *Description* `Contains text:` *Afghan*. After (4), branching occurs due to the mining type:

- **Occurrence mining:** (5) allows the definition the OM variant, and the filters for the mining rules for keywords and RSS feeds to be mined.
- **Value mining:** (6) requires the choice of RSS feed element to be filtered, and (7) is concerned with identifying the *value to mine* from the element chosen in (6).

(8) is a common stage for both mining types requiring a name to be given to the column-level set of mining rules: this name will in turn be used as the name of the column in the dataset's database table. (9) provides a summary of the mining rules for each column so far defined and allows them to be edited: other options allow the polling frequency to be modified and for new sets of column-level rules to be created. Finally, (10) allows dataset-level mining rules to be set, i.e. the dataset's name and polling dates, whereupon the mining rules are persisted to database storage (Section 5.6) and the dataset's table is created in the database. The new dataset can now be updated with data mined from RSS feeds during polling (Section 5.5).

In addition to Figure 5.1, Figure 5.2 and Table 5.2 illustrate the correspondence of the relationship of mining rules to columns of datasets during polling. In this illustration, a mock dataset has been compiled by combining the examples of OM (Algorithm 5.1) and VM (Algorithm 5.2) in Section 5.4.3. We can see in Figure 5.2 that the mock dataset consists of two columns, i.e. `Commonwealth` and `IBMprice`, which employ OM and VM respectively to mine data from three RSS feeds between 17 00 - 21 00 on 29 Sep 2009.

Table 5.2 lists the mining rules for each column of the dataset. Column `Commonwealth` employs OM and is updated every hour from RSS feeds *BBC news* and *Sky news*[1] with the collected frequency of keyword *Commonwealth* from the `<description>` element of each feed. On the other hand, column `IBMprice` uses VM to mine the value of IBM's stock from the `<description>` element of Nasdaq's RSS feed. To return to the relationship

[1]The RSS feed for *Sky news* is not actually present in Algorithm 5.1: it has been added here to emphasise the relationship of mining rules to columns of datasets.

of mining rules to columns of datasets, the actual definition of the column-level mining rules for the mock dataset would require two iterations of stages (1) to (9) in the process illustrated in the DFD in Figure 5.1, i.e. the first iteration to define rules for column `Commonwealth`, and the second iteration for `IBMprice`, before (10) persists these and the dataset-level mining rules to the database.

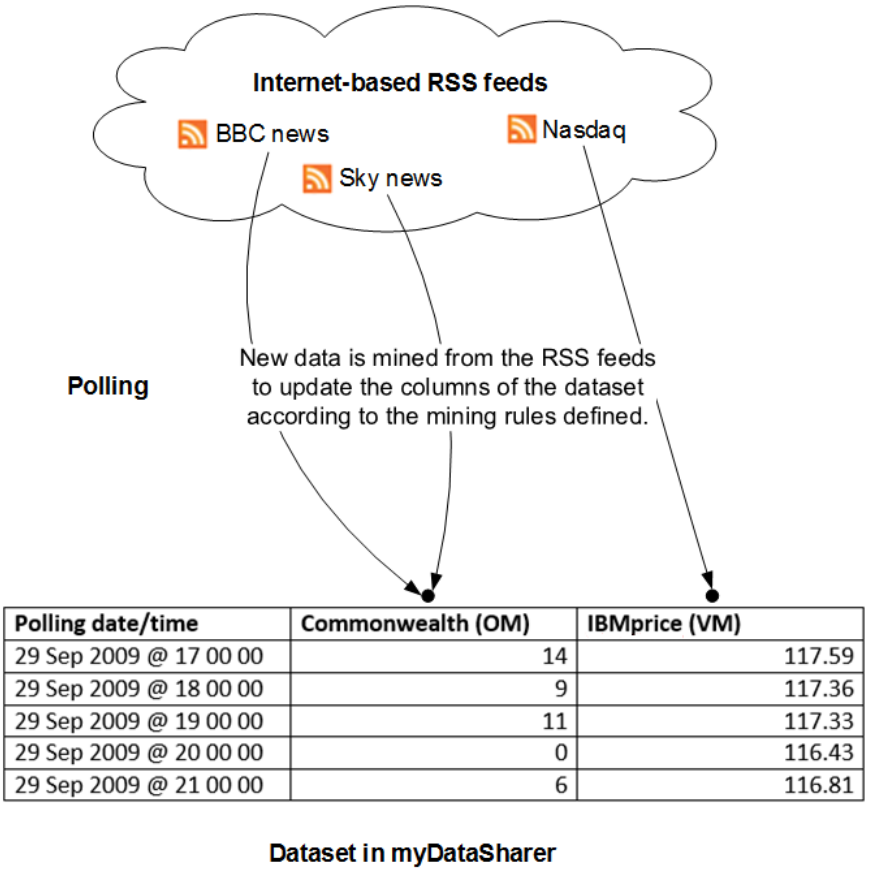


Figure 5.2: Sample data from polling RSS feeds of mock dataset: cf. *case study one*.

Column name	RSS feeds	RSS feed element	Mining type	Description
`Commonwealth`	*BBC news* *Sky news*	`<description>`	OM	Frequency of word *Commonwealth*
`IBMprice`	Nasdaq	`<description>`	VM	Value of IBM's stock

Table 5.2: Column-level mining rules of mock dataset: cf. *case study one*.

5.4.3 Filters

Function

Each column-level set of mining rules for a dataset makes use of case-sensitive filters. These filters are based upon the structural metadata (Appendix B.2) in RSS's format to identify elements of feeds to determine textual (OM) or numeric (VM) data for mining during polling (Section 5.5). XML for the filters is produced by myDS during the definition of mining rules and persisted to database storage as part of the resulting dataset created (Section 5.6). The principal XML elements used to define the filters and their related UML (Appendix B.2) classes, are listed in Table 5.3. This table should be read in conjunction with Figure 5.3 which also lists the attributes of the UML classes: methods are not displayed for clarity.

Element/attribute	Description	UML class
`<dataminer>`	A *container* encapsulating the `<columnminer>` elements in a dataset.	`DataMiner`
`<columnminer>`	Specifies the name and type of a set of column-level mining rules.	`ColumnMiner`
`<rssminer>`	For a `<columnminer>`, this provides common attributes for the `<rssoccurrenceminer>` (OM) or `<rssvalueminer>` (VM). `sortfield` and `sortascending` are attributes of `<rssminer>` to respectively sort and restrict RSS data during the definition of mining rules.	`RSSMiner`
`<rssoccurrenceminer>`	Stores the keywords and the RSS feed elements to be mined.	`RSSOccurrence-Miner`
`<rssvalueminer>`	Stores the RSS feed element to be filtered, and the JavaScript `<regexp>`, i.e. regular expression, which locates the *position* of the value to mine from that element.	`RSSValueMiner`
`<rssurl>`	The RSS feeds to be mined for a `<columnminer>`.	
`<polling_interval>`	The `<dataminer>`'s polling frequency.	

Table 5.3: Principal elements and attributes of XML filters for mining rules, and corresponding UML classes: cf. *case study one*.

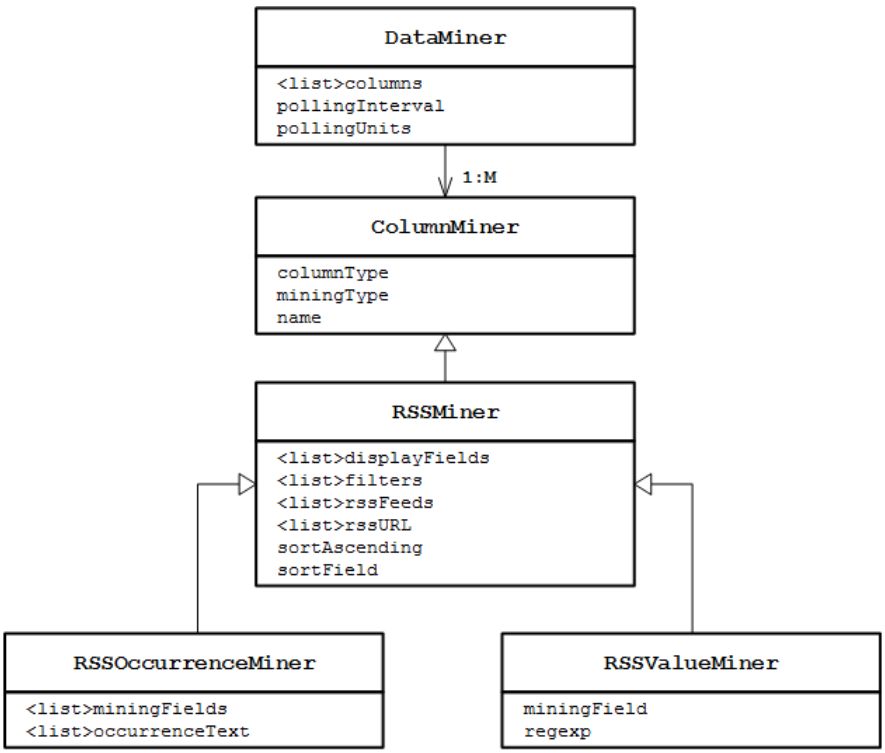


Figure 5.3: UML class diagram displaying cardinality and specialisation of principal occurrence and value mining classes (methods have been edited for clarity): cf. *case study one*.

Occurrence mining

Figure 5.4 displays a partial screen dump of the myDS page where column-level mining rules of OM's *Count occurrences* variant are being defined upon data published by the BBC in their web site's principal RSS news feed at `http://feeds.bbci.co.uk/news/uk/rss.xml`. There are two filters: (1) *Count Occurrences in Description Containing Commonwealth* which has already been defined, whereas (2) the second filter, involving the feed's `<title>` element, is shown in a partially defined state.

Stated simply, when the resulting column-level mining rules of filter *Count Occurrences in Description Containing Commonwealth* are persisted to myDS's database as part of a dataset, the XML created by myDS for this filter will be subsequently used during polling to identify and mine the count of the occurrences of keyword *Commonwealth* from the `<description>` element of the BBC's news feed.

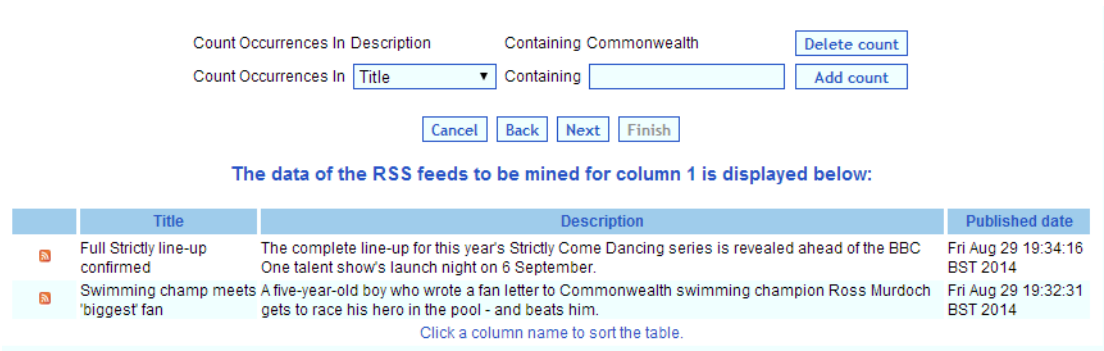


Figure 5.4: Partial screen dump of stage (5) in the DFD in Figure 5.1, displaying the definition of occurrence mining rule filters in myDataSharer: cf. *case study one*.

Algorithm 5.1 lists edited XML for the *Count Occurrences in Description Containing Commonwealth* filters where the `<dataminer>` element (1) serves as the encapsulating dataset-level *container* for the mining rules. The `<rssoccurrenceminer>` element (4) defines the use of OM to mine the frequency of *Commonwealth* from (9) `<rssoccurrencecount field="DESCRIPTION">Commonwealth</rssoccurrencecount>`, i.e. the `<description>` element of the BBC's news feed listed in the `<rssurl>` element (7): if it became necessary to mine data from additional RSS feeds for *Commonwealth*, extra `<rssurl>` elements would be needed here. Element `<columnminer>` (6) specifies the name and type of the column-level mining rules: these details will be used to create the column of the resulting database table which will be populated during polling according to the `<polling_interval>` element (3). (5) lists the attributes of the `<rssminer>` used to sort and restrict data from the RSS feed in the `<rssurl>` element (7) during the definition of mining rules. Lastly, if other sets of column-level mining rules were required in this example, each would have its own respective `<rssoccurrenceminer>` element filter.

```
 1: <?xml version="1.0" ?>
 2: <dataminer>
 3:     <polling_interval units="HOURS">1</polling_interval>
 4:     <rssoccurrenceminer>
 5:         <rssminer sortfield="PUBLISHEDDATE" sortascending="false"
                      pollfilterenabled="true">
 6:             <columnminer name="Commonwealth" type="INTEGER"></columnminer>
 7:             <rssurl>http://feeds.bbci.co.uk/news/uk/rss.xml</rssurl>
 8:         </rssminer>
 9:         <rssoccurrencecount field="DESCRIPTION">Commonwealth
            </rssoccurrencecount>
10:     </rssoccurrenceminer>
11: </dataminer>
```

Algorithm 5.1: Column-level occurrence mining rule filters (edited for clarity): cf. *case study one*.

Value mining

The edited XML listed in Algorithm 5.2 represents filters to record fluctuations in the price of *IBM* stock at `http://www.nasdaq.com/aspxcontent/NasdaqRSS.aspx?data=quotes&symbol=IBM`. The roles performed by elements `<dataminer>`, `<rssminer>`, `<columnminer>` and `<rssurl>` match their OM counterparts. The major difference between OM and VM is illustrated by the `<rssvalueminer>` element (4) which includes the `<regexp>` element (9) to locate the *position* in the `<description>` element of the value to mine during polling (Section 5.5).

```
 1: <?xml version="1.0" ?>
 2: <dataminer>
 3:     <polling_interval units="HOURS">1</polling_interval>
 4:     <rssvalueminer field="DESCRIPTION">
 5:         <rssminer sortfield="PUBLISHEDDATE" sortascending="false"
                    pollfilterenabled="true">
 6:             <columnminer name="IBMprice" type="FLOATING"></columnminer>
 7:             <rssurl>http://www.nasdaq.com/aspxcontent/NasdaqRSS.aspx?
                      data=quotes&amp;symbol=IBM</rssurl>
 8:         </rssminer>
 9:         <regexp>IBM Last ([-+]?[0-9, ]*\.[0-9][0-9 ]*) Change .* % Change
                  .*% Volume .* As of: .* EDT View: Stock Quote | News
            </regexp>
10:     </rssvalueminer>
11: </dataminer>
```

Algorithm 5.2: Edited column-level XML filters for value mining: cf. *case study one*.

Combining mining types and multiple RSS feeds

The examples of OM and VM filters listed above all implement columns in datasets relating to a single RSS feed. Nevertheless, mining rules for OM do permit multiple feeds to be mined for a given column, and more complex combinations of filters can be specified according to the relationship of mining rules to columns of datasets (Section 5.4.2): Section 5.6 describes an OM example of this. The XML filters for a column-level set of mining rules are also mutually exclusive, i.e. a column can have filters for OM or VM but not both, although a dataset can include columns for both, e.g. the mock dataset in Figure 5.2 and Table 5.2. Section 6.3.4 discusses patterns of mining types in case study one.

RSS feed data

Rome (Section 4.7.5) is used to parse the text of new RSS feed postings when mining rules are defined upon current RSS feed content, allowing it to be sorted and restricted.

5.5 Polling and mining

This section and Section 5.6 consider the implementation in myDS of the common components and terminology described in Section 4.7.1. In myDS, the polling process is run by Quartz Scheduler (Section 4.7.3) every hour. The polling frequency, i.e. 1 - 24 hours, for a given dataset determines when it will be included in the polling process: this single polling frequency applies to all the columns of a dataset (Section 5.4.2). When polling includes a dataset, the column-level XML filters created by myDS during the definition of mining rules (Section 5.4.3) are employed. Each RSS feed in the `<rssoccurrenceminer>` or `<rssvalueminer>` elements *contained* in the dataset's `<dataminer>` (Section 5.4.3), is polled consecutively for new postings, i.e. new RSS feed `<item>` elements. If new postings are found in the feed, Rome (Section 4.7.5) parses the relevant elements of each new `<item>` and, depending upon the mining rules of the columns in the dataset, new textual and numeric data is mined and persisted (Section 5.6) to database storage together with the polling date/time. Thus, mined and temporal data are stored together allowing both to be easily visualised. The original text of the RSS feeds is not retained.

During polling, mining rules of several datasets can refer to the same RSS feed. To prevent numerous requests to the feed's originating web site, myDS incorporates a cache to ensure that each RSS feed is only read once during polling. The cache is also cleared prior to a polling cycle. As each dataset (Appendix B.2) is polled consecutively, allowing for the cache, it is completely independent of every other: thus, the system is scalable.

5.6 Database persistence

The tables for datasets which persist (Appendix B.2) data mined from RSS feeds during polling in myDS are conventional SQL tables implemented in the application's database. Owing to variations in the columns of a dataset caused by the relationship of mining rules and columns (Section 5.4.2), myDS does not employ a single generic table to store mined data. For each dataset, a dedicated table is created dynamically in myDS's MySQL[2] database when the definition of mining rules is complete, but before the table can be populated from polling. In the examples of XML filters for OM and VM given earlier in this chapter, reference is made to the use of `<columnminer>` elements in mining rules. When the mining rules making up a dataset are persisted to database storage in the table created for the dataset, the name and type of a `<columnminer>` are used to specify the name and SQL data type for the corresponding column in the table.

[2]MySQL and its compliance with ANSI/ISO standards [203] are discussed in Appendix B.1.

Dataset 264 from case study one's assignment (Chapter 6) demonstrates the use of `<columnminer>` elements in naming the columns of dataset tables. In this example, the *Count occurrences* variant of OM was used to mine the `<description>` elements of three IT-industry related RSS feeds hourly for occurrences of company names *Amazon*, *Google* and *Microsoft*, and populate three columns of a database table, where each column counted the occurrences of one of the company names from the three feeds. Algorithm 5.3 lists edited XML filters of dataset 264's mining rules where three `<rssurl>` elements define the RSS feeds (7 - 9) for `<columnminer>` *MicrosoftCount* (6), to be mined for keyword *Microsoft* (10) to update the corresponding column of the table represented by the partial SQL in Algorithm 5.4, wherein (3) lists the attributes of column *MicrosoftCount*.[3]

```
 1: <?xml version="1.0" ?>
 2: <dataminer>
 3:     <polling_interval units="HOURS">24</polling_interval>
 4:     <rssocurrenceminer>
 5:         <rssminer sortfield="PUBLISHEDDATE" sortascending="false"
                    pollfilterenabled="true">
 6:             <columnminer name="MicrosoftCount" type="INTEGER"></columnminer>
 7:             <rssurl>http://www.theregister.co.uk/headlines.atom
                </rssurl>
 8:             <rssurl>http://www.engadget.com/rss.xml
                </rssurl>
 9:             <rssurl>http://feeds.feedburner.com/TechCrunch
                </rssurl>
10:             <rssoccurrencecount field="DESCRIPTION">Microsoft
                </rssoccurrencecount>
11:         </rssminer>
12:     </rssocurrenceminer>

        . . .

13: </dataminer>
```

Algorithm 5.3: Edited XML filters for myDataSharer's dataset 264: cf. *case study one*.

```
1: CREATE TABLE ds_264_96025423112009 (
2:     id INT(11) NOT NULL AUTO_INCREMENT,
3:     MicrosoftCount DOUBLE DEFAULT NULL,

       . . .

4:     PollDateTime TIMESTAMP NOT NULL,
5:     PRIMARY KEY (id)
6: )
```

Algorithm 5.4: Partial SQL of database table for dataset 264 in myDataSharer: cf. *case study one*.

[3]Column *MicrosoftCount* is a DOUBLE because INTEGER columns in the XML filters of mining rules are converted to SQL doubles due to the existence of decimal points in values mined during polling.

5.7 The diary

myDS's includes a *diary* which was originally intended to record logical, physical and temporal data of every action carried out by a user. Each page in myDS contains at least one diary event. These events, comprising simple alphanumeric codes, allow the diary to record the details of any operation carried out on application content, i.e. user, community, dataset or query, analysis or visualisation object, and the type of that operation.

Given that the assignment for case study one focused on mining rules, the diary's original purpose was not served, but it did provide data for the case study's research questions (Section 6.2.2).

5.8 Visualising data

Given myDS's original premise as a *data-sharing* application (Section 5.3), available visualisation types are divided between those for *textual* data and those for *numeric* data, the choice of which is determined by the source of a dataset's data or mining type.

For textual data, visualisation types include a word-cloud and a tree-map (Appendix B.2): the latter was provided by Js-Treemap (Appendix B.1). The majority of the pie and x - y charts for displaying numeric data mined from RSS were implemented in Google Charts. This product is described in Section 4.7.7 because of its use in both case studies for our first paradigm. myDS also employs a decision tree visualisation provided by Weka (Section 10.5).

5.9 Afterword

In this chapter we have described the myDS application which served as the platform for our first case study. Furthermore, this chapter has also profiled the two mining types which were implemented in myDS to define mining rules for our first paradigm. The specifics of myDS, with reference to the common components and terminology shared by the two applications written for our first paradigm (Section 4.7.1), are also described.

Chapter 6

Case study one: Mining and visualising textual and numeric data from RSS

6.1 Foreword

In Chapter 5 we presented our myDS software and its implementation of mining rules. In this chapter, we concentrate upon the use of myDS in case study one for our first RSS-mining paradigm (Section 1.4.1). Section 6.2 defines this case study, which formed an assessed assignment for a corpus of students, and the research questions we sought to answer concerning the feasibility of mining rules. The results of case study one are presented in Section 6.3: in doing this, we look at each research question individually and document examples, patterns and efficiencies in the definition of mining rules and visualisations. We also discuss difficulties encountered by our students during the case study. In Section 6.4 we appraise case study one according to its research questions and student corpus, and the effects of these upon our mining rules.

6.2 The assignment

6.2.1 Synopsis

Case study one sought to test our first paradigm's definition of mining rules upon RSS. In Nov - Dec 2009, a laboratory-based assessed assignment involved thirty-five part- and full- time students with "varying employment and experience backgrounds" (O'Shea and

Levene [288]): this corpus formed the class for a Masters-level module concerning search engines and web navigation. The assignment required each student to use the OM and VM mining types in our myDS software to respectively mine textual and numeric data, by defining mining rules upon small numbers of RSS feeds, and to visualise the data mined during polling.

6.2.2 Research questions

This case study employed the author's myDS application to answer the following research questions (cf. O'Shea and Levene [288]) concerning our first paradigm:

1. **Is it possible to define mining rules upon RSS to determine and visualise trends?** This was the most fundamental question of case study one which concerned whether our students could effectively use mining rules to mine and visualise textual and numeric data from RSS.

   In answering this question, we also document the *user experience* using examples of feedback received from our students concerning the difficulties they encountered.

2. **How efficient was the process to define mining rules upon RSS?** If our first research question addresses the usability of our technologies, a logical corollary of this is the efficiency, in terms of time taken, of the process to define mining rules upon RSS. Despite familiarity with any new process being engendered with frequent use, because of the unproven nature of the task and software at hand, we employed the simple metric of how long it took our students to define individual instances of mining rules during the assignment.

3. **Can the diary be used to model user behaviour?** This question was intended in conjunction with myDS's original aim of modelling user behaviour within a social *data-sharing* environment by using the application's diary (Section 5.7). The diary was subsequently used to provide data for our second research question. Therefore, its original function of modelling user behaviour is not considered in detail. Despite this, diary schedules for mining and visualisation during the assignment are used in this chapter.

6.2.3 The assignment

Figure 6.1 reproduces the assignment given to our student corpus. This assignment consisted of two mandatory components and a third optional one: the second and third

components required the defining of mining rules upon RSS in our myDS software. The stages are described below:

1. **Part one:** Each student was required to find two datasets of their own choice, to upload them to myDS, visualise them and to explain the meaning of the data and visualisation. This part of the assignment was intended to test data-sharing components of myDS (Section 5.3) and was not part of the case study: hence it is only included here for completeness.

2. **Part two:** Each student was allocated four RSS feeds, from a list pre-selected by the author, and asked to define mining rules of their own choice upon each feed using OM or VM (Section 5.4.1). We also asked the students to poll each feed for a period of four days and, at the end of each day, to produce a visualisation displaying the data mined from each feed.

3. **Part three:** This optional part of the assignment required the students to find two additional RSS feeds of their own, and to also define mining rules and poll them for four days. Visualisations of each day's data were also requested.

6.2.4 Setting-up

Selection and allocation of RSS feeds

A master list of sixteen RSS feeds (with a further seven in reserve) was used for the second part of the assignment (Appendix A.1.1). These feeds were selected because they covered popular but diverse themes: we also sought to avoid contentious issues such as race, religion and politics. Our thinking was that this would allow OM, and permit VM to be served by the sport and financial feeds included. We made no use of a formal taxonomy to classify our RSS feeds, and to avoid any potential ambiguity arising from them, we did not employ either of RSS's `<category>` element types (Section 2.2.3). Consequently, the following category names are for convenience of presentation in this chapter only:

- **Sport:** Yahoo football feed `http://sports.yahoo.com/sow.rss`.
- **Lottery results:** Links to the results of lottery feeds in the USA provided by web site `http://www.superpages.com/cities/lottery/index_rss.html`.
- **Financial:** `http://www.sloomedia.com/currency/feeds/USD.xml` supplied exchange rates for the US Dollar against many currencies.

05 Nov 2009

SEWN 2009 - myDS 2 (Assessed Coursework) – Extracting and visualising data in myDataSharer

Earlier in the module, we asked you to become familiar with the on-line community-based, data-sharing application, myDataSharer (myDS), which is being developed by Martin O'Shea for his PhD.

For this assignment, we would like you to use myDS to upload some datasets of your own, poll some RSS feeds, and generate visualisations from data mined from these; myDS will record this data and generate metadata which will later be analysed by Mark and Martin.

Part 1: Uploading and visualising data

You are to find and upload to myDS, two datasets of your own choosing. One of these datasets should be tabular, i.e. rows and columns, and the other should be narrative, i.e. a body of text.

To produce a meaningful visualisation of each dataset and to provide a short explanation of why you think this is so.

Part 2: Mining and visualising data from RSS feeds I

When you have completed part one above, to email Martin O'Shea who will provide you with four RSS feeds for mining. You will be notified of these feeds either through myDS's bulletin board or email.

We would then like you to enter these feeds into myDS and to select occurrence or value mining for each one (see Note 1 below) using parameters of your choice, e.g. which keywords to mine for occurrence mining.

Once you have done this, we would like each feed to be polled (see Note 2) for a period of 4 days and at the end of every 24 hours, we would like you to produce a visualisation (see Note 3) displaying the data derived from each feed (see Note 3).

Part 3: Mining and visualising data from RSS feeds II

[*Part 3 is* **not compulsory** *for this lab, but if you complete it you can score 'bonus' marks which will count towards the final coursework mark.*]

This is similar to Part 2 where we would like you to find two additional RSS feeds **by yourselves**, and set each one for either occurrence or value mining (see Note 1 below) for a polling period of 4 days using parameters of your own. And again, at the end of every 24 hours, we ask that you provide a visualisation of the data derived from each feed (see Note 3 below).

Notes

1. Further details about myDS are available on http://qzone.dcs.bbk.ac.uk:8080/myDataSharer/aboutGeneral and a user guide on mining and visualising RSS feeds in myDS is available from http://qzone.dcs.bbk.ac.uk:8080/myDataSharer/aboutDownloads/

   YOU ARE ADVISED TO READ THIS DOCUMENT.

2. myDS includes a scheduler which is set to automatically poll each RSS feed defined in an RSS tabular dataset every hour.

3. Visualisations and queries defined over RSS tabular datasets are automatically updated with new data from each hourly polling.

4. At the time of writing, myDS's query and visualisation facilities for tabular datasets and RSS tabular datasets lack any date / time filtering options which can be used to filter data (we will let you know if this changes).

   Visualisations covering more than 24 hours are acceptable here. But a work-around for this is that, for each day, you download and edit the daily data and upload it as a new dataset and base your visualisations on this data alone.

Figure 6.1: Case study one's assignment *Extracting and visualising data in myDataSharer*: cf. *case study one.*

- **Media:** The then-current top 100 popular music album releases listed by `http://feeds.musicchartfeeds.com/itunestop100albums`.
- **Information technology:** IT developers questions and answers web site `http://www.stackoverflow.com/feeds`.
- **News:** The BBC web site's principal news feed at `http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/front_page/rss.xml`.

The full allocation of RSS feeds to students is listed in Appendix A.1.2, and was randomised to avoid any pre-determined patterns which could have biased the case study. We also ensured that each student received at least one feed that we thought could be used for VM. Certain students were allocated either partially or wholly identical sets of feeds in order to provide a degree of consistency in the results. Some re-allocation of feeds was also performed during the case study because of difficulties encountered: we discuss these issues in Section 6.3.5.

Guidelines

From the start, it was our intention that the assignment and its submission requirements would allow us to automatically assess any results with the aim of answering our research questions (Section 6.2.2). By adopting this approach, we avoided the need for post-assignment student questionnaires or surveys. Instead we were able to focus directly upon the students' submissions to analyse results and for feedback, explicitly using the submissions, and implicitly via the quality of datasets and visualisations produced. The diary proved instrumental in this analysis by collating data about user activity in a similar way to that of web servers. This allowed us to gauge the case study's results according to the time taken to define mining rules for our second research question.

6.3 Results

6.3.1 Order of presentation

In presenting the results of case study one, we first describe the two categories employed to analyse the datasets and visualisations created by our students, and then we discuss each category's actual results. Both categories are then combined when we consider the time taken to define mining rules upon RSS during the tenure of the case study.

6.3.2 Categorising our results

Due to the relationship of mining rules to columns of datasets (Section 5.4.2) where, given the mining rules defined, a column in a dataset (Appendix B.2) can be made up of data mined from one or more RSS feeds, reference is made deliberately in this section to the numbers of datasets created in case study one, rather than the numbers of RSS feeds or columns defined in the mining rules of the datasets. We also use the term *mining rules* to *substantially* include both the dataset- and column- level rules described in Section 5.4. Additionally, to better report on the successes and failures encountered, we have divided the number of datasets into two major categories:

1. **Reported results:** These consist of the datasets and visualisations described in the submissions to the assignment received from our students, i.e. some 173 datasets (52.59%) of the total created during the case study.

2. **Unreported results:** These are datasets and visualisations defined by our students but which were not included in submissions: some 156 datasets representing 47.41% of the total.

6.3.3 Is it possible to define mining rules to RSS to determine and visualise trends?

The rationale of our first research question

The most fundamental question asked in case study one concerned whether our students could actually use our technologies to define mining rules to effectively mine and visualise textual and numeric data from RSS. To describe the results of this, we first consider mining rules, and follow this by describing subjects and examples. We then discuss the patterns found in the reported results and the nature of the unreported results.

Mining types

The 173 datasets reported by our students included those sourced from RSS feeds allocated to them for part two of the assignment and also those they originated for part three (Section 6.2). In all a total of 135 OM datasets (78.00%) and thirty-eight VM datasets (22.00%) were created: these figures are decomposed by parts of the assignment and mining types in Table 6.1. The difference in the number of datasets defined for each mining type is readily explained by the quantity of textual data which can be mined from typical RSS feeds by OM, whereas numeric data for VM can only be mined from feeds produced by

dedicated web sites which report modern forms of ticker-tape (Appendix B.2) data, such as financial movements, sports or lottery results.[1]

Assignment part	Number of OM datasets	Number of VM datasets
Part two	103	29
Part three	32	9
Total	135	38

Table 6.1: Breakdown of reported datasets created by mining types: cf. *case study one*.

Range of subjects mined

An analysis of the 173 reported datasets created by our students revealed that keyword frequencies in an extensive range of subjects were mined from the allocated and student-originated RSS feeds during the assignment. In presenting the results of this analysis below, as with Section 6.2.4, the categories of the feeds are used for convenience of presentation only, and examples illustrate variations in subjects mined:

- **Sport:** Examples of football included frequencies of postings in RSS feeds mentioning local or national (especially South American) teams, results of games, and counts of the sources of stories on Yahoo's RSS football feed at `http://sports.yahoo.com/sow.rss`, or from various news agencies, e.g. *Reuters* and *Associated Press*. *Formula One* postings concerned particular racing drivers.

- **Lottery:** Where datasets included frequencies of lottery draws or the frequency of a particular number drawn over a particular length of time.

- **Culture:** Including media criticism and film/television: (1) references to the film *Avatar*, (2) frequencies of seasonal references, e.g. *Christmas*, and (3) counting types of music or a particular singer such as *Shakira* or *Britney Spears*. Other instances concerned fashion, celebrities, education and language, e.g. counting the use of various prepositions in an RSS feed.

- **Information technology:** Social networking such as counting postings to Twitter [408] by *Stephen Fry*, whilst other frequencies focused on terms *blog* and *software*, and IT companies, e.g. software houses or internet service providers.

[1] Of the 329 datasets created during the case study, only a single unreported dataset combined OM and VM: unfortunately, only a small amount of data was collected.

- **Finance:** Recording various rates or trying to find links or correlations between foreign currencies, prices of precious metals and shares.
- **Climate:** Weather forecasting and earthquake magnitudes.
- **News:** Frequencies of prominent politicians such as *Barack Obama* or *Gordon Brown*, postings to news web sites, financial or technological stories, and postings about countries and cities. Other instances provided frequencies of how many postings on Google News [153] were sourced from UK newspapers *Daily Telegraph*, *The Times* and *The Guardian*.
- **Other:** Postings to the RSS feed of *Amnesty International*, and reporting statistical data, e.g. the distribution of numeric digits in RSS feeds and numbers of public holidays in different countries.

Table 6.2 lists the breakdown of the reported datasets by category for the second and third parts of the assignment.

RSS feed category	**Number of datasets: part two**	**Number of datasets: part three**
Sport	29	2
Lottery	8	1
Culture	31	3
IT	26	3
Finance	26	3
Climate	0	6
News	11	17
Other	1	6
Total	132	41

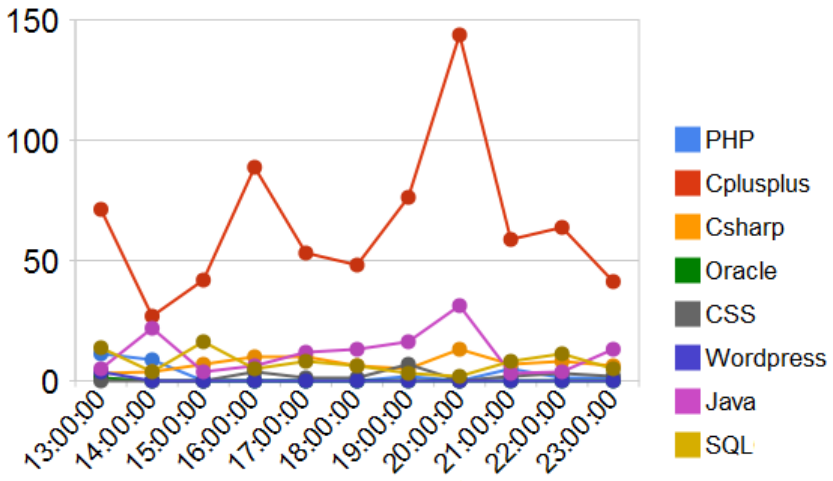
Table 6.2: Breakdown of reported datasets by category and assignment parts two and three: cf. *case study one*.

Occurrence mining and programming languages

Figure 6.2 displays an extract from dataset 577 listing the frequencies of questions posted hourly to RSS feed `http://www.stackoverflow.com/feeds` between 13 00 - 23 00 on 30 Nov 2009, for various programming languages. The originally submitted myDS visualisation is also displayed.

PHP	CPlusPlus	CSharp	Oracle	CSS	WordPress	Java	SQL	PollDateTime
11	71	3	1	0	4	5	14	2009-11-30 13:00:00
9	27	4	0	0	0	22	4	2009-11-30 14:00:00
0	42	7	0	0	0	4	16	2009-11-30 15:00:00
0	89	10	0	4	0	6	5	2009-11-30 16:00:00
0	53	10	0	1	0	12	8	2009-11-30 17:00:00
0	48	6	0	1	0	13	6	2009-11-30 18:00:00
2	76	5	0	7	0	16	3	2009-11-30 19:00:00
0	144	13	0	0	0	31	2	2009-11-30 20:00:00
5	59	7	0	2	0	3	8	2009-11-30 21:00:00
1	64	8	0	3	0	4	11	2009-11-30 22:00:00
1	41	6	0	2	0	13	5	2009-11-30 23:00:00

(a) Extract of data mined during polling.



(b) Time-series plot of data.

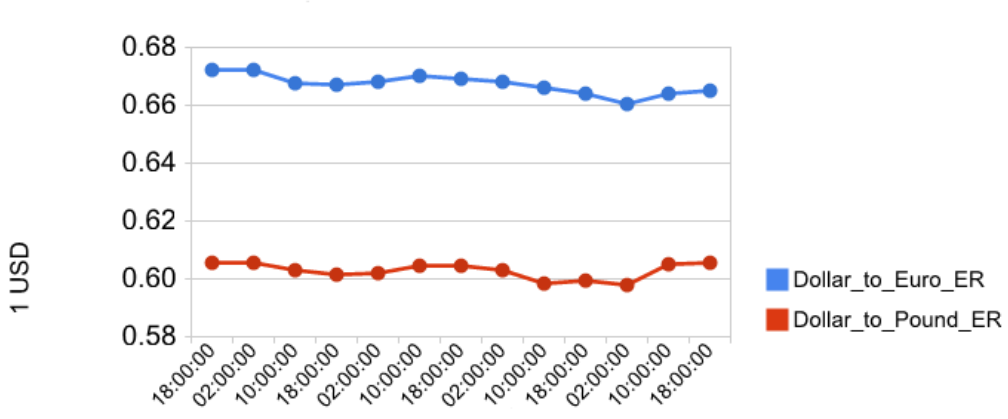
Figure 6.2: Extract of data and corresponding visualisation demonstrating use of occurrence mining in myDataSharer's dataset 577: cf. *case study one*.

Value mining and foreign exchange

Dataset 247 used VM to record *Sterling* and *Euro* foreign exchange fluctuations against the USA *Dollar* every eight hours. Data was mined using VM from the `<title>` element of each `<item>` (Section 2.2.3) in RSS feed `http://www.sloomedia.com/currency/feeds/USD.xml` every eight hours from 18 00 on 22 Nov 2009 to 18 00 on 26 Nov 2009. The data is listed and visualised in Figure 6.3.

Dollar_to_Euro_ER	Dollar_to_Pound_ER	PollDateTime
0.6724	0.6058	2009-11-22 18:00:00
0.6724	0.6058	2009-11-23 02:00:00
0.6676	0.6029	2009-11-23 10:00:00
0.6672	0.6015	2009-11-23 18:00:00
0.6682	0.6018	2009-11-24 02:00:00
0.67	0.6047	2009-11-24 10:00:00
0.6693	0.6046	2009-11-24 18:00:00
0.6683	0.6033	2009-11-25 02:00:00
0.6662	0.5983	2009-11-25 10:00:00
0.664	0.5993	2009-11-25 18:00:00
0.6604	0.5981	2009-11-26 02:00:00
0.6639	0.6052	2009-11-26 10:00:00
0.6651	0.6054	2009-11-26 18:00:00

(a) Extract of exchange rate data mined during polling.



(b) Time-series plot of exchange rate fluctuations.

Figure 6.3: Data extract and visualisation of exchange rate fluctuations from myDataSharer's dataset 247: cf. *case study one*.

6.3.4 Patterns of use

This section is concerned with various trends we noticed in the 173 reported datasets created by our students.

Preferred mining type

OM was preferred to VM by a significant margin, i.e. of the 173 datasets created in the reported results, some thirty-eight (22.00%) concerned VM, whilst 135 (78.00%) were derived from OM (Table 6.1).

Mining multiple RSS feeds for a single dataset

The relationship of mining rules to columns of a dataset (Section 5.4.2) allows multiple RSS feeds to be defined in a set of OM column-level mining rules. In the majority of the reported datasets the students used a single feed to define mining rules, where the feed populated a dataset (Appendix B.2) of only one column. We determined two exceptions to this:

1. Where students repeatedly re-used the same RSS feed: approximately 25.00% of reported datasets created mined the same feed more than once to populate more than one column of a dataset. An example of this was the use of OM's *Count all items* variant to mine Yahoo's football feed (`http://sports.yahoo.com/soccer/rss.xml`) for a dataset of sixteen columns for *top* European football clubs. The XML filters (Section 5.4.3) in the mining rules created by myDS for a given column of this dataset referred to the `<title>` element of the feed and was named, e.g. *BarcelonaOccYahSoc* or *BordeauxOccYahSoc*, in order to identify both team and feed.

2. In approximately 7.00% of reported datasets, counts were combined from more than one RSS feed into a single column of the dataset. The best example of this aggregation occurred in part three of the assignment (Section 6.2). In dataset 264 (Section 5.6) three IT-industry related RSS feeds were mined hourly for occurrences of company names *Amazon*, *Google* and *Microsoft*, i.e. the dataset's database table included three columns, each of which counted the occurrences of one of the company names from the three feeds. The student responsible considered this to provide "broader scope and higher reliability of the mining results" where high counts indicate that "web-sites are frequently updated and the topics are highly relevant to them."

Daily editing of data

A requirement of the assignment given to our students was to visualise the data mined from their RSS feeds every twenty-four hours of each four day polling period. In doing this, some students supplied screen dumps either for each day of the, or for the entire, polling period.

Use of visualisations

Table 6.3 summarises the distribution of the visualisations in the reported datasets. Where a student created multiple instances of the same visualisation type for a dataset, we have counted these as a single visualisation: *Mixed* refers to a collection of several different visualisation types for a single dataset.

Table 6.3 reveals that the high volume of column chart instances was because it was the default visualisation type, and this may also have been the most familiar type of x - y chart to our students. The pie chart proved ineffective in certain cases because of the number of *slices* resulting from mined data, and the lack of relationships in mined data reduced the efficacy of the scatter chart. The decision tree (Section 10.5) was sparsely used because of its more specific role in classification (Section 2.5), and the textual and numeric data mined in case study one did not lend itself to decisions and consequences. Similarly, the tree-map (Appendix B.2) was not used because mined data lacked any implicit hierarchy of subsets: it was also possible that our students were also unfamiliar with this type of visualisation.

Visualisation type	**Number of visualisations**
Column chart	70
Line chart	33
Mixed	25
Area chart	16
Bar chart	10
None	7
Pie chart	7
Decision tree	3
Scatter chart	2
Tree-map	0
Total	173

Table 6.3: Distribution of visualisation types: cf. *case study one*.

6.3.5 Discussion: explaining the unreported results

Several difficulties were encountered by our students during case study one which caused delays in producing the reported results (Section 6.3.2). These issues also led to mining rules being abandoned, and created datasets being unreported. In this section, we discuss these difficulties and several other issues.

RSS data variations

Polling in myDS is explained in Section 5.5. Despite pre-checking of the RSS feeds allocated to the students for the second part of the assignment, various feeds were not actually updated frequently during the polling periods defined by our students. This led to: (1) delays caused by requests for the allocation of a new feed, and (2) a minority of the reported datasets being sparsely populated. Repeated checking of an RSS feed by one student to record temperatures in London, indicated that all of its postings appeared to have been published at the time of polling, hence the student found that no useful data could be mined. Certain RSS feeds were also found to have their content updated during regular hours only, e.g. financial feeds would only update during business hours or sports feeds at weekends, affecting the use of VM. Future-dated postings were routinely excluded by the polling process.

System and user issues

- **Bugs:** Several bugs in our myDS software affected the successful definition of mining rules or mining of data during polling. An example of this was found in one of the RSS feeds initially allocated to eight students, which repeatedly caused a system *null pointer exception* during the definition of VM mining rules: other feeds were allocated in its place. Other bugs were reported by students during the case study: some of these were fixed or information about them was posted to the *About: Known issues* page in myDS.

- **The process of defining mining rules:** A small number of our students bemoaned the fact that once mining rules for a dataset had been defined, persisted to database and made ready for polling, the rules could not be changed at all, or the dataset could not be deleted. A consequence of this was that if a mistake was made when defining mining rules or polling failed to mine data, the students concerned had no choice but to re-enter the original, or define new, mining rules in a new dataset. Such delays caused one student to declare "I am losing patience with this assignment."

Other students stated that they found the process to define mining rules to be either "non-intuitive" or "counter-intuitive", although they did not provide specific details. myDS's interface was also described as "academic" and "sterile" because of a perceived lack of interactivity. In addition, several students reported finding myDS's interface difficult to use and therefore it took them longer to define mining rules for the assignment. There was an extreme case where a student admitted having failed to produce any significant datasets or visualisations because they did not understand the relationship between mining rules and the columns of datasets (Section 5.4.2) We also believe that in a small number of cases, students used VM instead of OM or vice-versa, or had combined data from feeds incorrectly.

The fourth stage of the process to define mining rules is used to display data from the RSS feeds already entered. It also provides various controls to allow users to restrict and sort the data, e.g. to show only RSS feed `<item>` elements where the *Description* `Contains text:` *Afghan*, before branching occurs due to the mining type (Section 5.4.2). The restrictions for stage four implicitly used boolean connective AND but did not allow OR and NOT to be specified: neither did they allow compound entries for multiple keywords, e.g. *Description* `Includes text:` *Obama, Afghan, house*. A further issue when defining mining rules was the use of a dedicated page for each stage of the process. myDS includes nine stages/pages per set of column-level mining rules (Figure 5.1), and then a further one to convert these rules into a dataset to be populated during polling. This is a lengthy process requiring a significant attention to detail on the user's part when a smaller number of pages could have proved more satisfactory, even if each page consisted of more than one stage, or where a summary page could have listed the full set of mining rules being defined. In connection with this point, at least one student reported finding it difficult to remember what they had entered when defining mining rules.

Value mining

VM's requirement for structured numeric data produced by dedicated web sites is described in Section 5.4.1. Given that only 22.00% of the reported datasets were value mined, the following examples describe some of the difficulties our students encountered with VM when setting the *value to mine*, i.e. the current value of an, e.g. exchange rate or stock price, when defining VM mining rules:

- **Masking other changing values:** A column-level set of mining rules for VM allows a numeric value to be mined from an element of an `<item>` of an RSS feed during polling. If a user wants the exchange rate between two currencies to be mined from a feed, and the `<description>` element of the feed contains the rate's current value, then this value must be entered by the user as the value to mine when mining rules are defined. This enables the *position* of the current value in the text of the `<description>` element to be recorded in the XML filters of the mining rules (Section 5.4.3). During polling this position is used to locate the rate's updated value in the `<description>` element and mine it. If the text of the element to be mined contains other changing values, e.g. dates or times, they must be *masked*, i.e. disabled, otherwise VM will not work correctly. This happened in a number of instances where students lacked knowledge of the changing values in the elements of the RSS feeds they used.

- **Incorrectly setting the value to mine:** A second issue with VM occurred when students defined the value to mine as static text, or even selected the wrong RSS feed element to mine data from. An example of this was dataset 319 where feed `http://themoneyconverter.com/PAB/rss.xml` was filtered to record fluctuations between Sterling and the Panamanian *Balboa*. As the partial screen dump in Figure 6.4 illustrates, the value to mine consisted of static text GBP from the `<title>` element of the feed rather than the `<description>` (Section 2.2.3) which contained the exchange rate applicable at the time of masking. Hence, only text *GBP* was mined during polling.

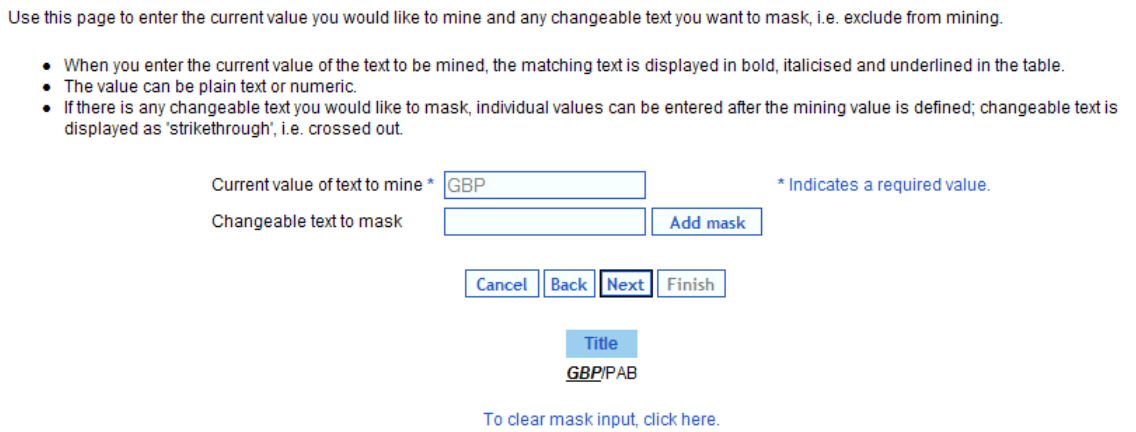


Figure 6.4: A partial screen dump of an incorrect definition of a value mining rule: cf. *case study one*.

6.3.6 How efficient was the process to define mining rules upon RSS?

Explaining our second research question

This question is concerned with the efficiency, in terms of time taken, of the process to define mining rules upon RSS. Despite familiarity with any new process being engendered with frequent use, because of the unproven nature of the task and software at hand, we employed the simple metric of how long it took our students to define individual instances of mining rules.

Timing the definition and application of mining rules

To determine if the time taken by our students to define mining rules reduced at all, we made use of data compiled from all 329 datasets created by our students, i.e. 173 reported in submissions and 156 that were not: of this number, 234 were for OM and ninety-five for VM. Timings were calculated for the maximum number of seconds recorded between two diary events (Section 5.7), which for convenience we refer to here as *D1*, i.e. the time a user visited the first of the pages required to define mining rules, and *D2*, i.e. the time when a dataset was created and polling could begin according to the mining rules defined. The diary extract in Table 6.4 demonstrates this where it took 93 52 (mm ss) between events 57045 (*D1*) and 58094 (*D2*) to create a set of OM mining rules for dataset 466: this is despite several intermediate *D1* events which may represent repeated visits to the first page before the definition of mining rules was started.

Event number	Event code	Date/time
57045	D1	27 Nov 2009 @ 21 57 37
57163	D1	27 Nov 2009 @ 22 05 49
57813	D1	27 Nov 2009 @ 23 09 29
57959	D1	27 Nov 2009 @ 23 20 27
58036	D1	27 Nov 2009 @ 23 27 14
58094	D2	27 Nov 2009 @ 23 31 29

Table 6.4: myDataSharer diary extract showing the time taken to define occurrence mining rules for dataset 466: cf. *case study one.*

Table 6.5 displays an example of the source data for a typical student, where start times for defining mining rules for four OM datasets and four VM datasets, and the duration of these definitions, are given. To better illustrate the times taken to define mining rules, the

D2 columns in Table 6.5 do not refer to the actual times recorded for the *D2* events, but to the number of minutes and seconds following a *D1* event before they, the *D2* events, occurred. The last row in the table lists the mean times for the datasets created.

Occurrence mining		**Value mining**	
D1 date/time	**D2 (mm ss)**	**D1 date/time**	**D2 (mm ss)**
19 Nov 2009 @ 23 41 49	17 25	19 Nov 2009 @ 22 54 42	24 44
20 Nov 2009 @ 00 52 58	66 09	19 Nov 2009 @ 23 21 16	19 39
22 Nov 2009 @ 00 43 25	66 28	22 Nov 2009 @ 01 50 31	06 51
25 Nov 2009 @ 01 52 43	16 36	24 Nov 2009 @ 23 10 47	33 09
Mean	41 39	**Mean**	21 05

Table 6.5: A typical student's timings to define mining rules: cf. *case study one*.

This process was then extended across all datasets created to calculate the overall times taken to define mining rules for OM and VM during the case study. Figure 6.5 displays the frequencies of the times taken, and reveals a significant positive skew for each mining type, where mining rules for a minority of datasets took much longer to define than the majority. We believe this was due to the reasons discussed in Section 6.3.5. Table 6.6 summarises the mean and standard deviation of each mining type.

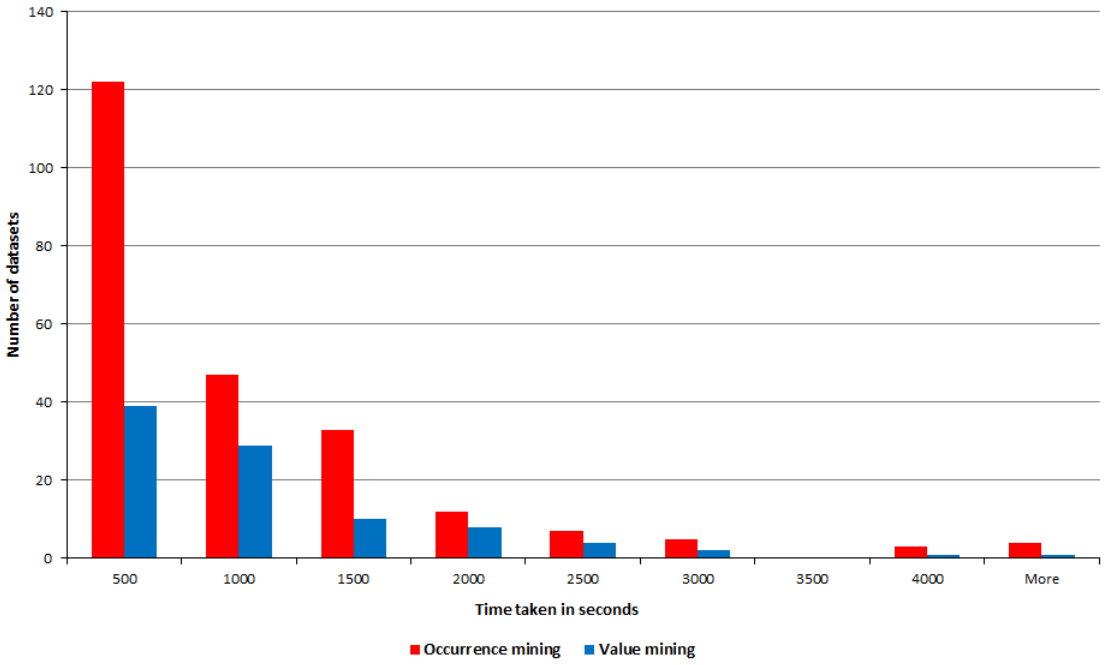


Figure 6.5: Distribution of mining rule timings by mining type: cf. *case study one*.

Mining type	Number of datasets	Mean (mm ss)	SD (mm ss)	Skew
OM	234	13 30	15 41	2.99
VM	95	14 51	14 41	3.01

Table 6.6: Statistics per mining type based upon all datasets created: cf. *case study one*.

In Section 6.3.5, we explained why our students could have found VM more complicated than OM. Accordingly, we expected that our statistics would reveal a lower mean and SD for OM compared with VM because of the shorter times taken to define mining rules. This did not actually occur, and a *2*-tailed z-test (Appendix B.2) confirmed no statistical significance between the means of the two populations.

A second z-test was applied to an edited number of datasets, which saw approximately 8.00% of the datasets removed from each population: this 8.00% consisted of twenty OM and eight VM datasets, where mining rule definition times were above 2,000 seconds, which we thought to be causing the previously observed positive skew. The second test's results showed a difference in significance with an *alpha* level of 0.05 (95.00%). The results listed in Table 6.7 show that the mean and SD for OM were lower than their VM counterparts, and that the skew had been significantly reduced.

Mining type	Number of datasets	Mean (mm ss)	SD (mm ss)	Skew
OM	214	09 44	07 40	1.03
VM	87	11 31	07 52	1.03

Table 6.7: Statistics per mining type based upon an edited number of datasets created: cf. *case study one*.

We were also interested to see if there was a reduction in the time taken by our students to define mining rules. Therefore, for each student, we took the time to define rules for the first dataset they created for a particular mining type, and compared it against the time taken for the last dataset defined from rules using the same type. Table 6.8 summarises the changes of timings. We also recorded reductions:

- **Occurrence mining:** In the majority of cases, an actual reduction was found for twenty-six (74.00%) of our thirty-five students as opposed to eight (23.00%) who saw an increase, or in one case where data was incomplete.

- **Value mining:** We saw a reduction for ten (29.00%) of our students: seven students (20.00%) saw an increase and data for the remaining eighteen was incomplete.

The average number of datasets created per student was 6.69 for OM and 2.71 for VM, which reflected the preference of OM in the datasets reported as per Table 6.1.

	Occurrence mining			Value mining		
Change	**No of students**	**Mean**	**SD**	**No of students**	**Mean**	**SD**
Decrease	26	14 09	12 27	10	17 46	21 53
Increase	8	11 06	19 12	7	08 53	06 04

Table 6.8: Changes in student timings when defining mining rules (mean and SD values are given in (mm ss) format: cf. *case study one*.

6.3.7 Timing visualisations

Section 5.8 describes the visualisation types in myDS to display data from a dataset. Defining a visualisation involves the selection of the type, e.g. bar chart or scatter chart, and options, e.g. which dataset columns to include, entry of the visualisation's name and final confirmation. Though our students were able to select from eight visualisation types, this section treats these types as one to determine the overall average time taken to define visualisations, and any reductions found.

Some 302 visualisations were created from the reported and unreported datasets from mined RSS data. In timing these, we used three diary events for the maximum number of seconds recorded. For convenience, we refer to these events as *V1* when a user visited the first of the pages to define a new visualisation, *V2* when a user visited the page of an existing visualisation and modified it to create a new visualisation, and *V3* when a visualisation was saved. The diary extract in Table 6.9 demonstrates this principle for a new visualisation for dataset 235.

Event number	**Event code**	**Date/time**
38072	V1	24 Nov 2009 @ 11 22 17
38074	V1	24 Nov 2009 @ 11 23 09
38122	V1	24 Nov 2009 @ 11 57 15
38124	V1	24 Nov 2009 @ 11 58 13
38130	V3	24 Nov 2009 @ 11 59 32

Table 6.9: myDataSharer diary extract showing the time taken to define a visualisation for dataset 235: cf. *case study one*.

In Table 6.9 event 38072 recorded the first *V1* for a new visualisation: later we see *V3* event 38130 showing the visualisation being created. The other *V1* events imply that the student revisited the first page between the first and last events, and could well have visited other pages. We recorded the timings for visualisations for the maximum number of seconds between the first *V1* (or *V2*) event and a *V3* event, where the dataset was common to both regardless of other page visits between these events.

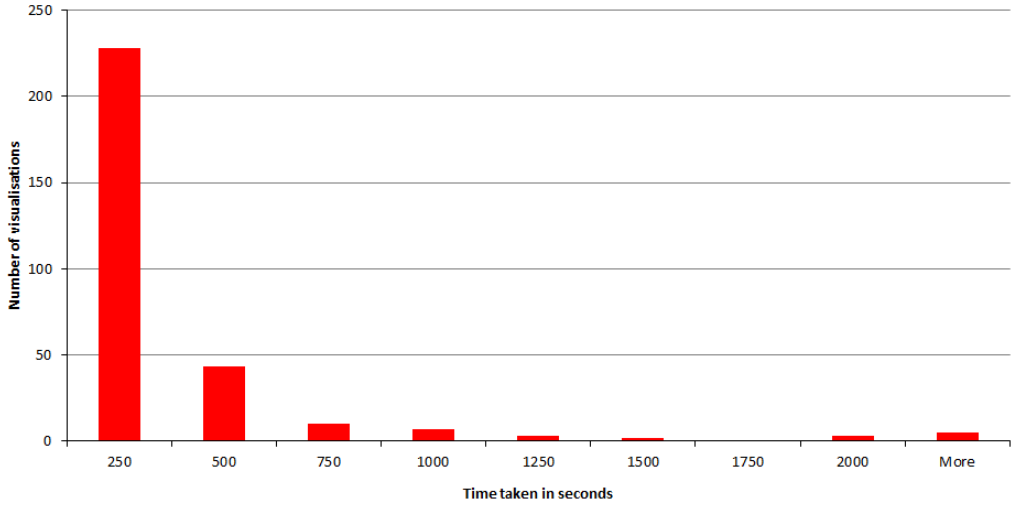


Figure 6.6: Histogram of the distribution of visualisation timings: cf. *case study one*.

For defining visualisations, the mean time was 04 28 (mm ss) with an SD of 08 34 (mm ss). Again, the variation was due to the positive skew evident in Figure 6.6, which displays a histogram of the frequency of visualisation definition times. Overall, there were average of 8.63 visualisations per student.

Comparisons between timings for the first and last visualisations for each student, summarised in Table 6.10, indicate that for twenty-three (66.00%) students, a reduction was found compared to eight (23.00%) who saw an increase. Timings for four students could not be determined because of a lack of data. Table 6.11 lists the types of the visualisations created.

Change	**Number of students**	**Mean (mm ss)**	**SD (mm ss)**
Decrease	23	02 59	03 24
Increase	8	11 13	24 37

Table 6.10: Changes in student timings when defining visualisations: cf. *case study one*.

Visualisation type	Number created
Column chart	149
Line chart	64
Area chart	38
Bar chart	18
Decision tree	13
Pie chart	12
Scatter chart	7
Tree-map	1
Total	302

Table 6.11: Breakdown of reported and unreported visualisations created by type: cf. *case study one*.

6.3.8 Can the diary be used to model user behaviour?

myDS's diary (Section 5.7) was originally intended to model user behaviour within a social *data-sharing* environment (Section 5.3). Given that the assignment for case study one focused on mining rules, the diary's original purpose is not considered in detail in this chapter, although it did provide data for our second research question. The diary was also used to produce data for the time-series (Section 2.8.3) plot in Figure 6.7 which displays the daily creation of datasets and visualisations during the case study.

We can see that initially, little work was done until 19 Nov 2009 and that Thursday falls within the busiest periods, i.e. 26 Nov 2009 for datasets and 03 Dec 2009 for visualisations: as this was when student *lab* sessions were held, it was not unexpected. We also see that Sunday 29 Nov 2009 was a busy mining day. From 03 Dec 2009, there was a drastic fall before 04 Dec 2009's original assignment deadline, and a final peak on 05 Dec 2009 due to an extended deadline. Table 6.12 summarises the busiest mining period of the assignment, i.e. where 26 - 29 Nov 2009 saw some 122 datasets created. This was followed by the busiest period for visualisations with thirty-four created on 27 Nov 2009 and forty-five on 03 Dec 2009.

Date	Number of OM datasets	Number of VM datasets
26 Nov 2009	33 (10.03%)	13 (3.95%)
27 Nov 2009	25 (7.59%)	9 (2.73%)
29 Nov 2009	28 (8.51%)	14 (4.25%)

Table 6.12: The busiest days for mining rules during the assignment: cf. *case study one*.

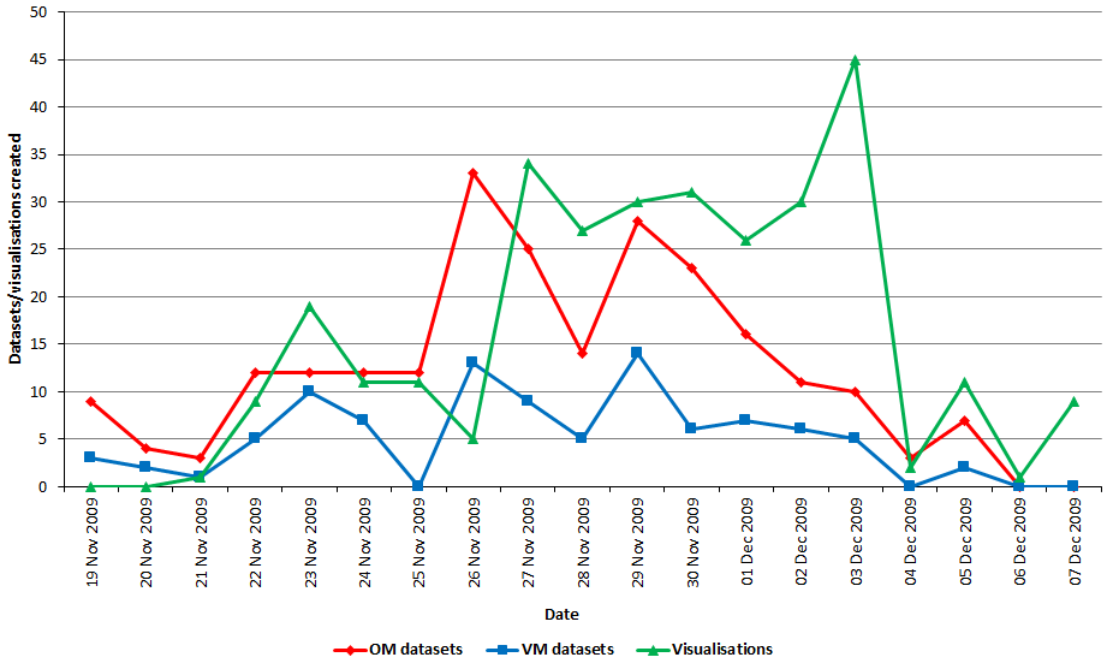


Figure 6.7: Time-series plot of the daily creation of datasets and visualisations: cf. *case study one*.

6.4 A posteriori appraisal of case study one

6.4.1 Comparing reported results and research questions

We have documented the results of case study one by dividing the datasets created by our students by those reported in submissions and those not reported. A total of 173 datasets were reported together with a comparable number of unreported datasets. In comparing the reported results back to our research questions, we determined that our mining types were able to mine textual and numeric data from RSS. We also saw significant reductions in time taken by a majority of our students to define mining rules. OM was proved to be the preferred mining type, i.e. 78.00% of datasets created were mined from OM compared to some 22.00% for VM. This is because the majority of feeds on the internet consist of textual rather than numeric data. We compare our first paradigm's use of mining rules with appropriate related work in Chapter 9.

6.4.2 Refining mining rules

Section 6.3.5 discusses the delays caused by, and difficulties encountered by some of our students with, the iterative and long-winded process to define mining rules, especially for those rules using VM. This necessitated a review of mining rules for our second case study. As a result of this, we decided to focus exclusively on mining only textual data from RSS for visualisation, by refining our textual mining rules to use simple, direct mining types.

myDS was originally written as a *data-sharing* application with an RSS-mining component: consequently the application included many now-redundant elements and a *cluttered* interface. Therefore, in tandem with the refining of our textual mining rules, we decided to implement our second case study in a new *lightweight* application called visualRSS (vRSS) which we present in Chapters 7 and 8.

6.4.3 The question of a pilot study

No pilot study of our myDS software or its implementation of mining rules was carried out prior to case study one. This was due to deadlines and the absence of an available corpus of testers during the application's development. Within the actual module for which the assignment of the case study was part of (Section 6.2.1), other assignments further precluded a pilot study carried out by some, or all of, the student corpus. myDS was thoroughly tested during its development stages prior to case study one and stress tests were carried out with Apache JMeter (Appendix B.1) appropriate to the number of expected students in the module. Nevertheless, we accept that a pilot study could have identified some of the issues we discuss in Section 6.3.5 concerning the unreported results, and that this may have reduced the work to refine the textual mining rules (Section 6.4.2) for case study two.

6.4.4 RSS feeds and student corpus demographics

In addition to the results of case study one, it is necessary to consider issues relating to the student corpus we employed (Section 6.2.1). Given that this corpus formed a class for a module taught by our main supervisor, i.e. Professor M. Levene, it was available to us at the time and we employed it homogeneously without considering demographics. Similarly, the RSS feeds making up the list used for the second part of the assignment (Section 6.2.4) were also selected on this basis, except where we sought to avoid contentious issues. Concerning the selection of RSS feeds by the students for the third part of the assignment, as with part two, no demographically- or ethically- based questions were

raised by the students concerning the feeds used, or the data mined and visualised. With reference to this latter point, at the beginning of case study one the assignment (Section 6.2) made it known to the corpus that "myDS will record this data and metadata which will be later analysed".

Had we undertaken a demographically-based allocation of RSS feeds to students in case study one, we acknowledge that our results may have varied along such lines, although such variations would inevitably have been constrained by the nature and size of the corpus. It is also possible that a demographically-based allocation of feeds could have given rise to the contentious issues we sought to avoid. We also accept that alternative demographically-based student or user, or RSS feeds and category, corpora may well have produced other, different results for this case study.

6.4.5 Loss of data

It is germane to state here that a small quantity of data was lost after case study one. This data, consisting of postings to the BBS supplied by jforum (Appendix B.1) and integrated into myDS, was created as part of the process to define mining rules (Section 5.4). No data concerning the actual mining rules, datasets and visualisations was lost.

6.4.6 Publication

The original version of case study one described in this chapter, together with the profile of myDS in Chapter 5, was published in O'Shea and Levene [288].

6.5 Afterword

In this chapter we have presented the research questions which formed the basis of our first case study. These questions concerned: (1) the actual definition of mining rules upon RSS to mine and visualise textual and numeric for trend analysis (Appendix B.2), (2) the efficiency of this process, and (3) whether myDS's diary could be used to model user behaviour. We have also discussed how our research questions were met according to the results of the case study. We have also appraised the case study and the refining of our textual mining rules for case study two.

Chapter 7

Case study two: The visualRSS application

7.1 Foreword

This chapter is the first of two presenting the second case study for our first RSS-mining paradigm (Section 1.4.1). We begin with Section 7.2 which briefly defines this case study, and we continue by explaining the concept of vRSS in Section 7.3. Section 7.4 describes the use of refined mining rules within vRSS as the application was originally implemented for case study two,[1] and Section 7.5 examines the anatomy of vRSS's mining types. The remaining sections of this chapter concern how vRSS implements the other common components and terminology of the two applications written for our first paradigm (Section 4.7.1). Section 7.6 focuses on polling and data storage, and Section 7.7 describes the calculation of keyword frequencies from polled data in vRSS. Lastly, Section 7.8 details the storage of RSS textual content, and data visualisation is illustrated in Section 7.9.

7.2 Case study two

An *alpha*-version of vRSS was tested in case study two by thirty-six part- and full- time Masters-level students in a second assessed assignment during Dec 2011. This version of vRSS included the textual mining rules refined from case study one (Section 6.4.2), which allowed us to research preferences of the mining types employing the rules, visualisations, distribution of categories of feeds visualised, and the common use of these amongst the mining types. The case study's assignment and results are described in Chapter 8.

[1] vRSS was later extended for case study three (Section 4.8.1).

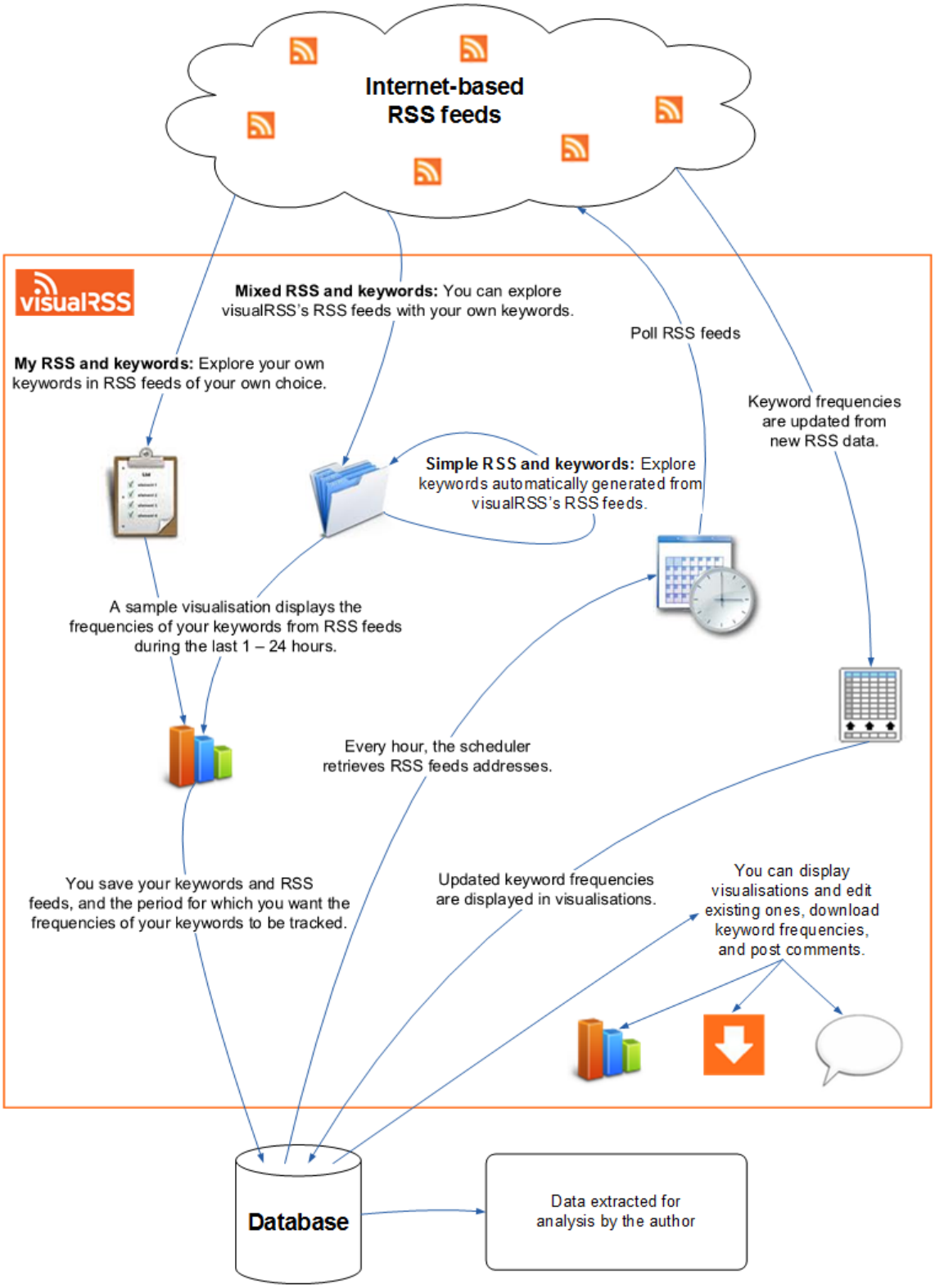


Figure 7.1: The conceptual representation of visualRSS as originally published (cf. O'Shea and Levene [289]) for *case study two*.

7.3 The concept of visualRSS

The *infographic* (Appendix B.2) representing the concept of vRSS in Figure 7.1, was originally published in 2012 (cf. O'Shea and Levene [289]) to illustrate the functionality of the principal components of the application. vRSS allows users to define mining rules to create visualisations, wherein the fluctuations in the frequencies of popular keywords present in the text of RSS feeds are mined during polling and visualised, in order to identify trends in the feeds. The implementation of this concept involved the refined textual mining rules, together with the realisation of the common components and terminology described in Section 4.7.1, according to the objectives of the case study.

7.4 Mining rules

7.4.1 The definition process

For our first case study, mining rules (Section 4.7.2) placed emphasis upon the relationship of the rules to the columns of datasets (Section 5.4.2). In case study two, the refined textual mining rules place this emphasis upon the visualisation of data (Section 7.3), by employing one of the three mutually exclusive mining types (Table 7.1). These simple, direct mining types form a balance between user-selection of individual RSS feeds and keywords on the one hand, i.e. *manual* mining, and the selection of system-generated keywords from system-indexed categories of RSS feeds in *automatic* mining on the other hand, with *semi-automatic* mining forming an intermediary.

Mining type	**Alternatively known as**	**Description**
Automatic	*Simple*	Selection by users of popular keywords in the *rsso-sphere* from system-indexed categories of RSS feeds.
Semi-automatic	*Mixed*	Users enter their own keywords and track these from system-indexed categories of RSS feeds.
Manual	*My*	Manual entry by users of RSS feeds and keywords, to perform a *granular* tracking of subjects.

Table 7.1: visualRSS mining types (italics reveal the name of each mining type seen by users): cf. *case study two.*

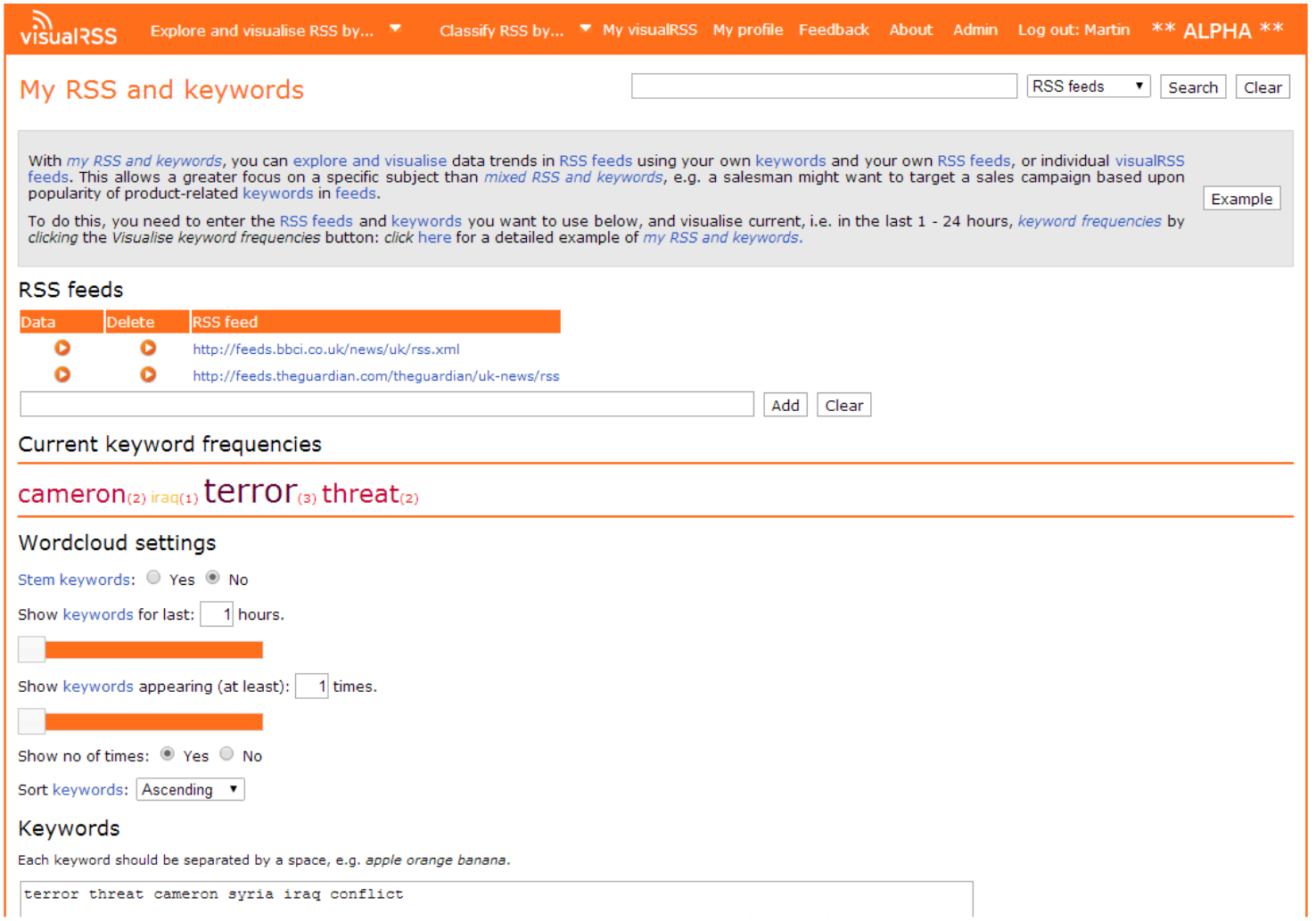


Figure 7.2: Screen-dump of *manual* mining in visualRSS: cf. *case study two*.

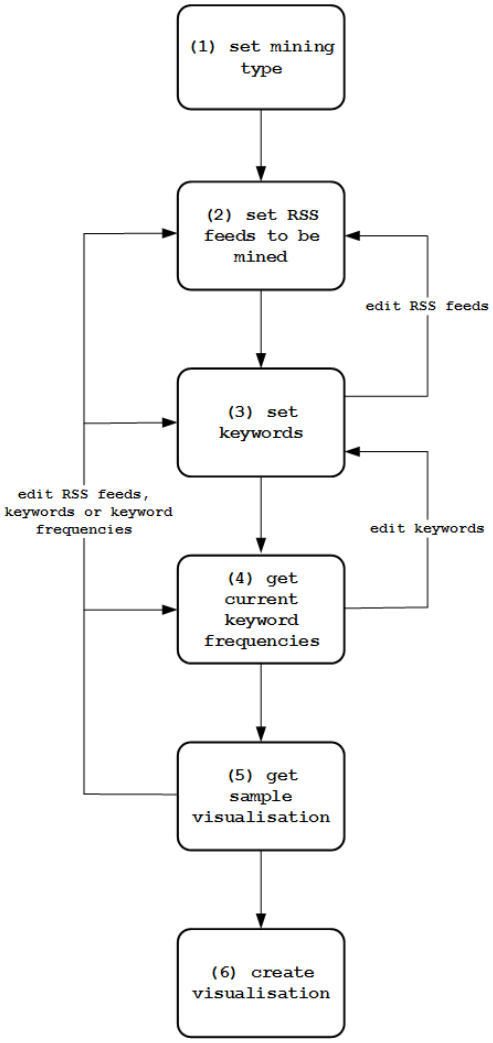


Figure 7.3: DFD of the process flow of *manual* mining in visualRSS (data stores and external entities have been edited for clarity): cf. *case study two*.

In all three mining types, keyword frequencies are updated by mining new postings made to RSS feeds during polling (Section 7.6). Section 7.5 examines the *anatomy* of the mining types, and in conjunction with this, the following paragraphs use Figures 7.2 and 7.3 to describe the manual mining process.

Figure 7.3 uses an edited DFD (Appendix B.2) to illustrate the stages of manual mining. Stage (1) requires the choice of the mining type which calls the appropriate page for the type. (2) and (3) concern the specification of keywords and RSS feeds. When an RSS feed is manually entered, its URL is displayed in the *RSS feeds* table at the top of the page in Figure 7.2. This allows the postings to the feed over the last 1 - 24 hours to be browsed according to the *Show keywords for last: x hours* control of the

word-cloud (Section 7.4.2). Keywords are entered through the HTML `<textarea>` at the bottom of the page. Frequencies of the keywords (4) are set, i.e. calculated, from the postings to the user-entered RSS feeds within the aforementioned hour control. Keywords and frequencies are then displayed in the word-cloud. Stages (2 - 5) are iterative whereby at any time, RSS feeds and keywords entered can be edited, or the word-cloud re-drawn.

Once a user is satisfied with their RSS feeds and keywords, the keyword frequencies from the word-cloud are displayed in a sample visualisation (5) on a new page. This sample visualisation provides the basis of a new *permanent* visualisation should the user decide to save the mining rules. Saving a set of mining rules corresponds to (6) in the DFD in Figure 7.3, where the various rules, i.e. the sample visualisation's type, RSS feeds, keywords and polling dates, are persisted to database storage (Appendix B.2). The frequencies of the keywords are subsequently updated during polling (Section 7.6), and are automatically included in the new visualisation when it is displayed in vRSS: a typical visualisation of data mined is illustrated in Section 7.9.

Other facilities in vRSS include a keyword-based search facility which allows users to browse vRSS's feeds and visualisations, a personalised list of a user's recent visualisations, and a series of *About*, i.e. help, pages.

7.4.2 The role of the word-cloud

During the definition of mining rules in vRSS, a customisable word-cloud (Appendix B.2) is constantly displayed to inform the user of the frequencies of the keywords they have entered or selected according to the use of RSS feeds, or categories thereof, by the mining type being used. The word-cloud format is employed because of its ability to visually represent the keyword frequencies as a bag-of-words (BoW) (Appendix B.2).

For convenience, Figure 7.4 reproduces the word-cloud from Figure 7.2. HTML controls can be seen allowing the number of keywords displayed to be changed, to show or hide keyword frequencies, sorting keywords and stemming (Appendix B.2): the stemming control was not actually used in case study two because it was not implemented in time for the case study. The control *Show keywords for last: x hours* varies the display of keyword frequencies in the word-cloud to the current date/time minus the number of hours between 1 - 24.

For automatic mining, the keywords displayed in the word-cloud are calculated using two elements (Section 2.2.3) of the RSS feeds in the selected system-indexed categories. This is a three-stage process: (1) the text of the `<title>` elements in the RSS feeds in the system-indexed categories are merged to produce a list of keywords, after which (2)

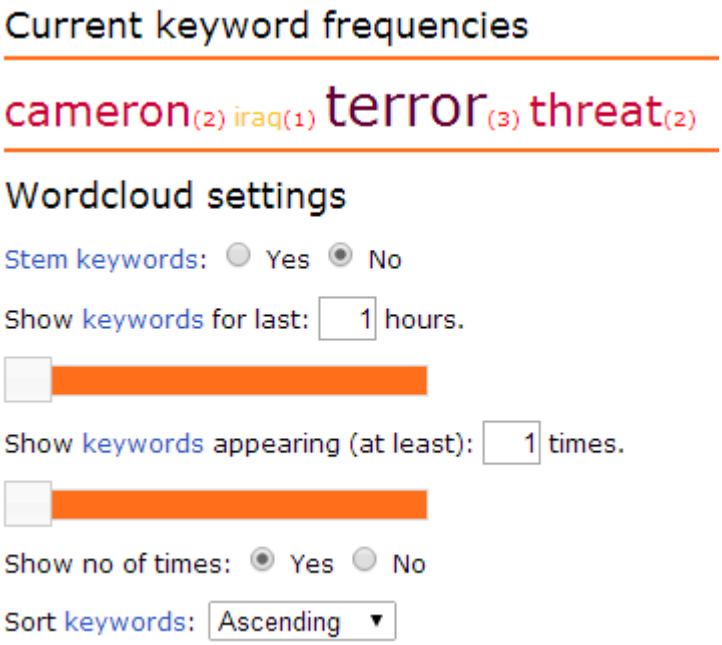


Figure 7.4: Sample word-cloud and HTML controls in visualRSS: cf. *case study two*.

the frequency of each keyword is calculated from the `<description>` elements of the feeds using Lucene (Section 7.7), and (3) a new word-cloud is produced displaying the popular keywords found and their frequencies. Semi-automatic employs an equivalent process except that no use is made of the `<title>` element in the selected system-indexed RSS feeds: keywords are entered by users instead, but keyword frequencies are calculated from the `<description>` elements as before. In the case of manual mining, illustrated in Figure 7.2, where the RSS feeds and keywords are manually entered by the user, the frequencies of the keywords displayed in the word-cloud are calculated only from the `<description>` elements of the feeds.

7.4.3 Keywords

For purposes of uniformity across the mining types, keywords (Section 4.6) in vRSS are simple, English language unigrams without context or meaning, and with a maximum length of fifty lowercase characters. During polling (Section 7.6), a minimal ETL cleanses RSS text, i.e. HTML is purged, non-alphanumeric characters are edited from keywords, stop words (Appendix B.2) are removed, and all numbers are treated as positive.

7.5 The anatomy of a mining type

Although each mining type in vRSS (Section 7.4) has a dedicated page, all the types employ the same basic mechanism to define mining rules upon RSS. This mechanism forms a simple hierarchy of Java classes maintaining the RSS feeds, current keyword frequencies and other mining rules. These classes,[2] preserving something of the naming and format of the mining types used in case study one (Section 5.4.1), are illustrated in Figure 7.5 using an edited UML (Appendix B.2) class diagram. The superclass of this hierarchy, i.e. `RSS_Feed_Miner`, includes dedicated naming elements and an `RSS_Feed_Polling` object. RSS feeds are stored in a series of parallel lists along with the elements, categories and the mining type to be used.

Objects of the `RSS_Feed_Occurrence_Miner` subclass use a key/value *map* to maintain the keyword frequencies which are displayed in the word-cloud (Section 7.4.2) on the page of the mining type being used. For each mining type, during the definition of mining rules, the frequency of a particular keyword in `map<keywordFrequencies>` is aggregated from the contents of the `list<rssFeeds>` attribute of the superclass: for the automatic and semi-automatic mining types, the `list<rssFeedCategories>` attribute is also used to record the RSS feed categories selected. Instances of these classes form the equivalent of the XML filters of the mining rules used in myDS (Section 5.4.3) in case study one.

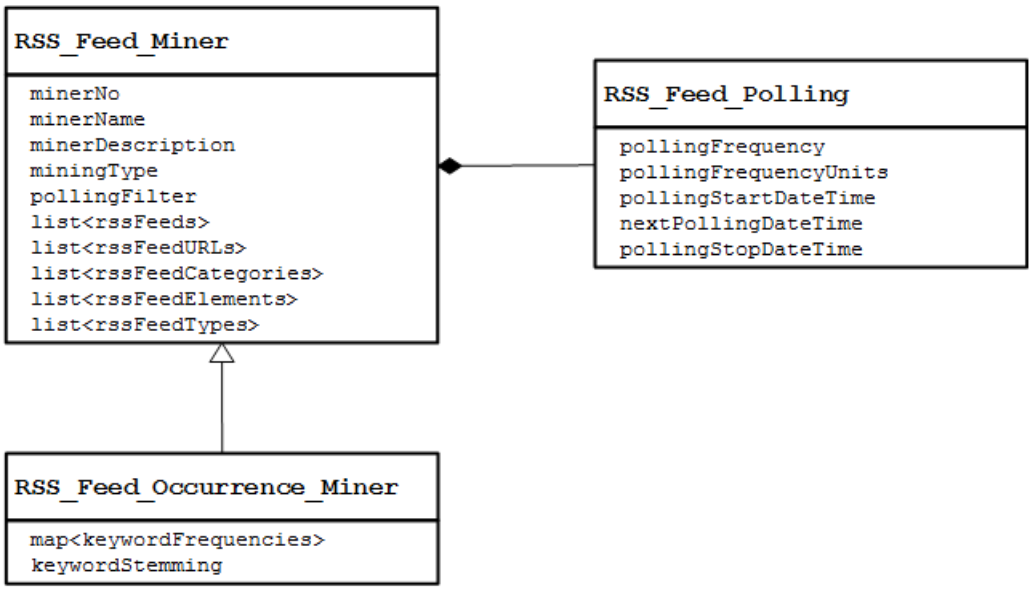


Figure 7.5: UML class diagram of visualRSS's class hierarchy for defining mining rules (methods have been edited for clarity): cf. *case study two*.

[2]This class hierarchy was later extended for our third case study (Section 4.8.3).

7.6 Polling RSS feeds and mining keywords

In Section 7.4.2, reference is made to the use of the `<description>` element of RSS feeds for the calculation of keyword frequencies during the definition of mining rules. vRSS employs Quartz Scheduler (Section 4.7.4) to poll system-indexed RSS feeds every hour. Algorithm 7.1 lists the pseudocode of the polling process wherein the keyword frequencies found in the `<description>` element of each new `<item>`, i.e. posting, in each feed are persisted to the keyword frequency index in the application's database. This index is structured as a series of *M:N* database relationships to store keyword frequencies mined from different `<item>` elements of RSS feeds (Section 2.2.3). Table 7.2 displays a representation of the index, although only frequencies of keywords from the `<description>` elements of RSS were used in our second case study because of time constraints affecting development.

Polling date/time	**Keyword**	**RSS feed**	**RSS element**	**Frequency**
29 Apr 2014 @ 11 00 00	$\texttt{keyword}_1$	$\texttt{rssFeed}_1$	`<title>`	4
29 Apr 2014 @ 11 00 00	$\texttt{keyword}_2$	$\texttt{rssFeed}_1$	`<description>`	2
29 Apr 2014 @ 11 00 00	$\texttt{keyword}_1$	$\texttt{rssFeed}_1$	`<title>`	6
29 Apr 2014 @ 12 00 00	$\texttt{keyword}_2$	$\texttt{rssFeed}_3$	`<description>`	1
29 Apr 2014 @ 12 00 00	$\texttt{keyword}_n$	$\texttt{rssFeed}_n$	`<description>`	3

Table 7.2: A representation of the keyword frequency index in visualRSS: cf. *case study two*.

Algorithm 7.1 lists pseudocode for the hourly polling process as it was implemented for case study two. The polling process first calculates the current polling date/time (1), i.e. the last hour, and follows this with two consecutive stages represented by the `for` loops (2, 16).

```
1: set pollingDateTime ← (now − 1 hour);
2: for each (rssFeed) do
3:     get <item> postings in rssFeed;
4:     for each (<item> in rssFeed) do
5:         set text ← null;
6:         if (<pubDate> of <item> ≥ pollingDateTime) then
7:             set text ← sanitise(<description>);
8:             get list of keyword frequencies from text;
9:             for each (keyword in list) do
10:                write keyword, frequency, pollingDateTime to index;
11:            end for
12:        end if
13:        write <item>, pollingDateTime to database;
14:    end for
15: end for
16: for each (visualisation) do
17:    read keywords of visualisation from database;
18:    for each (keyword in visualisation) do
19:        get frequency of keyword in pollingDateTime from index;
20:        increment frequency of keyword in visualisation;
21:        write keyword, frequency, pollingDateTime to database;
22:    end for
23:    add pollingDateTime to visualisation
24: end for
```

Algorithm 7.1: Pseudocode of visualRSS's polling algorithm: cf. *case study two.*

1. The first loop (2) mines data from each RSS feed indexed by vRSS. For a particular feed, its URL is polled and the postings of the feed are retrieved (3). For each posting (4), i.e. `<item>`, the published date/time, i.e. the `<pubDate>` element, (4) is checked to determine if the posting was made within the current polling period (6). If any new postings are found, the text of each posting's `<description>` element (7) is parsed by Rome (Section 4.7.5), and sanitised.[3] From this sanitised text, a list of keywords (8) is calculated by Lucene (Section 7.7), and each *mined* keyword together with its frequency and the polling date/time is written to the index (9, 10). Lastly, the `<item>` of the RSS feed is written to the database (13).

2. The second loop (16) works per visualisation. For each visualised keyword (17, 18), the number of instances mined, from all of the RSS feeds in the visualisation's mining rules during the current polling period, is retrieved from the index (19). The frequency of the keyword in the visualisation is then updated with the new instance count (20, 21). Finally (23), the visualisation is updated with the polling date/time.

[3]Sanitisation refers to the basic data cleansing described in Section 7.4.3.

7.7 Calculating keyword frequencies from RSS-mined data

vRSS makes use of the Java-based open-source, third-party product Lucene (Appendix B.1) to calculate the frequencies of popular keywords present in the text of RSS feeds, either in the word-cloud displayed during the definition of mining rules (Section 7.4.2) or during polling (Section 7.6). With reference to McCandless et al. [252], the four basic steps used by Lucene to achieve this are described in the list below (line number references are to Algorithm 7.2):

1. **Extraction:** The original content of a data source, e.g. the text of an RSS feed, `.pdf` file for Adobe Acrobat Reader [7] or `.doc(x)` file for Microsoft Word [265] is extracted (2).

2. **Analysis:** Depending upon requirements, the analysis stage prepares the customisable *analyzers* which will be used during indexing, e.g. Lucene's `StandardAnalyzer` for conversion of text to lowercase and the inclusion/exclusion of stop words (3 - 9).

3. **Indexing:** A `Directory` object is used to define the index's location either in memory or the platform's file system (10): vRSS employs memory-based indexes. A Lucene `IndexWriter` object, associated with the index, is created, and together with the extracted text, a `Document` is produced (12). The analyzers applied to the document are used to write a stream of *tokens* to the index, i.e. the document is added to the index, where the tokens are used as *lookup* keys for retrieval (13 - 15).

4. **Querying:** From (16) an `IndexReader` object is used to query the index based upon the lookup keys and to return output. (18, 19) show a `Text_TermVectorMapper` object used by the `IndexReader` to get the newly calculated keyword frequencies which are returned to vRSS (20), whereupon the index is closed (21).

The implementation of these steps for case study two is displayed in Algorithm 7.2 which lists an extract of Java code from the `Text_Stemmer_Indexer` class in vRSS. Despite the name of the class, as previously mentioned in Section 7.4.2, stemming was not implemented, hence its absence in Algorithm 7.2. In Sections 10.4.6 and 11.6.1 respectively, we describe the extension of the `Text_Stemmer_Indexer` class for the classification and sentiment analysis components of case study three.

```
 1: StandardAnalyzer standardAnalyzer = null; // Lucene analyzer to convert text
                                              // to lowercase and include/
                                              // exclude stop words.

    // 1. Extraction.
 2: this.setTextToStemIndex( /* Text previously extracted
                                from RSS feeds. */ );

    // 2. Analysis.
 3: if (this.includeStopWords()) {
 4:     standardAnalyzer = new standardAnalyzer(Version.LUCENE_30, "English");
 5: }
 6: else {
 7:     Set<String> stopWords = (Set<String>) Stop_Word_Listener.getStopWords();
 8:     standardAnalyzer = new standardAnalyzer(Version.LUCENE_30,
                                                "English", stopWords);
 9: }

    // 3. Create memory-based index.
10: Directory index = new RAMDirectory();
11: IndexWriter iw = new IndexWriter(index,
                                     standardAnalyzer, true,
                                     IndexWriter.MaxFieldLength.UNLIMITED);

    // Index analysed contents of document.
12: Document doc = new Document();
13: doc.add("Text", this.getTextToStemIndex(), Field.Store.YES,
            Field.Index.ANALYZED, Field.TermVector.YES);
14: iw.addDocument(doc);
15: iw.close();

    // Read index.
16: IndexReader ir = IndexReader.open(index);
17: Text_TermVectorMapper ttvm = new Text_TermVectorMapper();

    // 4. Querying.
18: int docId = 0;
19: ir.getTermFreqVector(docId, "Text"), ttvm);
20: this.setKeywordFrequencies(ttvm.getWordFrequencies());
21: ir.close();
```

Algorithm 7.2: visualRSS's `Text_Stemmer_Indexer` class code calling Lucene: cf. *case study two.*

7.8 Persisting RSS to database storage

In addition to the keyword frequency index and visualisations being updated with data mined from postings made to RSS feeds during polling (Section 7.6), the postings, i.e. `<item>` elements, are also persisted (Appendix B.2) to dedicated feed tables in vRSS's MySQL[4] database. This is illustrated in line (13) in Algorithm 7.1, and allows users to browse the content of RSS feeds indexed by vRSS at any time. Moreover, if a new RSS feed is entered by a user during the definition of manual mining rules: (1) category allocation of the feed is performed by vRSS's administrator upon receipt of an automated email from the application, (2) the feed is added to vRSS's system-indexed RSS feeds, and (3) a template is used to automatically create a table for the feed in the database. Algorithm 7.3 lists an example of the templated SQL for RSS feed 159's table, where its name is post-fixed with a unique identifier and the columns correspond to the elements of an RSS feed `<item>` described in Section 2.2.3.

```
 1: CREATE TABLE _rss_159_245774_29122010_1293628782822 (
 2:     id INT(11) NOT NULL AUTO_INCREMENT,
 3:     Title VARCHAR(1024) DEFAULT NULL,
 4:     Description VARCHAR(8192) DEFAULT NULL,
 5:     PublishedDate VARCHAR(1024) DEFAULT NULL,
 6:     Link VARCHAR(2048) DEFAULT NULL,
 7:     Authors VARCHAR(1024) DEFAULT NULL,
 8:     Categories VARCHAR(1024) DEFAULT NULL,
 9:     Contents VARCHAR(1024) DEFAULT NULL,
10:     UpdatedDate VARCHAR(1024) DEFAULT NULL,
11:     URL VARCHAR(2048) DEFAULT NULL,
12:     CreatedDateTime TIMESTAMP NOT NULL,
13:     UserNo VARCHAR(50) NOT NULL,
14:     RSSFeedNo INT(11) NOT NULL,
15:     PRIMARY KEY (id)
16: )
```

Algorithm 7.3: Example of templated SQL in visualRSS to dynamically create a dedicated database table for RSS feed 159: cf. *case study two*.

7.9 Visualising data mined from RSS

In vRSS, when a set of mining rules is persisted to database storage to create a new visualisation, the visualisation initially displays no data. The mining rules include two

[4]MySQL and its compliance with ANSI/ISO standards [203] are discussed in Appendix B.1.

dates during which polling (Section 7.6) will occur: it is only between these dates that the frequencies of the keywords to be displayed in the visualisation are updated. A typical visualisation includes two charts: (1) an aggregation displaying the frequencies of the keywords since polling began, where the chart type is the one selected by the user for a sample visualisation (Section 7.4.1) prior to a set of mining rules being persisted to the database. (2) a time-series (Section 2.8.3) plot depicts keyword frequency fluctuations during the period of the aggregation.

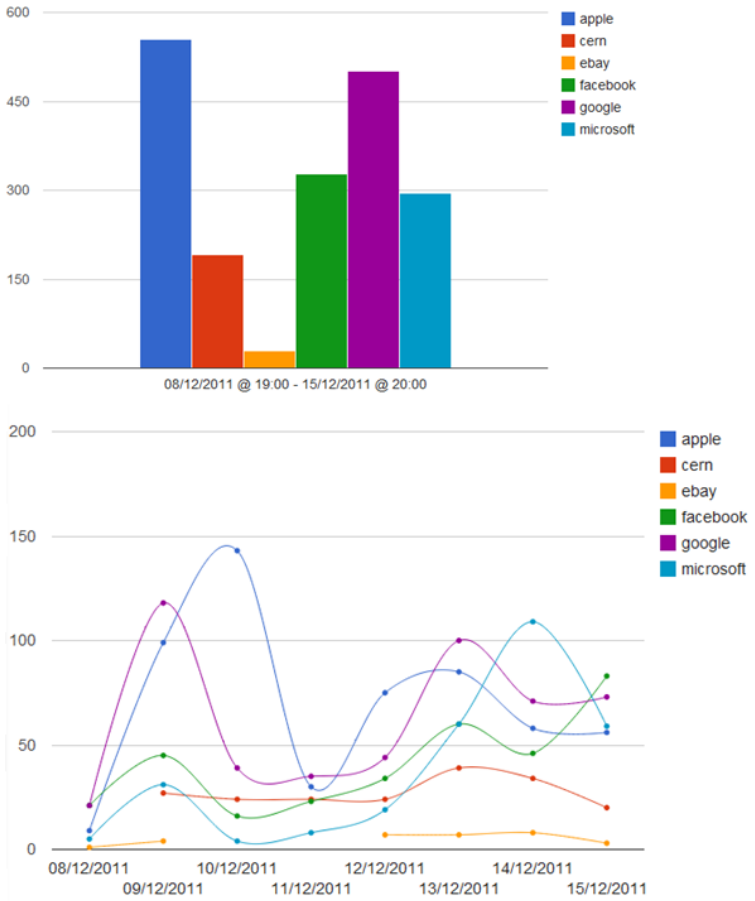


Figure 7.6: A typical visualRSS visualisation displaying the aggregation of keyword frequencies (top) in a user-selected column chart, and a time-series plot of the fluctuations in the keyword frequencies (bottom): cf. *case study two*.

Figure 7.6 illustrates a typical visualisation in vRSS. Several IT-related keywords are displayed for a week in Dec 2011: aggregated frequencies are shown in the user-selected type, i.e. in this example a column chart, and the time-series plot depicts keyword frequency

fluctuations during the aggregation's period. The page in vRSS displaying a saved visualisation also includes the keywords and RSS feeds from which the displayed keyword frequencies are calculated: these are not shown in Figure 7.6. With the exception of the word-cloud, the various pie, x - y charts, and tree-map visualisations in vRSS are implemented using Google Charts (Section 4.7.7).

7.10 Afterword

In this chapter we have presented the vRSS software written for our second case study. We have described how and why the application implements textual mining rules refined from case study one (Section 6.4.2) to mine data from RSS to be visualised. In conjunction with this, we have described how vRSS implements the common components and terminology of the two applications written for our first paradigm (Section 4.7.1). The use of vRSS in case study two, the last for this paradigm, is the subject of the next chapter, Chapter 8, before we compare mining rules with appropriate examples of related work in Chapter 9.

Chapter 8

Case study two: Mining and visualising data trends in RSS feeds

8.1 Foreword

In this chapter we present the results of our second case study which involved the use of our vRSS software (Chapter 7). We begin by defining the rationale and objectives of this case study in Section 8.2. Like our previous case study, this case study took the form of an assessed assignment for a second corpus of students: we describe the assignment, its research questions and the original RSS feed and category corpus used in Section 8.3. Section 8.4 summarises the results of the assignment, and in Section 8.5 we employ a student's submission to demonstrate the use of the refined mining rules in vRSS. We conclude with an appraisal of case study two in Section 8.6, where this case study represents the culmination of our work for our first RSS-mining paradigm (Section 1.4.1).

8.2 Rationale and objectives

Chapter 7 has described our vRSS software, and its use of textual mining rules refined from case study one (Section 6.4.2). In order to evaluate the refined rules in an *alpha*-version of vRSS, case study two formed an assessed assignment (Section 8.3) in Dec 2011 for the student corpus making up that year's class of the search engines and web navigation module previously used in case study one (Section 6.2.1). This corpus consisted of thirty-six part- and full- time Masters-level students, again of "varying employment and

experience backgrounds" (O'Shea and Levene [288]). Our objectives in this case study were to research preferences of the mining types employing the refined mining rules, visualisations, distribution of categories of feeds visualised, and the common use of these amongst the mining types (Section 7.4).

8.3 The assignment

8.3.1 Description

Figure 8.1 reproduces case study two's assignment. It consisted of three mandatory components:

1. **Part one:** Each student was required to register with vRSS and to familiarise themselves with the application's *About* pages, to look at the examples of the various mining types available and their use of keywords.

2. **Part two:** Each student was asked to select categories of RSS feeds from the corpus provided (Section 8.3.2), or to use groups of feeds of their own choice, and choose up to six unigram keywords from these feeds. These keywords were to be entered into vRSS using each of the three mining types. vRSS was then used to track the frequencies of the keywords for a seven day period.

3. **Part three:** This final part of the assignment called for each student to supply a report when their tracking period was over. The report was to list the feeds and keywords tracked, to explain why they were chosen, the mining types used, to visualise the results obtained, and to say why a specific type of visualisation was used to do this.

8.3.2 RSS Feeds and categories

We planned to employ data from an original corpus of fifty-seven RSS feeds arranged into seven categories (Appendix A.2.1). Each category was given a simple, generic label for identification, e.g. *Business, finance and economics* (BFE), in order to avoid any potential ambiguity arising from either type of `<category>` element found in the format of RSS (Section 2.2.3).

The RSS feeds in each category were chosen to be English language in content, global or regional rather than applicable to a specific country, and also to be wide-ranging and

01 Dec 2011 **SEWN 2011 - Labsheet 5 (Assessed Coursework) – visualising RSS**

Background

You will remember that earlier in the module, labsheet 2 concerned RSS feeds where we asked you to syndicate two feeds on a web page of your own, and to comment upon problems and solutions of aggregating multiple RSS feeds into one for end-user consumption.

This assignment builds upon labsheet 2, and requires you to track keywords in small numbers of RSS feeds for a period of seven days, using three related techniques, and also to report on the outputs created.

For this exercise, we will be using an application called visualRSS (vRSS) which has been developed by Martin, and which forms part of the Martin's PhD work. vRSS allows users to track and visualise keyword frequencies in RSS feeds as social data, which may be useful in roles varying from statistics to marketing and trend analysis, correlating or tracking topical issues, or for mining numeric financial and sporting data.

Thus this labsheet forms an experiment, which we would like you to participate in and support, where the outputs described in your submissions will record data and metadata which will later be analysed.

There are also bonus marks available for this labsheet as well!

So what do we want you to do?

1. **To register with visualRSS:** This is a simple process which will take a few minutes at http://qzone.dcs.bbk.ac.uk:8080/visualRSS/getRegistration/. No confidential information is held in vRSS although email addresses are required to be your Birkbeck emails.

   Upon registration, you are recommended to read vRSS's *About* pages, and to look at the examples given for the selection of keywords you would like vRSS to track from RSS feeds.

2. **Track and visualise keywords frequencies in RSS feeds:** vRSS contains some 50 RSS feeds arranged into a number of simple categories, which have been updated hourly since late-November and which will be updated hourly throughout the duration of this labsheet.

   We would like you to use categories of these feeds, or small groups of RSS feeds of your own choice (possibly including those you used for labsheet 2), to track keyword frequencies for a seven day period.

   This should be done by using up to half a dozen keywords in each of the three related techniques vRSS contains for selecting RSS feeds and keywords for tracking:

   a. **Simple RSS and keywords:** Using simple RSS and keywords, you can explore and visualise RSS feeds using keywords automatically generated from vRSS's RSS feeds. Simple RSS and keywords is useful for seeing the *current buzz* in the *rssosphere*.
   b. **Mixed RSS and keywords:** With mixed RSS and keywords, you can explore and visualise vRSS's RSS feeds using your own keywords, e.g. to track topical issues.
   c. **My RSS and keywords:** My RSS and keywords, allows you to explore and visualise RSS feeds with your own keywords and your own RSS feeds, or individual feeds from visualRSS. This allows a greater focus on a specific subject than permitted under mixed RSS and keywords.

   For example, you could use **my RSS and keywords** with keywords representing current stories in some IT-related RSS feeds of your own, and compare their frequencies against keywords generated automatically by vRSS from its equivalent category of feeds. Another example would be checking two independent collections of feeds to record the differences in frequencies of your keywords between them. Allocation of bonus marks will be based upon experimentation.

3. **Report on the outputs:** At the end of your 7 day tracking period, we would like you to write a report containing the following:

   a. To list the particular feeds and keywords you tracked, to explain why you chose them and how you implemented them in vRSS, and for each technique, a visualisation of the results obtained, and why this type of visualisation was selected.

   b. vRSS's *About* pages describe some possible scenarios of how it might be used and by whom. We would like you to describe two applications of where you think vRSS might be useful and why.

Figure 8.1: Student assignment *Visualising RSS*: cf. case study two.

RSS feed category	Number of RSS feeds	Sample RSS feed
Business, finance and economics (BFE)	5	http://feeds.bbci.co.uk/news/business/rss.xml
Fashion, celebrity and lifestyle (FCL)	6	http://www.entertainmentwise.com/rss/celebrity.rss
Film	6	http://www.denofgeek.com/index.rss
Music	7	http://www.billboard.com/rss/the-feed/
News and current affairs (NCA)	12	http://feeds.bbci.co.uk/news/world/rss.xml
Science, nature and technology (SNT)	9	http://feeds.technologyreview.com/technology_review_top_stories
Sport	5	http://www.espn.co.uk/rss/sport/story/feeds/0.xml?type=2

Table 8.1: Sample RSS feeds and categories: cf. *case study two*.

relevant in nature. Table 8.1 lists a sample RSS feed for each category in the corpus used. We sought to bias the corpus towards current affairs and scientific subjects, and also decided to use *genuine* RSS feeds rather than the outputs of aggregators or readers (Section 3.2), in order to maintain real-world data knowing that the categories were unbalanced. Furthermore, because our students were able to add RSS feeds of their own to vRSS during the assignment, imbalances in the numbers of feeds per category were inevitable.

The fifty-seven RSS feeds in the corpus were polled during the Jul - Nov 2011 period when we were gathering data for case study three (Section 10.3.2). During this time, seven of the feeds were withdrawn from the corpus (Appendix A.3.1) reducing the number of feeds to fifty before case study two began: the reasons for the withdrawal are more appropriately described in Section 10.7.1.

8.4 Results

8.4.1 Organisation

The assignment for case study two (Section 8.3) was deliberately *free* in format because we did not want to bias our students in any way that would affect results. Related to this was our preference to collect a wide variety of data for our research objectives. Therefore, our students were able to choose keywords, add new RSS feeds and use the mining types without restriction.

In discussing the results below, we initially pay attention to how the mining types were used and then focus upon the corpus of RSS feeds and categories. We subsequently comment upon the visualisations created by our students to display the frequencies of the keywords mined from the feeds during the assignment. We also discuss a specific example of vRSS's implementation of our first paradigm.

8.4.2 Mining rules

Use of mining types

For the second and third parts of the assignment, our students' submissions included some 135 visualisations in all. Given that the type of each visualisation was chosen as part of vRSS's mining rules process (Section 7.4.1) of selecting categories of RSS feeds and keywords, we were able to determine that the majority of the visualisations employed different permutations of RSS feed categories (Figure 8.5). In a small number of cases though, students used the same feeds and keywords for each mining type, e.g. one student used keywords *economy*, *recession*, *depression*, *war* and *apocalypse* "because of major events in current affairs", where semi-automatic mining proved the most successful mining type because "it tracked 4 keywords for 7 days". In terms of the mining types used to create the 135 visualisations, semi-automatic mining was the most popular type with some forty-eight (35.60%) instances created. Least popular was manual mining with forty-three instances (31.80%) produced.

Table 8.2 and Figure 8.2 display the distribution of the mining types and RSS feed categories. Semi-automatic mining proved to be the favourite type among our students with 170 (48.57%) instances used to define mining rules. Automatic mining received some 121 (34.57%) uses despite some dissatisfaction with this type because, according to one of our students, generic keywords convey "less meaning and are less indicative of specifics." However, with automatic mining intended to provide a *current buzz*, this was not surprising. Within the use of semi-automatic mining, the most popular RSS categories were BFE and NCA.

RSS feeds and categories

Case study two ended with some 202 RSS feeds, including those in new categories such as Travel and Astronomy, being polled hourly for new postings. The most popular categories were NCA with fifty-two feeds (25.74%), SNT with thirty-nine feeds (19.30%), and Sport (thirty-one feeds or 15.35%): least popular was the Miscellaneous category with two feeds

(1.00%), closely followed by EA with six feeds (2.97%) and Travel with five feeds (2.48%). Table 8.3 lists the distribution of the 202 RSS feeds according to their categories: for feeds added by the students when defining manual mining rules, category allocation was performed by the author (Section 7.8).

RSS feed category	Mining type		
	Automatic	Semi-automatic	Manual
Astronomy	0	1	1
Business, finance and economics (BFE)	20	28	12
Entertainment and arts (EA)	8	12	0
Fashion, celebrity and lifestyle (FCL)	8	11	0
Film	10	12	2
Gaming	8	13	1
Miscellaneous	3	7	1
Music	8	12	0
News and current affairs (NCA)	19	28	20
Science, nature and technology (SNT)	18	17	14
Sport	16	21	7
Travel	3	8	1
Total	121	170	59

Table 8.2: Tabular representation of mining types/RSS feed categories distribution: cf. *case study two*.

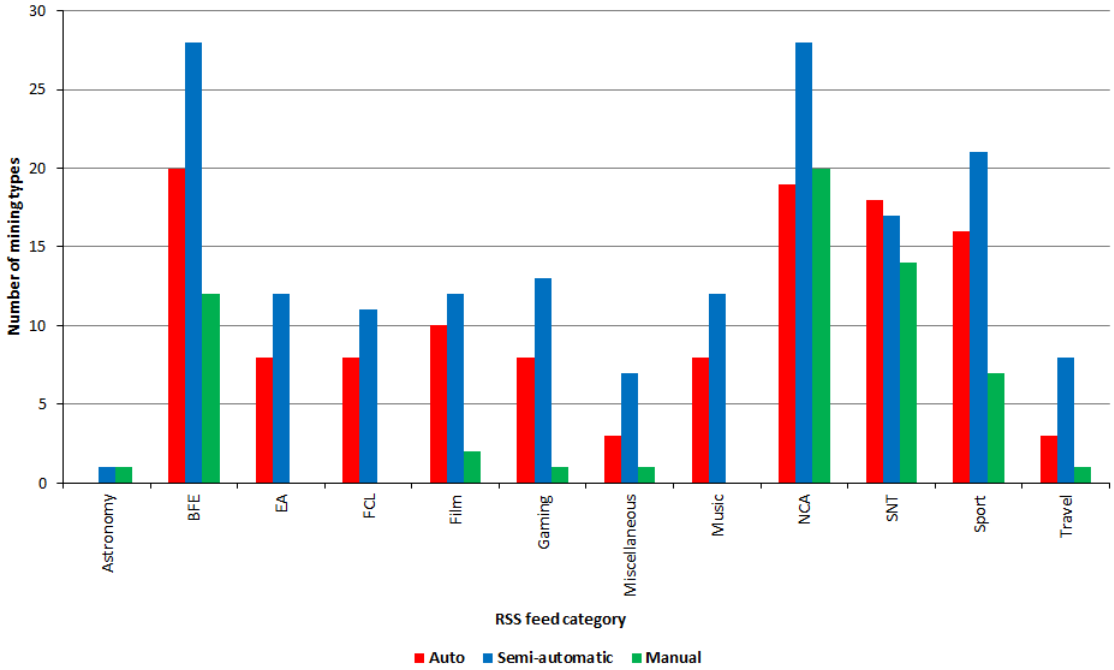


Figure 8.2: Graphical representation of mining types/RSS feed categories distribution: cf. *case study two*.

RSS feed category	No of RSS feeds
Astronomy	9
Business, finance and economics (BFE)	24
Entertainment and arts (EA)	6
Fashion, celebrity and lifestyle (FCL)	8
Film	13
Gaming	6
Miscellaneous	2
Music	7
News and current affairs (NCA)	52
Science, nature and technology (SNT)	39
Sport	31
Travel	5
Total	202

Table 8.3: Final corpus of RSS feeds and categories: cf. *case study two*.

8.4.3 Visualisations

vRSS's visualisation types, i.e. a pie and x - y charts, *word-cloud* and a *tree-map* were used to create 135 visualisations. Figure 8.3 shows that column and bar charts were the most popular visualisation types with 115 (85.19%) instances for all three mining types. We believe that the word-cloud and tree-map (Appendix B.2) types were unpopular because our students were unfamiliar with them, and also that these types do not associate words with specific colours to convey information.[1]

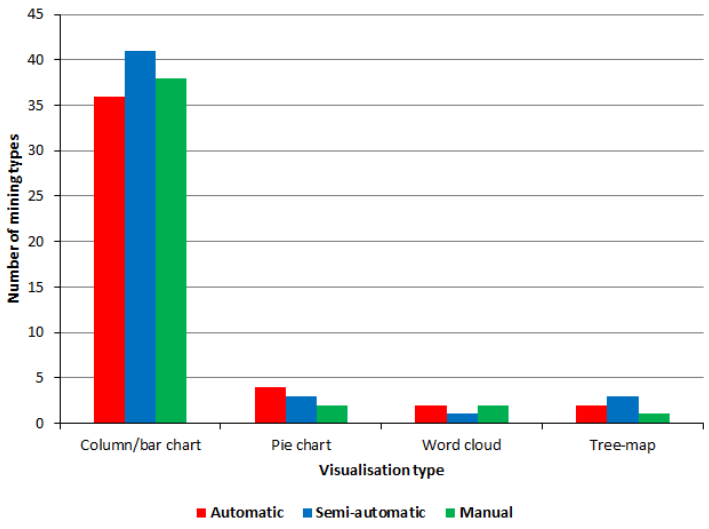


Figure 8.3: Distribution of visualisations per mining type: cf. *case study two*.

[1] cf. Hearst [172], who wrote that "treemaps have not been proven successful at showing textual data."

RSS feed category	Visualisation type			
	Col/bar chart	Pie chart	Word-cloud	Tree-map
Astronomy	1	0	1	0
Business, finance and economics (BFE)	50	3	4	3
Entertainment and arts (EA)	16	1	2	1
Fashion, celebrity and lifestyle (FCL)	12	2	3	2
Film	19	2	2	1
Gaming	17	2	2	1
Miscellaneous	10	0	1	0
Music	15	2	2	1
News and current affairs (NCA)	56	4	4	3
Science, nature and technology (SNT)	35	5	5	4
Sport	36	4	3	1
Travel	10	0	2	0
Total	277	25	31	17

Table 8.4: Tabular representation of the distribution of visualisation types and RSS feed categories: cf. *case study two*.

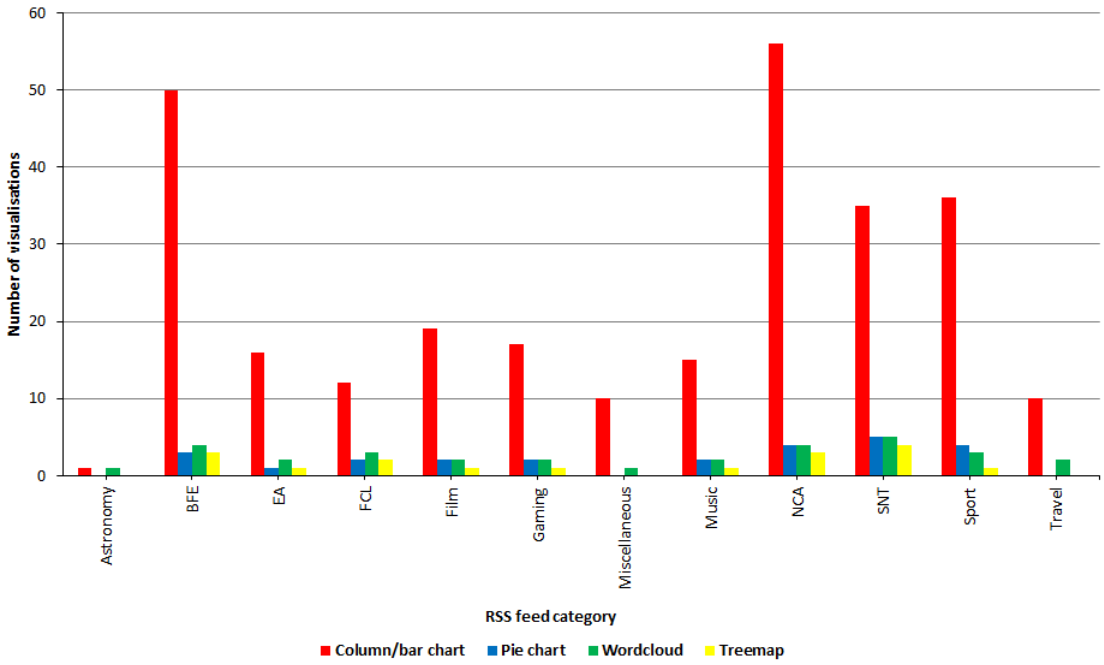


Figure 8.4: Graphical representation of the distribution of visualisation types and RSS feed categories: cf. *case study two*.

The distribution and types of the 135 visualisations created during case study two are illustrated in Table 8.4 and Figure 8.4 according to the RSS feed categories visualised.

We can see that the column and bar charts were the most popular visualisation types with 277 (79.14%) instances overall. This includes fifty-six uses of category NCA and fifty for category BFE. Related to this, the histogram in Figure 8.5 demonstrates the inverse relationship between the permutations of RSS feed categories used in visualisations and the number of visualisations.

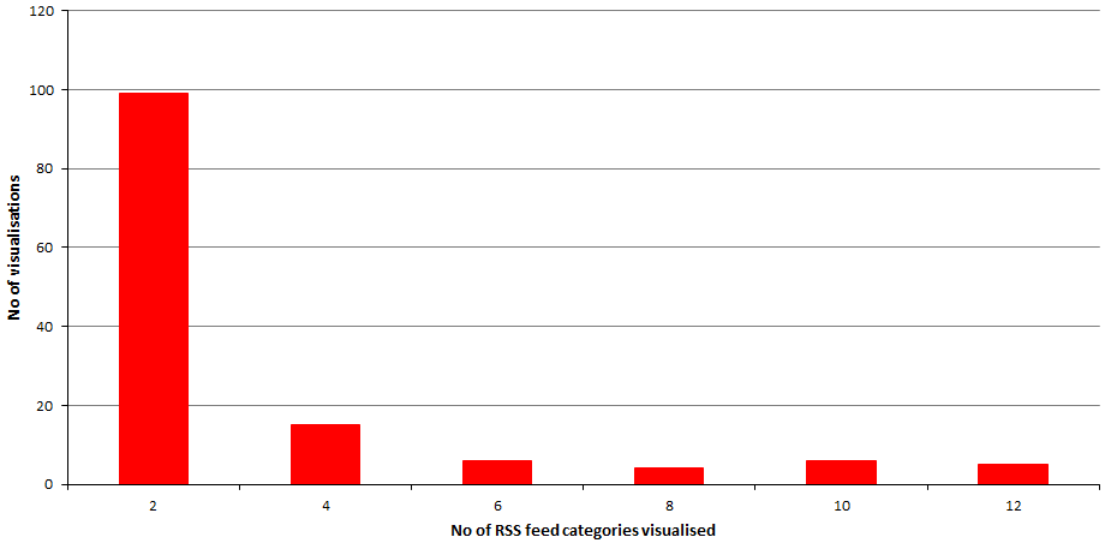


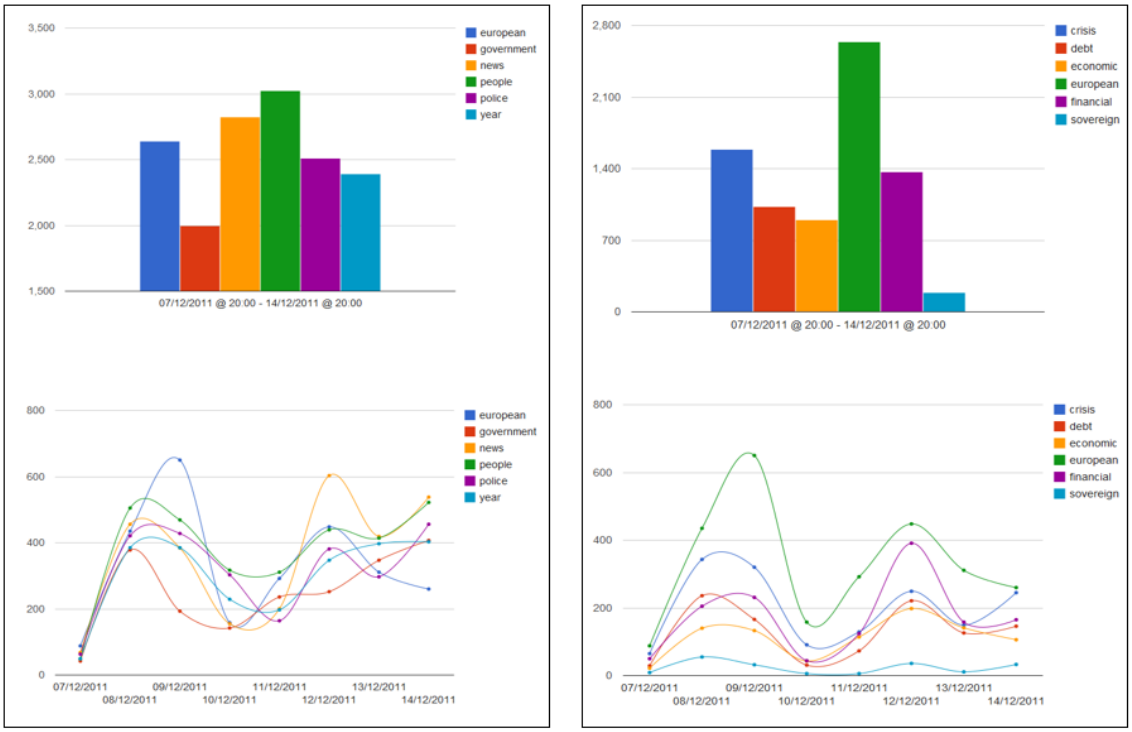
Figure 8.5: Histogram of RSS feed categories used per visualisation: cf. *case study two*.

8.5 Anatomy of a student submission: a demonstration of mining rules in visualRSS

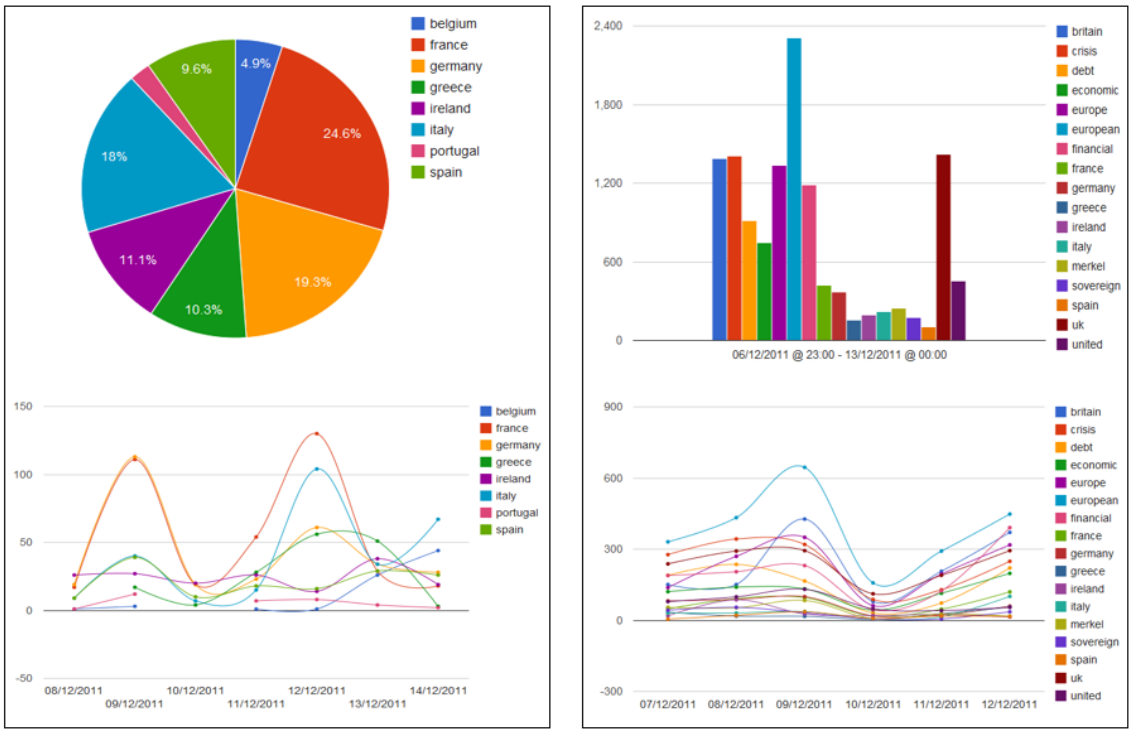
To complement Sections 7.4 and 7.9, this section provides an example of vRSS's implementation of our first paradigm's mining rules. Table 8.5 reproduces parts two and three of a submission for case study two's assignment by one of our students. The submission employed a *substantially* common set of keywords and RSS feeds taken from financial and current affairs feeds in categories BFE and NCA (Section 8.3.2) concerning the "global financial crisis" in late 2011. Figure 8.6 displays the visualisations supplied with the submission wherein column charts and a pie chart display aggregated frequencies, and time-series (Section 2.8.3) plots depict keyword frequency fluctuations during the polling, i.e. aggregation, period.

Visual'n	Mining rule	Description
1	Mining type	Automatic
	RSS feed categories	BFE, NCA
	Keywords	*european*, *government*, *news*, *people*, *police* and *year*
	Duration	07 Dec 2011 @ 20 00 00 - 14 Dec 2011 @ 20 00 00.
	Comments	The first visualisation used the automatic mining type where the student remarked that "It is no surprise that the terms *european* and *government* featured quite high on this list" and that the time-series plot "is a little frenetic as the word selection isn't especially focused."
2	Mining type	Semi-automatic
	RSS feed categories	BFE, NCA
	Keywords	*crisis*, *debt*, *european*, *economic*, *financial* and *sovereign*
	Duration	07 Dec 2011 @ 20 00 00 - 14 Dec 2011 @ 20 00 00.
	Comments	"The next visualisation (2) where the keywords are specifically chosen, is an attempt to show how the intensity of new items about the global financial crisis change over the course of a week." It was the "timeline that's most interesting here, as it shows the change in the level of reporting of the topic over time."
3	Mining type	Semi-automatic
	RSS feed categories	BFE, NCA
	Keywords	*belgium*, *france*, *germany*, *greece*, *ireland*, *italy*, *portugal* and *spain*
	Duration	08 Dec 2011 @ 20 00 00 - 14 Dec 2011 @ 20 00 00.
	Comments	This visualisation "works really well. Sticking with the Euro crisis topic, it is an attempt to see which countries were mentioned the most in News and Current Affairs. I chose a pie chart for this. Predictably *Germany*, *France* and *Italy* feature high."
4	Mining type	Semi-automatic
	RSS feed categories	NCA
	Keywords	*britain*, *crisis*, *debt*, *economic*, *europe*, *european*, *financial*, *france*, *germany*, *greece*, *ireland*, *italy*, *merkel*, *sovereign*, *spain uk* and *united*
	Duration	06 Dec 2011 @ 20 00 00 - 13 Dec 2011 @ 00 00 00.
	Comments	A semi-automatic case which was the student's "favourite. It's essentially the same as (2)" using "significantly more search terms". "Again it's the timeline that's most revealing. There's a really clear increase in euro-crisis news on Friday and a notable dip during the weekend."

Table 8.5: Mining rules in visualRSS for student submission: cf. *case study two*.



(a) Visualisations 1 and 2.



(b) Visualisations 3 and 4.

Figure 8.6: Visualisations in visualRSS for sample student submission: cf. *case study two*.

8.6 A posteriori appraisal of case study two

8.6.1 Reception

Case study two was *borne* out of the need to refine the textual mining rules used in case study one (Section 6.4.2). Therefore, case study two is distinct from its predecessor despite both studies being laboratory-based and involving corpora of student users: the research objectives of each case study also differ.

The implementation of mining rules in case study two focused less upon structural RSS metadata (Appendix B.2) inherent in the relationship of mining rules to columns of datasets (Section 5.4.2). Instead, the mining rules concentrated upon the application of simple, direct mining types (Section 7.4.1) to mine data from RSS for visualisation. In this respect, our work for case study two conforms to the *purist* approach used by Thelwall et al. [398] which we describe in Section 3.3.3.

Like myDS prior to case study one (Section 6.4.3), no pilot study was carried out for vRSS: this was again due to deadlines and the absence of an available corpus of testers during the application's development. Nevertheless, during the case study, many constructive comments were received concerning both vRSS and the assignment. In support of this, we cite the range of subject areas covered by the results presented in Section 8.4, and the range of applications suggested for vRSS in Section 8.6.3.

8.6.2 Students and RSS feeds

In Section 6.4.4 we described our homogeneous use of case study one's student corpus without regard to the demographics of the corpus, except where we sought to avoid contentious issues in the RSS feed selection and allocation. We adopted the same approach to the respective student, and RSS feed and category corpora, of case study two. The students (Section 8.2) were advised in the assignment that they were participating in an "experiment" (Section 8.3), where data and metadata would be analysed later. Moreover, no demographic or other issues were raised by the students concerning case study two even after they added their own RSS feeds to the original corpus (Section 8.3.2), bringing it to a total of 202 feeds (Table 8.3). We expect that if any issues had occurred during the case study, that the addition of feeds by the students would have been the most likely cause, and may also have led to other, different results.

8.6.3 Applications

In the third part of case study two's assignment, we asked our students to propose applications for vRSS. Many of the suggestions made confirmed the author's own opinions in subject areas such as:

- **Business intelligence:** As a data source for big-data analytics (Appendix B.2), or in using mining rules (Section 4.7.2) to turn semi-structured data into tabular form for use in data mining (Section 2.4) fact and decision tables.
- **Linguistics:** To reveal geographical, cultural or political bias in news reporting, or calculating n-gram (Appendix B.2) relationships between keywords to assist search engine results.
- **Tracking and trending:** Where an organisation might place *mouse-over* or other advertisements in web pages based upon popular keywords, or to track frequencies of keywords to determine market share.

More germane is the use of vRSS as a web service, e.g. as a browser extension or API to allow web sites to display vRSS's outputs *on the fly*, as illustrated in Figure 8.7.

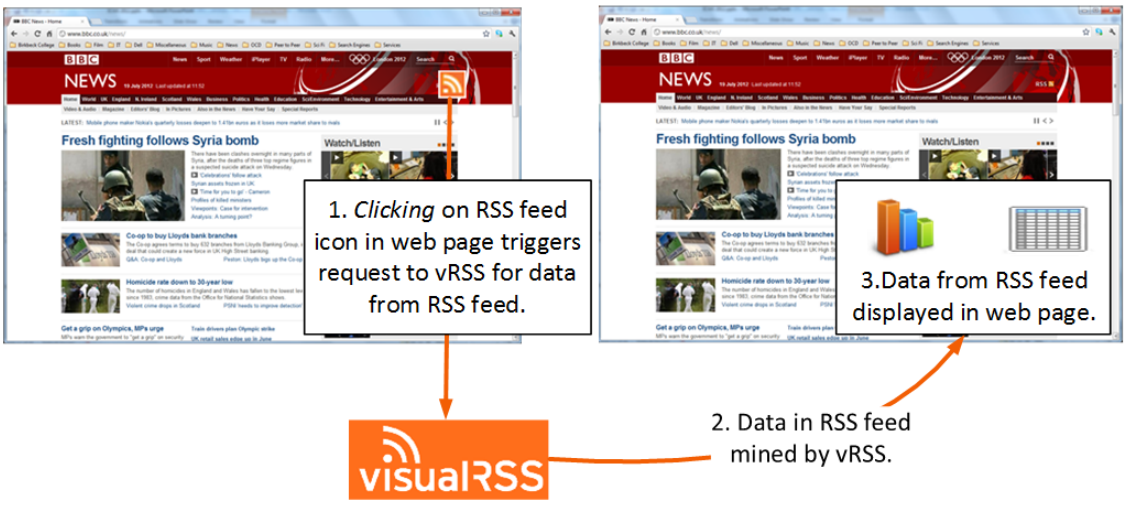


Figure 8.7: visualRSS as a web service: cf. *case study two*.

8.6.4 Publication

This chapter's description of case study two's results, together with the profile of vRSS in Chapter 7, are extended from their original publication in O'Shea and Levene [289].

8.6.5 Extensions to visualRSS

- **Android OS client:** For the mobile platform, a client *app* (Appendix B.2) was developed by Shema [354] during 2013 for our vRSS software: the client, written using the Android OS, is described in Appendix C.
- **Case study three:** vRSS was extended during 2013 - 2015 for case study three for our second paradigm (Section 4.8.1).

8.7 Afterword

In this chapter, we have described the rationale for our second case study which concerned the use of textual mining rules refined from case study one to allow a more straightforward mining of data from RSS for visualisation. We have presented the results of this case study based upon our students' use of vRSS's mining types, the final corpus of RSS feeds and categories and visualisations created. Furthermore, we have used a student's submission for the assignment to demonstrate mining rules in vRSS. This chapter also appraises case study two which represents the culmination of the work for our first paradigm. Chapter 9, the last of the five chapters in Part II, reviews this paradigm's use of mining rules and compares it with appropriate related work.

Chapter 9

Paradigm one and related work

9.1 Foreword

In this chapter we compare our first RSS-mining paradigm with those examples of related work described in Chapter 3 that we believe to be appropriate to the paradigm (Section 1.4.1). To this end, the organisation of this chapter is simple: Section 9.2 briefly reviews the two case studies for our first paradigm. This is followed by a detailed description in Section 9.3 of each example of related work, where we employ application context and use of RSS as the criteria for its comparison with our first paradigm's mining rules. Finally, Section 9.4 provides a final review of mining rules and the related work.

Each section in this chapter concerned with an example of related work is named for the title and author(s) of the example described therein.

We do not consider issues of software and application architecture in the related work described in this chapter:[1] nor do we consider the applications of RSS described in Section 3.2.

9.2 Paradigm one: a brief summary

Our first paradigm is based upon the premise that we can employ mining rules to produce from RSS data that is more *actionable and effective* than we currently see in the use of the technology (Chapter 3).

The definition of mining rules (Section 4.7.2) upon RSS to determine and visualise trends is intended to provide a straightforward means for users to specify how textual and

[1] Despite the focus of this chapter, an interesting fact is that two of the examples of related work, described in Sections 9.3.1 and 9.3.2 respectively, make use of Rome (Section 4.7.5) to parse RSS feeds.

numeric data is to be mined from feeds during polling to update and visualise the objects the rules become part of. We summarise the paradigm's two case studies as follows:

- **Case study one:** Chapters 5 and 6 described an assessed assignment held in late 2009 wherein a corpus of thirty-five Masters-level students were required to mine and visualise data from small numbers of RSS feeds allocated to them, as well as from their own feeds. This work, employing our myDS software, made use of the two mining types listed below:

  1. **Occurrence mining:** OM counts the occurrences of specified strings in the text of RSS feeds to explore trends or track issues. The three variants of OM are described in Section 5.4.1.
  2. **Value mining:** VM analyses RSS feeds which provide structured content. Such feeds report modern forms of ticker-tape (Appendix B.2) data, such as financial movements, sports or lottery results.

  In this case study, we sought to assess the feasibility of the mining types developed and the efficiency of their use. Mining rules were used to successfully create 173 reported datasets. Within this number, OM proved to be the preferred mining type with 78.00% of the datasets employing it, compared to 22.00% for value-based numeric data for VM. An overall total of 302 visualisations, covering a wide range of subjects, were also produced where standard x - y charts were the most popular types. We also observed efficiencies in the definition of mining rules because of reductions in the time taken despite the difficulties described in Section 6.4.2.

- **Case study two:** This case study, presented in Chapters 7 and 8, refined our previous use of textual mining rules from case study one, and employed a second corpus of thirty-six Masters-level students in another assessed assignment during Dec 2011. Each student was required to select up to six unigram keywords and system-indexed categories of RSS feeds, or groups of their own feeds. Thence, using our vRSS software, the students were required to track the frequencies of these keywords from the feeds for a seven day period, after which they were to report and visualise their results. The case study used three simple, direct mining types to mine RSS for data for visualisation. These mining types allowed a balance between manual selection of individual RSS feeds and keywords on the one hand, and selection of system-generated keywords from system-indexed categories of feeds on the other. A

further intermediate type allowed the use of system-indexed categories of feeds and user-entered keywords.

Our objectives were to research preferences of the mining types employing the refined mining rules, visualisations, distribution of categories of feeds visualised, and the common use of these amongst the mining types. A total of 135 visualisations were created, with semi-automatic mining proving the most popular mining type: column and bar charts were the most popular visualisation types with 85.19% instances created. In all, some 350 permutations of one or more RSS feed categories were used in the 135 visualisations. The original corpus of fifty RSS feeds and seven categories, which was extended by the feeds chosen by the students during the case study, gave a final total of 202 feeds grouped into twelve categories. News and current affairs (NCA) with fifty-two RSS feeds, was the most popular category compared to the Miscellaneous category with two feeds (Section 8.4.2).

9.3 Related work

9.3.1 *AtomsMasher: Personalised Context-Sensitive Automation for the Web* by Van Kleek et al.

AtomsMasher by Van Kleek et al. was the subject of a series of publications between 2007 - 2009 which included [414], [413] and [412]. According to [413], AtomsMasher allowed an author to write simple rules that defined *reactive behaviours*, i.e. behaviours in terms of relationships between entities and their current contextual state. These rules could be used to query, filter and specify behaviours on RSS feeds, email and weather forecasts, which were subsequently carried out by the application. AtomsMasher was considered by its authors to be a "personal automation tool" which was aimed at "a similar audience to that of most mashups and EUA". In connection with this, Van Kleek et al. [413] cited Hartmann et al. [168] by identifying this "audience" with "*the growing group of web designers and developers that are familiar with HTML and scripting languages*". AtomsMasher made use of an RDF data model (Appendix B.2) for its knowledgebase. Van Kleek et al. provided the following definition of the five elements of AtomsMasher's architecture:

1. The periodic retrieval of "external information" via RSS and Atom web feeds, API calls to web services, email and IM.

2. "Feed Prisms" to "create feed-specific import filters" in order to "distil information from packed and misappropriated source schema fields into RDF."

3. Feed rules which "reconcile new items produced by prisms with entities" already in the application's knowledgebase, "resolving references to entities mentioned in the new entity's properties".

4. State rules which "drive the state model by analyze [*sic*] incoming entries and setting state variables based on patterns in these items".

5. "Behaviour rules - execute reactive behaviours based on incoming items and state variable values, and causes actions to occur."

Examples of AtomsMasher's use documented by Van Kleek et al. concerned scenarios which could potentially benefit from the automation of processes used to retrieve, consult and consolidate multiple sources of information. The following example of a *context-based reminder*, i.e. to *remind me to call my mother when I get home*, illustrates the use of AtomsMasher's rules and syntax:

```
If //state/feed rule
    and(New.type.equals('plaze'), New.location.name.equals("Central Sq Apts"));
then
    my.location = Location("Home", {geo:New.geo});
------
if //query/behaviour rule
     my.location.equals(Location("Home"));
then
     showReminder("Call mom!");
```

In this example, AtomsMasher was instructed to look for incoming items of type *plaze* whose name equals *Central Sq Apts*. Having found such an item, a *Location* object is created, called *Home*, which is assigned geospatial coordinates from the item. *Home* is then stored in the *state table* (indicated by *my*) with the value of the *location* variable. The behaviour rule above is satisfied when the state variable *location* equals the *Home* object created beforehand.

No experimental results were documented in Van Kleek et al. [413], but the authors referred to ongoing work towards making the system more predictable and understandable, and accessible to non-programmers.

9.3.2 *RoSeS : A Continuous Query Processor for Large-scale RSS Filtering and Aggregation* by Creus et al.

Really Open Simple and Efficient Syndication, i.e. RoSeS, was a project by the *Agence Nationale de la Recherche*, i.e. the National Agency for Research, in France between Sep 2008 - Dec 2010. According to project documentation available on-line at `http://www-bd.lip6.fr/wiki/roses/start`, RoSeS was aimed at:

> "defining a set of web resource syndication services and tools for localizing, querying, generating, composing and personalizing RSS feeds available on the Web.
>
> The proposed approach is based on the observation that web content syndication can be considered as a particular large-scale distributed data management problem that might be solved by combining peer-to-peer data sharing infrastructures, XML data management and continuous query processing."

In 2011 Creus et al. [80] documented the prototype implementation of RoSeS as a "system for large-scale content-based RSS feed querying and aggregation." This prototype was part of the generic RoSeS framework which sought to implement "a range of services for crawling, filtering and aggregating of RSS feeds. A central goal of this framework is to enable large-scale RSS aggregation based on algorithms and data structures which are scalable in terms of the number of feeds, publications and subscription."

Rather than the "standard online RSS aggregation services", Creus et al. focused on "applying and extending current data stream management and continuous query processing solutions." Thus, RoSeS was based upon "declarative languages, views and multi-query optimization." A web-based interface allowed users to create "personalized feeds by defining and composing content-based filtering and aggregation queries on collections of RSS feeds."

The language implemented in RoSeS provided "instructions for *registering feeds* (register), *defining new feeds* (create) and *creating subscriptions* (subscribe)." To illustrate the use of these instructions, we reproduce elements of the example of RoSeS documented by Creus et al.:

> "Suppose Bob regularly organizes with his friends outings to rock concerts. He therefore defines a publication *RockConcertStream*, including items about concerts from feed *FollowedTwitterStream*, and rock concert announces from feed *EventAnnounces*. For this, he first registers the corresponding source streams in the system and creates a new publication *RockConcertStream*:"

The language of RoSeS made use of three clauses to perform this action: (1) a mandatory *from* clause, "which specifies the input feeds called *main feeds*", (2) a series of *join* clauses for "joining main feeds with other feeds called *annotation feeds* where secondary feeds only produce annotations (no output) to main feed elements", and (3) an optional *where* clause for "defining filtering conditions on main feeds and annotation feeds." An RSS feed called `RockConcertStream.rss` results from these actions, which we can see below in line `create feed RockConcertStream`:

```
register feed http://www.infoconcert.com/rss/news.xml as
    EventAnnounces;
register feed http://twitter.com/statuses/user_timeline
    /174451720.rss as FollowedTwitterStream;
create feed RockConcertStream
    from (EventAnnounces as $ca | FollowedTwitterStream) as $r
    where $ca[title contains 'rock'] and $r[description
    contains 'concert'];
```

`RockConcertStream` can now be subscribed to using:

```
subscribe to RockConcertStream output file 'RockConcertStream.rss'
every 10 minutes;
```

The Java-based prototype of RoSeS employed a three-level architecture: (1) *acquisition* which acquired and transformed "RSS documents into a continuous stream of RoSeS items", (2) *evaluation* which used an "*algebraic multi-query plan*" to evaluate the algebraic operations upon the incoming RoSeS items and manage the creation, changing and deletion of publication queries, and (3) *dissemination* which transformed RoSeS objects into various output formats which included RSS, SMS and email.

9.3.3 *RSS query algebra: Towards a better news management* by Getahun and Chbeir

In 2013 Getahun and Chbeir [137] proposed a "dedicated RSS algebra" that included operators which were "application domain specific and can be tuned according to the user preferences." The authors gave the basis for their algebra as being that existing "XML query algebras are not fully appropriate to retrieve RSS news", because of three principal reasons: (1) the need for "semantic-aware operators" where the text-rich content of RSS is "dependent upon the wording and verification of the author", (2) the need for a time-oriented retrieval because of the dynamic nature of news, and (3) the need for "identifying

relationships between items" because of the evolution and relatedness of news items. The authors defined their work as presenting:

- A "set of specialized RSS operators that takes into consideration the similarity and relationship between elements and textual values." This *similarity* was based upon string comparisons "between tag names and contents."
- A "highly expressive algebra that allows a user to formulate both simple and complex query expressions" including Query-by-Example (QBE) by Zloof [468].
- A "set of query equivalence rules that are used to optimize a query."
- A "prototype to validate our proposal."

Getahun and Chbeir used EBNF [202], together with symbols of their own, to define their RSS data model as a *tree* representing news items. Each element of the tree had two basic parts: a *tag name* of the element, and *content* which referred to text for either a *simple* element or other *complex* element. We illustrate this below by using the `<title>` and `<pubDate>` elements of an `<item>` (Section 2.2.3) published by the BBC in their web site's principal RSS news feed at `http://feeds.bbci.co.uk/news/rss.xml` on 02 Jan 2015:

```
<title>US sanctions North Korea over Sony</title>
<pubDate>Fri, 02 Jan 2015 20:07:38 GMT</pubDate>
```

These two elements of the `<item>` become the following *complex* element in the authors' notation:

$$\mathit{NewElement}\left(\text{“item”}, \left\{\begin{array}{l}\mathit{NewElement}(\text{“title”}, \text{“US sanctions North Korea over Sony”}),\\ \mathit{NewElement}(\text{“pubDate”}, \text{“Fri, 02 Jan 2015 20:07:38 GMT”})\end{array}\right\}\right)$$

In order to be used on RSS feeds, the algebra proposed by Getahun and Chbeir extended to the definition of a set of functions and operators that included *select*, *join*, *union*, *intersect*, *difference*, *merge* and *additive union* and *additive intersect*,[2] which operate through *windows*, i.e. "A window is a mechanism to extract [a] finite set of items from the infinite data stream."

[2]Getahun and Chbeir [137] defined a "similarity join" as a "binary operator defined on two windows." Therefore, the *additive union* and *additive intersect* operators worked by accepting as "input two windows each containing the results of a similarity-join operator." If we had two windows, i.e. w_1 and w_2, where each had been generated by a similarity-join operator, additive union would return "all elements in either w_1 or w_2", whereas additive intersect would return "all elements of w_1 having similar elements in w_2."

The authors tested their algebra using a desktop prototype application called *EasyRSS-Manager* against a dataset of 902 real news items from the AG corpus of news articles [162], and cited results where their "semantic query processing provides a number of relevant documents that could not be retrieved otherwise". The authors also compared their software with other principal RSS aggregators based upon keyword selection, similarity-based operators and merging, and reported that only their prototype provided a "customizable, adaptive and rule-based merging approach." Getahun and Chbeir concluded by considering future work including an on-line version of their software for evaluation and feedback.

9.4 Review

9.4.1 Application context and use of RSS

The examples of related work described in this chapter all employ the definition of rules upon RSS, some of which involve the specification of keywords or phrases for content filtering, or the use of structural metadata in RSS's format. These rules govern how data mined from RSS and used according to application context. We review these rules, together with the mining rules of our first paradigm, in Table 9.1 and in the following paragraphs.

Author(s)	**Application**	**Own syntax**	**Context and use of RSS**
Van Kleek et al.	*AtomsMasher*	Yes	Allowed users to write simple rules to query, filter and specify behaviours on RSS content.
Creus et al.	*RoSeS*	Yes	Data stream management and continuous query processing permitting users to create personalised RSS feeds by defining content-based filtering and aggregation queries on collections of feeds.
Getahun and Chbeir	RSS algebra	Yes	Functions and operators to query RSS content based upon the temporal nature, relatedness/similarity and relationships between items of content.
O'Shea	*myDataSharer*/ *visualRSS*	No	Defining mining rules upon RSS to determine and visualise trends from textual and numeric data.

Table 9.1: Summary of related work and mining rules: cf. *case studies one and two.*

In Section 9.3.1 we referred to Van Kleek et al. [413] citing Hartmann et al. [168] to define the target audience of AtomsMasher to include "*the growing group of web designers and developers familiar with HTML and scripting languages*". This function of AtomsMasher, in allowing this user base to write scripts for "reactive behaviours" based upon RSS content, largely precludes a comparison between AtomsMasher and our mining rules despite the similarity of rules based upon RSS.

RoSeS by Creus et al. [80] is, in its most basic form, an RSS feed aggregator (Section 3.2). This aggregation, illustrated by the example listed in Section 9.3.2, and which permits users to produce personalised feeds based upon filtering and querying RSS content, distinguishes it from the purpose of our mining rules.

In the RSS query algebra by Getahun and Chbeir [137], a complex set of operators are employed to focus upon the temporal nature, relatedness/similarity and relationships between items, of RSS content. Their prototype "allows a user to formulate RSS query [*sic*] using our operators." We cite the example of the unary *similarity selection* operator provided by the authors, i.e. "Let w be a window defined on CNN news published between 5 and 7 o'clock on December 3, 2009 and PT = {0.6,True}",[3] where we have to identify "all news in w having title element describing 'Bus explosion in Damascus' (with a similarity value of 0.6)." If w represents the results retrieved, it will not be restricted to "Bus explosion in Damascus". Rather, w will include `<item>` elements containing similar words, e.g. *coach* (similar to *bus*), *detonation* (similar to *explosion*) and *Syria* (similar to *Damascus*), because of "the semantic similarity value greater than or equal to 0.6." Similarly, the QBE operator in the algebra will, given a sample news item, retrieve "all news items in w similar to it."

When compared with the relational algebra by the authors, our mining rules provide a grouping of RSS feeds into categories only in our second case study. Mining rules do not include any provision for relatedness or similarity given our keyword conventions and characteristics (Section 4.6). Consequently, the *raison d'être* of the algebra presented by Getahun and Chbeir [137] has no specific connection with our first paradigm. Moreover, operators in their RSS algebra are based upon the retrieval from RSS feeds of those items satisfying the algebraic operators used.

It is this last point that distinguishes our mining rules from the three examples of related work we describe, i.e. none of the examples makes a dedicated use of the frequencies of popular keywords present in the text of RSS feeds, whereas mining these is at the heart

[3] In the RSS algebra by Getahun and Chbeir [137], for *selection* and top-n operators PT represents an equality threshold and TRUE determines the use of semantics or not.

of our work, especially the textual mining rules refined from case study one (Section 6.4.2).

9.4.2 Syntax

The example of registering an RSS feed in RoSeS by Creus et al. [80] in Section 9.3.2 is, in the author's opinion, the closest example of work related to the XML filters used in our original implementation of mining rules in case study one (Section 5.4.3). RoSeS also employs a dedicated *syntax*: this consideration also extends to the use of scripting in AtomsMasher by Van Kleek et al. [413] (Section 9.3.1), and the RSS algebra by Getahun et al. [137] described in Section 9.3.3.

The removal of the aforementioned XML filters when the original implementation of mining rules was refined for our second case study, allowed our final use of mining rules to employ standard HTML page controls (Section 7.4.2) and Java classes (Section 7.5) to provide a more straightforward means for users to specify how textual data is to be mined from RSS for visualisation. In this syntax-free implementation, our mining rules also differ to the related work described in this chapter.

9.5 Afterword

Despite the constraints of its case studies, described in Sections 6.4 and 8.6 respectively, our first paradigm's use of RSS produces the actionable and effective data that we describe in Section 2.9, together with visualisation as a representative medium of it for the benefit of users in differing domains. Thus, the paradigm's implementation of mining rules, and their syntax-free specification of RSS feeds and keywords, are distinct from the related work described in this chapter. This difference also extends to RSS aggregators cited in Section 3.2 where, although methods including keyword filtering are used to filter content for aggregation, the outputs are typically one or more feeds for user *consumption* via *readers*.

Part III

Paradigm 2: Classifying RSS according to the fluctuations in the frequencies of popular keywords and correlating this with sentiment

Chapter 10

Case study three: Category-based classification of RSS feeds

10.1 Foreword

Like Section 5.1, this foreword is concerned with introducing Part III of this thesis, and this chapter specifically. Part III is made up of three chapters for our second RSS-mining paradigm (Section 1.4.1). The first of these chapters concerns the first component of case study three, i.e. a semi-automated application of well-known classification techniques to RSS to classify feeds into categories according to the fluctuations in the frequencies of popular keywords present in their text. In the second component of our third case study (Chapter 11), we sought to determine a correlation between the changes in the keyword frequencies and sentiment and visualised the results. Part III concludes with a comparison of case study three (Chapter 12) with appropriate examples of related work from Chapter 3.

In presenting our third case study, this chapter and Chapter 11 are not concerned with detailed definitions of their respective subject areas of classification, data and text mining and sentiment analysis: these subjects are considered in Chapter 2. We have instead placed greater attention upon the classification and sentiment analysis techniques we employed, and the integration of the principal open-source, third-party products (Appendix B.1) used, into an extension of our vRSS software from case study two. Therefore, this chapter and Chapter 11 interleave details of the aforementioned integration, research work and results.

This chapter is made up of some six sections. In Section 10.2 the objectives of our classification work are defined, whilst software and the RSS feed and category corpus used

are described in Section 10.3. Section 10.4 is concerned with training and testing data wherein we discuss the pre-processing, organisation and use of RSS feeds in this data together with the algorithm used to produce it via semi-automated batch processing. Our choice of an open-source, third-party classification product and our use of it is the subject of Section 10.5, and this extends to profiles of the three classifiers we employed. This is followed by a description of the classification process and the presentation of summary and detailed results in Section 10.6. In the appraisal of our classification work in Section 10.7, we review issues with the format of, and published content in, RSS together with keyword miscellany.

10.2 The rationale for the keyword-based classification of RSS

The objectives of our work classifying RSS feeds into categories, based upon the fluctuations in the frequencies of popular keywords found in their text, were threefold:

1. To employ a DT classifier to produce a ballpark (Appendix B.2) result, which could be subsequently confirmed by other MNB and SVM classifiers. Therefore, we would have a *substantially* consistent set of results reached by the three classifiers which we did not need to maximise because of the *proof of concept* nature of our work.

2. To determine whether our feature selection for producing training/testing data, based upon permutations of the parameters described in Section 10.4.2, and different combinations of RSS feed elements (Section 10.4.4), would vary our classification results to any significant degree

3. To use the classification of RSS feeds and keywords to validate the use of semi-automated batch processing of RSS feeds at category-level for our sentiment analysis work (Chapter 11).

10.3 Setting-up

10.3.1 Software

The classification work described in this chapter made use of an extension to our vRSS software. Given that this extension employed the same architecture and database as the original application written for case study two (Chapter 5), we do not focus upon either

here, although the extension and other software characteristics concerning vRSS in case study three are described in Section 4.8. In this chapter, we restrict our description of software to: (1) the production of training/testing data for classification, (2) the process of classification proper, and (3) the principal open-source, third-party products used. In connection with this, although Quartz Scheduler (Section 4.7.4) was used to run the semi-automated batch processing for our classification work, we do not describe its use below: rather, we pay particular attention to the use of the following products:

- **Lucene:** Our classification work made use of permutations of keywords, e.g. stemming, stop words and n-grams to the level of trigram (Section 10.4.2). In Section 10.4.6 we document the extension of our previous use of Lucene in case study two for these elements.
- **Weka:** The choice of popular data mining tool Weka, its integration into our vRSS software, and the selection and use of DT, MNB and SVM classifiers using default parameters, is described in Section 10.5.

10.3.2 RSS feeds and categories

The original intention with our classification work was to employ the same corpus of RSS feeds and categories used in case study two (Section 8.3.2), but with data gathered during the Jul - Nov 2011 period. In Table 10.1, we reproduce the list of sample RSS feeds for each category.

RSS feed category	Number of RSS feeds	Sample RSS feed
Business, finance and economics (BFE)	5	`http://feeds.bbci.co.uk/news/business/rss.xml`
Fashion, celebrity and lifestyle (FCL)	6	`http://www.entertainmentwise.com/rss/celebrity.rss`
Film	6	`http://www.denofgeek.com/index.rss`
Music	7	`http://www.billboard.com/rss/the-feed/`
News and current affairs (NCA)	12	`http://feeds.bbci.co.uk/news/world/rss.xml`
Science, nature and technology (SNT)	9	`http://feeds.technologyreview.com/technology_review_top_stories`
Sport	5	`http://www.espn.co.uk/rss/sport/story/feeds/0.xml?type=2`

Table 10.1: Original sample RSS feeds and categories (reproduced from Table 8.1): cf. *case study three*.

This corpus originally included fifty-seven feeds, but issues encountered (Section 10.7.1) during the data gathering period reduced the count of feeds to fifty. Appendix A.3.1 lists the seven feeds in question. In all, the data used in our classification work involved a total of 128,886 `<item>` elements gathered from the fifty RSS feeds in the corpus. Furthermore, we made no use of the `<category>` element types in the format of RSS (Section 2.2.3) during our classification work: this was in order to avoid any potential ambiguity arising from them.

A second change to the corpus for case study three was due to initially poor DT classification results for RSS feed categories FCL, Film and Music in comparison with the other categories. This resulted in those categories being merged into a single, generic Entertainment and arts (EA) category, and another feed being relocated. Table 10.2 lists sample RSS feeds from each re-organised category, and the full corpus, used in all subsequent training/testing data produced for classification, is listed in Appendix A.3.2.

Category name	**Number of RSS feeds**	**Sample RSS feed**
Business, finance and economics (BFE)	5	`http://feeds.bbci.co.uk/news/business/rss.xml`
Entertainment and arts (EA)	18	`http://www.billboard.com/rss/the-feed/`
News and current affairs (NCA)	12	`http://feeds.bbci.co.uk/news/world/rss.xml`
Science, nature and technology (SNT)	10	`http://feeds.technologyreview.com/technology_review_top_stories`
Sport	5	`http://www.espn.co.uk/rss/sport/story/feeds/0.xml?type=2`

Table 10.2: Sample RSS feeds and categories after re-organisation: cf. *case study three*.

10.4 Training and testing data

The following sections of this chapter describe the components of the feature selection (Appendix B.2) for our training/testing data as a prelude to describing the algorithm to produce the data in Section 10.4.5.

10.4.1 Pre-processing

The classification component of case study three made use of a minimal ETL (Appendix B.2) involving:

- The removal of HTML from the text of `<title>` and `<description>` elements in the RSS feeds of our corpus. This was achieved by employing jsoup (Appendix B.1).
- The extension of the basic keyword conventions and characteristics described in Section 4.6 to *substantially* employ the following:
  - **Accents:** Accents, e.g. naïve or être, were removed.
  - **N-grams:** We used trigrams as the highest type of n-gram (Appendix B.2) for our keywords in order to balance computational efficiency against the number of n-grams produced. Each n-gram was further restricted to sixty-four keywords. Thus, if we generated a keyword list of bigrams, the number of n-grams totalled 128, i.e. sixty-four unigrams and sixty-four bigrams. Similarly, a keyword list of trigrams included sixty-four trigrams and 128 uni- and bi grams.
  - **Number of characters per RSS `<item>` element:** Within an element of an RSS feed `<item>`, we allowed keywords to span sentences where that element was composed of more than one sentence, but we did not allow keywords to be generated from text spanning more than one element of the same RSS `<item>`, or those preceding or succeeding it. We used the first 1,024 characters of a `<title>` element and the first 8,192 characters of a `<description>` element from any given RSS feed `<item>`.
  - **Numbers:** By default, numeric values, e.g. 75.00% or 300.00, were excluded.
  - **Stemming and stop words:** The use of keyword stemming and stop words (Appendix B.2) in our sentiment work is described in Section 10.4.6.

This sanitisation was performed dynamically during the generation of training/testing data (Section 10.4.5). In this way, we were able to base our feature selection upon permutations of the parameters described in Section 10.4.2, and different combinations of RSS feed elements (Section 10.4.4).

10.4.2 Tranches and parameter permutations

We organised the full corpus of RSS feeds and categories (Section 10.3.2) into four duration-based *tranches*, each of ten or thirty/thirty-one day periods, where a tranche was applied to a classification (Section 10.6.1) to define its duration. The tranche numbering used is listed below:

1. **Tranche 1:** Consecutive monthly 30 or 31 day periods, e.g. 01 - 30(31) Aug 2011.
2. **Tranche 2:** Consecutive monthly 10 day periods, e.g. 01 - 10, 11 - 20 and 21 - 30 (but not 31) of a given month.
3. **Tranche 3:** Cross monthly 10 day periods, e.g. 27 Jul - 05 Aug 2011.
4. **Tranche 4:** Cross monthly 30 day periods, e.g. 17 Aug - 15 Sep 2011.

Table 10.3 lists the permutations of stemming and stop words included in the tranches of training/testing data in order to determine if they would cause any significant variations in our classification results.

Permutation	**Parameters**		
number	**Stemming**	**Stop words**	**N-gram type**
1	`false`	`false`	`unigram`
2	`true`	`false`	`unigram`
3	`false`	`true`	`bigram`
4	`true`	`true`	`bigram`
5	`false`	`false`	`bigram`
6	`true`	`false`	`bigram`
7	`false`	`true`	`trigram`
8	`true`	`true`	`trigram`
9	`false`	`false`	`trigram`
10	`true`	`false`	`trigram`

Table 10.3: Parameter permutations applied to training/testing data: cf. *case study three*.

10.4.3 Segmentation

Our process to produce training/testing data for a classification is based upon the duration of the tranche applied to it (Section 10.4.2). Within the duration, the tranche is decomposed into a series of ten *segments*. For a thirty day month, segments are three days long, but in a month of thirty-one days the last segment is four days. In the case of a ten day duration, segments are daily. This allowed us to produce training data in nine of the segments and left the tenth segment for testing: the ten segments were rotated so that each segment was used once for testing.

The static nature of this rotation is depicted in Table 10.4 for a typical month where the column headings identify the number and days of each segment, and each row represents the training/testing data (shown in red) according to the segments. For any given classification, the first row was made up of testing data covering the three days of the first segment, whilst the remaining nine segments for days 04 - 30(31) formed the training data. The rotation of the segments is illustrated by the movement of the testing data along the top-left to bottom-right diagonal line of the table. Therefore, the testing data for the last row was made up from the days of segment ten, and the training data was made up from the days of the other nine segments.

This process produced ten dedicated *pairs* of training/testing data where each pair, i.e. each row of the table, was independent of the others. Therefore, in our classification work, this ordering allowed each training/testing data pair to form a cross-validation (Appendix B.2) *fold*, and permitted us to vary the inputs to our classifications with regard to the following feature selection criteria:

- The duration of the tranche applied to the classification, and permutations of the parameters described in Section 10.4.2.
- Nine training data segments and one testing data segment. Although the ordering used was fixed, we discuss the effects of changing it in Section 10.7.3.
- A combination of RSS feed elements (Section 10.4.4) for keyword frequency production.

Segments and days of tranche									
1: 01 -03	**2: 04 -06**	**3: 07 -09**	**4: 10 -12**	**5: 13 -15**	**6: 16 -18**	**7: 19 -21**	**8: 22 -24**	**9: 25 -27**	**10: 28 -30(31)**
TE	TR	TR	TR	TR	TR	TR	TR	TR	TR
TR	TE	TR	TR	TR	TR	TR	TR	TR	TR
TR	TR	TE	TR	TR	TR	TR	TR	TR	TR
TR	TR	TR	TE	TR	TR	TR	TR	TR	TR
TR	TR	TR	TR	TE	TR	TR	TR	TR	TR
TR	TR	TR	TR	TR	TE	TR	TR	TR	TR
TR	TR	TR	TR	TR	TR	TE	TR	TR	TR
TR	TR	TR	TR	TR	TR	TR	TE	TR	TR
TR	TR	TR	TR	TR	TR	TR	TR	TE	TR
TR	TR	TR	TR	TR	TR	TR	TR	TR	TE

Table 10.4: Use of *segments* for generating training/testing data for a classification, *TR* denotes training data and *TE* (shown in red) refers to testing data: cf. *case study three*.

10.4.4 RSS feed elements

Two combinations of `<title>` and `<description>` elements from each RSS feed `<item>` (Section 2.2.3) were used to produce training/testing data: the combinations are listed in Table 10.5.

Number of RSS elems	Notation	Keywords generated from RSS elements	Keyword freqs calculated from RSS elements
1	TxD	`<title>`	`<description>`
2	TDxTD	`<title>` and `<description>`	`<title>` and `<description>`

Table 10.5: Use of RSS feed elements to generate training/testing data: cf. *case study three*.

We henceforth use the convention, i.e. TxD and TDxTD, displayed in the *Notation* column in Table 10.5, to identify combinations of RSS `<title>` and `<description>` elements in our training/testing data. This notation is one of convenience: there is no mathematical or other significance inherent in it. We demonstrate this notation using the mock RSS `<item>` element listed in Algorithm 10.1.

```
1: <item>
2:     <title>Venusian intervention on Mars?</title>
3:     <description>Is Venusian intervention on Mars inevitable? That is the
                    question being asked by pundits today. Many expect the
                    president of Mars to declare martial law.</description>
4:     <pubDate>Fri, 02 Jan 2015 20:07:38 GMT</pubDate>
5: </item>
```

Algorithm 10.1: Mock RSS `<item>` to demonstrate the use of combinations of `<title>` and `<description>` elements: cf. *case study three*.

If the mock RSS feed `<item>` in Algorithm 10.1 is mined for unstemmed unigrams which exclude stop words, the resulting keywords (converted to lowercase) and frequencies produced by the alternative combinations of RSS feed elements are:

1. **TxD:** The keywords generated from the `<title>` element are *intervention*, *mars* and *venusian*. Frequencies calculated from the `<description>` element are *intervention* (1), *mars* (2) and *venusian* (1), because *intervention* and *venusian* occur once, but *mars* occurs twice.

2. **TDxTD:** Where both `<title>` and `<description>` elements are used for generating popular keywords *and* calculating their frequencies, the results are: *asked* (1), *declare* (1), *expect* (1), *inevitable* (1), *intervention* (2), *law* (1), *mars* (3), *martial* (1), *president* (1), *pundits* (1), *question* (1), *today* (1), and *venusian* (2). To take keyword *mars* as an example, it has a frequency of three because it occurs once in the `<title>` and twice in the `<description>`.

10.4.5 Algorithm

A two-stage process

Quartz Scheduler was used to run the semi-automated batch processing (Section 4.8.2) during the first component of third case study three in order to produce training/data for classification. This two-stage process involves: (1) generating popular keywords for the tranche's duration, and (2) calculating the daily frequencies of each popular keyword from each RSS feed within the tranche. For each classification, this process was carried out ten times, i.e. once for each of the ten pairs of training/testing data (Section 10.4.3).

The following paragraphs describe the process to generate popular keywords and calculate their frequencies for one of the ten pairs of training/testing data used in a given classification (Section 10.6.1): line number references correspond to the pseudocode of Algorithms 10.2 and 10.3. Java classes used in these algorithms to store keyword frequencies, and later re-used in our sentiment analysis work (Chapter 11), are described in Section 4.8.3. The use of Lucene in calculating keyword frequencies in the classification component of case study three duplicated the product's role in case study two (Section 7.7), but with the extensions described in Section 10.4.6 to handle bi- and tri- grams.

Data structures and variables

The principal data structures and variables used in Algorithms 10.2 and 10.3 are listed below:

- ***day*:** A day of the *tranche*.
- ***globalKeywords*:** The validated collection of up to sixty-four popular n-grams of each type in the training/testing data pair. *globalKeywords* is populated from *globalNgrams*.
- ***globalNgrams*:** The collection of all n-grams in the training/testing data pair for the duration of the tranche: populated from all *rssFeedElementNgrams* objects.

- ***rssFeed*:** An RSS feed belonging to an *rssFeedCategory*.
- ***rssFeedCategory*:** The category of feeds an *rssFeed* belongs to.
- ***rssFeedElement*:** Depending upon the combination RSS feed elements being used, the collection of all of the `<title>` or `<description>` elements from every *rssFeedItem* in an *rssFeed* for a *day* of the *tranche*.
- ***rssFeedElementNgrams*:** The collection of n-grams and frequencies for an *rssFeedElement* object.
- ***rssFeedItem*:** The `<item>` elements of an *rssFeed* for a *day*.
- ***tranche*:** The duration, i.e. number of days, of a training/testing data pair.

In addition to the items above, in stage two of the process, *rssFeedDailyNgrams* elements collect n-grams and frequencies from all *rssFeedElement* objects in an *rssFeed* for a *day* of the *tranche*.

Stage one: generating popular keywords

In Algorithm 10.2, in order to generate popular keywords for a training/testing data pair for the duration of the tranche (6), the *rssFeedElement* from which the frequencies are to be calculated, e.g. the `<title>` or `<description>` element (7), is read in the order of each *rssFeedCategory* and *rssFeed* (8). Therefore for every day of the *tranche* (9 - 11), if it is not a day used for producing testing data, for each *rssFeed* in the current *rssFeedCategory* (12), the text of every *rssFeedItem* is read from the *rssFeedElement* (15) and sanitised (Section 10.4.1). For the now-sanitised text of each *rssFeedItem*, n-grams are indexed, i.e. n-gram frequencies are calculated, by Lucene (16). Every n-gram is written to *rssFeedElementNgrams* (17 - 19) which collects the n-grams for the *rssFeedElement*. When more than one RSS feed element is employed to generate popular keywords, i.e. the `<title>` and `<description>`, this process is repeated for each element.

The n-grams from each *rssFeedElement* are written to *globalNgrams* as they are produced (25 - 27). When *globalNgrams* is fully populated from all of the *rssFeedElement* elements used, its n-grams are sorted in descending order of frequency and type (29). This determines the most popular n-grams overall, from which the sixty-four most popular keywords of each n-gram type are extracted and validated (30 - 34). These n-grams form the global keyword list *globalKeywords* for the current pair of training/testing data and are persisted (Section 10.4.7) to database storage (35). When producing testing data for

a pair (4), irrespective of the number of RSS feed elements used, the generation of the popular keywords is ignored because the training data for that pair (5) has already been produced and only has to be retrieved from the database to populate *globalKeywords*.

```
1: set globalKeywords ← null;
2: set globalNgrams ← null;
3: set mode ← (training or testing);
4: if (mode = testing) then
5:     read globalKeywords from database;
6: else
7:     for each (rssFeedElement) do
8:         for each (rssFeedCategory) do
9:             for each (day in tranche) do
10:                set rssFeedElementNgrams ← null;
11:                if (day not testing segment day) then
12:                    for each (rssFeed in rssFeedCategory) do
13:                        set text ← null;
14:                        for each (rssFeedItem in rssFeedElement) do
15:                            set text ← sanitise(rssFeedItem);
16:                            get index of ngrams in text;
17:                            for each (ngram not in index) do
18:                                add ngram to rssFeedElementNgrams;
19:                            end for
20:                        end for
21:                    end for
22:                end if
23:            end for
24:        end for
25:        for each (ngram in rssFeedElementNgrams) do
26:            add ngram to globalNgrams;
27:        end for
28:    end for
29:    set globalNgrams ← sort(globalNgrams in descending order);
30:    for each (ngram in globalNgrams) do
31:        if ((ngram = valid) and (countPerNgram ≤ 64)) then
32:            add ngram to globalKeywords;
33:        end if
34:    end for
35:    write globalKeywords to database;
36: end if
```

Algorithm 10.2: Pseudocode of the first stage of the algorithm to generate popular keywords for a pair of training/testing data: cf. *case study three*.

Stage two: calculating keyword frequencies

To calculate the frequency of each keyword for every RSS feed per day of the tranche for our training/testing data, we made use of the process described in Algorithm 10.3. We begin with each RSS feed category, i.e. *rssFeedCategory*, in line (1). For every day in the tranche, assuming that the current day is not excluded (2, 3), for each *rssFeed* in the *rssFeedCategory* (4), every *rssFeedItem* in the *rssFeedElement* (6) is read (8): this will depend upon the combination of RSS feed elements used (Section 10.4.4). When the text is read (9), it is sanitised before its n-grams frequencies are calculated by Lucene (10). Each n-gram (10) is then either added to *rssFeedDailyNgrams* (13), or if already present it is incremented (15). Lastly, lines (20 - 24) persist to the database daily frequencies of global n-grams in *rssFeedDailyNgrams* to be used in a classification.

```
 1: for each (rssFeedCategory) do
 2:     for each (day in tranche) do
 3:         if (day not testing segment day) then
 4:             for each (rssFeed in rssFeedCategory) do
 5:                 set rssFeedDailyNgrams ← null;
 6:                 for each (rssFeedElement) do
 7:                     set text ← null;
 8:                     for each (rssFeedItem in rssFeedElement) do
 9:                         set text ← sanitise(rssFeedItem);
10:                         get index of ngrams in text;
11:                         for each (ngram in index) do
12:                             if (ngram in rssFeedDailyNgrams) then
13:                                 increment ngram in rssFeedDailyNgrams;
14:                             else
15:                                 add ngram to rssFeedDailyNgrams;
16:                             end if
17:                         end for
18:                     end for
19:                 end for
20:                 for each (ngram in rssFeedDailyNgrams) do
21:                     if (ngram in globalKeywords) then
22:                         write ngram to database;
23:                     end if
24:                 end for
25:             end for
26:         end if
27:     end for
28: end for
```

Algorithm 10.3: Pseudocode of the second stage of the algorithm to calculate keyword frequencies for a pair of training/testing data: cf. *case study three*.

10.4.6 Keyword variations

Weka (Section 10.5), used for the classification of training/testing data, implements a `StringToWordVector` filter to convert text into a bag-of-words (BoW) (Appendix B.2). However, having used Lucene in our second case study (Section 7.7), we retained it to produce the following parameter permutations required of our training/testing data, i.e.:

- **Stemming:** This was enabled by Lucene's implementation of Porter's *Snowball* algorithm (Appendix B.2). The effects of stemming are described in Section 10.7.2.
- **Stop words:** As per case study two we used MySQL's list of stop words, and employed it to remove any n-grams composed purely of stop words.
- **N-grams:** Allowing bi- and tri- grams (Appendix B.2).

Algorithm 10.4 lists Java code from vRSS's `Text_Stemmer_Indexer` class, which was revised from our second case study (Section 7.7). The code listed permits keyword stemming, but equivalent code for non-use of stemming is not included. The first test in the code is for stop words (6). If stop words are used and n-grams larger than unigrams are required (8), a `SnowballAnalyzer` extends the use of `StandardAnalyzer` for stemming (9). To produce n-grams up to a particular size, a `ShingleAnalyzerWrapper`, *seeded* with the appropriate parameter value, is used (10). If only unigrams are required, a single `SnowballAnalyzer` is used (13). Line (17) onwards repeats this code but without stop words.

10.4.7 Database persistence

Irrespective of the combination of RSS feed elements (Section 10.4.4) used to produce training/testing data, the end result is the keyword frequencies for the duration of a tranche and its permutation of parameters (Section 10.4.2). The keyword frequencies, which consist of the ten pairs of training/testing data required for a classification, are *persisted* (Appendix B.2) to vRSS's database. In the database, the data for each pair is stored in separate, but related, tables where a typical training data table includes up to 192 columns depending upon the n-gram type used for the keywords, i.e. sixty-four per n-gram type, and 1,350 rows of data based upon fifty RSS feeds for twenty-seven days, i.e. nine segments, of the tranche's duration, or 1,400 rows for twenty-eight days. A testing data table could have as many columns but only 150 or 200 rows covering the three/four days of the training segment. These tables subsequently populate the `.arff` files used later by Weka during classification (Section 10.6.2).

```
SnowballAnalyzer snowballAnalyzer = null; // Lucene analyzer extends
                                          // StandardAnalyzer to allow
                                          // stemming.
ShingleAnalyzerWrapper shingleAnalyzer = null; // Lucene analyzer to
                                               // calculate n-grams.
int minnGramLength = 1;

// 1. Extraction.
this.setTextToStemIndex( /* Text previously extracted
                            from RSS feeds. */ );

// 2. Analysis.
if (this.includeStopWords()) {
    Set<String> stopWords = (Set<String>) Stop_Word_Listener.getStopWords();
    if (this.getnGramLength() > minnGramLength) {
        snowballAnalyzer = new SnowballAnalyzer(Version.LUCENE_30,
                                                stopWords);
        shingleAnalyzer = new ShingleAnalyzerWrapper(snowballAnalyzer,
                                                     this.getnGramLength());
    }
    else {
        snowballAnalyzer = new SnowballAnalyzer(Version.LUCENE_30,
                                                stopWords);
    }
}
else {
    if (this.getnGramLength() > minnGramLength) {
        snowballAnalyzer = new SnowballAnalyzer(Version.LUCENE_30, null);
        shingleAnalyzer = new ShingleAnalyzerWrapper(snowballAnalyzer,
                                                     this.getnGramLength());
    }
    else {
        snowballAnalyzer = new snowballAnalyzer(Version.LUCENE_30, null);
    }
}
```

Algorithm 10.4: visualRSS's `Text_Stemmer_Indexer` class extended for classification: cf. *case study three*.

10.5 Product and classifier selection

10.5.1 Product choice

The choice of product for our classification work was guided by the following criteria: (1) the use of an established product, (2) an integrated tool-set which could be readily integrated into vRSS without the need to learn a programming/scripting language, (3) the product being open-source, and (4) the availability of support. We found that online surveys and academic reviews of popular data mining tools, including but not limited to [45], [93], [156], [266], [293], [317] and [366], frequently referred to KNIME,[1] MATLAB,[2] Orange,[3] R,[4] RapidMiner[5] and Weka.[6]

We chose the Weka machine learning software (Appendix B.1), because several of our criteria were satisfied by the references to it in the surveys listed above. Moreover, Weka was categorised by Mikut and Reischl [266] as both a data mining "suite" and "library", where Weka components "have been integrated into many other open-source tools" including "RapidMiner and KNIME." Weka's popularity was further revealed by SourceForge at `http://sourceforge.net/projects/weka/files/stats/timeline?dates=2000-04-27+to+2016-03-08`, which reported that between 27 Apr 2000 - 08 Mar 2016, Weka had been downloaded in excess of seven million times.[7] [8]

10.5.2 Classifier choice

Our choice of DT, MNB and SVM classifiers was based upon their use/reference by/in work by [11], [72], [235], [243], [329], [349], [458], [424] and [462], as well as others listed in Section 2.7. To this end, our first use of Weka employed a DT classifier to provide a ballpark result which could subsequently be confirmed by other MNB and SVM classifiers. We employed the DT, MNB and SVM classifiers in Weka on a black-box basis using default parameters except where discussed below.

[1] `https://www.knime.org`.
[2] `http://uk.mathworks.com/products/matlab/`.
[3] `http://orange.biolab.si/community/`.
[4] `https://www.r-project.org`.
[5] `https://rapidminer.com`.
[6] `http://www.cs.waikato.ac.nz/ml/weka/`.
[7] We also made previous use of Weka in our first case study (Section 5.8).
[8] An interesting fact is that page `http://www.sentistrength.wlv.ac.uk/#About` of the web site for SentiStrength, used for the sentiment analysis component of case study three (Chapter 11), makes reference to Weka's `.arff` file format. We describe our use of this format in Section 10.6.2.

10.5.3 The decision tree (DT)

Weka's `J48` classifier is an implementation of Quinlan's C4.5 algorithm [313] to generate decision trees. Quinlan described the:

> "program C4.5 from which the whole system derives its name. This program generates a classifier in the form of a decision tree, a structure that is either
>
> - a *leaf*, indicating a class, or
> - a *decision node* that specifies some test to be carried out on a single attribute value, with one branch and subtree for each possible outcome of the test.
>
> A decision tree can be used to classify a case by starting at the root of the tree and moving through it until a leaf is encountered. At each nonleaf decision node, the case's outcome for the test at the node is determined and attention shifts to the root of the subtree corresponding to this outcome. When this process finally (and inevitably) leads to a leaf, the class of the case is predicted to be that recorded at the leaf."

We based our use of decision trees upon Drazin and Montag [96], who discussed how Weka's `J48` classifier used "*pruning* tactics" to "affect the classification accuracy of a testing set of data." Pruning is concerned with reducing the complexity of decision trees to optimise classification accuracy (Appendix B.2). There are two approaches: (1) post-pruning which concerns reductions in the complexity of *grown* trees, whereas (2) pre-pruning focuses instead on pruning a *growing* tree. Drazin and Montag, using the *soybeans* dataset by Michalski and Chilausky [258], found that post-pruning classification accuracy improved to a level of 92.80% as confidence[9] was raised to a maximum value of 0.50. For pre-pruning, accuracy fell when the minimum number of instances per leaf was increased over a range of one to forty.[10] Figure 10.1 displays a Weka decision tree, reformatted using GraphViz [160], of Fisher's *Iris* dataset [123]. The numbers in parenthesis per leaf represent the number of total instances in the current classification at this leaf (numerator)/the number of incorrect classifications at this leaf (denominator).

The objective in selecting an attribute for the partitioning of a decision tree is to reduce the uncertainty in the data to the maximum possible extent, i.e. to select the

[9] Confidence in Weka's (Appendix B.1) `J48` classifier is used to "test the effectiveness of post-pruning" [96] where "lowering the confidence factor decreases the amount of post-pruning."
[10] cf. Rajput and Arora [316].

attribute with the maximum expected reduction in entropy caused by partitioning the data according to the attribute. An attribute with a significant reduction will be preferred to other attributes. Entropy is the measure of the uncertainty associated with the selected attribute.

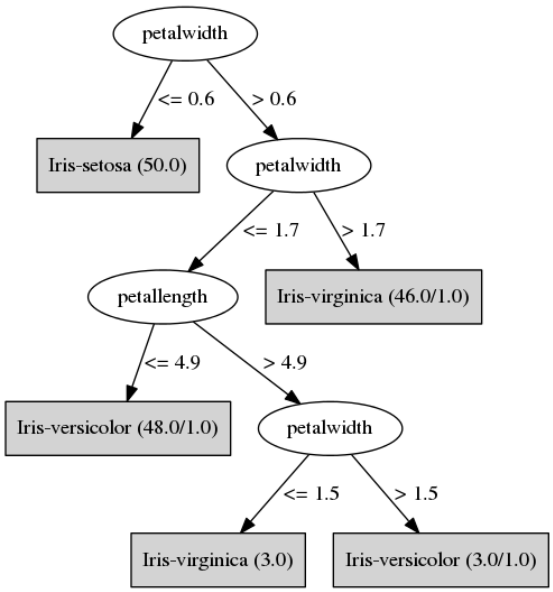


Figure 10.1: Sample Weka decision tree, formatted using GraphViz [160], displaying Fisher's *Iris* dataset [123]: cf. *case study three*.

10.5.4 Multinomial naïve Bayes (MNB)

Naïve Bayes is a probabilistic classifier based upon the premise that the existence or otherwise of a feature in a class is unrelated to the existence or otherwise of any other feature. Vectors of feature values are classified into one of a finite set of classes. When applied to document categorisation, a *document* is seen as a BoW, and a multinomial version of Bayes can be applied. Frank and Bouckaert [129] defined MNB as "the version of naive Bayes that is commonly used for text categorization problems." Furthermore:

> "In the MNB classifier each document is viewed as a collection of words and the order of words is considered irrelevant. The probability of a class value c given a test document d is computed as"

$$P(c|d) = \frac{P(c)\prod_{w\in d} P(w|c)^{n_{wd}}}{P(d)},$$

> "where n_{wd} is the number of times word w occurs in document d, $P(w|c)$ is the probability of observing word w given class c, $P(c)$ is the prior probability of class c, and $P(d)$ is a constant that makes the probabilities for the different classes sum to one. $P(c)$ is estimated by the proportion of training documents pertaining to class c and $P(w|c)$ is estimated as"

$$P(w|c) = \frac{1 + \sum_{d \in D_c} n_{wd}}{k + \sum_{w'} \sum_{d \in D_c} n_{w'd}},$$

> "where D_c is the collection of all training documents in class c, and k is the size of the vocabulary (i.e. the number of distinct words in all training documents). The additional one in the numerator is the so-called Laplace correction, and corresponds to initializing each word count to one instead of zero. It requires the addition of k in the denominator to obtain a probability distribution that sums to one. This kind of correction is necessary because of the zero-frequency problem: a single word in test document d that does not occur in any training document pertaining to a particular category c will otherwise render $P(c|d)$ zero."

We made use of Weka's default `NaiveBayesMultinomial` classifier as it was supplied, despite the existence of a `NaiveBayesMultinomialText` classifier. This latter classifier works on strings of text rather than the numeric keyword frequencies, i.e. features, produced by Lucene (Section 10.4.6) which made up our Weka `.arff` data (Section 10.6.2) files.[11]

10.5.5 The support vector machine (SVM)

The history of SVMs can be traced back to Boser et al. [47] and Cristianini and Shawe-Taylor [81]: here we refer to the definition provided by Manning et al. [243]:

> "An SVM is a kind of large-margin classifier: It is a vector-space-based machine-learning method where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)."

[11] The applicability to our classification work of the `NaiveBayesMultinomial` and `NaiveBayesMultinomialText` classifiers was discussed on 17 Apr 2014 on the Weka mailing list at `http://list.waikato.ac.nz/pipermail/wekalist/2014-April/060604.html`.

Where the maximal decision boundary, i.e. the *hyperplane*, separates the classes and has the largest distance between border-line data points, i.e. *support vectors*. With reference to biomedicine, Statnikov et al. [372] wrote that if such a hyperplane, i.e. "linear decision surface does not exist, the data is mapped into a much higher dimensional space ("feature space") where the separating decision surface is found". This non-linear classification is illustrated in Figure 10.2 where the feature space is enabled by the *kernel trick*, i.e. a similarity measure (Appendix B.2) or kernel function, based upon the distribution of similarities between a given data point and other data points around it.

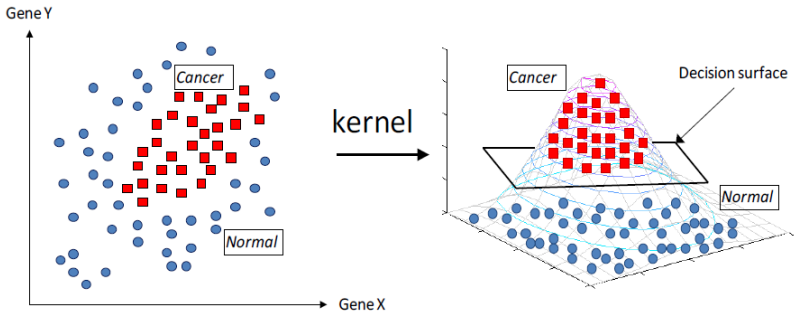


Figure 10.2: SVM-based detection of cancer cells (reproduced from Statnikov et al. [372]): cf. *case study three*.

According to Manning et al. [243], "SVMs are inherently two-class classifiers." Despite this, common multi-class SVM classification methods include the use of sets of "one-versus-rest" or "one-versus-one" classifiers. Our work employed the one-versus-one method because of the implementation of this in the Java version of the LIBSVM library [65] which Weka provides a wrapper for: there is no actual default SVM classifier in Weka.

We employed LIBSVM within Weka using default parameters except for the use of a linear kernel, which is more applicable to SVM-based text classification. Joachims [206] has cited four reasons for this: (1) "High dimensional input space" can handle "large feature spaces", (2) that most features are "irrelevant", (3) the sparse nature of vectors with "few entries which are not zero", and with reference to the classical *Reuters-21578* dataset (Lewis [225]), because (4) "most text categorization problems are linearly separable". The use of a linear kernel is also supported by: (1) Teng et al. [391] who reported "that we can achieve better classification results through the linear SVMs", (2) Rennie and Rifkin [324] who wrote that "in informal experiments, we also found that linear performs at least as well as non-linear kernels", and (3) Yang and Liu [458] who "obtained a slightly better result with the linear SVM than with the non-linear models."

10.6 Classification

10.6.1 Implementation

Our classification of RSS feeds involved two-stages run using Quartz Scheduler (Section 4.7.4): each stage is described below and corresponds with the appropriate process in the DFD displayed in Figure 10.3:

1. `perform sub-classifications:` In Section 10.4 we described the production of ten pairs of training/testing data for a classification. In the first stage of the classification process, Weka is used to classify each pair of training/testing data independently of the others. We refer to this process as *sub-classification* in order to distinguish it from the actual *parent* classification in stage two of the process. Thus, a classification is actually made up from a series of ten iterations, where each iteration sub-classifies a pair of training/testing data.

   Algorithm 10.5 lists Java code to call Weka to perform a sub-classification in vRSS for a distinct pair of training/testing data which acts as a *fold* in cross-validation (Appendix B.2). In this code, we can see the training/testing data `.arff` files (Section 10.6.2) being retrieved in lines in (1) and (3), where the class attribute, i.e. the RSS feed category, is set for each in (2) and (4). (5) and (6) concern the retrieval of the classifier and its parameters from the database, leading to (9) its *training*. Finally the classifier is evaluated (10) and the sub-classification is performed (11).

   The Weka output of a particular sub-classification is reproduced in Figure 10.5. In this case, an MNB classification of bigrams allowed keyword stemming and the popular keywords, which included stop words, were generated from the `<title>` element of the RSS feeds in our corpus, and frequencies were calculated from the `<description>` elements. In the sub-classification, 1,350 training instances covered nine segments for the period 04 - 30 Sep 2011, and were tested on 150 instances of testing data covering segment 01 - 03 Sep 2011. Of these, we can see 114 correctly sub-classified testing instances in Figure 10.5.

2. `calculate average result:` The outputs produced by the ten sub-classifications are averaged to produce a final result for the classification based upon the tranche, parameter permutation, combination of RSS feed elements, and training/testing data described in Section 10.4.

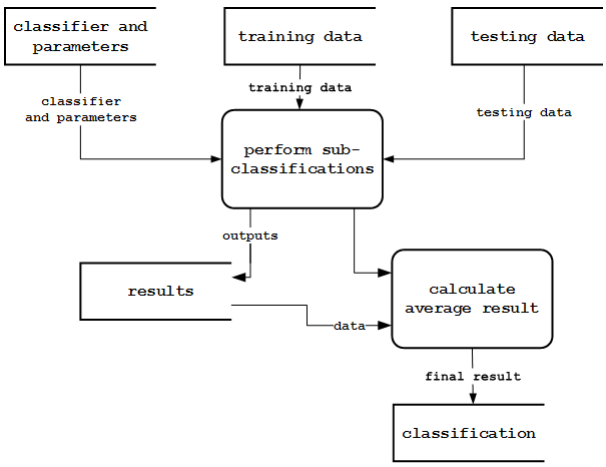


Figure 10.3: DFD of the classification process (*substantially* reproduced from Figure 4.6): cf. *case study three*.

```
   // Read training data from Weka .arff data file.
 1: Instances trainingInstances = new Instances(trainingData.arff);

   // Set last column as training data's class attribute.
 2: trainingInstances.setClassIndex(trainingInstances.numAttributes() - 1);

   // Read testing data from Weka .arff data file.
 3: Instances testingInstances = new Instances(testingData.arff);

   // Set last column as test data's class attribute.
 4: testingInstances.setClassIndex(testingInstances.numAttributes() - 1);

   // Get classifier and options.
 5: Classifier cls = // Set type of Weka classifier from database.
 6: String options = // Set Weka options for classifier from database.
 7: String[] optionsArray = options.split(" ");
 8: cls.setOptions(optionsArray);

   // Train the classifier.
 9: cls.buildClassifier(trainingInstances);

   // Evaluate the classifier on the testing data.
10: Evaluation eval = new Evaluation(trainingInstances);
11: eval.evaluateModel(cls, testingInstances);
```

Algorithm 10.5: visualRSS code implementing Weka for a sub-classification, i.e. iteration, of a *parent* classification: cf. *case study three*.

10.6.2 Data formats

The `.arff`, i.e. *attribute relationship file format*, was developed for use with Weka (Section 10.5). Figure 10.4 displays an extract of an `.arff` file where some minor formatting has been performed for clarity. The data lists the first ten unigrams of training dataset 19040, and the frequency of each unigram in each of the five RSS feeds of the Banking, finance and economics (BFE) category (Section 10.3.2) for the first day of the classification, i.e. 04 Sep 2011. The file has the following items:

- The `@relation` declaration provides a text name for the data.
- `@attribute` defines the name and data type of an attribute, i.e. keyword. The last `@attribute` listed identifies the class, i.e. `RSSFeedCategoryDescription`, and includes a list of the RSS feed categories.
- `@data` represents the data in the relation. Each row forms a single instance of training or testing data, i.e. the frequencies of the keywords, calculated from the appropriate elements of a particular RSS feed for a given day, and is terminated by the name of the class that the feed belongs to.

```
@relation _dm_19040_031925_06112013_1383748052958_Numeric

@attribute video numeric
@attribute kill numeric
@attribute stock numeric
@attribute world numeric
@attribute win numeric
@attribute bank numeric
@attribute libya numeric
@attribute europ numeric
@attribute back numeric
@attribute euro numeric

. . .

@attribute RSSFeedCategoryDescription {'BFE','NCA','SNT','Sport','EA'}

@data
0,0,0,0,0,4,0,0,0,0,'BFE'
0,3,0,4,0,0,2,1,0,1,'BFE'
0,0,0,0,0,0,0,0,0,0,'BFE'
0,0,2,0,0,7,0,2,1,1,'BFE'
0,0,0,0,0,0,0,0,0,0,'BFE'

. . .
```

Figure 10.4: Extract of training dataset 19040 in the `.arff`, i.e. *attribute relationship file format*, used by Weka: cf. *case study three*.

```
=== Summary ===

Correctly Classified Instances         114               76      %
Incorrectly Classified Instances        36               24      %
Kappa statistic                          0.6733
Mean absolute error                      0.1281
Root mean squared error                  0.2722
Relative absolute error                 42.5318 %
Root relative squared error             70.1404 %
Coverage of cases (0.95 level)          95.3333 %
Mean rel. region size (0.95 level)      46      %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.600    0.000    1.000      0.600    0.750      0.758    0.953     0.883     BFE
                 0.694    0.070    0.758      0.694    0.725      0.644    0.906     0.831     NCA
                 0.500    0.083    0.600      0.500    0.545      0.447    0.869     0.688     SNT
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Sport
                 0.926    0.188    0.735      0.926    0.820      0.712    0.920     0.841     EA
Weighted Avg.    0.760    0.101    0.767      0.760    0.753      0.676    0.918     0.828

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
  9  1  5  0  0 |  a = BFE
  0 25  4  0  7 |  b = NCA
  0  4 15  0 11 |  c = SNT
  0  0  0 15  0 |  d = Sport
  0  3  1  0 50 |  e = EA
```

Figure 10.5: Typical Weka results for a sub-classification, or iteration of a *parent* classification. Results for training dataset 19040 and testing dataset 20861 are displayed: cf. *case study three*.

In case study three, a typical `.arff` file for training data covering twenty-seven days of a month, i.e. nine segments, included some 1,350 (27 days * 50 RSS feeds) or 1,400 (28 days * 50 RSS feeds) `@data` rows and up to 192 columns for the keywords, i.e. sixty-four per n-gram type (Section 10.4). A testing data `.arff` file included some 150 (3 days * 50 RSS feeds) or 200 (4 days * 50 RSS feeds) `@data` rows.

For a given classification, twenty `.arff` files were required, i.e. one for the training data and one for the testing data of each sub-classification (Section 10.6.1). Figure 10.5 displays the results of the sub-classification of a pair of training/testing data: in this example, we see the results of training dataset 19040 on test dataset 20861. The *Summary* section of the results includes the number of correctly/incorrectly sub-classified instances, whilst the *Detailed Accuracy by Class* includes TP, TN, FP and FN figures used to calculate the precision, recall, and F-measure metrics per RSS feed category, i.e. class, used. The rows of the *Confusion Matrix* (Kohavi and Provost [212]) represent actual classes and the columns indicate predicted classes.[12] Correctly sub-classified instances are visible in the top-left to bottom-right diagonal line: in this case we can see 114 correctly sub-classified

[12]In Appendix B.2 we explain the precision, recall and F-measure metrics, and illustrate the calculation of the TP, TN, FP and FN figures for class NCA using the confusion matrix in Figure 10.5.

instances belonging to test dataset 20861, i.e. in the *Summary* we see that the confusion matrix includes 9 (BFE), 25 (NCA), 15 (SNT), 15 (Sport) and 50 (EA).

Once the series of ten sub-classifications for a classification has completed (Section 10.6.1), the second stage of our classification process averages the F-measure, precision and recall metrics, based upon their values per class in the *Detailed Accuracy by Class* part of the Weka outputs produced by each sub-classification, in order to produce a final result (Section 10.6.3).

10.6.3 Results

Metrics

We employed the popular F-measure, precision and recall metrics (Appendix B.2) to summarise our classification results. In both the summary and detailed results, we have also aggregated the tranching (Section 10.4.2) and RSS feed elements (Section 10.4.4).

Summary statistics

Table 10.6 summarises our results by listing the number of individual classifications made for each classifier, as well as F-measure and other metrics for each classifier averaged across their respective number of classifications. We can see a *substantially* consistent result for all three classifiers, but where MNB and SVM produced more comparable results despite a 0.02% difference between their F-measure (Appendix B.2) values. The lower DT result is explained by the large number of classifications we ran to do with *pruning* based upon Drazin and Montag [96] (Section 10.5). Given the tranches described in Section 10.4.2, reference to Drazin and Montag for tranche one resulted in a wider range of DT classifications and results, whereas default Weka parameters in the other tranches produced DT results comparable with those for MNB and SVM.

Classifier	No of classifications	Avg F-measure	Avg precision	Avg recall
DT	740	0.72	0.77	0.71
MNB	200	0.79	0.83	0.80
SVM	200	0.77	0.81	0.77

Table 10.6: Tabular representation of summary classification results: cf. *case study three*.

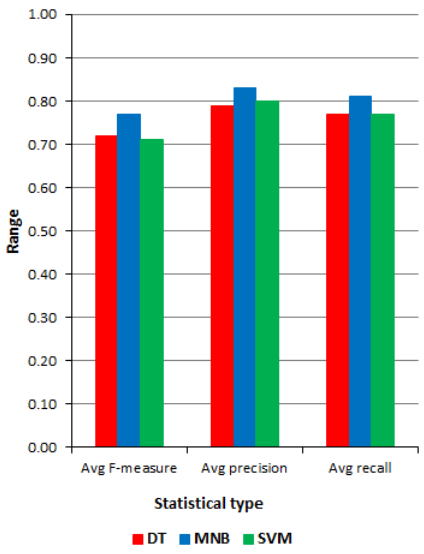


Figure 10.6: Graphical representation of summary classification results: cf. *case study three*.

Detailed breakdown

The detailed results in Table 10.7 include the parameter permutations in Section 10.4.2. We see that DT values were again lower than MNB and SVM, and classifications using keyword stemming were frequently outperformed by those without stemming, e.g. the MNB classification for bigrams with stemming and stop words returned an F-measure of 0.79, whilst unstemmed and stopped/unstopped classifications returned 0.80. Similarly, the DT classification using bigrams, with stemming but without stop words, gave an F-measure of 0.71, whilst those not using stemming scored 0.73 regardless of their use of stop words or not.

Par permutation				Classifier								
No	Stem	Stop	Len	DT (Avg)			MNB (Avg)			SVM (Avg)		
				Fm	Pr	Rc	Fm	Pr	Rc	Fm	Pr	Rc
1	F	F	1	0.72	0.78	0.71	0.79	0.83	0.80	0.77	0.81	0.76
2	T	F	1	0.70	0.74	0.69	0.79	0.83	0.80	0.77	0.81	0.77
3	F	T	2	0.73	0.78	0.72	0.80	0.83	0.80	0.77	0.81	0.76
4	T	T	2	0.71	0.75	0.70	0.79	0.83	0.80	0.77	0.81	0.77
5	F	F	2	0.73	0.78	0.72	0.80	0.84	0.80	0.77	0.82	0.77
6	T	F	2	0.71	0.76	0.71	0.80	0.83	0.80	0.78	0.82	0.77
7	F	T	3	0.73	0.78	0.72	0.79	0.83	0.80	0.76	0.81	0.76
8	T	T	3	0.72	0.76	0.71	0.79	0.82	0.80	0.77	0.81	0.77
9	F	F	3	0.73	0.78	0.72	0.80	0.84	0.80	0.77	0.82	0.77
10	T	F	3	0.71	0.76	0.71	0.80	0.83	0.80	0.77	0.81	0.77

Table 10.7: Tabular representation of detailed classification results: cf. *case study three*.

In Table 10.7 the left hand columns read: *No* refers to the number of the parameter permutation (Table 10.3), *Stem* is stemming, *Stop* refers to stop words and *Len* concerns the n-gram type's length. Right hand columns *Fm*, *Pr* and *Rc* refer to the F-measure, precision and recall metrics respectively. Figure 10.7 displays the detailed F-measure results.

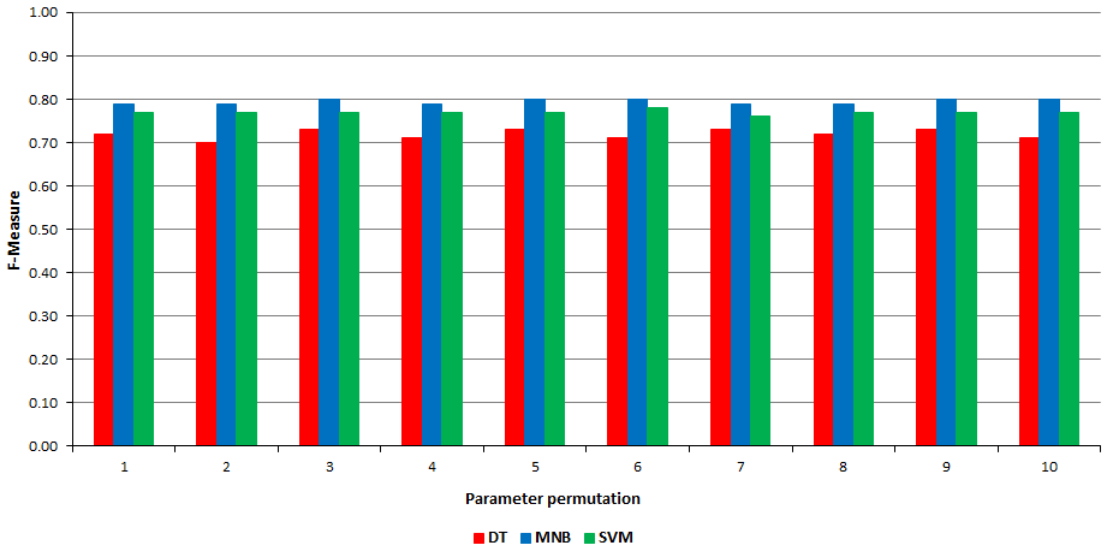


Figure 10.7: Graphical representation of detailed F-measure results: cf. *case study three*.

10.7 A posteriori appraisal of the classification component of case study three

10.7.1 The format of RSS

During the Jul - Nov 2011 data gathering period for case study three, we encountered a series of issues with data mined from our corpus of RSS feeds and categories (Section 10.3.2). This resulted in seven feeds being withdrawn from the corpus before case study two began in Dec 2011. These feeds are listed in Appendix A.3.1, and the reasons for their withdrawal concerned the incorrect population and optional nature of `<item>` elements (Section 2.2.3) when they were published, as the following examples demonstrate:

- **Incorrect population of elements:** Figure 10.8a shows RSS feed 146, i.e. film and television web site *Ain't it Cool News*, with the `<description>` element of each `<item>` populated with the `<pubDate>`.
- **Non-population of elements:** In RSS feed 180, i.e. *Scientific American* magazine, the non-population of the `<pubDate>` caused many duplicate postings, e.g. Figure 10.8b displays the same story for a period of four hours during polling.

ID	<title>	<description>	<pubDate>	PollingDateTime
80031	Matt Damon Looks Like He's Having Issue...	Wed, 10 Aug 2011 16:37:36 PM CDT		2011-08-11 00:00:00
80032	Watch the BLACK DYNAMITE Pilot from C...	Wed, 10 Aug 2011 16:06:47 PM CDT		2011-08-11 00:00:00
80033	Eddie Murphy Is HONG KONG PHOOEY!	Wed, 10 Aug 2011 15:52:12 PM CDT		2011-08-11 00:00:00
80034	All-Time Low Prices For SHANDLING'S, PA...	Wed, 10 Aug 2011 14:22:21 PM CDT		2011-08-11 00:00:00

(a) Extract from RSS feed 146 where the `<description>` element has been populated by the `<pubDate>` element.

ID	<title>	<description>	<pubDate>	PollingDateTime
266351	Pregnant Plesiosaur Fossil Suggests the ...	By Zoë Corbyn of Nature magazine A fossil of a...		2011-08-12 00:00:00
266403	Pregnant Plesiosaur Fossil Suggests the ...	By Zoë Corbyn of Nature magazine A fossil of a...		2011-08-12 01:00:00
266453	Pregnant Plesiosaur Fossil Suggests the ...	By Zoë Corbyn of Nature magazine A fossil of a...		2011-08-12 02:00:00
266504	Pregnant Plesiosaur Fossil Suggests the ...	By Zoë Corbyn of Nature magazine A fossil of a...		2011-08-12 03:00:00

(b) Extract from RSS feed 180 where the `<pubDate>` element for each `<item>` is not-populated.

Figure 10.8: Issues with incorrect population and optional nature of RSS feed `<item>` elements: cf. *case study three*.

Whilst the *optionality* of elements granted by the format of RSS provides much flexibility, it follows that the publisher of a feed is also responsible for the content of their feed. *We believe that the format of RSS can be criticised in this respect.* This is because optionality allows publishers to not populate elements in feeds which are of use to clients, and which

may be required by aggregators and readers (Section 3.2) to function correctly, e.g. the <pubDate> element of each <item> in a feed. Similarly, with reference to: (1) the *actual* instances of incorrectly populated <item> elements described above, and (2) the work by Liu et al. [231] and Longe and Salami [236] which concerns the populated contents of the <category> element types in RSS (Section 3.3.2), it is also the responsibility of the publisher to ensure the correctness of their feed's content. Where our paradigms and their case studies are concerned, we: (1) do not propose any amendments to the format of RSS, (2) have not contacted the relevant feed publishers concerning the issues we have described: our corrective action was instead to remove from our RSS corpus those feeds affected by the issues described, and (3) have made no use of RSS's <category> element types in our classification work because of the potential ambiguity described in Section 2.2.3.

10.7.2 Keyword miscellany

Despite a *substantially* consistent set of results produced by our DT, MNB and SVM classifiers (Section 10.6.3), we were aware that the extensions to the keyword conventions and characteristics (Section 4.6) for our classification work (Section 10.4.1) led to the inclusion in our results of instances of the names of news agencies, slogans or *buzz* words, e.g. *reuters*, variations of *bbc world news* or *minute world news*. The following list identifies keyword-related issues that we found in our training/testing data (Section 10.4):

- **Repeated RSS feed postings:** Where a publisher posted identical <item> elements to their RSS feeds more than once on the same day resulting in double (or more) counting of keyword frequencies. Figure 10.9a illustrates an instance of this marginal, but unavoidable duplication in RSS feed 107, where the <title> and <description> elements carry the same content, but the <pubDate> elements list the different times at which the same content was repeatedly published on 21 Aug 2011. This is similar to the duplication caused by the absence of a populated <pubDate> element per <item> illustrated in Figure 10.8b.

- **Non-alphabet characters:** In our use of the product, Lucene (Section 7.7) broke hyphenated words into separate keywords, e.g. *blu-ray* became *blu* and *ray*. Similarly, when calculating keyword frequencies from the text of RSS feeds, Lucene automatically filtered commas and other punctuation characters. This caused *pseudo-keywords*, e.g. *mubarakgoeson* was produced from the text of the <description> element in RSS feed 175 on 02 Aug 2011 which contained text "CAIRO (Reuters)

- Egypt's fallen leader, Hosni Mubarak, goes on trial Wednesday over his role in killing protesters, in a stark message to Arab rulers elsewhere that they too may one day be held to account."

ID	<title>	<description>	<pubDate>	PollingDatetime
27634	Libya rebels push towards Tripoli	Bouyed by gains on Saturday, rebel forces push towards the ...	Sun Aug 21 10:53:16 BST 2011	2011-08-21 11:00:00
27637	Libya rebels push towards Tripoli	Buoyed by gains on Saturday, rebel forces push towards the ...	Sun Aug 21 11:41:50 BST 2011	2011-08-21 12:00:00
27641	Libya rebels push towards Tripoli	Buoyed by gains on Saturday, rebel forces push towards the ...	Sun Aug 21 12:34:31 BST 2011	2011-08-21 13:00:00
27646	Libya rebels push towards Tripoli	Buoyed by gains on Saturday, rebel forces push towards the ...	Sun Aug 21 13:05:27 BST 2011	2011-08-21 14:00:00

(a) Extract from RSS feed 107 on 21 Aug 2011 displaying repeating content.

ID	<title>	<description>	<pubDate>	PollingDateTime
27385	The Pentagon's Next Big Battle	Bloomberg Government Insider examines the hard choices faci...	Sat Sep 03 03:41:30 BST 2011	2011-09-03 04:00:00
27386	Top Goat: What the Pentagon Buys		Sat Sep 03 03:41:30 BST 2011	2011-09-03 04:00:00
27387	The Pentagon's Next Big Battle	Bloomberg Government Insider examines the hard choices faci...	Sat Sep 03 04:42:28 BST 2011	2011-09-03 05:00:00
27388	Top Goat: What the Pentagon Buys		Sat Sep 03 04:42:28 BST 2011	2011-09-03 05:00:00

(b) Extract from RSS feed 159 on 03 Sep 2011 potential *pseudo-keywords*.

ID	<title>	<description>	<pubDate>	PollingDateTime
16277	Western Libya rebels strike north towar...	SHALGHOUDA, Libya (Reuters) - Libyan rebels said they had ca...	Thu Aug 11 23:29:47 BST 2011	2011-08-12 00:00:00
16278	Libyan rebels capture part of Brega, pus...	SHALGHOUDA/BENGHAZI, Libya (Reuters) - Libyan rebels said ...	Fri Aug 12 00:54:52 BST 2011	2011-08-12 01:00:00

(c) Extract from RSS feed 175 on 12 Aug 2011 for keyword *libyan rebel*.

Figure 10.9: RSS feed `<item>` element and keyword issues: cf. *case study three*.

- **Spanning:** Had we allowed keywords to span `<item>` elements of RSS feeds and if they had proved popular enough, they could have biased the generation of valid keywords, e.g. Figure 10.9b displays `<title>` elements of RSS feed 159 which would have resulted in *pseudo-keywords* such as *battle top goat*.

- **Stemming:** We found that stemming can be affected by possession, e.g. *murdoch's* became *murdochs* if not stemmed. Similarly, *murdoch's* became *murdoch* if stemmed.

  A further issue with keyword stemming concerned pluralisation. In Figure 10.9c stemmed keywords were generated, and their frequencies were calculated, from the `<title>` and `<description>` elements of RSS feed 175, i.e. *Reuters news agency*, for 12 Aug 2011. Using these criteria, the calculated frequency for bigram *libyan rebels* was three after all the keywords were automatically converted to lowercase. However, the phrase *libya rebels* in the `<title>` element of `<item>` 16,277 was excluded from this frequency because *libyan* does not stem to *libya* by default using the implementation of Porter's *Snowball* algorithm (Appendix B.2) in Lucene (Section 10.4.6). In Section 10.6.3 we describe results of classifications using stemming to be consistently lower compared to those without it.

Although many of these issues could have been addressed by using a corrective ETL (Appendix B.2) to cleanse the text of our RSS feeds, and therefore improve the performance of our classifiers, doing so would have matched the *pragmatic* approach adopted by Thelwall et al. [398]. We did not do this because such an approach is contrary to the correspondence of our paradigms (Section 1.4.2), and also to the *proof of concept* nature of our classification work which did not require us to maximise our results.

10.7.3 Training and testing data segmentation

Section 10.4.3 describes the *static* rotation of segments of training/testing data for a classification based upon the duration of the tranche applied to it. We expect that if the segments were re-ordered, the results of a new classification based upon this re-ordering would be consistent with those produced by the original static rotation, if not identical to them. This expectation is predicated upon the fact that the underlying keyword frequencies for each of the ten pairs of training/testing data would remain unchanged by the re-ordering. Therefore the outputs of the sub-classifications and averaging of them (Section 10.6.1) would remain *substantially* the same as before.

Table 10.8 demonstrates this expectation, e.g. to take the fifth row, we see the test data set to segment nine which covers days 25 - 27, and the remaining days form the nine training segments: in the original ordering in Table 10.4 we see an identical sequence for row nine: similarly, the dates for the other segments remain as they were before.

Segments and days of tranche									
1: 01 -03	**2: 04 -06**	**3: 07 -09**	**4: 10 -12**	**5: 13 -15**	**6: 16 -18**	**7: 19 -21**	**8: 22 -24**	**9: 25 -27**	**10: 28 -30(31)**
TR	TR	TR	TR	TR	TE	TR	TR	TR	TR
TR	TE	TR	TR	TR	TR	TR	TR	TR	TR
TR	TR	TR	TR	TR	TR	TR	TR	TR	TE
TR	TR	TR	TE	TR	TR	TR	TR	TR	TR
TR	TR	TR	TR	TR	TR	TR	TR	TE	TR
TE	TR	TR	TR	TR	TR	TR	TR	TR	TR
TR	TR	TR	TR	TR	TR	TE	TR	TR	TR
TR	TR	TR	TR	TR	TR	TR	TE	TR	TR
TR	TR	TR	TR	TE	TR	TR	TR	TR	TR
TR	TR	TE	TR	TR	TR	TR	TR	TR	TR

Table 10.8: Re-ordered *segments* for generating training/testing data for a classification, *TR* denotes training data and *TE* (shown in red) refers to testing data: cf. *case study three*.

10.8 Afterword

This chapter has presented the classification component of our second paradigm. We began this chapter by defining the corpus of RSS feeds and categories, and subsequently discussed in detail the nature of our training/testing data: this also included details of the principal open-source, third-party products used. We have described the classifiers and process used to produce a *substantially* consistent set of classification results, despite the maximisation of these not being our purpose. In the same vein, we have also cited issues with the format of RSS and keywords, which if addressed, could improve the results of the classifications. This work presented in this chapter also validates the use of semi-automated batch processing of RSS feeds at category-level for our sentiment analysis work (Chapter 11).

Chapter 11

Case study three: Correlating keyword frequencies with sentiment in RSS feeds

11.1 Foreword

This chapter focuses on the final component of case study three for our second RSS-mining paradigm (Section 1.4.1) where we correlated changes in the keyword frequencies in RSS feeds with sentiment (Section 2.7). We begin with the objectives of this work in Section 11.2. Our selection of a sentiment analysis tool is discussed in Section 11.3, and the other *apparatus* of this work, i.e. the RSS feed and category corpus and its organisation, is the subject of Section 11.4.

Section 11.5 describes the algorithm we employed to produce raw keyword frequency/sentiment data. In connection with this, Section 11.6 focuses upon the integration into our software and implementation of the principal open-source, third-party products used for our sentiment analysis work: attention is paid here to the scoring of sentiment as well as linguistic issues. Section 11.7 describes the processing of the raw data and the selection from it of candidate keywords for correlation: this correlation is demonstrated in Section 11.8 by a set of time-series plots. Lastly, Section 11.9 provides an appraisal of the second component of our third case study.

11.2 Objectives

The specific objectives of the sentiment analysis component of case study three sought to determine a correlation between: (1) the sentiment contained in the text of RSS feeds, and (2) the fluctuations in the frequencies of keywords representing named entities in the feeds, and to visualise the results for trend analysis (Appendix B.2). Figure 11.1 provides an example of this correlation in a mock time-series (Section 2.8.3) plot where: (1) the red line represents the fluctuations in the aggregated frequencies of an arbitrary keyword, and (2) the blue line illustrates the average positive sentiment related to the keyword, in the postings to a particular RSS feed for ten days in Aug 2011.

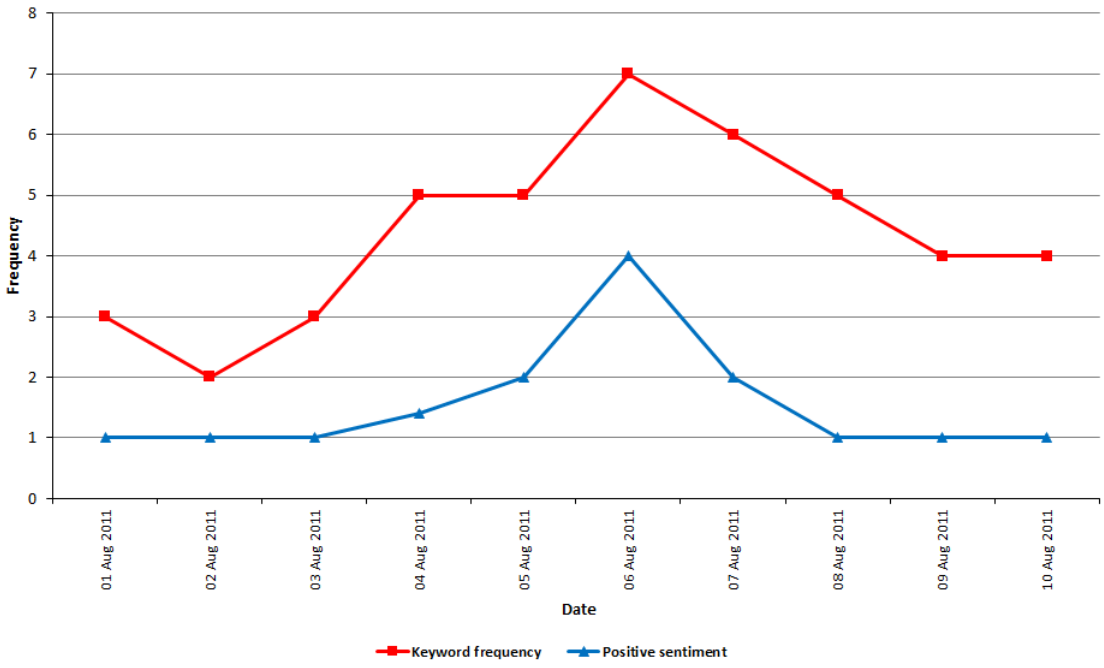


Figure 11.1: Mock time-series plot of positive keyword frequency/sentiment correlation: cf. *case study three*.

For our sentiment analysis work, we used semi-automated batch processing (Section 11.5) to produce raw keyword frequency/sentiment data. We then selected from this data a series of candidate keywords and made use of Pearson's correlation coefficient (Appendix B.2) to plot the correlation between the fluctuations in the frequencies of each keyword and the positive and negative sentiments related to these. It is necessary to state here that determining this *linear* correlation was the sole purpose of this component of case study three. We were not concerned with a deeper analysis of the underlying distribution of the keywords in the text of our RSS feeds: we regard this as a direction for the future

research work discussed in Section 13.7.1.

The aforementioned batch processing superseded a set of web pages that were originally developed for manual selection of RSS feeds and keywords for sentiment analysis, and to download or email results. These manual facilities were discontinued and consequently, we do not discuss them in this chapter, although they are referred to in Section 4.8.1. In addition to this, an example of the first page for selecting keywords and categories of RSS feeds is displayed in Section 4.8.4 where we comment upon the interface design of our vRSS software.

11.3 Sentiment analysis and sentiment analyser

Stated simply, sentiment analysis refers to determining the positive, neutral or negative sentiment in a piece of text, i.e. a document, sentence or aspect thereof. We do not provide a more detailed description of sentiment analysis, also called *opinion mining*, in this chapter because this subject is discussed in Section 2.7. Moreover, we review related RSS-based sentiment analysis work in Section 3.3.3. Nor did the black-box nature of our work require the development of a sentiment analyser: the software written for our sentiment analysis work comprised an extension of vRSS from case study two which is described, together with software characteristics of this, in case study three in Section 4.8.

In preference to products reviewed in recent surveys (Section 2.7), Weka (Section 10.5) or other data mining tools providing sentiment analysis facilities, we employed a Java-based sentiment analyser called *SentiStrength* (Appendix B.1), because of the product's ease of integration into the extension of our vRSS software. SentiStrength has also been used in related work including: (1) Nielsen [280] for comparing word lists used in sentiment analysis of microblogs, (2) Zhou et al. [466] for micro-blogging analysis, (3) Kucuktunc et al. [215] to analyse sentiment in Yahoo *Answers*, and (4) Giannopoulos et al. [139] for "diversifying user comments on news articles" according to such criteria as "similarity, sentiment expressed within comments, article's named entities also found within comments and commenting behavior of the respective users." Details concerning other applications and academic research employing SentiStrength are listed on the product's web site (Appendix B.1).

In Section 2.7, we explained *aspect*-level sentiment analysis. This approach focuses on a *word* or *feature* level in text to gauge positive or negative sentiment, and a target, i.e. the *subject* of the opinion. SentiStrength employs a *lexicon-based* approach in its adoption of aspect-level sentiment analysis, i.e. it uses a series of weighted word lists. We made

use of SentiStrength to produce sentiment related to keywords in the text of RSS feed `<title>` and `<description>` elements (Section 2.2.3).

According to the SentiStrength Java Manual [396], with regard to the "polarity detection on longer texts", e.g. the text of RSS feeds, SentiStrength can produce a *binary*, i.e. positive-negative, or a *trinary*, i.e. positive-neutral-negative, result. We discuss our use of the latter *scale* and the use of linguistics in SentiStrength in Section 11.6.2, as well as the product's *interface*, and its integration into the aforementioned extension of our vRSS software.

11.4 Apparatus

In this section we describe the use of RSS feeds and categories in the algorithm described in Section 11.5 to generate raw keyword frequency/sentiment data.

11.4.1 RSS feed and category corpus

For our sentiment analysis work, we employed the corpus of fifty RSS feeds organised into five categories[1] previously used for the classification component of case study three (Section 10.3.2). Although this corpus covered the three month period between Jul - Sep 2011, we focused upon the month of Aug 2011.

11.4.2 Tranche organisation

As with our classification work (Section 10.4.2), we partitioned our data into a series of tranches:

- **Tranche 1:** Fixed monthly period, i.e. 01 - 31 Aug 2011.
- **Tranche 2:** Fixed 10 day periods, i.e. 01 - 10, 11 - 20 and 21 - 31 Aug 2011.
- **Tranche 3:** Cross monthly 10 day periods, i.e. 27 Jul - 05 Aug 2011 and 27 Aug - 05 Sep 2011.
- **Tranche 4:** Cross monthly 30 day periods, i.e. 17 Jul - 15 Aug 2011 and 17 Aug - 15 Sep 2011.

[1] We made no use of the `<category>` element types in the format of RSS during our sentiment analysis work because of the potential ambiguity described in Section 2.2.3.

11.4.3 RSS feed categories

Within a given tranche, we made use of the categories of RSS feeds (Section 11.4.1) in two distinct ways:

1. **Single category:** Firstly, on an individual basis, each category was used to generate a *global* list of popular keywords from its feeds, thence each `<item>` in each feed in the category was tested for sentiment related to each keyword.

2. **All categories:** An all-categories analysis produced a *global* list of popular keywords from *all* of the RSS feeds in *all* of the categories, where each `<item>` in each feed in each category, was then tested for sentiment related to each keyword.

Each sentiment analysis was also repeated for the combinations of RSS feed elements in Section 11.4.4, to give an overall total of ninety-six individual analyses (Section 11.7.2).

11.4.4 RSS feed elements

Our sentiment analysis process employed RSS feed `<item>` elements in a similar manner to our classification work[2] described in Section 10.4.4, i.e.:

1. **One RSS feed element per `<item>` (TxD):** Where popular keywords were generated from the text of the `<title>` elements of the RSS feeds involved, and the `<description>` elements were used for keyword frequency calculation and sentiment analysis.

2. **Two RSS feed elements per `<item>` (TDxTD):** `<title>` and `<description>` elements were used to generate a set of popular keywords, and for the calculation of frequencies and analysis of sentiment.

11.4.5 Keywords and named entity recognition (NER)

In an extension of our keyword conventions and characteristics described in Section 4.6, we employed a basic NER (Appendix B.2) to identify *proper* nouns for use as keywords. Stated simply, each keyword had to be comprised entirely of uppercase characters, e.g. *BBC*, or satisfy one of three mutually exclusive alternatives: (1) where the first character of each word was in uppercase, e.g. *British Broadcasting Corporation*, (2) the first character of

[2] As per Section 10.4.4, the TxD and TDxTD notation describing the combinations of RSS feed elements is one of convenience: there is no mathematical or other significance inherent in it.

any given word was in uppercase, e.g. *according to British*, or (3) where the first character of the first word was in uppercase, e.g. *City of London*.

We made use of the final option because of its *inclusivity*. This option rejects *according to British* which consists of two lowercase words, both of which are also stop words (Appendix B.2), before a proper noun is encountered, but it accepts *British Broadcasting Corporation* because the leading character of the first word *British* is in uppercase: similarly, it accepts *City of London*. We allowed twelve keywords per *n-gram* type to the level of trigram, to be generated in our sentiment analyses (Section 11.7.2): the use of bi- and tri-grams in SentiStrength is described in Section 11.6.2. Use of stop words was allowed, but any keyword composed purely of them, e.g. *This is the*, was disallowed. Similarly, keyword stemming (Appendix B.2) was not permitted.

11.5 Algorithm

11.5.1 Order of presentation

In this section we describe the production of raw keyword frequency/sentiment data according to the apparatus described in Section 11.4. This involved using Quartz Scheduler (Section 4.7.4) to run the semi-automated batch processing and the re-use of Java objects from our classification work to store keyword frequencies (Section 4.8.3). The other principal open-source, third-party products used are described in Section 11.6.

11.5.2 Generating popular keywords

We do not describe or provide pseudocode for this stage of the sentiment analysis process. This is because it is practically identical to its classification counterpart (Section 10.4.5) to generate popular keywords for a tranche's duration. There are two exceptions: (1) there is no production of training or testing data, and (2) the need for additional keyword validation for NER and stop words described above.

11.5.3 Calculating keyword frequencies

Calculating keyword frequencies is very similar to the equivalent process in Section 10.4.5 for our classification work, but with the necessary extension for sentiment analysis. Pseudocode for this algorithm is listed in Algorithm 11.1, and the list below describes the principal data structures and variables used:

- ***day*:** A day of the *tranche*.
- ***globalKeywords*:** The collection of generated popular keywords for the *tranche* in a sentiment analysis: this collection is produced by the first stage of the sentiment analysis algorithm.
- ***rssFeed*:** An RSS feed belonging to an *rssFeedCategory*.
- ***rssFeedCategory*:** The category of feeds an *rssFeed* belongs to.
- ***rssFeedElement*:** Depending upon the combination of RSS feed elements being used, the collection of all of the `<title>` or `<description>` elements from every *rssFeedItem* in an *rssFeed* for a *day* of the *tranche*.
- ***rssFeedItem*:** The `<item>` elements of an *rssFeed* for a *day*.
- ***rssFeedDailyNgrams*:** The collection of n-grams and frequencies indexed from all *rssFeedElement* objects in an *rssFeed* for a *day*.
- ***tranche*:** The duration, i.e. number of days, of a sentiment analysis.

With reference to the line numbers in Algorithm 11.1, we begin with the RSS feed category or categories (Section 11.4.3), i.e. *rssFeedCategory*. For each *rssFeedCategory* (1), for every day in the tranche, for each *rssFeed* in the *rssFeedCategory* (3), every *rssFeedItem* in each *rssFeedElement* in the *rssFeed* (5) is read (7): this depends upon the combination of elements used (Section 11.4.4). When the text is read (8), it is sanitised[3] before its n-grams are indexed, i.e. n-gram frequencies are calculated (9) by Lucene (Section 11.6.1). Each n-gram (10) is then either added to *rssFeedDailyNgrams* (14), or if already present it is incremented (12).

Lines (16 - 28) are concerned with sentiment analysis, after which the algorithm continues as per classification. In line (16), if an n-gram from the index of *text* created in (9) is found in *globalKeywords*, we know that *text* contains an n-gram which can be used for keyword-based sentiment (Section 11.3). Two pre-requisite steps are needed before sentiment analysis can occur:

1. The need to concatenate the individual words of bi- and tri- grams to remove spaces is dictated by SentiStrength (Section 11.6.2). This is element *concatNgram* in the **if**...**endif** in lines (20 - 25).

[3]The sanitisation of the text of an *rssFeedItem* is *substantially* consistent with the description given in Section 10.4.1 with the exception of any variations required of the apparatus described in Section 11.4.

2. If the concatenation of a bi- or tri- gram occurs, then any instances of this keyword in the text of the `<item>` element to be analysed for sentiment must also be replaced with the concatenated version, i.e. element *concatText* in (22).

```
1: for each (rssFeedCategory) do
2:     for each (day in tranche) do
3:         for each (rssFeed in rssFeedCategory) do
4:             set rssFeedDailyNgrams ← null;
5:             for each (rssFeedElement) do
6:                 set text ← null;
7:                 for each (rssFeedItem in rssFeedElement) do
8:                     set text ← sanitise(rssFeedItem);
9:                     get index of ngrams in text;
10:                    for each (ngram in index) do
11:                        if (ngram in rssFeedDailyNgrams) then
12:                            increment ngram in rssFeedDailyNgrams;
13:                        else
14:                            add ngram to rssFeedDailyNgrams;
15:                        end if
16:                        if (ngram in globalKeywords) then
17:                            set concatNgram ← null;
18:                            set concatText ← null;
19:                            set result ← null;
20:                            if (ngram not unigram) then
21:                                set concatNgram ← concatenate(ngram);
22:                                set concatText ← replace(ngram in text with
                                   concatNgram);
23:                            else
24:                                set concatNgram ← ngram;
25:                            end if
26:                            set result ← analyse(sentiment of concatNgram
                               in concatText);
27:                            write result to database;
28:                        end if
29:                    end for
30:                end for
31:            end for
32:            for each (ngram in rssFeedDailyNgrams) do
33:                if (ngram in globalKeywords) then
34:                    write ngram to database;
35:                end if
36:            end for
37:        end for
38:    end for
39: end for
```

Algorithm 11.1: Pseudocode of the second stage of the sentiment analysis algorithm: cf. *case study three.*

(26) represents the occurrence of the analysis by SentiStrength, i.e. analysing *concatText* for sentiment related to *concatNgram*: the *result* of this analysis is then *persisted* (Appendix B.2) to the database (27). The outputs of the algorithm are considered in Section 11.7 after we describe the use of SentiStrength in vRSS in Section 11.6.2. Lines (32 - 35) concern the writing to the database of the daily frequencies of those n-grams which sentiment was actually related to, i.e. unigrams, and bi- and tri- grams concatenated to *concatNgram* in lines (20 - 25), as they were found in $globalKeywords$.

11.6 Third-party tools

11.6.1 Customising Lucene

Our previous use of Lucene to calculate keyword frequencies is described in Section 7.7 for case study two, and in Section 10.4.6 for the classification component of case study three. Both of these instances employed lowercase keywords which made use of the default conversion of uppercase to lowercase performed in Lucene's `StandardAnalyzer`. In order to use uppercase in our sentiment work, vRSS's use of Lucene required: (1) the customisation of Lucene's `StandardAnalyzer` and `SnowBallAnalyzer` by disabling case conversion code, and (2) adjusting vRSS's `Text_Stemmer_Indexer` class to accommodate the customised *analyzers*, now called `StandardUpperCaseAnalyzer` and `SnowBallUpperClassAnalyzer` respectively. We do not reproduce any code concerning these customisations in this thesis.

11.6.2 Using SentiStrength

Interface

Several versions of SentiStrength are available for use. These include: (1) the product's web site (Appendix B.1), (2) an interactive Microsoft Windows [263] *executable* file, and a pre-compiled Java library for programmatic use. Whichever version is used, SentiStrength has been described in Thelwall et al. [397] as:

> "a lexicon-based classifier that uses additional (non-lexical) linguistic information and rules to detect sentiment strength in short informal English text. For each text, the SentiStrength output (for both version 1 and version 2) is two integers: 1 to 5 for positive sentiment strength and a separate score of 1 to 5 for negative sentiment strength. Here, 1 signifies no sentiment and 5 signifies strong sentiment of each type. For instance, a text with a score of 3, 5 would contain moderate positive sentiment and strong negative sentiment. A neutral

text would be coded as 1, 1. Two scales are used because even short texts can contain both positivity and negativity and the goal is to detect the sentiment expressed rather than its overall polarity."

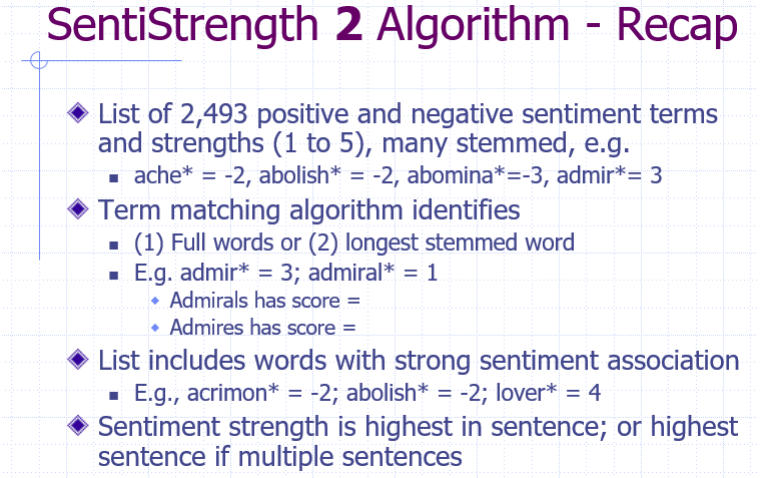


Figure 11.2: The SentiStrength algorithm (reproduced from [395]): cf. *case study three*.

Figure 11.2, which is reproduced from the materials for course *Sentiment strength detection for the Social Web* [395], provides an overview of the algorithm used by SentiStrength. A more concise description is given by Thelwall [394]:

> "When SentiStrength reads a text, it splits it into words and separates out emoticons and punctuation. Each word is then checked against the lexicon for matching any of the sentiment terms. If a match is found then the associated sentiment score is retained. The overall score for a sentence is the highest positive and negative score for its constituent words and for multiple sentences and [*sic*] the maximum scores of the individual sentences is taken."

According to Thelwall [394], the word scores can be modified by: (1) "a list of emoticons together with human-assigned sentiment scores", (2) a "list of idioms with sentiment strength weights." Moreover, SentiStrength "incorporates a number of rules to cope with special cases."[4] Table 11.1 lists the sentiment methods, reproduced from the *Sentiment strength detection for the Social Web* [395] course materials, employed by SentiStrength.

[4]SentiStrength (Appendix B.1) maintains a series of text files which contain weighted word lists for its lexicon and other sentiment methods. For our purposes, these files were placed within the folder structure of our vRSS software.

Method	Example
sentiment word strength list	terrify=-4
spelling corrected	nicce -> nice
booster words alter strength	very happy
negating words flip emotions	not nice
repeated letters boost sentiment/+ve	niiiice
emoticon list	:) =+2
exclamation marks count as +2 unless ve	hi!
repeated punctuation boosts sentiment	good!!!
negative sentiment ignored in questions	h8 me?

Table 11.1: Summary of sentiment methods employed by SentiStrength (reproduced from [395]): cf. *case study three*.

Linguistics

In Thelwall [394] reference is made to experimental work evaluating SentiStrength. Thelwall concluded that "SentiStrength has near-human accuracy on general short social web texts but is less accurate when the texts often contain sarcasm, as in the case of political discussions. The accuracy of SentiStrength can be enhanced by extending its lexicon and altering its mood setting for sets of texts with a narrow topic focus. As the case studies illustrate, SentiStrength can be used to analyse large scale sentiment patterns in the social web in addition to its commercial uses."

Table 11.1 lists the "grammatical information" [394] used by SentiStrength, although the product "does not attempt to use grammatical parsing (e.g., part of speech tagging) to disambiguate between different word senses. This is because it is designed to process very informal text from the social web and so, unlike typical linguistic parsers, does not rely upon standard grammar for optimal performance."

Our selection of candidate keywords for correlation is described in Section 11.7.4. Despite the use of a basic NER (Section 11.4.5), and the concatenation of bi- and tri- grams described below, this selection made no provision for linguistic context or semantics such as irony or sarcasm, beyond SentiStrength's own aforementioned "grammatical information" [394].

Outputs

All versions of SentiStrength produce the same type of output when analysing text for sentiment. To illustrate this, according to the apparatus described in Section 11.4, we re-use our previous example with concatenated text *I love StarTrek but I really hate StarWars*, and the concatenated keyword list of *StarTrek* and *StarWars*. This results in:

```
3 -5 -1 I love[3]StarTrek but I really hate[-4][-1 booster word]StarWars
[sentence: 3,-5] [result: max + and - of any sentence]
```

3 -5 -1 provides an overall summary of the results, i.e. a positive sentiment of 3 is recorded for the word *love* in relation to *StarTrek*, and a negative rating of -5 for the use of the word *hate* in connection with *StarWars*. The negative rating for *StarWars* is calculated from an initial rating of -4, given to the word *hate* and a further -1 because of the *booster* word *really*. The final number in the sequence 3 -5 -1 is the neutral rating, i.e. 3 - 4 = -1. In a longer piece of text consisting of several sentences, the result of an analysis is based upon the maximum and minimum sentiment found, i.e. [result: max + and - of any sentence].

Reference is made in Section 11.3 to our use of SentiStrength's *trinary* scale for "polarity detection on longer texts", such as the text of RSS feeds: the SentiStrength Java Manual [396] further states that this scale works in similar fashion to Taboada's *SOCAL* program [381], where "the total positive sentiment is calculated and compared to the total negative sentiment. If the total positive is bigger than 1.5* the total negative sentiment then the classification is positive, otherwise it is negative. Why 1.5? Because negativity is rarer than positivity, so stands out more (see the work of Maite Taboada)."

Implementation

In order to implement SentiStrength, we incorporated the aforementioned pre-compiled Java version of the product directly into our extended vRSS software (Section 4.8.1).

Algorithm 11.2 lists the basic Java code we implemented in vRSS in order for SentiStrength to produce a *trinary*, i.e. positive-neutral-negative, result for "polarity detection on longer texts" [396]. Line (1) contains a comma-separated list of keywords that sentiment will be based upon. (2) sets the parameters for the analysis, i.e. *explain* will add an explanation to the results of the analysis, *trinary* concerns the *scale* of the result, and *keywords* is the prefix of the comma-separated string of keywords in (1). For example, using our previous concatenated text *I love StarTrek but I really hate StarWars*, the concatenated keyword list (1) consists of *StarTrek* and *StarWars*: therefore, in the sentiment

analysis, the word *love* relates to *StarTrek* but *hate* refers to *StarWars*.[5] (3) and (4) instantiate an instance of SentiStrength and set its parameters to execute the analysis: this action calls SentiStrength's method `computeSentimentScores` (5), *seeded* with the concatenated text previously extracted from RSS feeds. The *trinary* results are acquired in lines (6 - 9), after which they are persisted to database storage.

```
1: String keywords = ( /* Keywords previously generated in algorithm. */ );

   // Initialise parameters.
2: String[] pars = {"sentidata", "explain", "trinary", "keywords", keywords};

   // Instantiation of SentiStrength.
3: SentiStrength ss = new SentiStrength();
4: ss.initialise(pars);

   // Analyse text.
5: String explanation = ss.computeSentimentScores( /* Text previously extracted
                                                      from RSS feeds. */ );

   // Get results.
6: String[] results = explanation.split(" ");
7: int positiveResult = Integer.parseInt(results[0]);
8: int neutralResult = Integer.parseInt(results[2]);
9: int negativeResult = Integer.parseInt(results[1]);
```

Algorithm 11.2: Use of SentiStrength in visualRSS: cf. *case study three*.

Use of bi- and tri- grams

The parameter permutation supplied to SentiStrength for an analysis in line (2) of Algorithm 11.2, includes the value *keywords* for keyword-based sentiment. Although, we have used *keywords* for this purpose, it is germane to state the product was designed only for unigrams. We know this from an email conversation between Professor M. Thelwall, i.e. SentiStrength's writer, and the author on 05 May 2015:[6] to quote the former:

> "I am sorry but this is only possible for unigrams. Perhaps as a workaround you could preprocess the texts to convert the phrases into words. For example *Star Wars* becomes *StarWars*, and then use *StarWars* as the keyword? Is this practical for you?"

As suggested, it was necessary to concatenate the individual words of bi- and tri- grams to remove spaces, e.g. *City of London* became *CityofLondon*, similarly golfer *Tiger* Woods became *TigerWoods*, to permit their use as keywords in SentiStrength.

[5] According to the SentiStrength Java Manual [396], parameters to "classify text near specified keywords" have default values of four.

[6] The full transcript of this conversation is available from the author upon request.

11.7 Post-algorithm data processing

11.7.1 Sentiment analysis 71

In describing the next stages of our sentiment analysis work, we use bigram *Tiger* Woods as a keyword to represent golfer Eldrick Tont *Tiger* Woods, in RSS feed 133, i.e. *Reuters* news agency's feed for sport, from sentiment analysis 71. This analysis employed the feeds in the Sport RSS feed category (Section 11.4.1) for a ten day duration of tranche 2 between 01 - 10 Aug 2011 (Section 11.4.2). Both `<title>` and `<description>` elements of RSS feed 133 were used for generating popular keywords and calculating their frequencies.

11.7.2 Raw keyword frequency/sentiment data

We ran a total of ninety-six individual sentiment analyses. Of this number, eighty of the analyses were for a single RSS feed category and the remaining sixteen were for all categories. We arrive at the total of eighty for the single category analyses with reference to the apparatus described in Section 11.4. Although four tranches were used, tranches 2, 3 and 4 included several durations of different date ranges. Therefore, with a total of eight durations overall, for five RSS feed categories, we have forty analyses: extending this to include the two combinations of RSS feed elements, we have eighty analyses. To summarise: eight durations for the tranches, multiplied by five RSS feed categories, multiplied by two RSS feed element combinations, i.e. ((1 + 2 + 3 + 2) * 5) * 2 = 80. A further analysis was also run for each duration including all five RSS feed categories: therefore, allowing for eight durations and two RSS feed element combinations, we have another sixteen analyses.

For every analysis, the algorithm described in Section 11.5 produced raw keyword frequency/sentiment data based upon its permutation of the apparatus described in Section 11.4, i.e. where according to the combination of RSS feed elements used in each analysis, the instances of these elements in every `<item>` in every RSS feed in the category or categories used were, for each day of the tranche, analysed for sentiment relating to every popular keyword generated.

Table 11.2 lists a sample of the raw keyword frequency/sentiment data of sentiment analysis 71, where each row: (1) corresponds to either a `<title>` or `<description>` element of an `<item>` posted to RSS feed 133, i.e. *Reuters* news agency, on 04 Aug 2011 containing the text *Tiger* Woods, and (2) lists SentiStrength's explanation for the analysis of sentiment relating to *Tiger* Woods in the text of the element. Text concerning *Reuters* and generic SentiStrength result text, e.g. [result: max + and - of any sentence], has been

edited from the explanations for clarity. Moreover, in Table 11.2 we also see an example of our concatenation of bi- and tri- gram keywords for SentiStrength (Section 11.6.2).

Row No	**Explant-ion**	**Published date/time**	**Keyword frequency**	**Sentiment**		
				Pos've	**Neu'l**	**Neg've**
1		04 Aug 2011 @ 02 24 17	1	1.00	-1.00	-4.00
	1 -1 -4 Motivated *TigerWoods* a 'scary' prospect on [sentence: 1,-4]					
2		04 Aug 2011 @ 02 24 17	1	1.00	0.00	-1.00
	1 -1 0 *TigerWoods* may have been sidelined [sentence: 1,-1]					
3		04 Aug 2011 @ 20 07 52	1	1.00	0.00	-1.00
	1 -1 0 on Thursday shortly before *TigerWoods* teed off to make [sentence: 1,-1]					
4		04 Aug 2011 @ 21 14 48	1	1.00	0.00	-1.00
	1 -1 0 on Thursday shortly before *TigerWoods* teed off to make [sentence: 1,-1]					
5		04 Aug 2011 @ 21 35 06	1	3.00	1.00	-1.00
	1 -1 0 *TigerWoods* flashed his trademark smile [sentence: 3,-1]					

Table 11.2: Raw keyword frequency and sentiment analysis outputs for concatenated keyword *TigerWoods* in RSS feed 133 on 04 Aug 2011 (edits have removed references to *Reuters* news agency and generic sentiment analysis result text): cf. *case study three*.

11.7.3 Aggregation

We next aggregated the raw keyword frequency/sentiment data per sentiment analysis. Daily frequencies for each keyword per RSS feed were added together whilst the positive, neutral and negative sentiments were averaged. Table 11.3 lists the daily aggregations of `<item>` elements of RSS feed 133 for the period 01 - 10 Aug 2011 for *Tiger* Woods in sentiment analysis 71. The contents of Table 11.2 form the row for 04 Aug 2011 in Table 11.3, i.e. the addition of the keyword frequencies give a result of five, and the average of the positive sentiment over the five rows gives (1.00 + 1.00 + 1.00 + 1.00 + 3.00) / 5 = 1.40. Figure 11.3 provides a time-series plot of the same data.

11.7.4 Candidate keyword selection

Following aggregation, for each sentiment analysis we calculated the total number and percentage of days each of its keywords appeared in each RSS feed for the duration of the tranche for the analysis. This principle formed the basis for our selection of candidate keywords for correlation with sentiment, because we could identify individual instances of keywords according to the apparatus (Section 11.4) used to produce the raw keyword frequency/sentiment data for any analysis. Nevertheless, in doing this many keywords

were routinely discarded because of: (1) the effects of the keyword miscellany[7] described in Section 10.7.2, and where (2) tests of random batches of ten keywords revealed the inclusion of 40 - 50.00% of "non-useful terms" (Thelwall et al. [398]).

Date	Keyword frequency	Sentiment		
		Positive	Neutral	Negative
01 Aug 2011	1	1.00	0.00	-1.00
02 Aug 2011	2	1.50	0.00	-1.00
03 Aug 2011	3	1.00	-0.33	-2.00
04 Aug 2011	5	1.40	0.00	-1.60
05 Aug 2011	4	2.00	1.00	-1.00
06 Aug 2011	7	1.43	1.00	-1.86
07 Aug 2011	2	2.00	1.00	-1.00
08 Aug 2011	1	1.00	0.00	-1.00
09 Aug 2011	3	1.00	-0.67	-1.67
10 Aug 2011	4	1.00	-0.25	-1.25

Table 11.3: Tabular representation of aggregated daily keyword frequencies and averaged sentiment of keyword *Tiger* Woods in RSS feed 133 for the period 01 - 10 Aug 2011: cf. *case study three*.

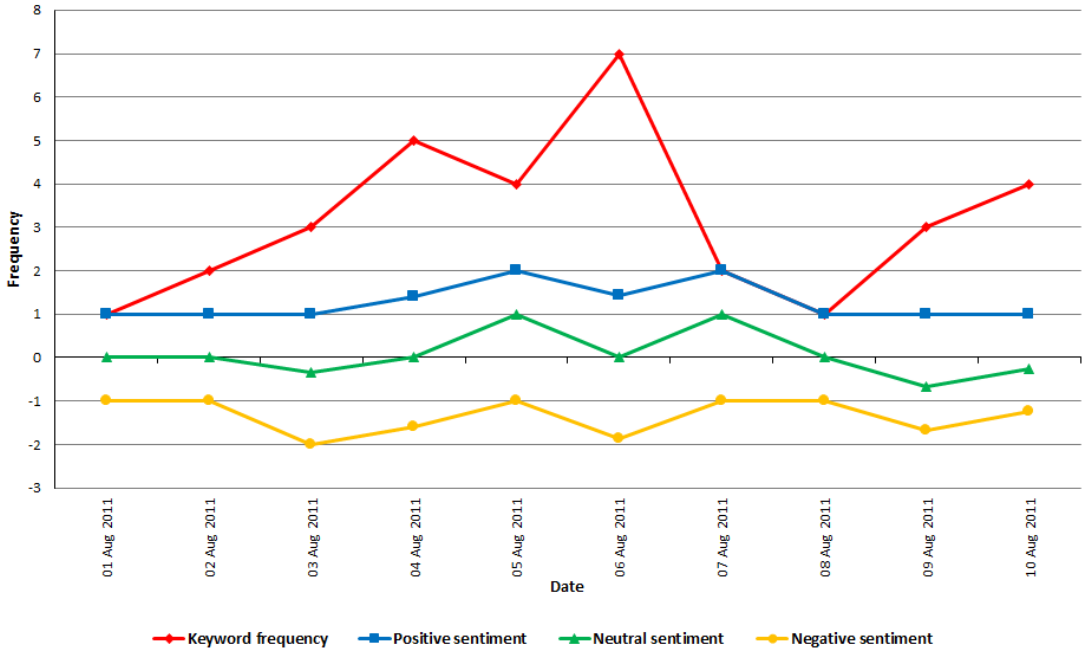


Figure 11.3: Time-series representation of aggregated daily keyword frequencies and averaged sentiment of keyword *Tiger* Woods in RSS feed 133 between 01 - 10 Aug 2011: cf. *case study three*.

[7]The third and fourth rows of Table 11.2 provide an example of this *keyword miscellany* where repeating content has been published in RSS feed 133 on 04 Aug 2011.

The majority of our candidate keywords comprised valid *proper* nouns such as: *European Stocks* (BFE), *Dark Knight Rises* (EA), *Afghan* (NCA), *Samsung Electronics* (SNT) and *Arsenal* (Sport), where respective RSS feed categories are included in parentheses. An extract from our final selection of candidate keywords is given in Appendix A.3.3: the list below *substantially* defines the selection criteria employed:

- **Translations of names:** We treated *Muammar Gaddafi* and *Moammar Gadhafi* as alternative bigrams.
- **Personal names:** References to a surname or full name identified individual people, e.g. *Kardashian* for *Kim Kardashian*, or *Steve Jobs* but not *Steve*. We also: (1) applied this principle to the use of titles where *President Barack Obama* was used as a distinct trigram, and (2) included titles such as *Prime Minister* or *President* as distinct keywords.
- **Names of political or corporate entities:** We employed full names such as *European Central Bank* in place of *European Central*.
- **Keyword stemming:** The non-use of stemming (Appendix B.2) resulted in keywords such as *Libya* and *Libyan* being treated distinctly.

We can see in Table 11.3 that keyword *Tiger* Woods appeared in RSS feed 133 every day of the tranche for sentiment analysis 71. Therefore, given this 100% percentage inclusion margin, *Tiger* Woods became a candidate keyword. A second example of a candidate keyword is given by unigram *Apple* in RSS feed 148, i.e. IT news web site *Eweek*, in sentiment analysis 102 which covered the full month of Aug 2011, i.e. tranche 1, and employed the `<title>` and `<description>` elements of the RSS feeds in the Science, nature and technology (SNT) category for generating popular keywords and calculating their frequencies (Section 11.4.4).

11.8 Results

11.8.1 Perspective

In Section 11.8.4 we provide a set of four time-series plots, and a further eight are included in Appendix A.3.4. Each of the twelve plots illustrates an example of our keyword frequency/sentiment results. We employed the Apache Commons Mathematics Library (Appendix B.1) to calculate Pearson correlations (Appendix B.2), i.e. for each candidate

keyword, positive, neutral and negative correlations were calculated. With reference to the two aggregation examples described above, *Tiger* Woods for RSS feed 133 for every day of sentiment analysis 71 resulted in a positive Pearson coefficient of 0.28, and *Apple* gave a positive coefficient of 0.53 for RSS feed 148 in sentiment analysis 102 for Aug 2011.

11.8.2 Expectations

Predicated upon raw keyword frequency/sentiment data (Section 11.7.2) and the apparatus of RSS feeds, categories and elements, and differing tranche durations of days (Section 11.4), our expectations of correlation amongst our ninety-six analyses were that there would be no *median* result according to these criteria or subsets thereof. We, more plausibly, expected individual candidate keywords to demonstrate either a positive or negative correlation, based upon the aforementioned apparatus and the stories contained therein.

11.8.3 Plot criteria

Keywords and RSS feed categories

The keyword frequency/sentiment time-series plots below, and in Appendix A.3.4 are concerned with the aggregated frequencies of a single candidate keyword and either the averaged positive or negative sentiment from a single RSS feed. Moreover, both ranges are normalised, and neutral sentiment (Section 11.3) has been edited from the plots for clarity.

We have used a single candidate keyword from a single RSS feed in our plots because the majority of the ninety-six individual analyses making up our raw keyword frequency/sentiment data (Section 11.7.2) were based upon a single RSS feed category: therefore, in each analysis the candidate keyword originated from one of the feeds in that category. Where a candidate keyword from one of the sixteen analyses for all fifty RSS feeds and five categories in our corpus (Section 11.4.1) is illustrated in a plot, only the positive or negative sentiment for that keyword, in a single RSS feed in one of the RSS feed categories in the analysis, is displayed. For example, the plot for keyword *President* (Figure 11.5) displays the frequency and sentiment in RSS feed 107 for the thirty day tranche from 17 Jul - 15 Aug 2011 of the particular sentiment analysis, although the keyword *President* and related sentiment occurred in thirty different RSS feeds within the same analysis.

Appendix A.3.3 lists an extract from our final candidate keyword selection together with the RSS feed and category displayed in each plot. The twelve plots in Section 11.8.4 and Appendix A.3.4 represent a cross-section of the candidate keyword selection to illustrate different patterns of raw keyword frequency/sentiment data and alternative

permutations of the apparatus of RSS feeds, categories and elements, and differing tranche durations of days described in Section 11.4.

Normalisation

In each time-series plot, the aggregated keyword frequencies and averaged sentiment have been normalised to display values between 0 and 1, or 0 and -1, depending upon the type of sentiment displayed. This re-scaling has been calculated using the following formula:

$$z_i = \frac{x_i \text{ - } \min(x)}{\max(x) \text{ - } \min(x)}$$

Therefore, the majority of the time-series plots consist of two graphs: (1) the first graph displays the fluctuations in the aggregated keyword frequency and the averaged positive sentiment, and (2) the second graph plots the average negative sentiment and the aggregated but *inverted*, i.e. multiplied by -1, keyword frequency.

Plot annotations

Each time-series plot below, and in Appendix A.3.4, is annotated as follows:

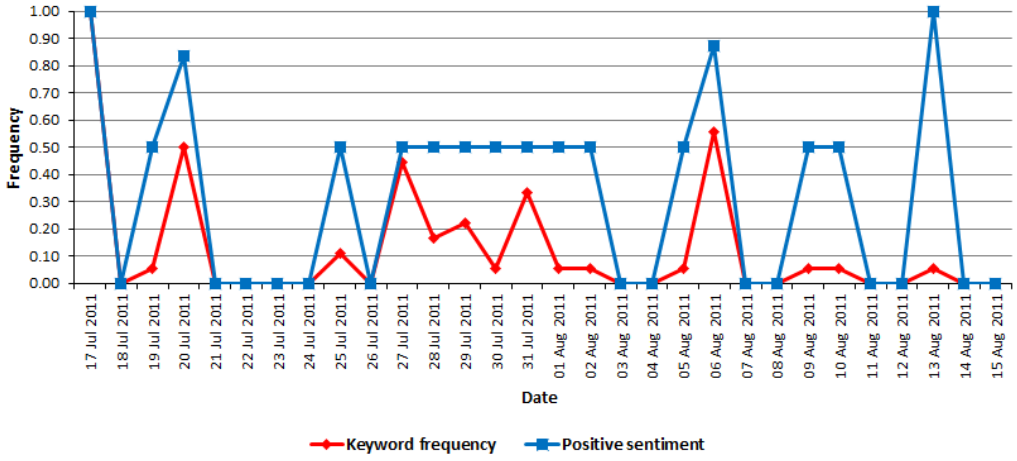
- **Keyword:** The candidate keyword.
- **N-gram length:** The length of the keyword's n-gram type.
- **RSS feed number:** The RSS feed from which the keyword was taken.
- **RSS feed category:** The category of the RSS feed in the sentiment analysis or *All*, if an analysis involved all categories in our corpus.
- **RSS feed elements:** As Section 11.4.4.
- **Duration:** The number of days an analysis ran for, i.e. the duration of the tranche (Section 11.4.2).
- **Margin:** The percentage of the number of days in the duration the keyword appeared in the RSS feed.
- **Correlation:** Pearson correlation(s) for the averaged sentiment.
- **Comments:** A brief explanation of the plot.

11.8.4 Keyword frequency/sentiment correlation plots

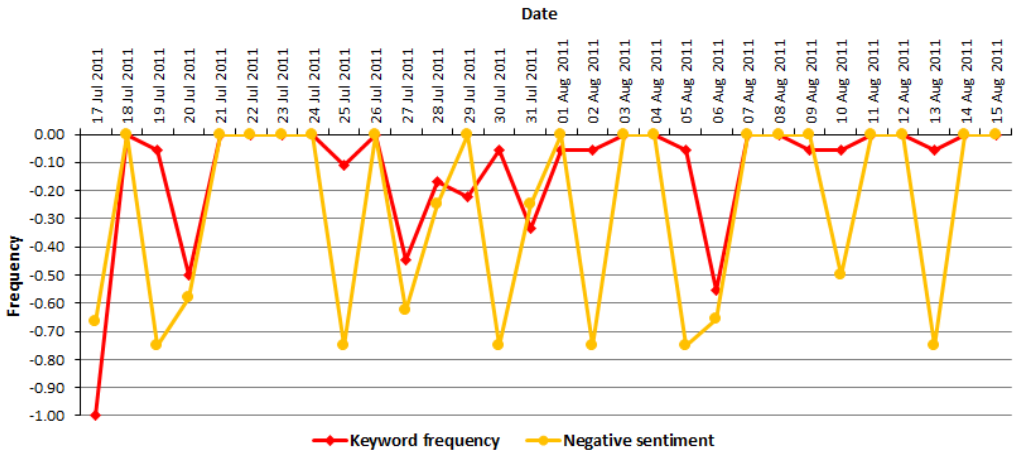
Keyword: *Afghan*

N-gram length	1
RSS feed	Number: 107, Category: NCA, Elements: TxD
Duration	Dates: 17 Jul - 15 Aug 2011, Number of days: 30, Margin 53.33%
Correlation	Positive: 0.70, Negative: -0.33
Comments	Negative sentiment regarding events in Afghanistan.

Table 11.4: Keyword: *Afghan*: cf. *case study three.*



(a) Positive sentiment.



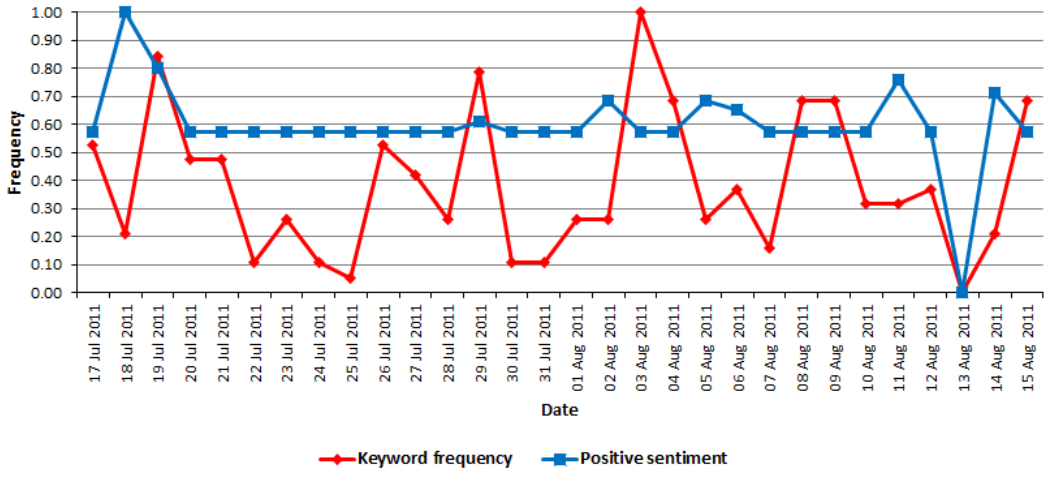
(b) Negative sentiment.

Figure 11.4: Keyword: *Afghan*: cf. *case study three.*

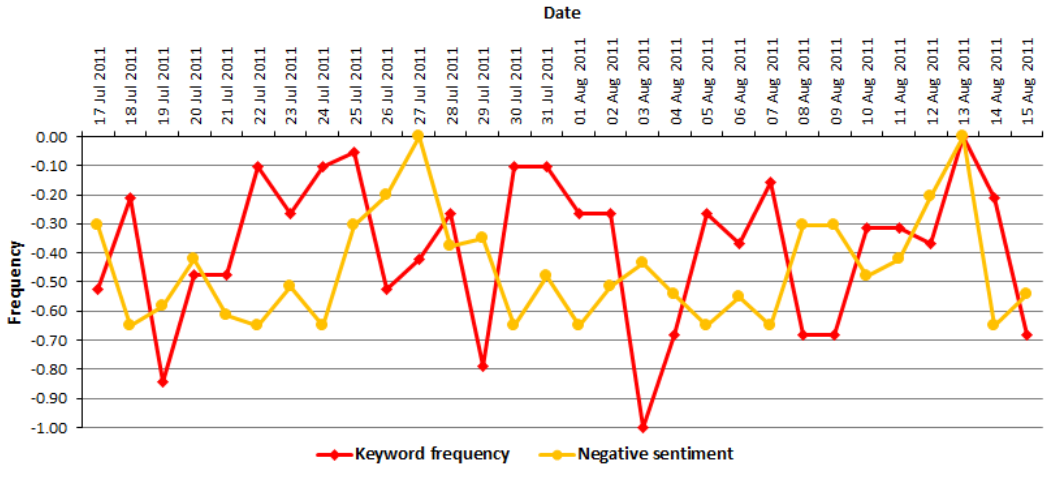
Keyword: *President*

N-gram length	1
RSS feed	Number: 107, Category: All, Elements: TDxTD
Duration	Dates: 17 Jul - 15 Aug 2011, Number of days: 30, Margin 96.67%
Correlation	Positive: 0.18, Negative: -0.41
Comments	Negative sentiment for various presidents globally.

Table 11.5: Keyword: *President*: cf. *case study three.*



(a) Positive sentiment.



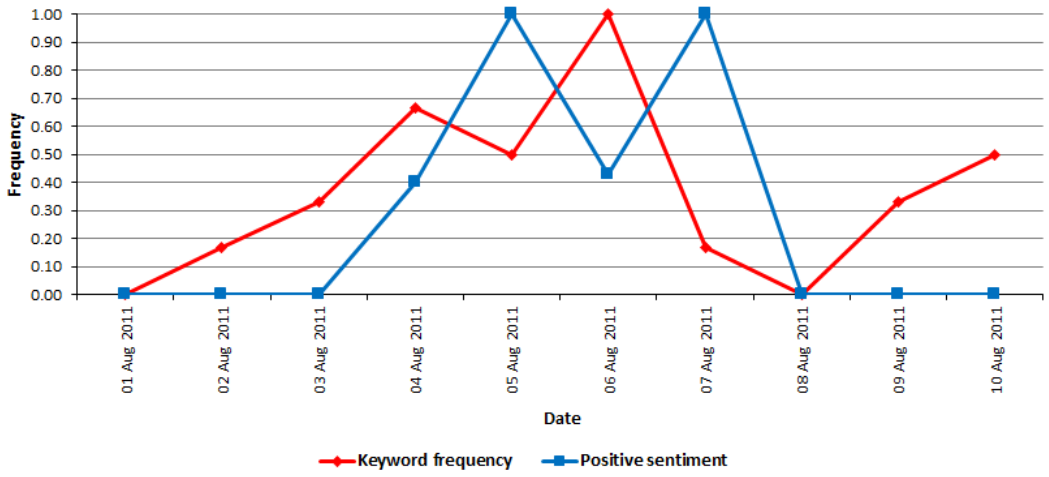
(b) Negative sentiment.

Figure 11.5: Keyword: *President*: cf. *case study three.*

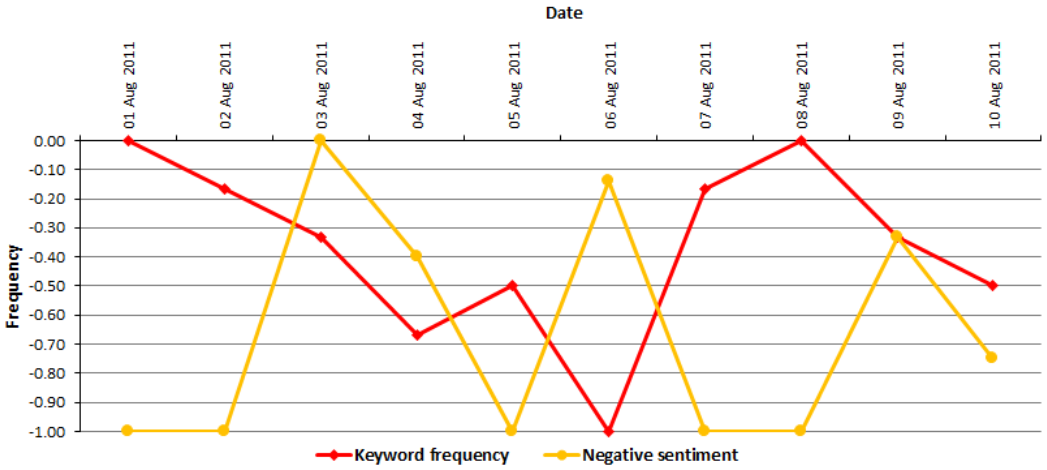
Keyword: *Tiger* Woods

N-gram length	2
RSS feed	Number: 133, Category: Sport, Elements: TDxTD
Duration	Dates: 01 - 10 Aug 2011, Number of days: 10, Margin 100.00%
Correlation	Positive: 0.28, Negative: -0.62
Comments	Balanced sentiment at the 2011 WGC-Bridgestone golf tournament.

Table 11.6: Keyword: *Tiger* Woods: cf. *case study three*.



(a) Positive sentiment.



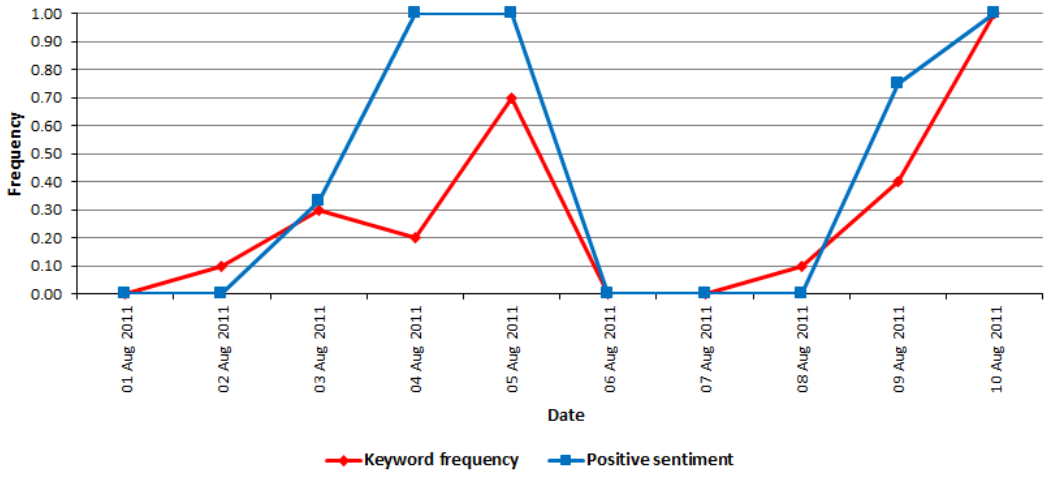
(b) Negative sentiment.

Figure 11.6: Keyword: *Tiger Woods*: cf. *case study three*.

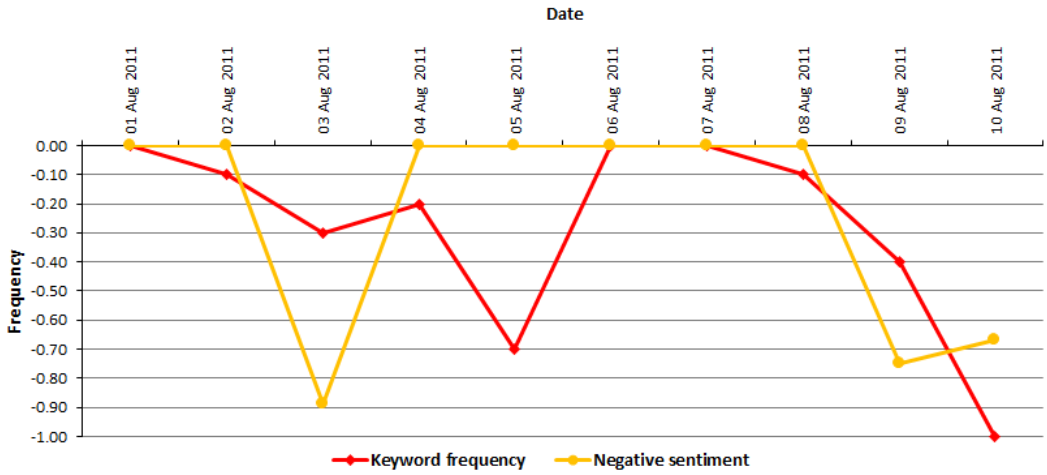
Keyword: *Wall Street*

N-gram length	2
RSS feed	Number: 121, Category: All, Elements: TDxTD
Duration	Dates: 01 - 10 Aug 2011, Number of days: 10, Margin: 70.00%
Correlation	Positive: 0.81, Negative: -0.50
Comments	Stock values fall on Wall Street in early Aug 2011.

Table 11.7: Keyword: *Wall Street*: cf. *case study three.*



(a) Positive sentiment.



(b) Negative sentiment.

Figure 11.7: Keyword: *Wall Street*: cf. *case study three.*

11.9 A posteriori appraisal of the sentiment analysis component of case study three

11.9.1 Patterns of correlation

The plots in Section 11.8.4 and Appendix A.3.4 demonstrate that the correlation determined between the sentiment and the fluctuations in the frequencies of keywords representing named entities in RSS feeds, can be visualised in time-series plots. Furthermore, the plots represent a cross section of our final selection of candidate keywords (Appendix A.3.3), and illustrate that keyword frequency/sentiment correlation can occur in different patterns. We see examples of these patterns in the varying lengths of duration of 10 or 30(31) days, and in the alternative combinations of RSS feed elements used, i.e. *Kardashian* in Figure A.10. The plots also employ specific and generic keywords referring to an individual or to a title, e.g. *President Barack Obama* (Figure A.6) and *President* (Figure 11.5) respectively. In addition, two of the plots are based upon analyses covering multiple subject areas because of their use of all of the RSS feed categories in our corpus (Section 11.4.1). The following sections of this appraisal identify constraints in our approach to keyword frequency/sentiment correlation.

11.9.2 Keyword temporality

We note the presence of *gaps* in our keyword frequency/sentiment plots to do with the absence of keywords. An example of this is provided by Figure 11.4 for keyword *Afghan*, where the gaps include the four day interval 21 - 24 Jul 2011. The gaps in this and other examples are present because of: (1) keyword stemming, (2) the absence of postings to the appropriate RSS feed for the period in question or for the particular named entity, or (3) a lack of postings from which sentiment can be produced. With reference to keyword *Afghan*, the following two examples illustrate the first and third reasons:

1. Figure 11.4 records no data for RSS feed 107 for period 03 - 04 Aug 2011. Two postings were actually recorded by vRSS for this period: their respective `<description>` elements read:

    - "One Nato soldier in eastern Afghanistan is shot dead by a man in police uniform, while another is killed in a militant attack, officials say."
    - "Afghans get new satire inspired by hit UK comedy".

In the first posting, keyword *Afghan* is not present as a unigram despite its forming the *root* of word *Afghanistan*, and in the second posting, our non-use of stemming precluded *Afghans* which would otherwise have stemmed to *Afghan*.

2. According to Figure 11.7, no postings were made to RSS feed 121 on the weekend of 06 - 07 Aug 2011 for keyword *Wall Street*.

11.9.3 Keyword relatedness

In each of the keyword frequency/sentiment plots in Section 11.8.4 and Appendix A.3.4, the positive and negative sentiment correlations refer to a single candidate keyword drawn from a single RSS feed, as described in Section 11.8.3. Therefore, the plots do not allow any form of keyword *relatedness* (Section 4.6), nor do the candidate keywords (Section 11.7.4) allow for the "grammatical information" [394] described in Section 11.6.2.

11.9.4 Other issues

- **Seasonal variations:** Our corpus of RSS feeds and categories is circa Aug 2011. Summer is traditionally a passive, *holiday* period in certain industries in comparison with autumn and winter, e.g. arts and entertainments.

- **SentiStrength:** The default scores used by SentiStrength (Section 11.3) for the lack of positive or negative sentiment in a piece of text are 1 and -1 respectively. These are, in the author's opinion, susceptible of misinterpretation, i.e. either score could be seen as indicating a weak sentiment where none actually exists.

- **Uniformity:** The absence of a basis for a comparative analysis to gauge the success or otherwise of the correlations for each n-gram (Appendix B.2) type used, where we plausibly expected individual candidate keywords to demonstrate either a positive or negative sentiment (Section 11.8.2).

11.10 Afterword

In this chapter, we have found that a correlation between sentiment in the text of RSS feeds and the fluctuations in the frequencies of keywords can be determined. We have described the apparatus and process employed, and have employed of a set of time-series plots to illustrate different keyword frequency/sentiment patterns of correlation: in appraising this correlation, we have also identified several applicable constraints. Nevertheless, the two components of case study three do complement each other, e.g. in the development of the regular semi- or fully- automated batch processing facility to classify and analyse RSS data for sentiment referred to in Section 13.7.2. Subject to comparing case study three with appropriate related work in Chapter 12, this chapter concludes the presentation of our second RSS-mining paradigm.

Chapter 12

Paradigm two and related work

12.1 Foreword

This chapter compares our second RSS-mining paradigm (Section 1.4.1) with examples of related work from Chapter 3 that we believe to be appropriate to the paradigm. We begin with a brief summary of the paradigm and its constraints in Section 12.2, before decomposing the components of case study three into two further sections. Each of these sections focuses on one of the components of the case study, although this chapter is principally concerned with our sentiment analysis work rather than classification. Section 12.3 summarises our classification work and briefly references related work. On the other hand, Section 12.4 gives a précis of our sentiment analysis work, before Section 12.5 profiles four appropriate examples of related work. Section 12.6 provides a summary comparison between our sentiment work and the related work cited.

For consistency with Chapter 9: (1) each section in this chapter concerned with an example of related work is named for the title and author(s) of the example described therein, and (2) we retain application context and use of RSS as the criteria for the comparison of related work with our second paradigm. Moreover, we do not consider issues of software and application architecture in the related work described in this chapter: nor do we consider the applications of RSS described in Section 3.2.

12.2 Paradigm two

12.2.1 Summary

Our second paradigm, to classify RSS according to the fluctuations in the frequencies of popular keywords and correlating this with sentiment, builds upon our first paradigm.

If basic *actionable and effective* data (Section 2.9) can be produced from RSS by the case studies of our first paradigm, the logical extension of this is to apply data mining techniques to RSS in order to further enhance the social utility of the technology.

12.2.2 Constraints

Constraints specific to the classification component of case study three have been described in Section 10.7. Similarly, constraints applicable to the case study's sentiment analysis component are the focus of Section 11.9. Given the particular nature of these constraints, we do not repeat them here.

12.3 Classification

12.3.1 Paradigm 2

Chapter 10 describes the classification component of case study three which consisted of a semi-automated application of well-known classification techniques to RSS in order to classify feeds into categories according to the fluctuations in the frequencies of popular keywords present in their text. A DT classifier was initially implemented in Weka (Appendix B.1) to provide a ballpark (Appendix B.2) result which could be subsequently confirmed by other MNB and SVM classifiers. We did not seek to maximise our classification results because of the *proof of concept* nature of our work. Nevertheless, at the end our classification work, which used data gathered during Jul - Nov 2011 from our corpus of fifty RSS feeds organised into five categories, average F-measure results per classifier were between 70.00 - 80.00%. We were also interested to determine whether our classification results would be affected by: (1) a series of tranches varying in length from ten to thirty/thirty-one days, and (2) feature selection (Appendix B.2) based upon parameter permutations of RSS feed elements and categories, stop words, stemming, and naïve n-grams to the level of trigram. In addition to this, we sought to use the classification of RSS feeds and keywords to validate our use of semi-automated batch processing of RSS feeds at category-level for our sentiment analysis work.

12.3.2 Review

Section 2.7 refers to a series of surveys of products for, and the application of well-known classification techniques to, sentiment analysis. For this reason, and because of the objectives of our classification work described above, this chapter does not make any comparison

between the related RSS-based classification work described in Section 3.3.2 and the classification component of case study three (Chapter 10). This stipulation specifically applies to the related work by Banos et al. [26], Cingiz and Diri [72], Saha et al. [342] and Teng et al. [391], and others because of their use of NB, MNB and SVM classifiers.

Nevertheless, although their contexts and methodologies vary, two cases of related RSS-based classification work, which we have described in Section 3.3.2, do document issues concerning the format of RSS which are equivalent to those we encountered during our own classification work:

1. Contextual comments made by Liu et al. [231] concern the absence in RSS of "uniform standards for categorization." As a result of this, "news sites determine how to categorize news articles by themselves." The authors further described variations in the categories used by publishers to classify content, where the content was not always sufficiently decomposed, and where the categories would not "satisfy every individual user."

2. The work by Longe and Salami [236] where the authors focused on the mismatch between the pre-defined category of an RSS feed and actual feed content, and reported that their SVM classifier placed "68% of the feed content retrieved" in a different category compared to the category specified by the feed's publisher.

These issues concern the `<category>` element in either the `<channel>` of an RSS feed, or in its `<item>` elements. We agree with the findings by Liu et al. and Longe and Salami regarding the `<category>` element types, and in order to avoid any potential ambiguity they may have caused (Section 2.2.3), we made no use of them in the case studies for our paradigms. Concerning our classification work, in Section 10.7.1 we discuss issues to do with the incorrect population and *optionality* of other `<item>` elements in RSS.

12.4 Sentiment analysis

12.4.1 Paradigm 2

The second component of case study three sought to determine a correlation between the fluctuations in the frequencies of popular keywords present in the text of RSS feeds and sentiment. To do this we made use of a lexicon-based sentiment analyser called SentiStrength (Section 11.3) to calculate sentiment related to *proper* nouns identified by a basic NER. A set of time-series plots visualise the correlation using varying numbers of days and different combinations of RSS feed elements during Aug 2011.

12.5 Related work

We consider four examples of related work in this section which we regard as appropriate to the sentiment analysis component of our second paradigm (Chapter 11). These examples demonstrate the *landscape* of the related work described in Section 3.3.3, where this extends from sentiment analysis into the subject areas of visualisation and trend, topic or event detection, or a combination thereof. We provide a detailed description and review of each example, and conclude with a collective summary.

12.5.1 *Are Raw RSS Feeds Suitable for Broad Issue Scanning? A Science Concern Case Study* by Thelwall et al.

In Thelwall et al. [398], the authors referred to "two relevant traditions of web data analysis": (1) "The purist approach is to analyse an Internet phenomenon as it is found, seeking to describe it as accurately as possible", and (2) the "pragmatic approach is to analyse a phenomenon from the perspective of attempting to gain information about an underlying phenomenon, rather than the Internet data (i.e. for indirect research)." The "key difference between the two approaches is that the former typically does not use any data cleansing whereas the latter tends to use extensive data cleansing."

In an experiment in 2006, the authors sought "to assess whether a *purist* approach to RSS feeds (i.e. using the raw feeds without data cleansing) is suitable for broad issue scanning, using a co-word frequency time series approach." The authors defined the term *broad issue scanning* to "describe the task of identifying and tracking important public debates arising within a given broad issue, such as public science concerns."

Co-word frequencies were used to track stories in 19,587 RSS feeds. The authors first identified items containing words in the "science-relevant" set, i.e. {*science*, *scientist*, *scientists*, *scientific*, *research*, *researcher*, *researchers*, *researching*, *researched*}. Additionally, items were identified relating to "public science concerns if they also contained one of the following set of concern words", i.e. {*argue*, *argued*, *fear*, *afraid*, *worry*, *worried*, *concern*, *concerned*, *frightened*, *scare*, *scared*, *risk*, *risked*, *risky*}. The authors stated that the "word lists were constructed by introspection and scanning postings judged to be about science debates. The collection of items containing at least one word in each of the two sets is labelled the *science concern corpus*. The words in this corpus are co-words in the sense of co-occurring with one 'science' word and one 'concern' word."

Thelwall et al. referred to the "low success rate" of their experiment despite their use, with reference to Gruhl et al. [161], of four methods of identifying words using varying

numbers of days, i.e. *spikes* of one day, *short bursts* of three days, *medium* bursts of five days, and *long bursts* of nine days. Of their results, "only two genuine debates" concerning *ozone* and the Terri Schiavo[1] case were found in the top 1,000 "burst" words. The authors used this result to conclude that "useful information is available in RSS feeds, once topics have been identified, and also that our broad issue scanning method was only able to identify a fraction of postings on this topic: hence the full set of feeds should be used to investigate topics, once identified." Thelwall et al. also wrote that "The domination of the results by non-useful terms" shows that data cleansing is "necessary for efficient broad issue scanning. Raw RSS feeds are unsuitable because some feeds carry extensive and repetitive content."

Thelwall et al. [398] differ from the sentiment component of case study three in two respects: (1) the authors focused on a single theme of public fears about science. In contrast, a diversity of named entities is identified in the corpus of RSS feeds and categories used for our sentiment analysis work, and (2) the use of co-word frequencies relating to these fears, which is counter to our use of the fluctuations in the frequencies of popular keywords present in the text of RSS feeds.

More fundamentally, it is necessary to address the concluding comment by Thelwall et al. that "Raw RSS feeds are unsuitable because some feeds carry extensive and repetitive content." We *disagree with this conclusion in connection with both of our paradigms*, despite issues with keywords described in Sections 10.7.2 and 11.9. We base this conviction upon the correspondence of the paradigms which describes their logical consistency according to their use of RSS (Section 1.4.2).

12.5.2 *Visual Sentiment Analysis of RSS News Feeds Featuring the US Presidential Election in 2008* by Wanner et al.

In a case study published in 2009, Wanner et al. [427] employed a combination of sentiment and visualisation techniques to reveal the emotional content of news stories concerning the US presidential election in 2008.

Data was gathered from the `<title>`, `<description>` and `<pubDate>` elements (Section 2.2.3) of fifty different RSS feeds at half-hourly intervals between 09 Oct - 10 Nov 2008. The texts of the `<title>` and `<description>` elements of each `<item>` were then concatenated, and filtering removed "those documents that contained none of the following signal words:" *Obama*, *McCain*, *Biden*, *Palin*, *Democrat* and *Republican*. The authors

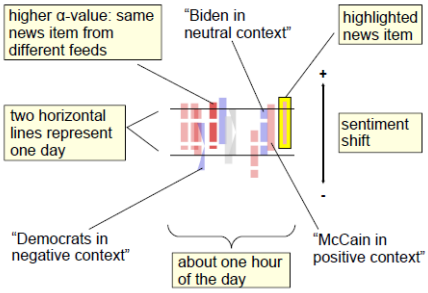
[1] A legal struggle in the USA during the period 1990 - 2005 concerning the right-to-die.

reported that more than 23,000 news items "contained at least one of the six strings." The authors also calculated "Pairwise similarities between news items" to count the "number of non-stopwords that two items have in common (normalized by the length of the larger item)."

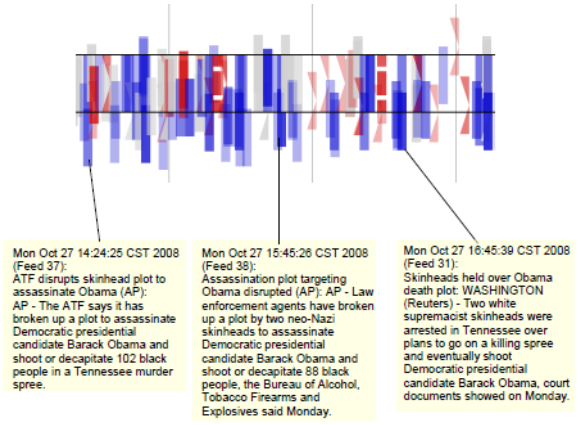
In order to determine the sentiment of each RSS `<item>` in their dataset, the authors relied upon "a freely available list of words that evoke positive or negative associations." The counts of the "number of positive and negative words" were then used to "evaluate the whole news item as rather positive if it contains in total more positive words than negative." Similarly, an item with more negative words than positive words was considered negative. "The absolute relation of positive against negative words normalized by the item's length, provides our sentiment score."

Figure 12.1a illustrates the *semantics* used by Wanner et al. to visualise their dataset. An interactive series of lines was used, i.e. a line per day, where "each colored object depicts one news item." A news item's emotional score was encoded by a vertical scale, and different symbols were used to identify news items according to the aforementioned "signal words". The translucently coloured symbols, i.e. blue (Democrats), red (Republicans) and grey for both, illustrated whether the text of an `<item>` mentioned either political party or both of them. Furthermore, the shape of the objects showed "whether the first candidate, the second candidate or only the name of the political party itself was mentioned." Figure 12.1b reproduces what Wanner et al. referred to as *Bad news for the Democrats* data: approximately one week before the election, many news items included negative references to a plot to assassinate Barack Obama in a murder spree, and to a corruption scandal involving a *Democrat* party senator.

Wanner et al. [427] wrote that their "timeline visualization builds upon three basic elements": (1) colour denoting the political party, (2) different shapes to identify "the discussed persons", and (3) that the "emotional score of each RSS news article resulted in the vertical position of the representative symbol on the time line." The authors concluded that their visualisation technique could be used to monitor public emotional discussions, and evaluate product reviews and public opinion on a particular subject. Reference was also made to potential uses in public relations and media concerning the offering of "sentiment analysis functionality of a multitude of large RSS feeds in real-time", and to people taking early action "before a topic dominates news coverage."



(a) Visualisation semantics.



(b) *Bad news for the Democrats.*

Figure 12.1: The US presidential election in 2008 (reproduced from Wanner et al. [427]): cf. *case study three.*

Two *key* differences distinguish Wanner et al. from the sentiment component of case study three. The first of these is the data domain: Wanner et al. focused exclusively upon a single event, i.e. the presidential election in the USA in 2008, in contrast to the diversity of named entities in our corpus of RSS feeds and categories. The second difference concerns the method used to analyse RSS feed `<item>` elements for sentiment. Wanner et al. based polarity upon the count of words for one sentiment type being greater than the number of words for the other: this differs from our use of sentiment related to the popular keywords present in the text of RSS feeds.

12.5.3 *Multiple coordinated views for searching and navigating Web content repositories* by Hubmann-Haidvogel et al.

Media Watch on Climate Change (MWCC) was described by Hubmann-Haidvogel et al. [188] in 2009 as a "domain-specific news aggregation portal" that "combines a portfolio of

semantic services with a visual information exploration and retrieval interface." The authors wrote that MWCC, which is available at `http://www.ecoresearch.net/climate/`, addresses questions relating to: (1) on-line content repositories and the means to visualise/search them for data, (2) effective and scalable methods to identify and assess the importance of (specifically geospatial) phrases in text, (3) the generation of controlled vocabularies based upon semantic associations within large document collections, and (4) the integration and visualisation of annotated data for searching.

MWCC "aggregates, filters and visualizes environmental Web content from several sources including 150 Anglo-American news media sites." Incoming content is annotated in three ways: (1) *geospatially* to do with geography, (2) *temporally* which adds timestamps for events reported, and (3) *semantically* by computing significant phrases and keywords. A log-likelihood ratio (Appendix B.2) is used to determine significant phrases and keywords in the corpus, keyword frequencies are determined using chi-square, and an ontology component suggests concepts and relations automatically from a controlled vocabulary (Appendix B.2). This dictionary is constructed from co-occurrences of words, key phrases and the use of WordNet [449] for disambiguation.

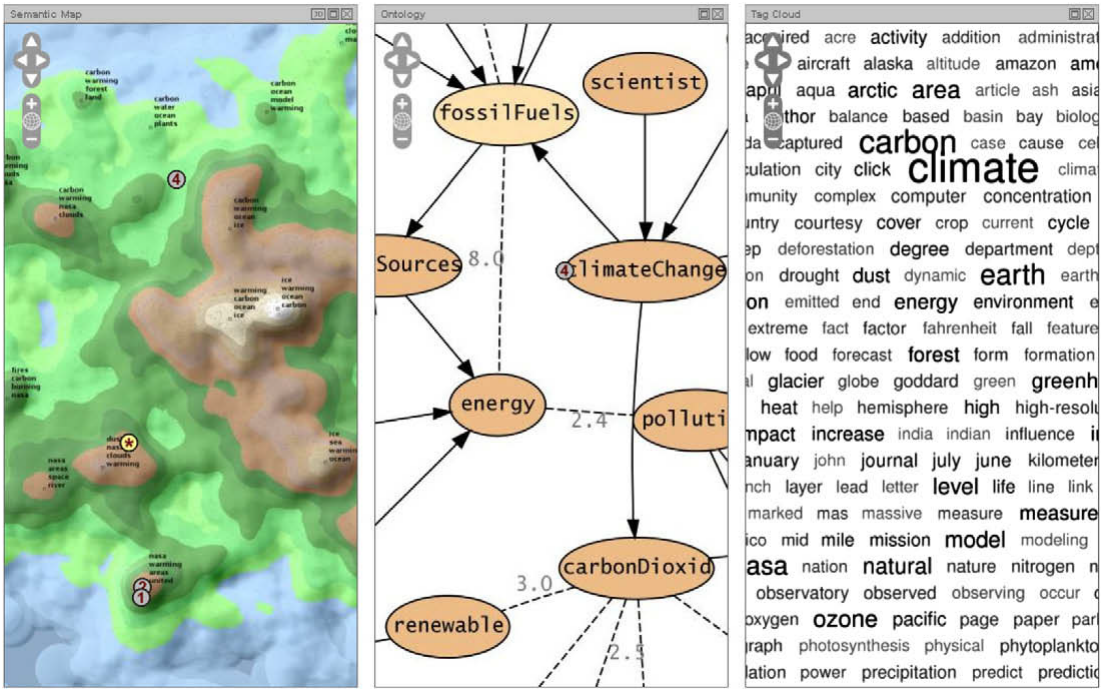


Figure 12.2: The MWCC interface (reproduced from Hubmann-Haidvogel et al. [188]): cf. *case study three.*

MWCC's interface, illustrated in Figure 12.2, integrates two-dimensional geographic and semantic maps, domain ontologies, and word-clouds to allow visual and textual content to be displayed and navigated. Moreover, MWCC automatically calculates and averages sentiment for a selected period, which according to Scharl et al. [347], with reference to Weichselbraun et al. [435], is based upon "aggregated polar opinions identified in the document." Since its inception, further work on MWCC was documented by Scharl et al. [347] in 2013 concerning the application's evolution into an on-line "domain-specific portal" for "environmental stakeholders."

MWCC's use of RSS-sourced data and its visualisation of popular keywords and sentiment analysis, is *analagous* to the sentiment analysis component of our third case study. In addition to this, the placement of its functionality in a web application employing a Java-based architecture, makes MWCC [188] the closest example of related work to our own. Nevertheless, we are able to balance these similarities against two significant distinguishing features of our work: (1) that given its range of data sources and interface, MWCC is addressed to a broader application context than our *proof of concept* RSS-based keyword/sentiment correlation work, and (2) that MWCC is a domain-specific tool concerning the subject area of climate change: on the other hand, we employ simple generic labels, e.g. *News and current affairs*, to categorise our RSS feed corpus, without any use of a controlled vocabulary (Appendix B.2) beyond the use of a basic NER in our sentiment analysis work (Section 11.4.5).

12.5.4 *Narratives: A Visualization to Track Narrative Events as they Develop* by Fisher et al.

This section concerns work by Fisher et al. [122] and their *Narratives* application. It is necessary to state that Narratives did not produce sentiment analysis of the trends, topics or events it visualised, but we include it because of the application's use of keyword correlation.

Narratives was presented by Fisher et al. in 2008 as a means to help users to "place news stories in their historical and social context by understanding how the major topics associated with them have changed over time." The application was based on a platform developed under a former internet technologies initiative by Microsoft, i.e. *LiveLabs*,[2] to

[2]The manifesto for LiveLabs, which ended in 2010, is available at `https://web.archive.org/web/20081115081705/http://livelabs.com/blog/the-live-labs-manifesto/`.

acquire real-time data from "many social media content types, including weblogs." Content was monitored for news stories using a "scraper", and a classifier checked URLs to determine whether an article was news or not, and each article was summarised for "significant keywords" which were also extracted. This "stream of input of *events*" was the principal data source for Narratives, where each event consisted of "the URL to a *referring blog*", the target article's URL and up to ten keywords. Narratives employed time-series (Section 2.8.3) plots to display fluctuations in keyword popularity.

Fisher et al. wrote that "Narratives is distinct" from tools such as *ThemeRiver* by Havre et al. [170] (Section 2.8.2), which allow "side-by-side" [122] comparisons between keywords, because of the ways it, i.e. Narratives, allowed users to investigate correlations between keywords: (1) *date-based correlation*, i.e. "what else was happening on this date around this particular keyword?" The top ten keywords for the date in question were ordered by frequency using fonts sized accordingly. (2) *numerical correlation* computed and displayed a Pearson correlation (Appendix B.2) "between every pair of terms visible onscreen." (3) *most correlated terms* where, for a particular word, Narratives collected a list of all articles that the requested keyword was associated with, and then collected the frequencies of all keywords associated with those articles. (4) *dependent correlation* where for each correlated item, Narratives presented "the curves for the conjunction of the two terms: thus, it shows only articles where both terms occurred." The authors wrote that "This distinction between types of correlation" was a means to "discover when co-occurrences between a pair of words are coincidental, and when they are part of the story."

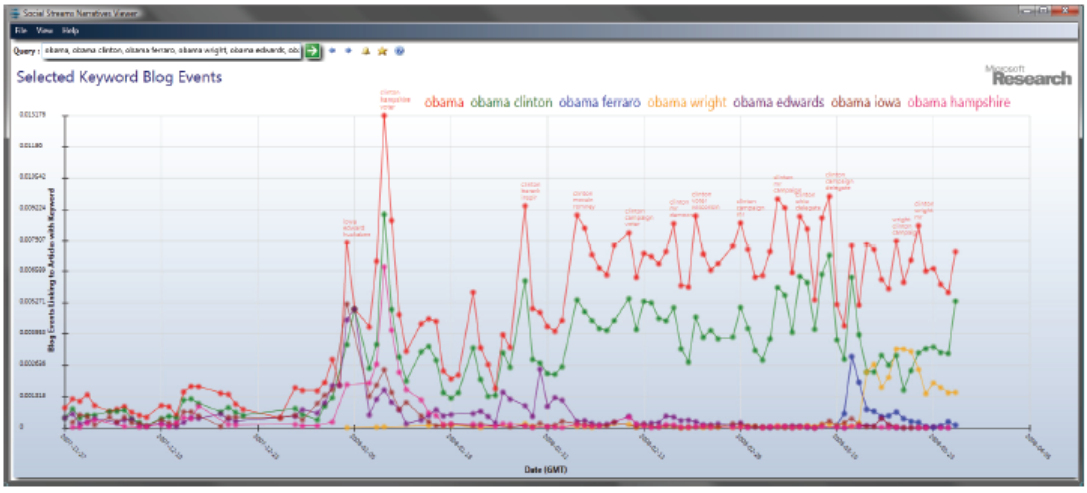


Figure 12.3: The *Narratives* interface displaying Barack Obama and relevant keywords between Nov 2007 - Mar 2008 (reproduced from Fisher et al. [122]): cf. *case study three*.

Two case studies were presented by the authors to evaluate Narratives. The first of these concerned Barack Obama between Nov 2007 - Mar 2008 prior to the US presidential election in 2008. This case study is illustrated in Figure 12.3 which displays *many* changes in keyword frequencies. The second case study examined "the amount of discussion around one company." Fisher et al. expected Google's [148] "most related terms to be linked to the company's products." Instead, attention at the time was focused on a communications related bid in an auction which Google lost, and Microsoft's attempt to purchase Yahoo [455].

The keyword correlation in Narratives distinguishes it from the sentiment analysis component of our third case study, i.e. Narratives used correlation to identify related keywords, whereas on the other hand, we correlate fluctuations in the frequencies of popular keywords present in RSS text with sentiment.

12.6 Summary

Table 12.1 lists a collection of simple criteria which we use in Table 12.2 to summarise the application context and use of RSS of the four examples of appropriate related work described in the previous sections of this chapter. Table 12.2 also includes the sentiment analysis component of our third case study, but *we emphasise that this is not for comparison of functionality but to illustrate application context.*

Criterion	**Description**
RSS	Employs RSS as the principal data source.
Other data sources	Are other data sources used in addition to RSS?
Domain-specific data	Does the related work concern a specific subject area, e.g. climate change?
Data constraints	Within the domain, do constraints apply, e.g. a controlled vocabulary or keyword co-occurrences?
Topic, trend or event detection	Are topics, trends or events detected?
Sentiment analysis	Is data analysed for sentiment?
Visualisation	Are visualisation facilities provided by the software?

Table 12.1: Application context criteria for appropriate related work: cf. *case study three.*

Authors	**Criteria**							**Context and use of RSS**
	RSS	Other data sources	Domain-specific data	Data constraints	Topic, trend or event detection	Sentiment analysis	Visualisation	
Thelwall et al.	✓		✓	✓	✓	✓		Public fears about science using "broad issue scanning", and data cleansing.
Wanner et al.	✓		✓		✓	✓	✓	Sentiment and visual analysis of news concerning the US election in 2008.
Hubmann-Haidvogel et al.	✓	✓	✓	✓	✓	✓	✓	MWCC "aggregates, filters and visualizes environmental Web content".
Fisher et al.	✓	✓			✓		✓	Visualisation of events and keyword-based correlation related to these.
O'Shea	✓				✓	✓		Proof of concept correlation of keyword frequencies and sentiment analysis.

Table 12.2: Summary of application context of appropriate related work (each quotation is according to the example of related work described in the respective section of this chapter): cf. *case study three*.

We consider the four examples of related work to be appropriate to the second component of our third case study because they, like our sentiment analysis work, extend into the subject areas of visualisation and trend, topic or event detection, or a combination thereof. Nevertheless, we have cited differences between our own sentiment analysis work and each example of related work on an individual basis. Despite these differences, in Table 12.2 we can see that similarities of context and use of RSS also exist, specifically with respect to Hubmann-Haidvogel et al. [188] in Section 12.5.3. Given the topicality of current research concerning sentiment analysis and visualisation, this is not surprising.

On the other hand, there are two generic differences that distinguish our own work. The first of these concerns the use by three of the four examples of related work of sentiment analysis within their data specific domain, regardless of whether this involves constraints. In contrast, our use of sentiment analysis is effectively *unbounded*, and as the set of time-series plots in Section 11.8.4 and Appendix A.3.4 demonstrates, keyword frequency/sentiment correlation can occur in different patterns (Section 11.9.1), given that the plots include varying lengths of duration or alternative combinations of RSS feed elements, specific and generic keywords, as well as one or more categories of RSS feeds concerning different data domains.

Secondly, at their most fundamental level, each of the examples of related work makes use of RSS (and in some cases of other types) as a source of data according to their application context. On the other hand, the defining objective of our own sentiment analysis work is to *enhance* the social utility of RSS as a technology, by producing data from feeds which is actionable and effective, rather than using RSS as a source of data for a particular classification of other type of operation. We describe this data and our use of visualisation as a representative medium of it, for the benefit of users in differing domains in Section 2.9.

12.7 Afterword

This chapter has compared our second paradigm with four *landscape*-defining examples of related work. We have identified similarities and differences between each example and our own work with reference to application context and use of RSS. These review criteria extend to the use/enhancement of RSS where the latter of these, and the different patterns of our keyword frequency/sentiment correlation, epitomise the differences between our sentiment analysis work and the related work described.

Part IV

Conclusion and closing comments

Chapter 13

Conclusion

13.1 Foreword

This chapter presents a summary of this thesis. To this end, Section 13.2 concerns the *raison d'être* and motivation of our work, from which we developed the hypothesis and objectives of this thesis. Individual objectives are the subjects of Sections 13.3, 13.4 and 13.5 respectively, wherein we assess their achievement: Section 13.5 extends this assessment by describing the findings of the case studies within the context of our RSS-mining paradigms (Section 1.4.1), and also the real-world domains which could benefit from the *actionable and effective* data produced from RSS in our case studies.

We highlight the constraints and risks inherent in our work in Section 13.6, together with our own reflections upon the PhD programme. Finally, in Sections 13.7 and 13.8 respectively, we discuss the directions for future work and concomitant requirements of a *beta*-version of our vRSS software.

13.2 Summary of research basis

This thesis is concerned with the function, i.e. *utility*, of RSS within the context of social media and how this can be enhanced. In spite of the applications and academic research employing RSS described in Chapter 3, RSS feeds are typically delivered to users via a browser, or by *readers*, where content is presented as *headline*, *story* and *snapshot(s)* (Section 2.2.4), and *consumed*. Predicated upon the motivation of producing from RSS data that is more actionable and effective than this journalistic style, which originated from a combination of personal and professional interests (Section 1.2), the hypothesis of this thesis, which we reproduce from Section 1.3, stated that:

> *Data of an actionable and effective nature can be produced from the fluctuations in the keyword frequencies present in the text of RSS feeds to enhance the social utility of the technology, where this data can benefit users in real-world scenarios, varying from statistics to marketing and trend analysis, correlating or tracking topical issues, or for mining financial and sporting data.*

We further defined three inter-related objectives for this thesis in order to validate this hypothesis:

1. **Definition:** The need to review the current function of RSS within the context of social media, and the utility gained from this.
2. **Production:** To develop the appropriate methods, i.e. paradigms, to mine the text of RSS in order to produce data of an actionable and effective nature.
3. **Demonstration:** To formulate and carry out a series of specifically designed and implemented case studies to demonstrate the paradigms and to use an appropriate medium to represent their outputs.

We judge the contributions made to web engineering and text mining by this thesis as being based upon the demonstration of the *proof of concept* of our objectives by our paradigms (Section 1.4.1) and their case studies. We define the contributions in the following sections of this chapter.

13.3 Definition

The definition of actionable and effective data (Section 2.9) is not a demonstrable objective and therefore not a contribution made by this thesis. Nevertheless, for completeness, we include it here as our first objective because of the basis it provides for our paradigms and their case studies in the subject areas discussed in Chapter 2, i.e. data and text mining, sentiment analysis and the use of visualisation to represent data.

In Chapter 2 we also reviewed the applications and services provided by social media in order to identify the nature of the utility they provide. According to Gallion [133], this utility concerns *socialising*, *entertainment*, *self-status seeking* and *information*. This analysis builds upon the definition by Hammersley [165] of the "social, spiritual and mercenary" reasons why organisations and individuals syndicate content in feeds. Other benefits have also been identified by Singh and Sahu [360].

Therefore, with regard to the above, we define our enhancement of RSS's utility within the context of social media as the result of employing RSS itself to produce a comprehensible and explainable representation of data which is inherent within the fluctuations in the keyword frequencies present in the text of RSS feeds. Consequently, the resulting data is more actionable and effective than that of the aforementioned journalistic style of presentation.

13.4 Production

The *n*-tiered Java JSP/servlet-based web application architecture employed by our myDS and vRSS software is described in Section 4.5. In order to implement our case studies, the *alpha*-versions of each application integrate many open-source, third-party products[1] [2] from the Java *ecosystem*. These products are defined in Appendix B.1 and include, but are not limited to: (1) hourly polling by Quartz Scheduler, (2) to have Rome parse the text of new postings made to RSS feeds, and (3) the calculation from the mined text of keyword frequencies by Lucene, (4) the visualisation of these keyword frequencies by Google Charts, and the use of (5) Weka to data mine them, and (6) SentiStrength to analyse them for sentiment, where this software and processing is *contained* in the (7) Apache Tomcat web server, which also communicates with the relational databases implemented in (8) MySQL.

The successful integration of these products into a coherent and innovative *alpha*-version prototype software, based upon and documented according to the web engineering/application principles and content of this thesis, provides the *vehicle* by which we have been able to realise the case studies which demonstrate the *proof of concept* of our paradigms. It is this development work that constitutes the contribution made by this thesis to web engineering, because without it, our paradigms would have remained purely theoretical.

[1] The distribution of the principal open-source, third-party products in our myDS and vRSS software for our case studies is described in Sections 4.7.1 and 4.8.1 respectively.

[2] Approximate counts of the number of lines of code in the principal open-source, third-party products used in our software, together with equivalent counts of the number of *indigenous* lines of code in myDS and vRSS, are listed in Appendix D.2.

13.5 Demonstration

13.5.1 The relationship of RSS-mining paradigms, case studies and software

For context, the relationship between our paradigms, case studies and software is illustrated in Figure 13.1, which is reproduced from Figure 1.1.

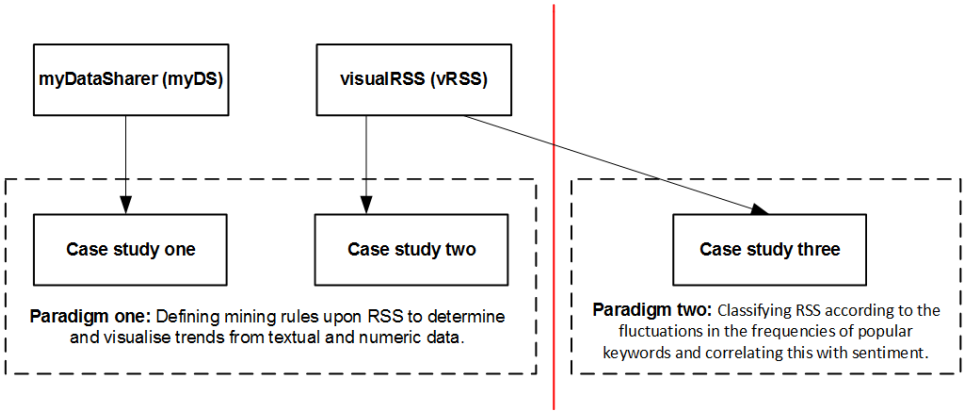


Figure 13.1: The relationship of our RSS-mining paradigms, case studies and software (reproduced from Figure 1.1).

13.5.2 Paradigm one

The premise of our first paradigm (Section 1.4.1) is to demonstrate that we can use mining rules to produce from RSS data that is more actionable and effective than we currently see in the use of the technology (Chapter 3). The definition of mining rules (Section 4.7.2) upon RSS to determine and visualise trends is intended to provide a straightfoward means for users to specify how textual and numeric data is to be mined from feeds during polling to update and visualise the objects the rules become part of. We summarise each case study for this paradigm and their principal findings below:

1. **Case study one:** Held in late 2009, this case study is described in Chapters 5 and 6. In an assessed assignment, a corpus of thirty-five Masters-level students were allocated small numbers of RSS feeds to mine and visualise data from: the students were also asked to do this with their own feeds. This work employed the two mining types listed below which were implemented in our myDS software:

    - **Occurrence mining:** OM counts the occurrences of specified strings in the text of RSS feeds to explore trends or track issues. The three variants of OM

are described in Section 5.4.1.

- **Value mining:** VM analyses RSS feeds which provide structured content. Such feeds report modern forms of ticker-tape (Appendix B.2) data, such as financial movements, sports or lottery results.

The original research questions of this case study (Section 6.2.2) concerned the feasibility of our first paradigm in terms of the mining types developed and the efficiency of their use. We found that:

- Mining rules were successfully used to create 173 datasets which were documented in the submissions by the student corpus.
- A total of 302 visualisations of the data mined from RSS feeds during polling, based upon the definition of mining rules, were produced and cover a wide range of subjects.
- We observed efficiencies in the definition of mining rules because of reductions in the time taken.
- The mining of textual data from RSS using OM proved it to be the preferred mining type with 78.00% of the reported datasets employing it, compared to 22.00% for value-based numeric data for VM. This difference is explained by the majority of RSS feeds on the internet consisting of frequently updated text-based items, rather than numeric data.
- Standard x - y charts were the most popular types of visualisations (Section 6.3.4).

2. **Case study two:** This case study, presented in Chapters 7 and 8, refined our previous use of textual mining rules in case study one (Section 6.4.2) because of difficulties encountered by our students, and employed a second corpus of thirty-six Masters-level students in an assessed assignment during Dec 2011. Each student was required to select categories of system-indexed RSS feeds, or to use groups of feeds of their own choice, and up to six unigram keywords from these feeds, and enter them into vRSS using the refined textual mining rules. vRSS then tracked the frequencies of these keywords for a seven day period. At the end of this period, the students were to report and visualise their results. Case study two employed:

   - Refined mining rules focused less upon structural RSS metadata (Appendix B.2) inherent in the relationship of mining rules to columns of datasets (Section

5.4.2) in case study one. Instead, the rules concentrated upon the application of simple, direct mining types to mine data to be visualised.

- These mining types allowed a balance between user-selection of individual RSS feeds and keywords on the one hand, i.e. *manual* mining, and the selection of system-generated keywords from system-indexed categories of RSS feeds in *automatic* mining on the other hand. A third type, i.e. *semi-automatic* mining, formed an intermediary allowing the use of system-indexed categories of feeds and user-entered keywords.
- The original corpus of fifty RSS feeds and seven categories, which was extended by the feeds added by the students during the case study, gave a final corpus of 202 feeds grouped into twelve categories. The most popular category in terms of the number of RSS feeds was NCA with fifty-two RSS feeds, least popular was the Miscellaneous category with two feeds.

Our objectives in case study two were to research preferences of the mining types employing the refined mining rules, visualisations, distribution of categories of feeds visualised, and the common use of these amongst the mining types. The most salient findings were:

- 135 instances of mining rules were used to create 135 visualisations.
- Semi-automatic mining proved to be the most popular mining type with some forty-eight (35.60%) instances created. Least popular was manual mining with forty-three instances produced: (31.80%) of the total.
- Column and bar charts were the most popular visualisation types with 115 (85.19%) instances created.
- The 135 visualisations created include some 350 permutations of RSS feed categories, i.e. where a visualisation displayed data sourced from one or more RSS feed categories. The most popular choice was the use of column and bar charts to visualise the NCA category of feeds: we recorded fifty-six instances. Overall, the column and bar charts amounted to 277 (79.14%) of the total number of permutations.
- Five visualisations employed ten or more of the twelve RSS feed categories.
- Many constructive comments were received during the case study for both the vRSS application and the nature of the assignment.

In Chapter 9 we described three cases of related work appropriate to our first paradigm. Each example required the definition of rules upon RSS to govern how data in feeds was used according to application context. Stated simply, these rules were concerned with: (1) querying, filtering and specifying behaviours on RSS content (Van Kleek et al. [413]), (2) data stream management and continuous query processing to create personalised RSS feeds (Creus et al. [80]), and (3) an algebra to query RSS based upon temporal and other relationships between items of content (Getahun and Chbeir [137]). In contrast to these examples, our mining rules present a syntax-free means to specify how textual and numeric data is to be mined from RSS.

Given this contextual difference between the mining rules of our first paradigm and the related work referred to above, and also with Chapter 3's review of the applications and academic research employing RSS, to the best of our knowledge, our approach of defining mining rules as a means of mining textual and numeric data from RSS to produce actionable and effective data, is unique. Therefore, considering the results of case studies one and two which demonstrate the paradigm, we can infer that mining rules are able to successfully mine and visualise information from RSS, and that our first paradigm establishes a contribution to text mining.

13.5.3 Paradigm two

Our second paradigm (Section 1.4.1) concerns a semi-automated application of well-known classification techniques to RSS in order to classify feeds according to the fluctuations in the frequencies of popular keywords present in the text of feeds, and to determine a correlation between this and sentiment. The rationale of this paradigm is that if our first paradigm can produce basic actionable and effective data from RSS, the logical extension of this is to apply data mining techniques to RSS in order to further enhance the social utility of the technology. We summarise each component of the paradigm's single case study, i.e. case study three, and its findings below:

- **Classification:** The classification component of our third case study is described in Chapter 10. This work consisted of the semi-automated classification of RSS feeds into categories according to changes in the keyword frequencies present in their text.

  We modified the RSS feed and category corpus from case study two into five categories and fifty feeds. A series of tranches, varying in length from ten to thirty/thirty-one days employing data gathered during Jul - Nov 2011, was used to generate training and testing data with a feature selection (Appendix B.2) based upon parameters

of RSS feed elements, stemming, stop words and naïve n-grams to the level of trigram. Having selected and integrated Weka (Appendix B.1) into an extension of our vRSS software, we applied a DT classifier to permutations of the aforementioned parameters to provide a ballpark result which could be subsequently confirmed by other MNB and SVM classifiers. The DT classifier produced an overall F-measure result of 0.72. The MNB and SVM classifiers achieved slightly superior F-measures of 0.79 and 0.77 respectively: this difference between the DT and other classifiers was due to our use of DT pruning according to Drazin and Montag [96].

Our classification work demonstrates that RSS feeds can be classified based upon the frequencies of popular keywords. We did not seek to maximise our results because of the *proof of concept* level of our work, but we were interested to determine whether our results would be affected by variations in the feature selection, and to use the classification of RSS feeds and keywords to validate the use of semi-automated batch processing of RSS feeds at category-level for our sentiment analysis work.

- **Sentiment analysis:** In the final component of case study three, we correlated changes in the keyword frequencies in the text of RSS feeds with sentiment. Proper nouns were identified in our corpus of feeds and categories for Aug 2011 using a basic NER (Appendix B.2), and a lexicon-based sentiment analyser called SentiStrength (Appendix B.1) was employed to calculate sentiment related to them.

  The principal result of this work is the visualisation, in Section 11.8.4 and Appendix A.3.4, of the correlation in a set of time-series plots. These plots illustrate that keyword frequency/sentiment correlation can occur in different patterns, given that the plots include varying lengths of days, alternative combinations of RSS feed elements, and the use of specific and generic keywords referring to individuals or to a title, or to multiple data domains according to the categories of RSS feeds used. This differs to those examples of related work employing sentiment based upon a single domain which we compared with our sentiment analysis work in Chapter 12.

  It is this, and the defining objective of our sentiment analysis work to enhance the social utility of RSS by producing data from feeds which is actionable and effective, that distinguishes our second paradigm from other work. This use of RSS as a technology rather than a source of data presents, to the best of our knowledge, a unique contribution to text mining.

13.5.4 Application

In presenting our paradigms and their case studies, we have identified several real-world domains where users could benefit from actionable and effective data in either a raw form or a visual representation of it. These domains include correlating or tracking topical issues for trend analysis in business intelligence, statistics, politics, market research and related subject areas, or for mining modern forms of ticker-tape data, such as financial movements, sports or lottery results. An alternative use of our vRSS software as a browser plug-in is illustrated in Section 8.6.3. Despite these domains, which we identify according to the achievement of our objectives for this thesis (Section 13.5), there are a series of constraints which apply to our work: we consider these in Section 13.6.1.

13.6 Reflections

13.6.1 The PhD programme

In Section 13.2 the motivation, hypothesis and objectives of this thesis, as they were originally defined in Chapter 1, are restated. These elements, the necessary work involved and contributions made thereof, constitute an instance of the *PhD programme* [393]. We take this opportunity to briefly reflect upon some of the circumstances and choices, constraints and justifications made regarding the *proof of concept* demonstration of our paradigms by their case studies during *our* PhD programme: within these reflections, we also identify directions for future work. Inevitably, these reflections overlap because of the inextricably connected nature of our work: for this reason, the relevant subject areas are simply listed in alphabetical order below:

- **Case study results:** Despite the references in Section 13.5 to the documented results of our case studies, we are aware that these results could have proved otherwise. In case studies one and two, such differences could have been caused by alternative student, or RSS feeds and category, corpora: for this reason we expand upon these constraints elsewhere in this section. For our third case study, other classifiers or sentiment analysis methods/tools may also have produced different results, and as a result, we consider this as a direction for the future work we discuss in Section 13.7.1.

- **Data flow diagrams:** We employed DFDs (Appendix B.2) in this thesis: (1) in order to present a concise and *lightweight* representation of the processes and flows

of data in our myDS and vRSS software, and (2) because of authorial familiarity. We acknowledge that UML *activity*, *information flow* and *use case* behavioural diagrams could have been used as an alternative.

- **Keyword frequency and sentiment causation:** We do not consider the issue of *causation* in the sentiment analysis component of case study three (Chapter 11), i.e. do keyword frequency fluctuations cause changes in sentiment or vice-versa? It is our opinion that this is a *bidirectional* relationship, i.e. an event or story can cause sentiment which can be related to keywords therein, but sentiment can also result in a story with keywords. We cite two stories from Jul 2016 to demonstrate this:

  1. **Story causing sentiment:** The Chilcot enquiry was held in the UK to determine that country's role in the Iraq War in 2003. When the enquiry published its report, the widespread national and international news media coverage of this story included much criticism of the UK's then-prime minister Tony Blair. Thereafter, the content of RSS and other social media frequently included keywords such as *Tony Blair* or *Blair*, and negative sentiment therein related to the former prime minister as its subject. This example demonstrates a story causing sentiment, or exacerbating pre-existing sentiment.
  2. **Sentiment causing story:** The example we cite here concerns negative sentiment allegedly expressed towards popular singer Taylor Swift in the lyrics of a song titled *Famous* written by Kanye West. The reporting of this sentiment demonstrates that an opinion can trigger media coverage, i.e. where the expression of a sentiment or nature thereof, becomes the basis of a story, which will include the names of the affected individuals, so enabling their use as keywords.

- **Keywords and linguistics:** Although the conventions and characteristics of our keywords (Section 4.6) do permit the use of abbreviations, mining rules in our first paradigm, and system-generated keywords in our second and third case studies, do not consider issues of *relatedness*. Nor do they include keywords consisting partially or wholly of *emoticons*, i.e. simple text or graphical representations of sentiment such as `:-)` for *happiness* or `:'(` for *crying*, despite provision for these in SentiStrength in the sentiment analysis component of case study three (Section 11.6.2). Moreover, our keywords do not include contractions such as `CUL8R`, i.e. *See you later*, typical of *texting* or *tweeting* in social media (Section 2.3) where the character length of a posting is restricted. We were also aware that, in the classification component of

case study three, our keywords included named entities and slogans (Section 10.7.2), despite our later use of a basic NER in that case study's sentiment analysis work.

The currency of these and other types of keywords, or of managerial- or technological-originated jargon, e.g. the term *reboot* to describe the re-launching of a film franchise or TV series, in natural language and their concomitant *social* effects is beyond the *proof of concept* level of this thesis. Nevertheless, we regard some of the aforementioned linguistic issues as directions for the future research work we describe in Section 13.7.1.

- **Mining rules:** In Section 6.4.2 the refining of textual mining rules as a result of difficulties in our first case study is described. We believe that the subsequent use of the refined rules in case study two (Chapter 8), given the positive feedback received, makes our first paradigm's definition of mining rules upon RSS more amenable to a wider population.

- **Order of RSS-mining paradigms and case studies:** The order in which the work for our paradigms and case studies was carried out concerns the definitions of our RSS-paradigms and their correspondence (Section 1.4). For our first paradigm, the order of its two case studies was because of the need to demonstrate that mining rules could be successfully defined upon RSS. Paradigm two is a logical progression of paradigm one, i.e. could we not also apply data mining techniques to RSS? Other influencing factors, described elsewhere in this section, include student corpora and the laboratory-based nature of the case studies.

- **Pilot studies:** We carried out no pilot studies for the case studies for our first paradigm: this is discussed in Sections 6.4.3 and 8.6.1 respectively. The use of semi-automated batch processing of RSS feeds in case study three (Section 4.8.2) precluded the need for a pilot study therein. Upon reflection, given this generic absence, we conclude that it is not possible to retrospectively gauge the influence that pilot studies may have had on individual case studies as they were developed and carried out.

- **RSS:** There are three pertinent issues:

    1. **Social media:** In Section 1.2, we defined *ubiquity*, *longevity* and *diversity* as the reasons for the selection of RSS as an example of social media (Section 2.3) for our work. Given: (1) the utility of RSS (Section 2.2.4), (2) the range of

applications and academic research employing the technology (Chapter 3), and (3) the results of the case studies for our paradigms, we infer that this choice has been vindicated.

2. **Corpora:** This principally concerns demographics and the use of more extensive corpora, i.e. in terms of larger numbers of feeds and categories of them. Both issues are directions for the future work we discuss in Section 13.7.1.
3. **Format:** Although Section 10.7 concerns the classification component of our third case study, the issues described therein to do with the format of, and published content in, RSS are *substantially* applicable to all three case studies for our paradigms.

- **Software:** The development of our myDS and vRSS software as web applications (Section 4.3) for the case studies for our first paradigm was necessary to provide a consistently available medium to allow each case study's student corpus to complete its respective assessed assignment. The extension of vRSS for case study three was a logical consequence of this development, especially when we consider that our original intention was to use the discontinued manual facilities written for our sentiment analysis work: we refer to these facilities in Section 4.8.1.

  We do regret the non-use of big-data (Appendix B.2) products in our work, the popular availability of which did not coincide with the development periods of myDS or vRSS. Related to this, we also acknowledge the availability and use of the array of open-source, third-party products (Appendix B.1) in our software because of the absence of a *bespoke* tool for our particular use of RSS.

- **Student corpora:** The submissions for the assessed assignments of our first and second case studies were produced by the classes of a Masters-level module in 2009 and 2011 respectively, the subject of which concerned search engines and web navigation. Each class comprised an available corpus of students of "varying employment and experience backgrounds" (O'Shea and Levene [288]). In Sections 6.4.4 and 8.6.2 respectively, we describe our homogenous use of the student corpora where they provided sufficient numbers of users to test the mining rules for our first paradigm. We did not consider the demographics of either student corpus in these case studies.

  As a consequence of this, we cannot exclude the possibility that alternative student or user corpora, based upon demographics may have produced other, different results. Given the absence of users for case study three because of its use of semi-automated

batch processing of RSS feeds, we cannot speak in the same terms for the classification and sentiment analysis work for our second paradigm. A second, related consequence of using such corpora was the laboratory-based nature of case studies one and two: our third case study was by its nature laboratory-based. We consider demographics and limited real-world commercial use of our software as directions for the future research work discussed in Section 13.7.1.

13.6.2 Advice for a potential PhD student

The following list defines our advice for a potential PhD student which extends to cover the full range of the PhD programme where: (1) it is incumbent upon the student to be certain of their chosen research subject, and also its relevance within its *native* academic, commercial or other *environment*. In connection with this, it may be advisable that the student has several research subjects in mind at this time. With this basis established, the student must (2) carefully select their supervisor(s). Both supervisor(s) and student must now (3) define the boundaries of their work within the aforementioned environment. It follows that (4) a plan of work must be developed within the first year. Henceforth, the student must remain focused upon this plan and not be side-tracked by unnecessary or excessive detail which may cause delay. This can be prevented by the student (5) maintaining regular contact with their supervisor(s) to ensure that the plan is being followed, but the student should also (6) be prepared to disagree with their supervisor(s) given good reason. It is also sensible for the student to (7) maintain an amicable relationship with their supervisor(s). The student should also (8) be aware of work related to their own, and in connection with this, (9) seek to build up a network of contacts through published work in journals and conferences. This *networking* requires that the student must be prepared, and understand the need, to (10) criticise related work and accept criticism of, and be ready to defend, their own work especially when their *viva voce* takes place. It may also be useful (11) for the student to familiarise themselves with the work of the external examiners adjudicating their viva.

Further advice to a potential PhD student concerns the necessary application, patience and tolerance required to complete the PhD programme, despite the inevitable doubts, and consequent temptation to abandon their work. The student should also be aware of the seemingly ever-increasing amount of time required by the PhD programme for research work and thesis writing/revising up to the moment of final submission.

13.7 Directions for future work

We divide the directions for future work, beyond the *proof of concept* of our paradigms and their case studies, into two inter-related areas described in the following sections of this chapter.[3]

13.7.1 Research

We identify the following key five subject areas for future research work:

1. **Commercial use:** Research based upon limited commercial use.

2. **Demographics:** We reflect upon our use of student corpora in our case studies in Section 13.6.1. Alternative approaches for future research include giving greater attention to demographics in student, or other user, corpora. This could be based upon the allocation of: (1) generic or tailored RSS feeds and category corpora to specific demographic groups, or (2) generic corpora of RSS feeds and categories to different demographic parties, in order to determine variations in mining rules or sentiment according to contentious issues such as race, religion and politics.

3. **Keywords and linguistics:** This concerns future work requiring a more thorough mining and processing of keywords. Although some of the issues listed below are specific to case study three for our second paradigm, we believe that other items are generally applicable to both of our paradigms and their case studies.

   Issues include: (1) addressing the keyword anomalies described in Section 10.7.2, (2) more extensive use of NER for keyword disambiguation, (3) further use of stemming and stop words, (4) the use of NLP (Appendix B.2) techniques in order to detect keyword context or semantics in the text of RSS feeds, (5) the issue of *relatedness* in our keywords (Section 4.6), (6) the application of popular classification techniques, e.g. MNB and SVM (Section 10.5), as alternative methods of sentiment analysis to the use of a lexicon-based product, e.g. SentiStrength (Appendix B.1), in case study three, (7) comparing human/machine sentiment especially where non-lexicon methods of sentiment analysis (Section 2.7) are concerned,[4] and (8) the use of statistical techniques to explore the distribution of keywords in the text of RSS feeds,

[3] With the exception of the loss of a small quantity of data after case study one (Section 6.4.5), all software written for, and data created during, the cases studies for our RSS-mining paradigms has been archived for storage.

[4] This is the subject of a proposed post-PhD paper by the author and supervisors.

e.g. TF-IDF (Appendix B.2), beyond our present correlation of keyword frequencies and sentiment (Chapter 11).

4. **RSS feeds and categories:** The use of more extensive corpora in terms of larger numbers of feeds and categories, or to base corpora, or allocation thereof, upon demographics.

5. **Mobile client:** Including the client *app* developed by Shema [354] for the Android OS (Appendix C).

13.7.2 Facilities

The future research work and *beta*-version of vRSS, described in Sections 13.7 and 13.8 respectively, would both be served by the future development in the application of the following:

- The recommendation of RSS content to users. The recommendation methods described in Section 2.3.2, could be added to our software to suggest RSS feeds, visualisations or analyses of sentiment which may be of interest to users.

- The provision of an archive of RSS feed content together with search and retrieval functionality.

- Given related work concerning trend, topic or event detection, or a combination thereof in RSS feeds (Section 3.3.3), the incorporation into our software of detection facilities to allow interactive visualisation and sentiment analysis of timelines.

  There are three applicable corollaries:

  1. In Section 11.2 we referred to discontinued work in vRSS for manually running sentiment analyses. The collection of web pages written provides the *nucleus* for an interactive facility allowing users to select RSS feeds and keywords for sentiment analysis, where this could be enhanced through the use of boolean connectives AND, OR and NOT.
  2. A regular semi- or fully- automated classification and sentiment analysis of RSS data for sentiment to produce time-series (Section 2.8.3) plots.
  3. A greater use of open-source, third-party products and web services for the keyword and linguistic issues referred to in Section 13.7.1, e.g. the use of software available from the Stanford Natural Language Processing Group [371].

- Where the XML-based format of RSS (Section 2.2.3) and the technology's use as a delivery mechanism, enable our paradigms to be extended to feed types other than RSS.

13.8 A *beta*-version of visualRSS

Based upon the description of the *alpha*-version of our vRSS software in Chapter 7, we identify the following technical considerations for a *beta*-version of vRSS:

- **Architecture:** The retention of the existing n-tiered Java JSP/servlet-based web application architecture described in Section 4.5 but with: (1) a more extensive use of creational, structural and behavioural design patterns (Gamma et al. [134]), (2) a greater use of AOP for cross-cutting concerns (Appendix B.2), and (3) web services.
- **Database:** There are two alternatives here: (1) vertically partitioning the database into components, i.e. dedicated databases, or database partitions, for RSS feeds, keywords, mining rules, datasets and visualisations, or (2) by horizontal *sharding* (Appendix B.2) requiring the use of multiple databases based upon a common property, e.g. geography. The use of NoSQL products (Appendix B.2) must also be considered as an option here.
- **Scalability:** To guarantee performance, more hardware could be added: (1) vertically, by increasing the number or speed of application processors, or (2) using more physical/logical servers.

Other requirements of a *beta*-version of vRSS concern a (better) integration of the application's two halves. As a result of its original development for case study two, its extension (Section 4.8.1) and the use of batch processing in case study three, vRSS lacks a single, all-inclusive working interface. A result of an improved integration would provide the trend, topic or event detection, semi- or fully- automated classification and the sentiment facilities identified in Section 13.7.2.

Lastly, limited commercial use of a *beta*-version of vRSS, by users in one or more of the domains referred to in Section 13.5.4, would be of considerable benefit to the future work we describe above.

Part V

Appendices

Appendix A

Case study reference materials

A.1 Case study one

A.1.1 RSS feed corpus

RSS feed number	RSS feed URL	Mining type
1	http://www.stackoverflow.com/feeds	OM
2	http://sports.yahoo.com/sow.rss	VM
3	http://www.denofgeek.com/index.rss	OM
4	http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/front_page/rss.xml	OM
5	http://www.superpages.com/cities/lottery/index_rss.html	VM
6	http://feeds.boingboing.net/boingboing/iBag	OM
7	http://news.cnet.com/2547-1_3-0-5.xml	OM
8	http://www.scripting.com/rss.xml	OM
9	http://feeds.musicchartfeeds.com/itunestop100albums	OM
10	http://feeds2.feedburner.com/TheNextWeb	OM
11	http://www.premierleague.com/rss/ptv/page/ArticleIndex/0,,12306~2233528,00.xml	VM
12	http://www.football.co.uk/news/rss.aspx	VM
13	http://www.scorespro.com/rss/live-soccer.xml	VM

14	`http://www.scorespro.com/rss/live-formula.xml`	VM	
15	`http://www.metacritic.com/rss/movie/film.xml`	OM VM	or
16	`http://conor.net/feeds/rss/tube.xml`	OM	
17	`http://www.thinkbroadband.com/rss/full/`	OM	
18	`http://twitter.com/statuses/user_timeline/12719612.rss`	VM	
19	`http://feeds.instyle.com/instyle/thisjustin`	OM	
20	`http://www.sloomedia.com/currency/feeds/USD.xml`	VM	
21	`http://conor.net/feeds/rss/tube.xml`	OM	
22	`http://themoneyconverter.com/OMR/rss.xml`	VM	
23	`http://themoneyconverter.com/PAB/rss.xml`	VM	

Table A.1: Corpus of RSS feeds: cf. *case study one*.

A.1.2 Allocation of RSS feeds to students

Student number	**RSS feed number**			
	One	**Two**	**Three**	**Four**
1	10	12	14	19
2	2	5	9	20
3	10	12	14	19
4	3	4	6	22
5	6	9	12	20
6	2	3	6	13
7	2	7	10	23
8	5	11	17	19
9	3	6	19	23
10	4	7	19	23
11	3	6	14	22
12	6	9	11	23
13	12	14	22	27
14	2	4	8	23

15	3	6	13	22
16	1	7	8	22
17	4	6	7	12
18	5	11	17	19
19	1	8	17	22
20	4	6	19	22
21	2	5	9	20
22	2	5	9	22
23	6	11	17	23
24	4	5	19	23
25	4	7	10	23
26	3	6	17	22
27	2	11	19	22
28	1	8	19	22
29	5	11	17	19
30	2	5	10	23
31	1	4	13	22
32	3	5	9	20
33	1	4	5	11
34	4	8	13	19
35	3	6	13	22

Table A.2: Student allocation of RSS feeds: cf. *case study one*.

A.2 Case study two

A.2.1 Original RSS feed and category corpus

RSS feed no	RSS feed URL	Category
106	http://feeds.abcnews.com/abcnews/worldnewsheadlines	NCA
107	http://feeds.bbci.co.uk/news/world/rss.xml	NCA
108	http://rss.cnn.com/rss/cnn_world	NCA
109	http://www.nytimes.com/services/xml/rss/nyt/GlobalHome.xml	NCA
112	http://www.msnbc.msn.com/id/3032506/device/rss/rss.xml	NCA
113	http://english.pravda.ru/export.xml	NCA
114	http://www.france24.com/en/monde/rss	NCA
115	http://www.denofgeek.com/index.rss	Film
118	http://www.zdnet.com/search?t=1,7&mode=rss	SNT
119	http://feeds.bbci.co.uk/news/technology/rss.xml	SNT
121	http://feeds.bbci.co.uk/news/business/rss.xml	BFE
123	http://www.ok.co.uk/rss/32/okfashion	FCL
124	http://rss.feedsportal.com/c/592/f/7507/index.rss	Film
125	http://www.billboard.com/rss/the-feed/	Music
127	http://rss.feedsportal.com/c/375/f/434908/index.rss	FCL
128	http://evilbeetgossip.film.com/feed/	FCL
129	http://newsrss.bbc.co.uk/rss/sportonline_world_edition/front_page/rss.xml	Sport
130	http://rss.cnn.com/rss/edition_sport	Sport
131	http://www.espn.co.uk/rss/sport/story/feeds/0.xml?type=2	Sport
132	http://www.skysports.com/rss/0,20514,12040,00.xml	Sport
133	http://feeds.reuters.com/reuters/sportsNews	Sport
134	http://feeds.wired.com/wired/index	SNT
140	http://feeds.feedburner.com/totalfilm/news/	Film
141	http://www.mtv.com/rss/news/movies_full.jhtml	Film
143	http://feeds.feedburner.com/thr/film	Film
144	http://www.hollywoodnews.com/feed/	Film

148	http://www.eweek.com/rss.xml	SNT
149	http://hosted.ap.org/lineups/WORLDHEADS-rss_2.0.xml?SITE=WHIZ&SECTION=HOME	NCA
153	http://www.vanityfair.com/services/rss/feeds/everything.xml	FCL
158	http://www.infoworld.com/rss.xml	SNT
159	http://www.businessweek.com/rss/bwdaily.rss	BFE
160	http://rss.cnn.com/rss/money_topstories.rss	BFE
161	http://feeds.reuters.com/reuters/businessNews	BFE
162	http://online.wsj.com/xml/rss/3_7432.xml	BFE
163	http://feeds.people.com/people/headlines	FCL
165	http://www.music-news.com/rss/news.asp	Music
166	http://loft965.com/feed/	Music
167	http://www.mojo4music.com/blog/index.xml	Music
168	http://www.musicweek.com/rss.asp?navcode=232	Music
170	http://www.theengineer.co.uk/XmlServers/navsectionRSS.aspx?navsectioncode=186	SNT
171	http://feeds.bbci.co.uk/news/science_and_environment/rss.xml	SNT
172	http://feeds2.feedburner.com/nmecom/rss/newsxml	Music
173	http://www.spotify.com/uk/feed/	Music
174	http://www.prospectmagazine.co.uk/category/magazine/feed/	NCA
175	http://feeds.reuters.com/reuters/worldNews	NCA
176	http://www.spiegel.de/international/index.rss	NCA
177	http://newsfeed.time.com/feed/	NCA
178	http://www.msnbc.msn.com/id/3032117/device/rss/rss.xml	FCL
179	http://feeds.reuters.com/reuters/technologyNews	SNT
181	http://feeds.technologyreview.com/technology_review_top_stories	SNT

Table A.3: Original RSS feed and category corpus (Section 8.4.2): cf. *case study two*.

A.3 Case study three

A.3.1 Withdrawals from RSS feed and category corpus

RSS feed number	RSS feed URL	Category	Reason withdrawn	Withdrawn date
139	http://feeds.feedburner.com/variety/headlines	Film	<pubDate> element not populated.	01 Nov 2011
146	http://www.aintitcool.com/node/feed?category=	Film	<description> element populated by <pubDate> element.	01 Nov 2011
150	http://www.economist.com/rss/international_rss.xml	NCA	RSS feed not populated.	01 Nov 2011
151	http://feeds.foxnews.com/foxnews/world	NCA	Unknown.	01 Nov 2011
152	http://feeds.feedburner.com/newint	NCA	<pubDate> element not populated.	01 Nov 2011
169	http://news.sky.com/sky-news/rss/home/rss.xml	NCA	<pubDate> element not populated.	01 Nov 2011
180	http://rss.sciam.com/ScientificAmerican-Global	SNT	<pubDate> element not populated.	01 Nov 2011

Table A.4: RSS feeds withdrawn from feed and category corpus during the Jul - Nov 2011 data gathering period (Section 10.3.2): cf. *case studies two and three*.

A.3.2 Re-organised RSS feed and category corpus

RSS feed number	RSS feed URL	Initial category	Final category	Number of RSS <item> elements retrieved during Jul - Sep 2011
106	http://feeds.abcnews.com/abcnews/worldnewsheadlines	NCA	NCA	1,053
107	http://feeds.bbci.co.uk/news/world/rss.xml	NCA	NCA	11,107
108	http://rss.cnn.com/rss/cnn_world	NCA	NCA	602
109	http://www.nytimes.com/services/xml/rss/nyt/GlobalHome.xml	NCA	NCA	10,025
112	http://www.msnbc.msn.com/id/3032506/device/rss/rss.xml	NCA	NCA	3,144
113	http://english.pravda.ru/export.xml	NCA	NCA	563
114	http://www.france24.com/en/monde/rss	NCA	NCA	2,413
115	http://www.denofgeek.com/index.rss	Film	EA	332
118	http://www.zdnet.com/search?t=1,7&mode=rss	SNT	SNT	3,005
119	http://feeds.bbci.co.uk/news/technology/rss.xml	SNT	SNT	481
121	http://feeds.bbci.co.uk/news/business/rss.xml	BFE	BFE	3,514
123	http://www.ok.co.uk/rss/32/okfashion	FCL	EA	138
124	http://rss.feedsportal.com/c/592/f/7507/index.rss	Film	EA	1,333
125	http://www.billboard.com/rss/the-feed/	Music	EA	862
127	http://rss.feedsportal.com/c/375/f/434908/index.rss	FCL	EA	702
128	http://evilbeetgossip.film.com/feed/	FCL	EA	1,037

129	http://newsrss.bbc.co.uk/rss/sportonline_world_edition/front_page/rss.xml	Sport	Sport	5,747
130	http://rss.cnn.com/rss/edition_sport	Sport	Sport	542
131	http://www.espn.co.uk/rss/sport/story/feeds/0.xml?type=2	Sport	Sport	4,613
132	http://www.skysports.com/rss/0,20514,12040,00.xml	Sport	Sport	22,298
133	http://feeds.reuters.com/reuters/sportsNews	Sport	Sport	1,806
134	http://feeds.wired.com/wired/index	SNT	SNT	1,218
140	http://feeds.feedburner.com/totalfilm/news/	Film	EA	931
141	http://www.mtv.com/rss/news/movies_full.jhtml	Film	EA	18
143	http://feeds.feedburner.com/thr/film	Film	EA	1,105
144	http://www.hollywoodnews.com/feed/	Film	EA	1,791
148	http://www.eweek.com/rss.xml	SNT	SNT	1,480
149	http://hosted.ap.org/lineups/WORLDHEADS-rss_2.0.xml?SITE=WHIZ&SECTION=HOME	NCA	NCA	7,411
153	http://www.vanityfair.com/services/rss/feeds/everything.xml	FCL	EA	19
158	http://www.infoworld.com/rss.xml	SNT	SNT	828
159	http://www.businessweek.com/rss/bwdaily.rss	BFE	BFE	8,258
160	http://rss.cnn.com/rss/money_topstories.rss	BFE	BFE	3,832
161	http://feeds.reuters.com/reuters/businessNews	BFE	BFE	5,437
162	http://online.wsj.com/xml/rss/3_7432.xml	BFE	BFE	828
163	http://feeds.people.com/people/headlines	FCL	EA	1,171
165	http://www.music-news.com/rss/news.asp	Music	EA	3,798
166	http://loft965.com/feed/	Music	EA	663

167	http://www.mojo4music.com/blog/index.xml	Music	EA	94
168	http://www.musicweek.com/rss.asp?navcode=232	Music	EA	451
170	http://www.theengineer.co.uk/XmlServers/navsectionRSS.aspx?navsectioncode=186	SNT	SNT	626
171	http://feeds.bbci.co.uk/news/science_and_environment/rss.xml	SNT	SNT	608
172	http://feeds2.feedburner.com/nmecom/rss/newsxml	Music	EA	115
173	http://www.spotify.com/uk/feed/	Music	EA	20
174	http://www.prospectmagazine.co.uk/category/magazine/feed/	NCA	NCA	67
175	http://feeds.reuters.com/reuters/worldNews	NCA	NCA	6,279
176	http://www.spiegel.de/international/index.rss	NCA	NCA	562
177	http://newsfeed.time.com/feed/	NCA	NCA	1,373
178	http://www.msnbc.msn.com/id/3032117/device/rss/rss.xml	FCL	SNT	2,172
179	http://feeds.reuters.com/reuters/technologyNews	SNT	SNT	1,945
181	http://feeds.technologyreview.com/technology_review_top_stories	SNT	SNT	469

Table A.5: RSS feed and category corpus following re-organisation (Section 10.3.2): cf. *case study three*.

A.3.3 Corpus of candidate keywords (extract)

Column *Doc* identifies those candidate keywords visualised in the time-series plots in Section 11.8.4 and Appendix A.3.4.

Keyword	**N-gram length**	**No days**	**Margin %**	**RSS feed no/ category/elements**	**Doc**
Afghan	1	30	53.33	107/NCA/TxD	Yes
Anders Behring Breivik	3	10	70.00	175/NCA/TDxTD	Yes
Android	1	10	70.00	178/SNT/TxD	No
Apple/Microsoft	1	31	90.32	148/SNT/TDxTD	Yes
Aston Villa	2	10	80.00	132/Sport/TxD	No
China	1	31	80.65	159/BFE/TDxTD	No
Dark Knight Rises	3	10	30.00	143/EA/TDxTD	No
David Cameron	2	10	50.00	107/NCA/TDxTD	No
England	1	10	90.00	107/All/TDxTD	No
Google	1	31	77.42	118/SNT/TDxTD	No
Guillermo del Toro	3	10	50.00	140/EA/TDxTD	Yes
Kardashian	1	10	70.00	144/EA/TxD and TDxTD	Yes
Libyan rebel/rebels	2	10	50.00	149/NCA/TxD	No
London	1	10	80.00	131/All/TxD	No
Manchester City/United	2	10	70.00	129/Sport/TDxTD	No
NATO	1	10	90.00	175/NCA/TxD	Yes
President	1	30	96.67	107/All/TDxTD	Yes
President Barack Obama	3	10	50.00	107/All/TDxTD	Yes
Stakes at York	3	10	90.00	132/Sport/TDxTD	No
Sir Alex Ferguson	3	30	70.00	132/Sport/TDxTD	No
Syrian troops	2	10	60.00	149/NCA/TxD	No
Syrian forces	2	30	50.00	175/NCA/TxD	Yes
Tiger Woods	2	10	100.00	133/Sport/TDxTD	Yes
UK	1	30	73.00	165/EA/TDxTD	No
United States	2	31	67.74	109/NCA/TDxTD	Yes
Wall Street	2	10	70.00	121/All/TDxTD	Yes
Wesley Sneijder	2	10	90.00	132/Sport/TDxTD	No
World Cup	2	10	100.00	129/Sport/TxD	No

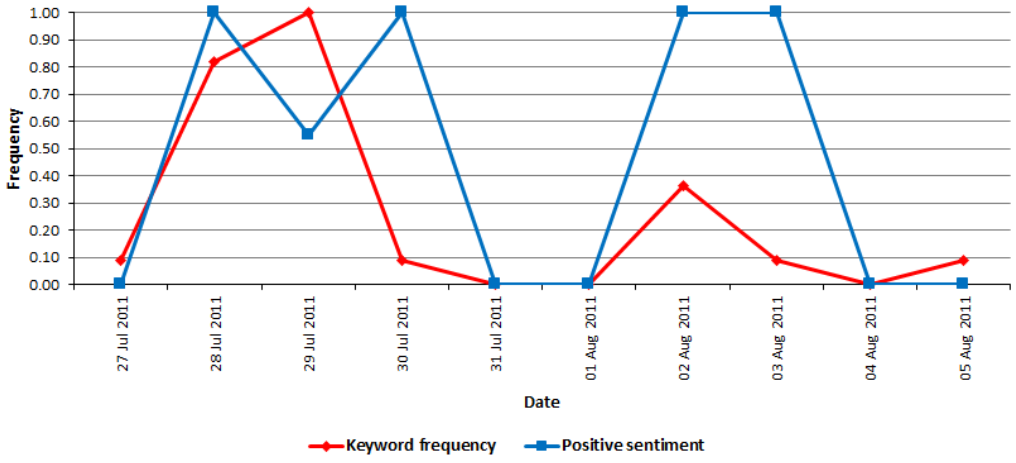
Table A.6: Extract of candidate keyword selection used for keyword frequency/sentiment correlation (Section 11.7.4): cf. *case study three*.

A.3.4 Additional keyword frequency/sentiment correlation plots

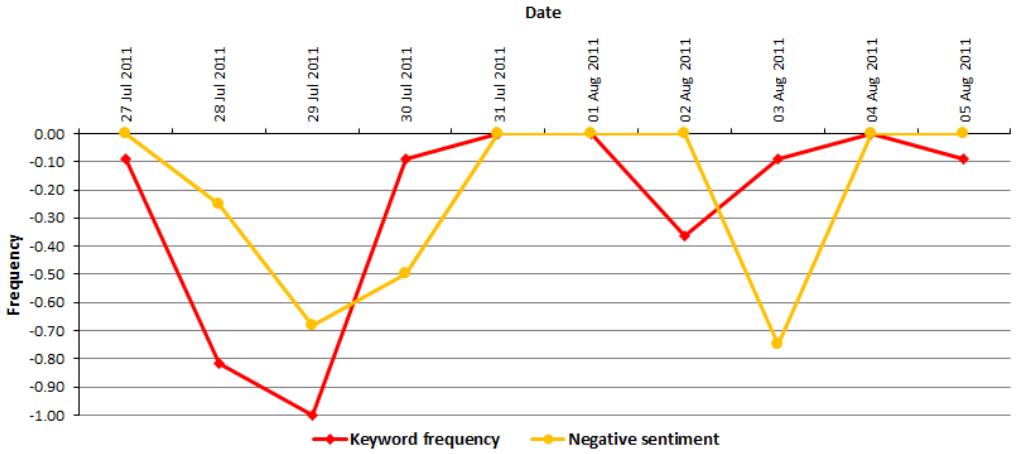
Keyword: *Anders Behring Breivik*

N-gram length	3
RSS feed	Number: 175, Category: NCA, Elements: TDxTD
Duration	Dates: 27 Jul - 05 Aug 2011, Number of days: 10, Margin: 70.00%
Correlation	Positive: 0.46, Negative: -0.55
Comments	Negative sentiment concerning the Jul 2011 killings in Norway.

Table A.7: Keyword: *Anders Behring Breivik*: cf. *case study three.*



(a) Positive sentiment.



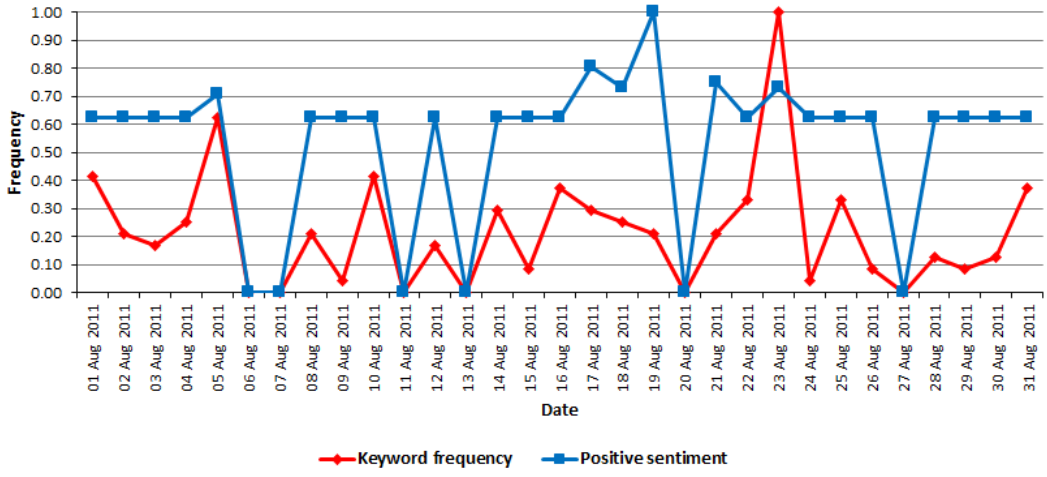
(b) Negative sentiment.

Figure A.1: Keyword: *Anders Behring Breivik*: cf. *case study three.*

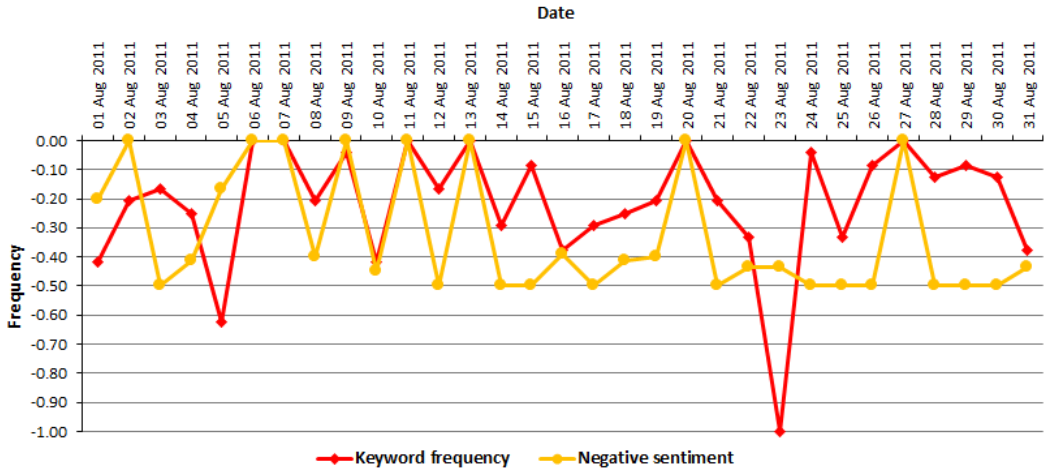
Keyword: ***China***

N-gram length	1
RSS feeds	Number: 159, Category: BFE, Elements: TDxTD
Duration	Dates: 01 - 31 Aug 2011, Number of days: 31, Margin: 80.65%
Correlation	Positive: 0.53, Average negative: -0.49
Comments	Balanced sentiment for Chinese financial and economic affairs.

Table A.8: Keyword: *China*: cf. *case study three.*



(a) Positive sentiment.



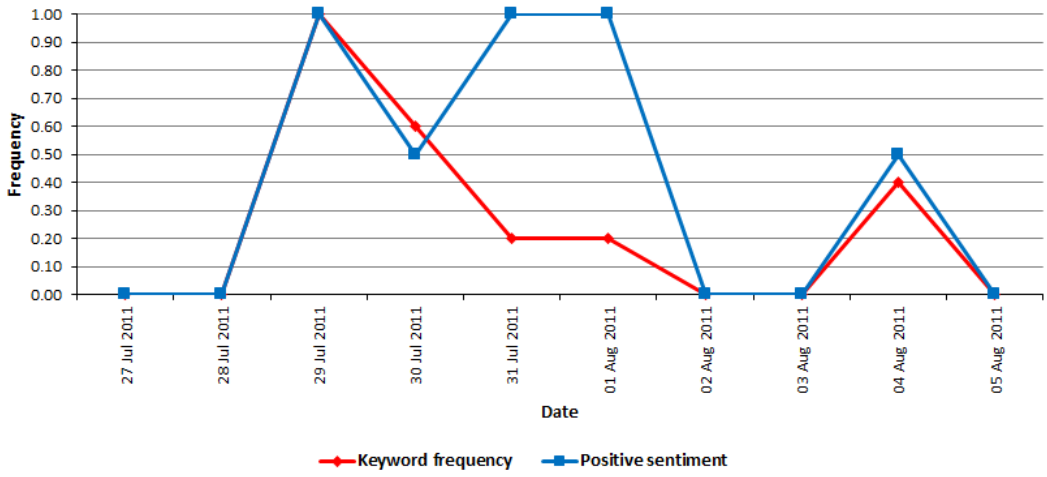
(b) Negative sentiment.

Figure A.2: Keyword: *China*: cf. *case study three.*

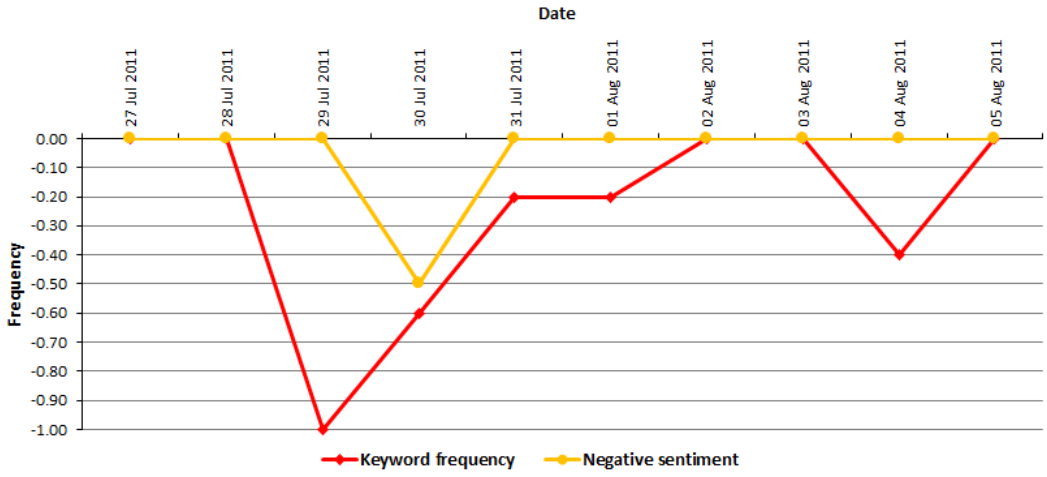
Keyword: ***Guillermo del Toro***

N-gram length	3
RSS feed	Number: 140, Category: EA, Elements: TDxTD
Duration	Dates: 27 Jul - 05 Aug 2011, Number of days: 10, Margin: 50.00%
Correlation	Positive: 0.67, Negative: -0.68
Comments	Positive sentiment *pitching* movies with director Guillermo del Toro.

Table A.9: Keyword: *Guillermo del Toro*: cf. *case study three.*



(a) Positive sentiment.



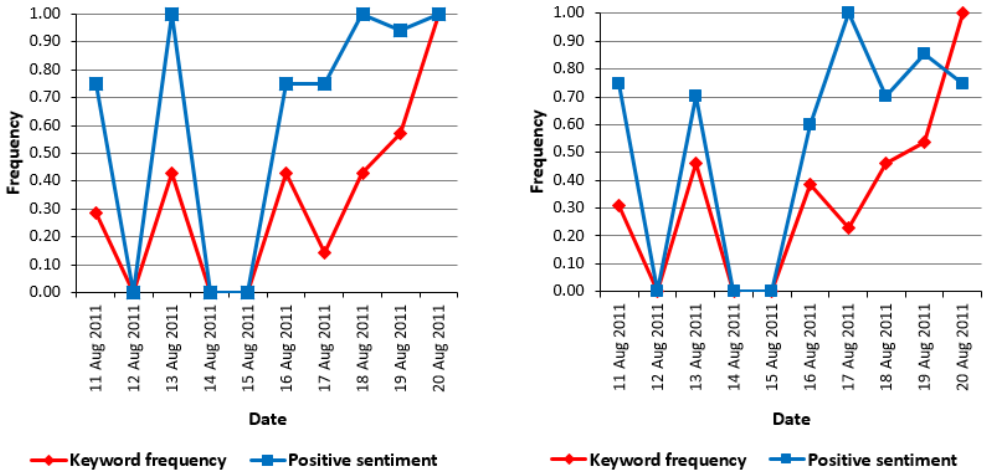
(b) Negative sentiment.

Figure A.3: Keyword: *Guillermo del Toro*: cf. *case study three.*

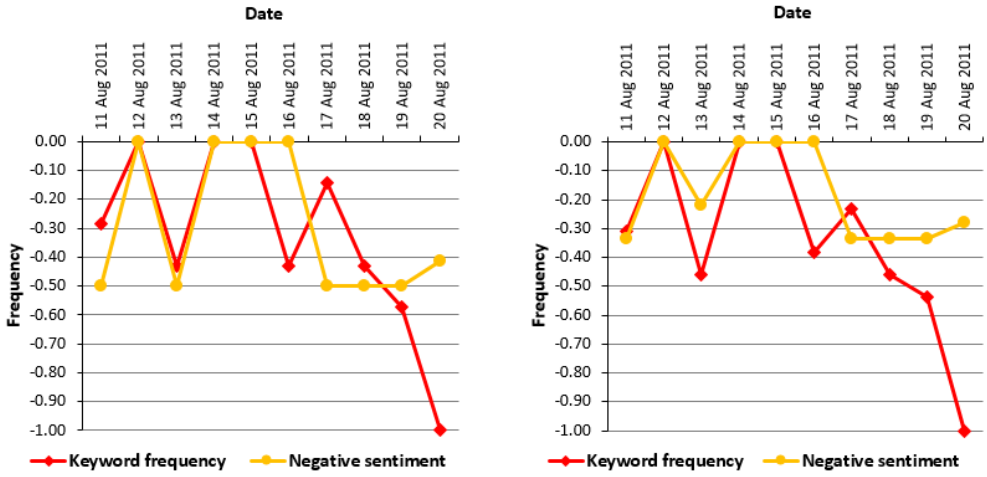
Keyword: *Kardashian*

N-gram length	1
RSS feed	Number: 144, Category: EA, Elements: TxD and TDxTD
Duration	Dates: 11 - 20 Aug 2011, No of days: 10, Average margin: 70.00%
Correlation	Average positive: 0.75, Average negative: -0.69
Comments	The wedding of media personality Kim Kardashian.

Table A.10: Keyword: *Kardashian*: cf. *case study three.*



(a) Positive sentiment TxD (left), TDxTD (right).



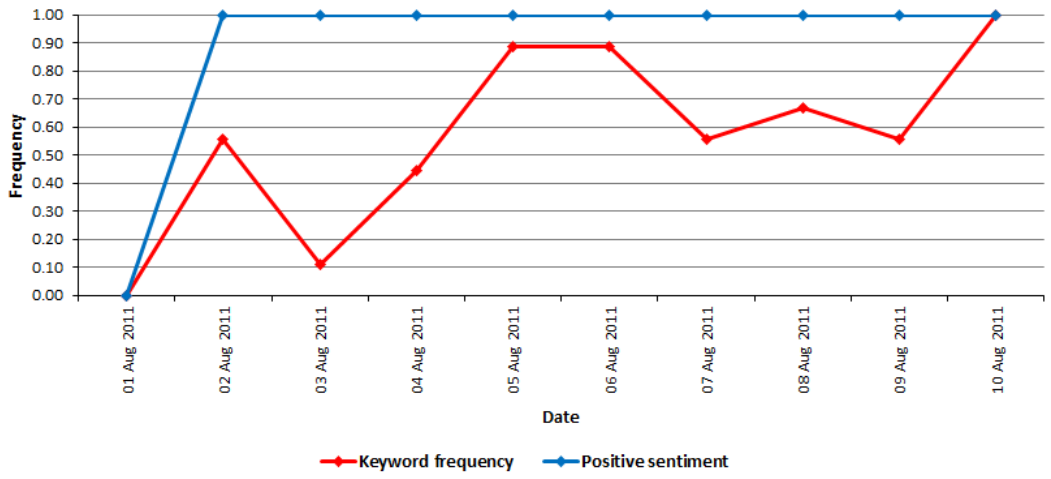
(b) Negative sentiment TxD (left), TDxTD (right).

Figure A.4: Keyword: *Kardashian*, with varying RSS elements: cf. *case study three.*

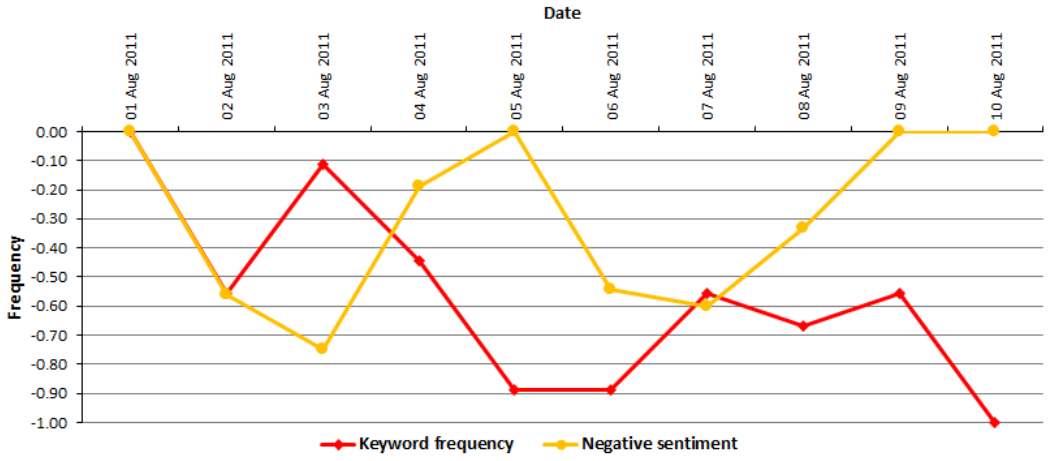
Keyword: *NATO*

N-gram length	1
RSS feeds	Number: 149, Category: SNT, Elements: TxD
Duration	Dates: 01 - 10 Aug 2011, Number of days: 10, Margin: 90.00%
Correlation	Positive: 0.61, Negative: -0.72
Comments	Negative responses to NATO operations in Afghanistan and Libya.

Table A.11: Keyword: *NATO*: cf. *case study three.*



(a) Positive sentiment.



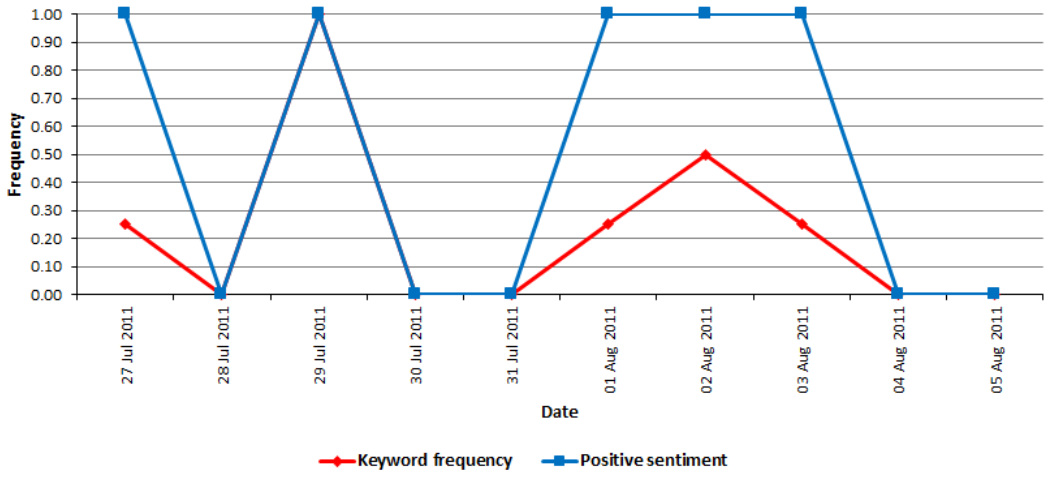
(b) Negative sentiment.

Figure A.5: Keyword: *NATO*: cf. *case study three.*

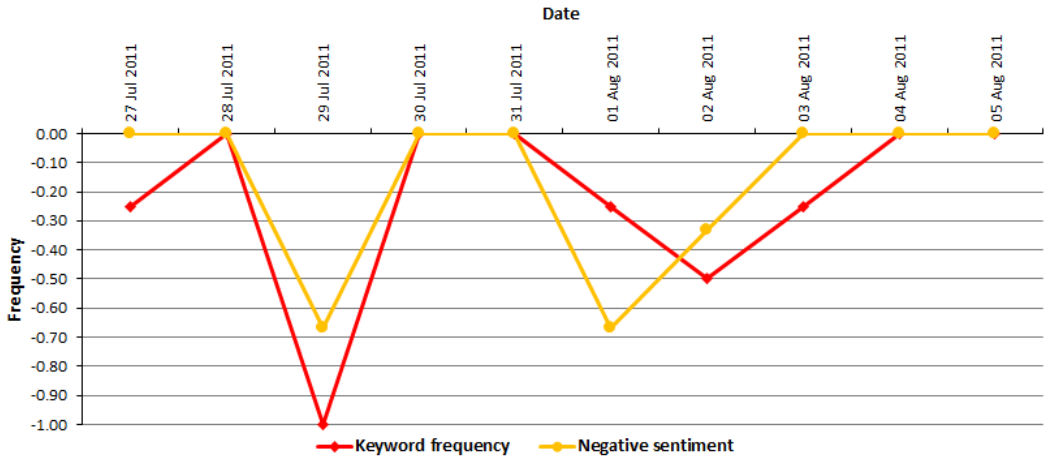
Keyword: ***President Barack Obama***

N-gram length	3
RSS feed	Number: 107, Category: NCA, Elements: TDxTD
Duration	Dates: 27 Jul - 05 Aug 2011, Number of days: 10, Margin: 50.00%
Correlation	Positive: 0.74 Negative: -0.42
Comments	Domestic and foreign troubles for President Barack Obama.

Table A.12: Keyword: *President Barack Obama*: cf. *case study three.*



(a) Positive sentiment.



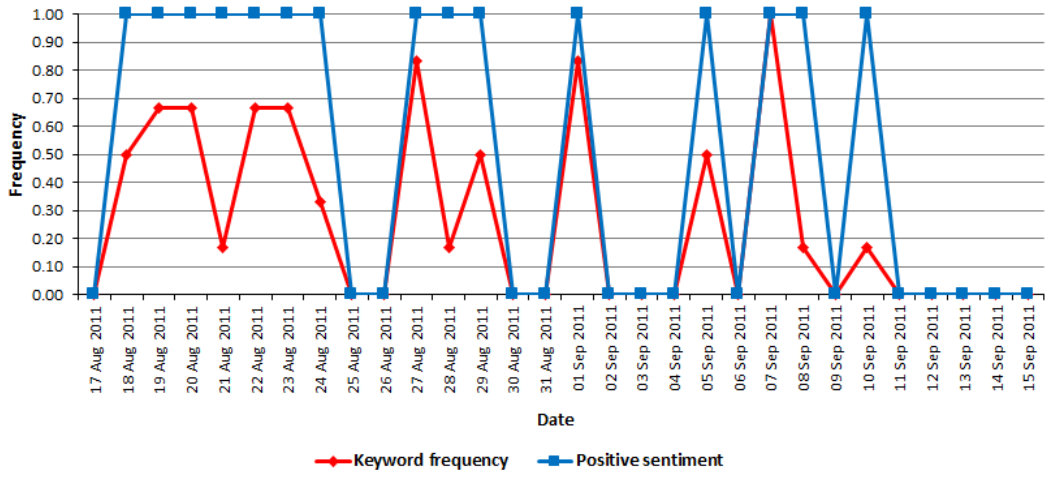
(b) Negative sentiment.

Figure A.6: Keyword: *President Barack Obama*: cf. *case study three.*

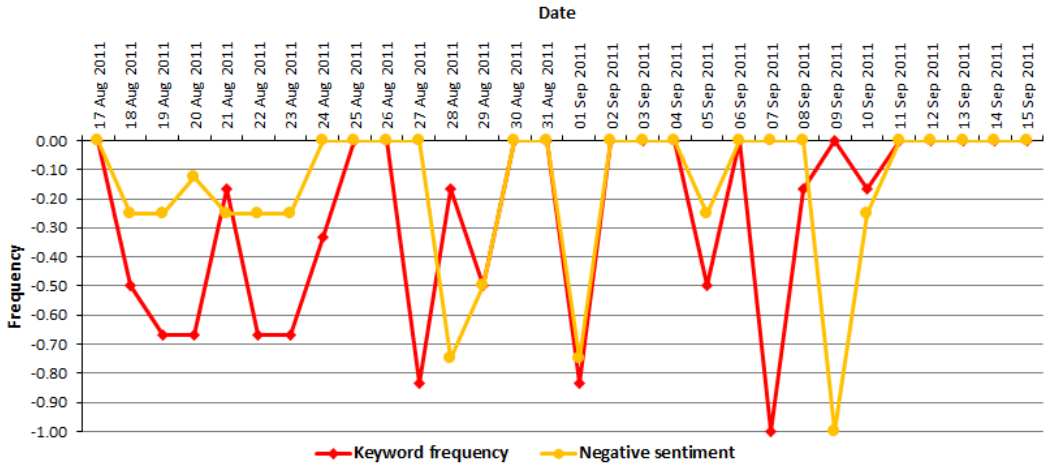
Keyword: *Syrian forces*

N-gram length	2
RSS feeds	Number: 175, Category: NCA, Elements: TxD
Duration	Dates: 17 Aug - 15 Sep 2011, Number of days: 30, Margin: 50.00%
Correlation	Positive: 0.81, Negative: -0.77
Comments	Anti-government violence and the response by the state in Syria.

Table A.13: Keyword: *Syrian forces*: cf. *case study three.*



(a) Positive sentiment.



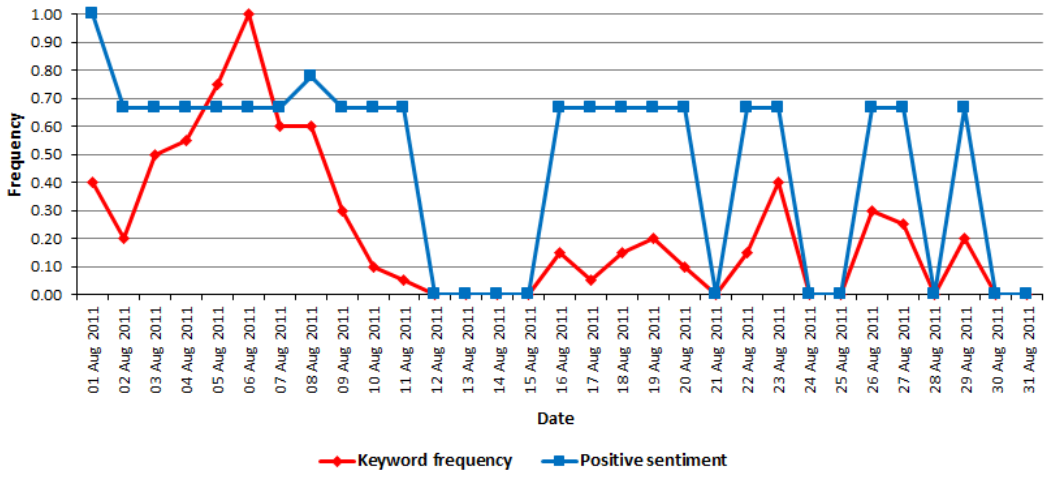
(b) Negative sentiment.

Figure A.7: Keyword: *Syrian forces*: cf. *case study three.*

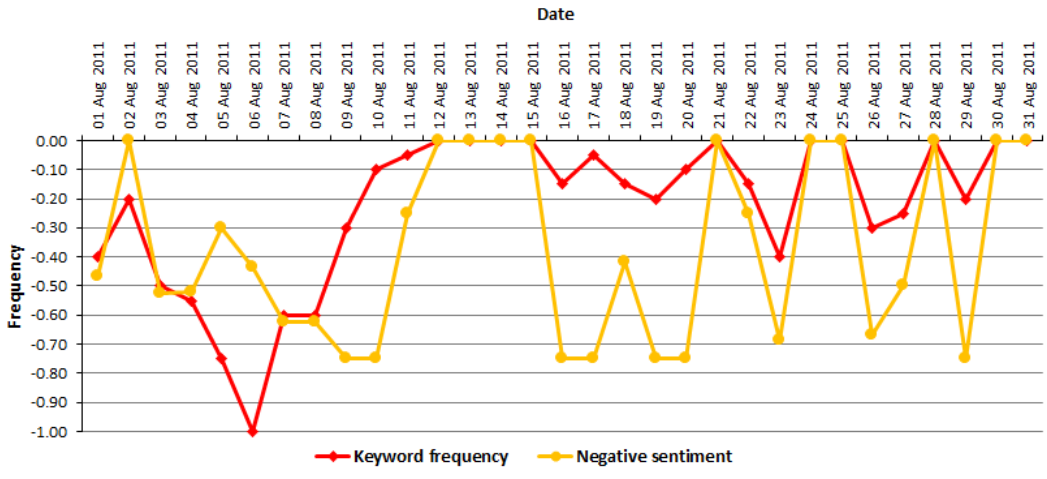
Keyword: ***United States***

N-gram length	2
RSS feed	Number: 109, Category: NCA, Elements: TDxTD
Duration	Dates: 01 - 31 Aug 2011, Number of days: 31, Margin 67.74%
Correlation	Positive: 0.61, Negative: -0.54
Comments	Domestic, economic and foreign difficulties in the USA.

Table A.14: Keyword: *United States*: cf. *case study three*.



(a) Positive sentiment.



(b) Negative sentiment.

Figure A.8: Keyword: *United States*: cf. *case study three*.

Appendix B

Glossaries

B.1 Glossary of products

This glossary lists the open-source, third-party products[1] [2] from the Java *ecosystem* that have been used in our myDS and vRSS software on a black-box, mash-up basis. Certain products used in compiling the appendices of this thesis are also listed herein. If a listed product is not open-source, it is otherwise freely available for use from its publishers. Descriptive quotations are also taken from the publishers' web sites.

Android The Android OS (`https://www.android.com/`) for mobile devices produced by Google [148].

Apache Commons Mathematics Library A self-contained library of "mathematics and statistics components addressing the most common problems not available in the Java programming language", available at `https://commons.apache.org/proper/commons-math/`.

Apache JMeter Available at `http://jmeter.apache.org/`, Apache JMeter "is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions."

[1] The distribution of the principal third-party products in our myDS and vRSS software for our case studies is described in Sections 4.7.1 and 4.8.1 respectively.

[2] Approximate counts of the number of lines of code in the principal third-party products used in our software, together with equivalent counts of the number of *indigenous* lines of code in myDS and vRSS, are listed in Appendix D.2.

Apache Tomcat Apache Tomcat, available at `http://tomcat.apache.org/`, is an "open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies." Version 6.0.26 of Apache Tomcat is used by myDS and vRSS as web server (Section 4.5).

Black Duck Open Hub "Black Duck Open Hub (formerly Ohloh.net) is an online community and public directory of free and open source software (FOSS), offering analytics and search services for discovering, evaluating, tracking, and comparing open source code and projects." Black Duck Open Hub can be found at `https://www.openhub.net/`.

CLOC CLOC "counts blank lines, comment lines, and physical lines of source code in many programming languages." It is available at `https://github.com/AlDanial/cloc`.

Google Charts A service provided by Google, available at `https://developers.google.com/chart/?hl=en`, which allows data to be visualised within a web page using a "rich gallery of interactive charts and data tools."

jforum jforum is a Java-based BBS (Appendix B.2) available from `http://jforum.net/`. Version 2.1.8 is used in myDS.

jQuery A "fast, small, and feature-rich JavaScript library" available at `https://jquery.com/`: version 1.5.1 was employed by vRSS for *sliders* on pages.

Js-Treemap A JavaScript implementation of a tree-map (Appendix B.2) by Cowie, available at `http://js-treemap.sourceforge.net/`.

jsoup jsoup is a "Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods." jsoup is available at `http://jsoup.org/`.

Lucene Lucene "is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform." Lucene is available at `https://lucene.apache.org/core/` and version 3.0.2 is used in our vRSS software.

MySQL MySQL (`http://mysql.com/`) is a popular RDBMS supported by Oracle (`http://www.oracle.com/`). Several versions of MySQL were used during the development of our myDS and vRSS software, and their use in the case studies for our paradigms. Version 5.5.16 of MySQL, which includes extensions to the SQL standard, was the earliest version used and its compliance with ANSI/ISO standards [203] is described on the product's web site at `http://dev.mysql.com/doc/refman/5.5/en/compatibility.html`.

NetBeans NetBeans is an open-source IDE primarily intended for Java, but which supports several other programming languages. NetBeans is sponsored by Oracle (`http://www.oracle.com/`), and several versions were used in the development of our software. NetBeans is available at `https://netbeans.org/`.

Quartz Scheduler Quartz Scheduler, available from Terracotta at `https://quartz-scheduler.org/`, is an:

> "open source job scheduling library that can be integrated within virtually any Java application - from the smallest stand-alone application to the largest e-commerce system."

Version 1.8.3 of Quartz Scheduler is used in myDS and vRSS.

Rome Rome, available at `http://rometools.github.io/rome/`, "is a set of RSS and Atom Utilities for Java" that provides:

> "ROME includes a set of parsers and generators for the various flavors of syndication feeds, as well as converters to convert from one format to another. The parsers can give you back Java objects that are either specific for the format you want to work with, or a generic normalized SyndFeed class that lets you work on with the data without bothering about the incoming or outgoing feed type."

SentiStrength SentiStrength is "a lexicon-based classifier" [397] written in Java. SentiStrength 2, available at `http://sentistrength.wlv.ac.uk/`, was implemented in vRSS for the sentiment analysis component of case study three (Chapter 11).

Simple XML Simple XML is a "high performance XML serialization and configuration framework for Java. Its goal is to provide an XML framework that enables rapid development of XML configuration and communication systems."

Version 2.7 of Simple XML, available at `http://simple.sourceforge.net/home.php`, is used for the REST (Appendix B.2) interface which resides between vRSS and its Android OS client (Appendix C).

TinyMCE TinyMCE is a "platform-independent web-based JavaScript WYSIWYG HTML editor control". Version 3.4.4, available at `http://www.tinymce.com/`, was employed by vRSS to allow entry of visualisation-related comments by the student corpora during case study two (Chapter 8).

Weka Weka is a "collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes."

Version 3.7.x of Weka was used during the classification component of our third case study, and is available at `http://www.cs.waikato.ac.nz/ml/weka/index.html`.

B.2 Glossary of terminology

This glossary concerns IT/industry terminology used in connection with the work presented in this thesis.

Atomicity, consistency, isolation and durability (ACID) The four ACID properties of the classical relational database model (Codd [75]), described by Haerder and Reuter [164] with regard to database transactions, i.e. a *transaction* is defined as "a short sequence of interactions with the database", for CRUD operations, are: (1) *atomicity* which requires that each transaction is "all-or-nothing", i.e. if one part of the transaction fails, then the entire transaction fails and is (if necessary) rolled back. (2) *consistency* where a transaction reaching its normal end, commits its results and "preserves the consistency of the database." (3) *isolation* refers to the consistency of the database's state where the events "within a transaction must be hidden from other transactions running concurrently." (4) *durability* where once "a transaction has been completed and has committed its results to the database, the system must guarantee that these results survive

any subsequent malfunctions."

Accuracy Defined by Thealing's *Data Mining Glossary* at `http://www.thearling.com/glossary.htm` as "A measure of a predictive model that reflects the proportionate number of times that the model is correct when applied to data." Accuracy is determined by the formula:

$$\mathrm{P} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

TP, TN, FP, and FN are defined elsewhere in Appendix B.2.

App The term *app* is defined by Search Mobile Computing at `http://searchmobilecomputing.techtarget.com/definition/app` as:

> "an abbreviated form of the word "application." An application is a software program that's designed to perform a specific function directly for the user or, in some cases, for another application program."

The term *app* is frequently used in conjunction with the mobile platform (Appendix B.2).

Aspect-oriented programming (AOP) According to Liu [232], in application software "Tasks like logging, performance profiling, validation, error checking and handling, transaction management and so on, are called *crosscutting concerns*, meaning that they are often intermingled with the main business logic code." AOP concerns the use of *aspects* to modularise crosscutting concerns which apply to more than once class, i.e. "an *aspect* is essentially a class that can be applied to altering the behavior of base code by *applying advice* (additional crosscutting behavior)".

Association rules Association rules (Aggarwal and Yu [10]) concern strong links between apparently unrelated items in a dataset (Appendix B.2). Such information is useful for organisations to decide customer targeting, shelving and sales promotions, e.g. according to Martin [248], the "classic application is market basket analysis: find items that are frequently purchased together by a customer." Such a rule might take the form that "customers who buy item A also often buy item B", and can be "identified by checking that the rule has some minimum confidence and support."

The *support* of a rule $X \Rightarrow$Y is defined by Aggarwal and Yu [10] as "the fraction of transactions which contain both X and Y." The *confidence* of a rule $X \Rightarrow$Y is "the fraction of transactions containing X, which also contain Y. Thus, if we say that a rule has 90% confidence then it means that 90% of the tuples containing X also contain Y."

Atom Atom is an XML-based medium for syndicating content which was written as an alternative to Winer's RSS 2.0 (Section 2.2.2). The IETF is currently responsible for the Atom 1.0 standard which was defined in 2005 and is available at `http://tools.ietf.org/html/rfc4287`.

Bag-of-words (BoW) The term *bag-of-words* was defined by Salton and McGill [344] as an "orderless document representation", where the frequencies of terms in a group of documents are recorded in vectors irrespective of the position of each term in the original document(s). Silić and Bašić [358] described BoW as an instance of VSM (Appendix B.2).

To demonstrate BoW, with apologies to the traditional English folk song *Greensleeves*, documents may be formed from the text of each line of the chorus:

"Greensleeves was all my joy,
Greensleeves was my delight,
Greensleeves was my heart of gold,
And who but my lady Greensleeves."

Giving us a corpus of thirteen words (frequencies are listed in brackets): (1) $greensleeves_{(4)}$, (2) $was_{(3)}$, (3) $all_{(1)}$, (4) $my_{(4)}$, (5) $joy_{(1)}$, (6) $delight_{(1)}$, (7) $heart_{(1)}$, (8) $of_{(1)}$, (9) $gold_{(1)}$, (10) $and_{(1)}$, (11) $who_{(1)}$, (12) $but_{(1)}$ and (13) $lady_{(1)}$. The column vectors below represent the word frequencies in each line of the chorus:

Document	**Word frequencies**												
Line no	Greensleeves	was	all	my	joy	delight	heart	of	gold	and	who	but	lady
1	[1	1	1	1	1	0	0	0	0	0	0	0	0]
2	[1	1	0	1	0	1	0	0	0	0	0	0	0]
3	[1	1	0	1	0	0	1	1	1	0	0	0	0]
4	[1	0	0	1	0	0	0	0	0	1	1	1	1]

Ballpark Investopedia at `http://www.investopedia.com/terms/b/ballpark-figure.asp` defines *ballpark* as:

> "A rough numerical estimate or approximation. Ballpark figures are commonly used by accountants, salespersons and other professionals to estimate current or future results. A stockbroker could use a ballpark figure to estimate how much money a client might have at some point in the future, given a certain rate of growth."

Big-data Frequently associated with NoSQL tools (Appendix B.2), big-data is defined by Techopedia at `https://www.techopedia.com/definition/27745/big-data` as:

> "a process that is used when traditional data mining and handling techniques cannot uncover the insights and meaning of the underlying data. Data that is unstructured or time sensitive or simply very large cannot be processed by relational database engines. This type of data requires a different processing approach called big data, which uses massive parallelism on readily-available hardware."

Boilerplate According to Techopedia at `http://www.techopedia.com/definition/1259/boilerplate`, boilerplate "is any form of writing that can be or is reused multiple times with minimal changes to the original content." Thus, in terms of computer programming:

> "this term refers to boilerplate code, which is code that has proved to efficient [*sic*] and can be extended to many applications. Code to produce standard mathematical operations, template programs, and most notably, open-source codes may all be considered boilerplate code."

Bulletin board system (BBS) Defined by Techopedia at `http://www.techopedia.com/definition/2481/bulletin-board-system-bbs`, the term BBS:

> "refers to text-based online communities that users can log into over the Internet using dedicated software. The bulletin board system predates the World Wide Web and was a popular application for Telnet users. Bulletin board systems were an early example of the Internet's ability to foster large online communities."

Chi-square (χ^2) A chi-square test, often written as χ^2, is designed to analyse categorical data. Manning and Schütze [244] wrote that the "χ^2 test is applied to 2-by-2 tables" where "The essence of the test is to compare the observed frequencies in the table with the frequencies expected for independence. If the difference between observed and expected frequencies is large, then we can reject the null hypothesis of independence."

Class diagram A class diagram provides a *static* view of an application. Tutorials Point at `http://www.tutorialspoint.com/uml/uml_class_diagram.htm` describes class diagrams for:

> "visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.
>
> The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.
>
> The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a *structural diagram*."

Controlled vocabulary Defined by the American Society for Indexing at `http://www.taxonomies-sig.org/about.htm`, a controlled vocabulary is:

> "an authoritative list of terms to be used in indexing (human or automated). A controlled vocabulary for a project might actually include multiple authority files for different kinds of terms.
>
> Controlled vocabularies are used to ensure consistent indexing, particularly when indexing multiple documents, periodical articles, web pages or sites, etc. They may also be used when indexing a single work, such as a [*sic*] encyclopedia, by multiple indexers.
>
> Controlled vocabularies do not necessarily have any structure or relationships between terms within the list. Controlled vocabularies are often used for name authorities (proper nouns), such as persons, organization names, company names, etc."

Cosine similarity Defined by Huang [187] where if "documents are represented as term vectors, the similarity of two documents corresponds to the correlation between the vectors.

This is quantified as the cosine of the angle between vectors, that is, the so-called cosine similarity. Cosine similarity is one of the most popular similarity measure applied to text documents, such as in numerous information retrieval applications and clustering too."

CRON `CRON` is described by Wikipedia (`https://en.wikipedia.org/wiki/Cron`) as a "time-based job scheduler" found in the UNIX operating system.

Cross-validation Cross-validation is defined by the authors of Weka (Appendix B.1), i.e. Witten and Eibe [447], as:

> "In cross-validation, you decide upon a fixed number of *folds*, or partitions of the data. Suppose we use three. Then the data is split into three approximately equal partitions and each in turn is used for testing and the remainder is used for training. That is, use two-thirds for training and one-third for testing and repeat the procedure three times so that, in the end, every instance has been used exactly once for testing."

Witten and Eibe [447] describe ten-fold cross-validation as the convention:

> "Why 10? Extensive tests on numerous datasets, with different learning techniques, have shown that 10 is about the right number of folds to get the best estimate of error, and there is also some theoretical evidence that backs this up. Although these arguments are by no means conclusive, and debate continues to rage in machine leaning and data mining circles about what is the best scheme for evaluation, 10-fold cross-validation has become the standard method in practical terms."

Data flow diagram (DFD) A DFD is defined by Pressman [309] as a "graphical technique that depicts information flow and the transforms that are applied as data moves from input to output." We have employed Yourdon's DFD notation [460] in this thesis, illustrated in Figure B.1, to concisely depict the processes and data flows in our myDS and vRSS software.

Dataset A dataset is a collection of related information made up of distinct elements concerning a given subject. According to Renear et al. [323], "four basic features can be identified as common to most definitions" of a dataset: (1) *grouping* refers to datasets as being "data treated collectively, as a unit." (2) *content* in terms of "the constituents of a

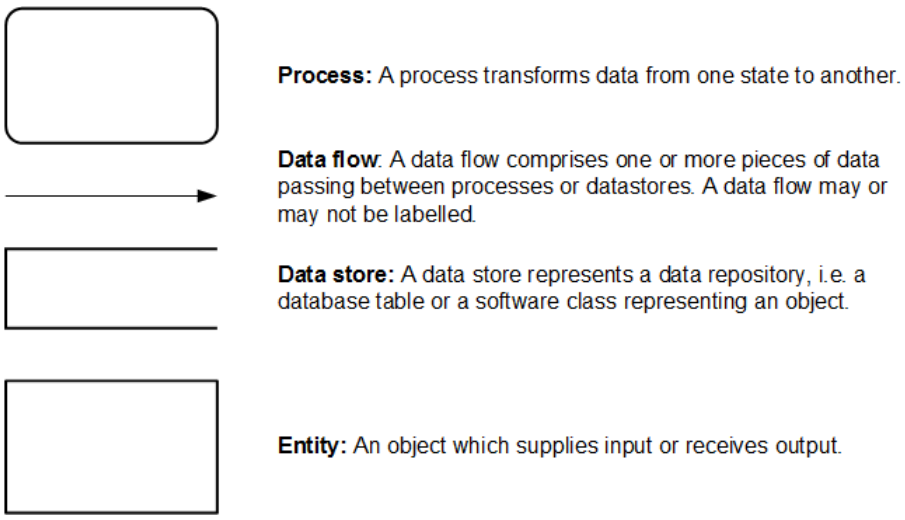


Figure B.1: Yourdon's DFD notation [460].

dataset are things of some particular kind." (3) *relatednesss* identifies "the grouped things as all being of the same general kind of entity", and (4) *purpose* where the dataset has an "intended application".

Entity relationship diagram (ERDM) An ERDM (Chen [66]) provides a visual representation of the tables and relationships forming the data model of a database implemented in an RDBMS. Although several alternative styles of notation exist for an ERDM, in this thesis we use the *crow's feet* and *boxes* notation by Martin [247]. Alternative notations for ERDMs are discussed by Song and Froelich [367].

Extract, transform and load (ETL) ETL is a process used to migrate data between data sources or application components, where the data from the source is often *cleansed*, i.e. transformed, to make it compatible with the target before being loaded into the target.

F-measure The F-measure metric, originated by Rijsbergen [326], is defined in the following form:

$$\text{F-measure} = \frac{2 * (\text{precision} * \text{recall})}{\text{precision} + \text{recall}}$$

Which combines precision and recall, both of which are frequently used in document classification. Precision, recall, TP, TN, FP and FN are defined elsewhere in Appendix B.2.

Feature selection At `https://en.wikipedia.org/wiki/Feature_selection`, feature selection is defined as "the process of selecting a subset of relevant features (variables, predictors) for use in model construction."

Heat-map First used by Kinney (`https://en.wikipedia.org/wiki/Cormac_Kinney`) in 1991, a heat-map is defined by the BusinessDictionary (`http://www.businessdictionary.com/definition/heatmap.html`) as a "graphical representation of data using colors to indicate the level of activity, usually using darker colors to indicate low activity, and brighter colors to indicate high activity. For example a heatmap could indicate the number of foreclosures in a geographical area during a set period of time."

Infographic With reference to Mashable at `http://mashable.com/category/infographics/`, infographics are:

> "graphic visual representations of information, data or knowledge. These graphics present complex information quickly and clearly such as in signs, maps, journalism, technical writing, and education."

Information gain (IG) Information gain is a method of feature selection (Appendix B.2) which, according to Yang and Pederson [459], measures the "number of bits of information obtained for category prediction by knowing the presence or absence of a term in a document."

Java database connectivity (JDBC) The JDBC API is defined by [286] as "the industry standard for database-independent connectivity between the Java programming language and a wide range of databases—SQL databases and other tabular data sources, such as spreadsheets or flat files. The JDBC API provides a call-level API for SQL-based database access."

***k*-nearest neighbour (KNN)** KNN, a supervised clustering algorithm, is defined by Wikipedia at `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`, wherein an object is classified "by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor."

KNN is frequently used in statistical estimation and pattern recognition, but it is not related to the k-means unsupervised clustering method.

Log-likelihood ratio test Defined by IBM at `http://www-01.ibm.com/support/knowledgecenter/SSLVMB_21.0.0/com.ibm.spss.statistics.cs/mixed_diet_intro_04.htm`, a log-likelihood ratio test is a "test of the sufficiency of a smaller model versus a more complex model. The null hypothesis of the test states that the smaller model provides as good a fit for the data as the larger model. If the null hypothesis is rejected, then the alternative, larger model provides a significant improvement over the smaller model."

The resulting ratio can be used to calculate a p-value (Appendix B.2) to decide whether to reject the null hypothesis in favour of the alternative hypothesis.

Mash-up A mash-up is defined by Pressman and Lowe [308] as:

> "a *hybrid* Web application that integrates content from multiple (usually third-party) sources in order to create a novel synergistic outcome. Mash-ups usually access the content-rich environment of major providers (such as Google, Amazon, and eBay) using a simple public interface. Even when there is no public interface, it is often possible to utilize third-party data through simple "screen-scraping" of the data available on websites. The rich interactivity of many mash-ups typifies the evolving direction of Web 2.0."

Metadata According to Bretherton and Singley [53], metadata "typically describes the structure of a data set or the interpretation to be placed on collections of similar items within that data set, rather than focusing on the individual instances usually regarded as primary data." The authors distinguished between two types of metadata:

1. **Control metadata:** This is concerned with organisation "directly affecting database or other computer system operations, though it is desirable that it be intelligible to scientists and/or database managers."
2. **Guide metadata:** This is "intended solely for use by humans and is expressed in natural language."

Bretherton and Singley also wrote that the "boundary between these two kinds of metadata is not fixed." These types broadly correspond with NISO [275] where: (1) *administrative*

metadata concerns resources and includes such properties as when and how an object was created, authorities and other technical information, (2) *descriptive* metadata "describes a resource for purposes such as discovery and identification." This includes elements such as a title, abstract, author, and keywords describing a periodical. (3) *structural* metadata concerns structure, e.g. "how pages are ordered to form chapters."

In our paradigms and their case studies, we are concerned with the semi-structured format of RSS (Section 2.2.3). Therefore, the XML elements and attributes of RSS form *structural* metadata because of the organisational role they perform. Use of *guide* metadata in our software is less *clear* with regard to the definition provided by Bretherton and Singley [53]. For this reason we consider labels of user interface (UI) controls and *About* pages in myDS and vRSS to be more readily described as *descriptive* metadata, as defined by NISO. We also cite the word-cloud and tree-map visualisation types, illustrated in Figure B.2, as examples of *descriptive* metadata because both types employ the frequency of each keyword displayed to represent its prominence in the dataset being visualised. An example of *administrative* metadata in our work concerns the `<channel>` element in an RSS feed, or the *created by* elements of datasets and visualisations in myDS and vRSS for case studies one and two for our first paradigm.

Mobile platform Defined by Techopedia (`https://www.techopedia.com/definition/30688/mobile-application-platform`):

> "A mobile application platform supports mobile application development using various tools for different programming languages and provides an application programming interface to allow interactivity between software packages. These tools include those used for testing applications, measuring mobile analytics and creating interfaces for profiling application performance.
>
> Vendors usually offer a mobile application platform to clients that want to go mobile or enter the mobile market. The platform includes migration tools and resources that support a mobile interface, or a development environment that allows creating new apps aimed at the Apple and Android markets. A platform approach to mobile application development helps provide a comprehensive model with tool suites that are accessible to developers and other users."

Mutual information (MI) Manning and Schütze [244] defined MI as "a symmetric, non-negative measure of the common information" in two variables. Moreover, MI is a

measure of independence because: "It is 0 only when two variables are independent", but for "two dependent variables, mutual information grows not only with the degree of dependence, but also according to the entropy of the variables."

N-gram An n-gram is defined by Wikipedia at `http://en.wikipedia.org/wiki/N-gram` as:

> "In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called shingles.
>
> An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram". Larger sizes are sometimes referred to by the value of n, e.g., "four-gram", "five-gram", and so on."

Table B.1 demonstrates the generation of n-grams from a simple piece of text to the level of bigrams, i.e. *RSS is a dialect of XML*. In this example, stop words (Appendix B.2) have been edited.

N-gram	N-gram length	Frequency
dialect	1	1
dialect xml	2	1
rss	1	1
rss dialect	2	1
xml	1	1

Table B.1: N-grams (to bigram level) generated from phrase *RSS is a dialect of XML*, with stop words edited.

Named entity recognition (NER) According to the Stanford Natural Language Processing Group [371], NER "labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names."

Natural language processing (NLP) Defined by Search Content Management at `http://searchcontentmanagement.techtarget.com/definition/natural-language-processing-NLP`, NLP is "the ability of a computer program to understand

human speech as it is spoken." Furthermore, "computers traditionally require humans to "speak" to them in a programming language that is precise, unambiguous and highly structured or, perhaps through a limited number of clearly-enunciated voice commands. Human speech, however, is not always precise -- it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context."

"Current NLP approaches are based upon machine learning" involving linguistic tasks such as text mining (Section 2.6) and NER.

NoSQL The technologies used to analyse and persist big-data are generically known as NoSQL, i.e. *no SQL*, *not only SQL* or *not relational*. This term refers to the non-use of traditional relational data structures (Section 4.5.7), and as described by Cattell [61], the use of key/value, document and extensible record stores, as well as "RDBMSs that provide horizontal scaling", as "categories" of data stores to deal with the volume of data generated.

***P*-value** A p-value is used to test the validity of a hypothesis in order to test the significance of its results. The hypothesis to be validated is often known as the *alternative* hypothesis, whereas the *null* hypothesis assumes no relationship between the two samples being measured.

If a p-value is below 0.05 it means that the alternative hypothesis can be interpreted to be *substantially* correct and the null hypothesis is rejected: a p-value above 0.05 means that the alternative hypothesis is unreliable so the null hypothesis is accepted.

Part-of-speech (POS) POS is defined by the Stanford Natural Language Processing Group [371] as the reading of text by software which "assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'."

Pearson's correlation coefficient Formulated by Pearson [294] in 1895, the product-moment correlation coefficient measures the strength of the linear relationship between two variables within a range between +1 (*positive* or *strong*) and -1 (*negative* or *weak*). At `https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient`, Wikipedia defines the calculation as being based upon the "covariance of the two variables divided by the product of their standard deviations." The formula is given as:

$$P_{x,y} = \frac{cov(x,y)}{\sigma_x \sigma_y}$$

Where *cov* is the covariance, and σ_x gives the SD of x. When plotted on a typical x - y chart, a higher positive coefficient value indicates that the value of y increases as the value of x increases: similarly a lower negative value sees y decrease as x increases. A coefficient of zero represents no correlation.

According to Lærd Statistics at `https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php#linear`, there are five assumptions "that are made with respect to Pearson's correlation". These assumptions concern: (1) variables being "either interval or ratio measurements", (2) that "the variables must be approximately normally distributed", (3) the presence of a "linear relationship between two variables", (4) the minimisation or removal of outliers, and (5) *homoscedasticity* where a typical x - y chart of a correlation demonstrates a *substantially* straight line because of a common variance amongst the variables.

Persistence In Driscoll et al. [97], a data structure is considered to be *persistent*:

> "if it supports access to multiple versions. The structure is partially persistent if all versions can be accessed but only the newest version can be modified, and fully persistent if every version can be both accessed and modified."

We define persistence to apply to the writing of the current *state* of a data structure or an object, i.e. the current set of values of its defining attributes, to a platform's file system or to database storage. As defined by Date [88], persistence suggests "that database data differs in kind from other, more ephemeral, data such as input data, output data, control statements, work queues, software control blocks, intermediate results, and more generally any data that is transient in nature." We consider this *partial* persistence to be applicable to our software, where only the current condition or state of an object can be readily "accessed and modified" (Driscoll et al. [97]) following persistence.

Podcast Defined by the OED at `http://www.oxforddictionaries.com/definition/english/podcast`, a *podcast* is "A digital audio file made available on the Internet for downloading to a computer or mobile device, typically available as a series, new instalments of which can be received by subscribers automatically."

Precision In the documents relevant to a query, precision "is the proportion of the predicted positive cases that were correct" (Bellaachia [31]). Precision is calculated by the formula:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

TP and FP are defined elsewhere in Appendix B.2.

Proof of concept As defined by Techopedia at `http://www.techopedia.com/definition/4066/proof-of-concept-poc`, a *proof of concept* is:

> "a demonstration, the purpose of which is to verify that certain concepts or theories have the potential for real-world application. POC is therefore a prototype that is designed to determine feasibility, but does not represent deliverables.
>
> Proof of concept is also known as proof of principle."

Query-expansion Manning et al. [243] wrote that "In *query expansion*, on the other hand, users give additional input on query words or phrases, possibly suggesting additional query terms. Some search engines (especially on the web) suggest related queries in response to a query; the users then opt to use one of these alternative query suggestions." The authors further wrote that the "most common form of query expansion is global analysis, using some form of thesaurus."

Random forest A random forest is defined by Ho [178] as a "method for increasing generalization accuracy through systematic creation and use of multiple trees." This need arises to prevent decision trees being too complex to overfit their training data.

Recall Recall concerns the documents relevant to a query that are successfully retrieved, i.e. the "proportion of positive cases that were correctly identified" (Bellaachia [31]). Recall is calculated using the formula:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

TP and FN are defined elsewhere in Appendix B.2.

Representational state transfer (REST) REST provides web services which employ HTTP, HTTPS or other TCP/IP application layer protocols. REST allows CRUD operations to be executed on a server via simple HTTP requests, where a *document* is dynamically exchanged between client and server using the request/response pattern implemented in web applications (Section 4.5.5). REST is also stateless, i.e. no client context is stored on the server between requests, and it is seen as an alternative to the more-complex SOAP and WSDL-based web services. The acronym REST originates from Fielding's doctoral thesis *Architectural Styles and the Design of Network-based Software Architecture*, from the University of California in 2000 [120].

Resource description framework (RDF) We summarise RDF, according to the description given at `http://www.w3schools.com/xml/xml_rdf.asp`: RDF is written in XML and is designed to "provide a common way to describe information so it can be read and understood by computer applications." RDF allows computers to use URIs to identify WWW-based "resources with properties and property values": these resources may describe "price and availability" of goods, schedules for events, metadata about HTML pages, and "content for search engines" and on-line libraries. The RDF language is also part of the W3C's DATA ACTIVITY *Building the Web of Data* [422] vision where WWW-based information has "exact meaning" and can be understood and processed by computers. This is based upon the *semantic web* proposed by Berners-Lee et al. [35].

Serialisation/deserialisation A Java mechanism defined by Tutorials Point at `http://www.tutorialspoint.com/java/java_serialization.htm`:

> "where an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object.
>
> After a serialized object has been written into a file, it can be read from the file and deserialized that is, the type information and bytes that represent the object and its data can be used to recreate the object in memory.
>
> Most impressive is that the entire process is JVM independent, meaning an object can be serialized on one platform and deserialized on an entirely different platform."

Sharding Defined by Tee [389] where:

> "Database sharding isn't anything like clustering database servers, virtualizing datastores or partitioning tables. It goes far beyond all of that. In the simplest sense, sharding your database involves breaking up your big database into many, much smaller databases that share nothing and can be spread across multiple servers. These small databases are fast, easy to manage, and often are much cheaper to use as they are often implemented by using open source licensed databases.
>
> And how do you do it? Well, there's a variety of different approaches, but essentially, it's just a matter of taking a look at your database and essentially 'horizontally partitioning' your data into logically related rows, as opposed to the types of columnizing of data that you do with a typical relational database. The logical rows that you come up with get isolated and deployed into their own database, and as a result, data interaction becomes much faster and more responsive."

Similarity measures We define similarity measures to be those properties of objects or labour which can be compared, i.e. quantified, for similarity, e.g. competing software products for the same business opportunity could be compared by methodology and technology used, performance, usability or outputs. Moreover, Boriah et al. [46], defined similarity measures to:

> "include clustering (k-means), distance-based outlier detection, classification (knn, SVM), and several other data mining tasks. These algorithms typically treat the similarity computation as an orthogonal step and can make use of any measure."

Stemming Stemming is defined by the Stanford Natural Language Processing Group [371] as a means to "reduce inflectional forms and sometimes derivationally related forms of a word to a common base form."

In our second case study, keyword stemming was a mining rule available in vRSS but it was not used (Section 7.4.2). For the classification component of case study three (Chapter 10), stemming was performed by Lucene's (Appendix B.1) implementation of the *Snowball*, or *Porter2*, algorithm described at `http://tartarus.org/martin/PorterStemmer/`, and written by Martin Porter, instead of the original English-language *Porter1* algorithm [304]. This was because:

> "The Porter stemmer should be regarded as 'frozen', that is, strictly defined, and not amenable to further modification. As a stemmer, it is slightly inferior to the Snowball *English* or *Porter2* stemmer, which derives from it, and which is subjected to occasional improvements. For practical work, therefore, the new Snowball stemmer is recommended. The Porter stemmer is appropriate to IR research work involving stemming where the experiments need to be exactly repeatable."

Two simple examples of stemming using the *Snowball* algorithm are: (1) *consonant* which stems to *conson*, and (2) *knocking* which becomes *knock*.

Stop words Defined by Pederson at `http://www.d.umn.edu/~tpederse/Group01/WordNet/wordnet-stoplist.html`, stop words are:

> "words that are excluded from some language processing task, usually because they are viewed as non--informative or potentially misleading. Usually they are non--content words like conjunctions, determiners, prepositions, etc. These are often called function words."

Our case studies made use of a list of 500+ stop words provided by MySQL (`http://dev.mysql.com/doc/refman/5.5/en/fulltext-stopwords.html`). Examples of frequently used stop words include: *according*, *indeed*, *sure*, *the* and *whatever*.

Suffix tree Introduced by Weiner [436], a suffix tree is a data structure that indexes all the suffixes in a string of characters. Individual suffixes are used as keys and values consist of positions in the text.

Term frequency - inverse document frequency (TF-IDF) Defined at `http://www.tfidf.com/`, TF-IDF is "a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus." IDF is attributed to Spärck Jones [369], and is frequently used in recommendation and ranking applications. TF-IDF is calculated as follows:

$$w_{x,y} = tf_{x,y} * \log_{10}\left(\frac{N}{df_x}\right)$$

Where for keyword x in document y:

- $tf_{x,y}$ is the frequency of x in y.
- df_x is the number of documents containing x.
- N is the total number of documents.

Concept frequency - inverse document frequency (CF-IDF), has been proposed by Goossen et al. [157] as a variation of TF-IDF "based upon TF-IDF and a domain ontology."

Ticker-tape Defined by Investopedia (`http://www.investopedia.com/terms/t/tickertape.asp`) as:

> "A device that shows stock symbols and numbers to convey information about trades. The ticker tape is electronic today, but gets its name from the ticking sound the original mechanical machine made and from the long, narrow pieces of paper that stock quotes were printed on."

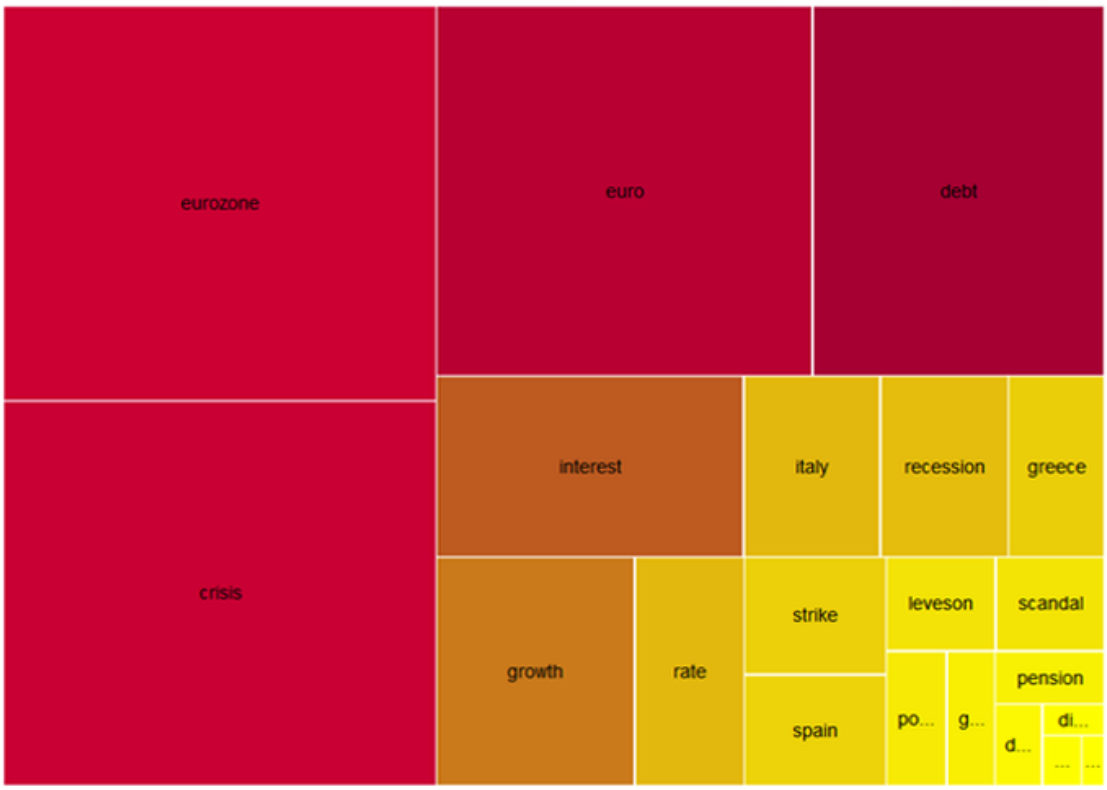
Topic modelling Topic models are defined by Mallet (`http://mallet.cs.umass.edu/topics.php`) as providing:

> "a simple way to analyze large volumes of unlabeled text. A "topic" consists of a cluster of words that frequently occur together. Using contextual clues, topic models can connect words with similar meanings and distinguish between uses of words with multiple meanings."

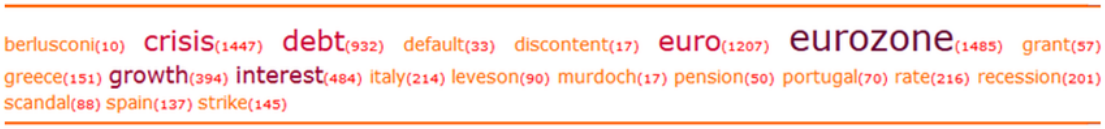
Tree-map Tree-maps, originally developed by Johnson and Shneiderman [207]:

> "partition the display space into a collection of rectangular bounding boxes representing the tree structure. The drawing of nodes within their bounding boxes is entirely dependent on the content of the nodes, and can be interactively controlled. Since the display size is user controlled, the drawing size of each node varies inversely with the size of the tree (i.e., # of nodes)."

Johnson and Shneiderman listed four objectives for the tree-map visualisation type: (1) efficient utilization of the display space, (2) interactivity, (3) comprehension, and (4) aesthetics. Figure B.2a displays a tree-map produced by vRSS during our second case study. The prominence of the rectangles for keywords *crisis*, *debt*, *euro* and *eurozone* reflects their higher frequencies in the underlying data and corresponds with the increased font size and darker colour of their equivalents in the word-cloud illustrated in Figure B.2b.



(a) Tree-map.



(b) Word-cloud.

Figure B.2: Tree-map and word-cloud visualisation types in visualRSS: cf. *case study two*.

Trend analysis This is defined by the BusinessDictionary at `http://www.businessdictionary.com/definition/trend-analysis.html` as a:

> "Method of time series data (information in sequence over time) analysis involving comparison of the same item (such as monthly sales revenue figures) over a significantly long period to (1) detect general patter [*sic*] of a relationship between associated factors or variables, and (2) project the future direction of this pattern."

True positive (TP), true negative (TN), false positive (FP) and false negative (FN) With reference to Bellaachia [31], the confusion matrix (Kohavi and Provost [212]) in Table B.2 illustrates the derivation of the TP, TN, FP and FN metrics.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	TP	FN
	Negative	FP	TN

Table B.2: Derivation of TP, TN, FP and FN metrics from a confusion matrix.

These metrics are explained in Table B.3. Moreover, in conjunction with the confusion matrix in Figure B.3, Table B.3 lists the calculation for class, i.e. RSS feed category, News and current affairs (NCA) during the sub-classification described in Section 10.6.2.

Metric	**Definition**	**Result**
True positive (TP)	An *item* is correctly classified as positive.	25 items correctly classified as NCA.
True negative (TN)	An item is correctly classified as negative.	106 items which are not TP, FP or FN, i.e. 9 + 5 + 15 + 11 + 15 + 1 + 50.
False positive (FP)	An item is incorrectly classified as positive.	8 items, i.e. 1 BFE, 4 SNT and 3 EA have been misclassified.
False negative (FN)	An item is incorrectly classified as negative.	11 items where 4 NCA have been misclassified as SNT, and 7 as EA.

Table B.3: Metrics and results for class NCA (Section 10.6.2): cf. *case study three*.

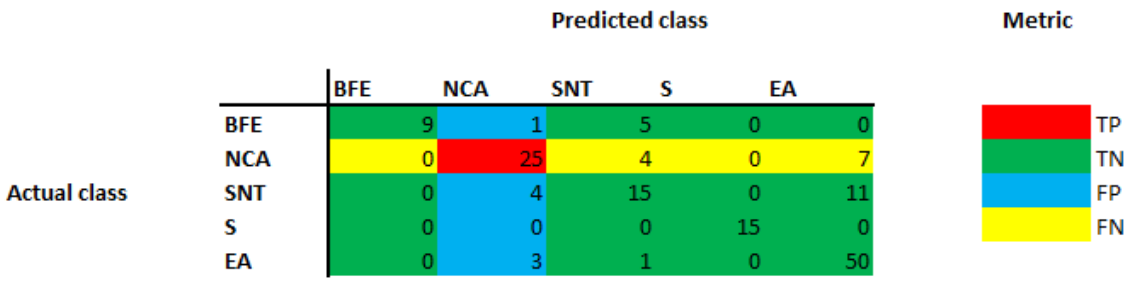


Figure B.3: Confusion matrix distribution and metrics for class NCA (Section 10.6.2): cf. *case study three*.

Twitterbot Defined by WhatIs at `http://whatis.techtarget.com/definition/Twitterbot`, a *Twitterbot* is a "software program that sends out automated posts on Twitter." A second use of a Twitterbot is to follow *tweets* made by users of Twitter [408].

Unified modelling language (UML) UML provides a series of graphical constructs and document artefacts where the objective, according to the Object Management Group (OMG [282]), is "to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes."

Uniform resource identifier (URI), uniform resource locator (URL) and uniform resource name (URN) The definitions below are summarised from [34]:

- **Uniform resource identifier:** A URI is a "compact string of characters for identifying an abstract or physical resource." A URI can be classified as "a locator, a name or both."
- **Uniform resource locator:** The term URL "refers to the subset of URI that identify resources via a representation of their primary access mechanism (e.g., their network "location"), rather than identifying the resource by name or by some other attribute(s) of that resource."
- **Uniform resource name:** URN refers to "the subset of URI that are required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable."

In this thesis the use of abbreviations *URI* and *URL* is *substantially* interchangeable.

Vector space model (VSM) VSM, originated by Salton et al. [345], has been defined by Manning et al. [243] as "The representation of a set of documents as vectors in a common vector space is known as the *vector space model* and is fundamental to a host of information retrieval (IR) operations including scoring documents on a query, document classification, and document clustering."

Waterfall model The classical Waterfall model of software development, attributed to Bell and Thayer [30], and Royce [331], is illustrated in Figure B.4. The model has a linear sequence of steps, each with its own set of goals, and development flows like a waterfall as the steps progress.

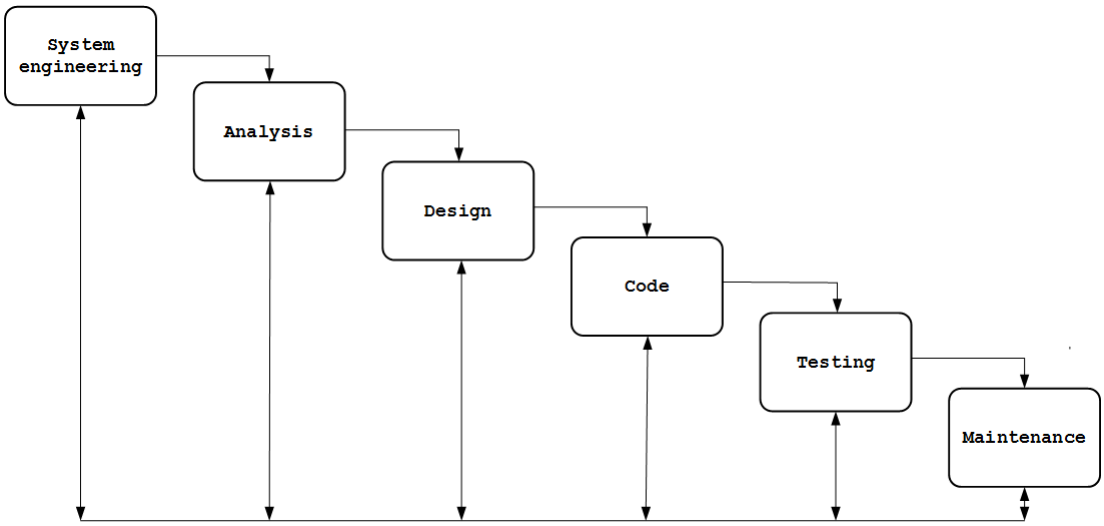


Figure B.4: The classical *Waterfall* model of software development (adapted from Pressman [309]).

Web 2.0 Web 2.0 refers to participatory web services that include social networking, dynamic sharing of content or blogging, compared with older, static web site functionality. *Web 2.0* is not a software architecture, development methodology or technology: the term refers to a set of conventions originally defined by O'Reilly in 2005 at `http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=1`. We summarise these as: (1) the WWW as platform, (2) the use of collective intelligence, (3) data as the next *Intel Inside*, (4) the end of the software release cycle, (5) lightweight and innovative programming models, (6) software above the level of a single device, and (7) a rich user experience. In the author's opinion, the term *Web 2.0* has become something of a media *buzzword* given its contemporary usage.

Word-cloud The basic format of a word-cloud, otherwise known as a tag-cloud, is described by Viégas and Wattenberg [415] as "a combination of many different type sizes in a single view—goes back at least 90 years to Soviet Constructivism. Beyond the surface style, however, a tag cloud usually has a particular purpose: to present a visual overview of a collection of text." A recent review of word-cloud generators is provided by Smitty [362].

A word-cloud produced by vRSS during our second case study is displayed in Figure B.2b. The increased font size and darker colour used to display keywords *crisis*, *debt*, *euro* and *eurozone* is because of their higher frequencies (shown in brackets) in the underlying data. These higher frequencies are also visible in the corresponding tree-map in Figure B.2a which displays the same data.

***Z*-test** A test for statistical significance, defined by Investopedia at `http://www.investopedia.com/terms/t/two-tailed-test.asp` as:

> "A statistical test in which the critical area of a distribution is two-sided and tests whether a sample is either greater than or less than a certain range of values. If the sample that is being tested falls into either of the critical areas, the alternative hypothesis will be accepted instead of the null hypothesis. The two-tailed test gets its name from testing the area under both of the tails of a normal distribution, although the test can be used in other non-normal distributions."

A *p*-value (Appendix B.2) is used to measure the statistical significance of the test.

Appendix C

The Android OS client *app* for visualRSS

C.1 Outline

In 2013, using the Android OS (Appendix B.1) for the *mobile platform*, Shema [354] developed a client *app* for our vRSS software. Although the client was not used in any of our case studies, the requirement was to:

> "implement a mobile client for the vRSS application. The client can consist of a user interface, application logic mainly to do web service calls on the vRSS RESTful interface, and data visualisation functionality. The client user interface should allow for three input modes mirroring vRSS three mining types and present the results as instructed by the user using a custom visualisation engine."

Shema described the core of the client as:

> "its data integration model. This layer enables the Android client to seamlessly interact with vRSS and exchange data without the client knowing anything about vRSS's underlying data model. The vRSS Android client works by exchanging XML objects with the main system, each of which corresponding to vRSS concepts such as mining rules, recent visualizations and active (polling) visualizations. These concepts gave rise to three main XML document structures (schema) which constitute the domain model of this application: the mining rules schema, the recent visualizations schema and the visualization schema."

The remaining sections of this appendix describe the client's implementation of the principal components of vRSS (Section 7.3).

C.2 Defining mining rules

The three mining types in vRSS (Section 7.4.1) are supported by the client: Figure C.1 displays *automatic* and *manual* mining.

Figure C.1: Defining mining rules in visualRSS's Android OS client *app*: the word-cloud (Section 7.4.2) displaying keyword frequencies in *automatic* mining (left) and entry of an RSS feed's URL in *manual* mining (right).

C.3 Polling and data storage

No polling of RSS or database persistence of RSS-mined data is carried out by the client because:

> "The vRSS Android client has no persistent data storage. Moreover, it has no direct access to the data objects that form the vRSS domain model. A web service interface on vRSS allows the client to collect data from the user, send

> it across to vRSS and retrieve processed results. Internally, the Android client maintains its state using java objects mapped to the XML schemas by means of an XML serialization framework called Simple. In this way, the Android client is able to deliver the functionality of vRSS without requiring access to vRSS [*sic*] underlying data store."

Communication between client and server makes use of XML. This is enabled by a REST (Appendix B.2) interface provided by SimpleXML (Appendix B.1) and implemented in vRSS. A HTTP request from the client is received as an XML document by the interface and *serialised* (Appendix B.2) into Java objects for vRSS to execute the request. The response by vRSS is reciprocally *deserialised* from Java objects into XML before it is returned to the client.

C.4 Visualising RSS-mined data

The client implements the five visualisation types in vRSS (Section 7.9). Figure C.2 displays the page to *browse* recent visualisations and a sample pie chart visualisation.

Figure C.2: Browsing recent visualisations (left) and a sample pie chart visualisation (right) in visualRSS's Android OS client.

Appendix D

Miscellaneous

D.1 Additional resources

The following list is made up of web sites that supplied advice or fragments of source code during the development of our myDS and vRSS software, but which are not acknowledged elsewhere in this thesis:

- **Avilyne Technologies:** `http://avilyne.com`.
- **The BalusC Code:** `http://balusc.blogspot.com`.
- **DaniWeb:** `http://www.daniweb.com/`.
- **DevX:** `http://www.devx.com/`.
- **Dynamic Drive:** `http://www.dynamicdrive.com/`.
- **High Scalability:** `http://highscalability.com/`.
- **IBM Developer Works:** `http://www.ibm.com/developerworks/`.
- **Java Code Geeks:** `http://www.javacodegeeks.com/`.
- **JavaWorld:** `http://www.javaworld.com/`.
- **Java.net:** `https://today.java.net/`.
- **JavaScript Toolbox:** `http://www.javascripttoolbox.com/`.
- **Jenkov.com:** `http://jenkov.com/`.

- **JavaScript Tricks, Samples, Tutorials & Howtos:** http://www.jtricks.com/.
- **JodaTime:** http://www.joda.org/joda-time/.
- **Johann Burkard:** http://johannburkard.de/.
- **kaptcha:** https://maven-repository.com/artifact/com.google.code.kaptcha/kaptcha/2.3.
- **KodeJava:** http://kodejava.org/.
- **LuceneTutorial.com:** http://www.lucenetutorial.com/.
- **Lucid Imagination:** http://www.lucidimagination.com/.
- **Pat's Place:** http://www.hunlock.com/.
- **Pete Freitag:** http://www.petefreitag.com/.
- **Program Creek:** http://www.programcreek.com.
- **Real's How To:** http://www.rgagnon.com/.
- **Rose India Technologies:** http://www.roseindia.net/.
- **Shining Star Services:** http://www.shiningstar.net/.
- **StackOverflow:** http://www.stackoverflow.com/.
- **Sun Corporation:** http://forums.sun.com/.
- **Texcount:** http://app.uio.no/ifi/texcount/.
- **Trent Richardson:** http://trentrichardson.com/.
- **Vikasing:** http://www.vikasing.com/.
- **Viral Patel:** http://viralpatel.net/.
- **Web Development Learnings:** http://stevethomas.com.au/.

Microsoft Excel [259] was used to produce many of the graphs and time-series plots in this thesis.

Lastly, the author apologises for any source of advice or code used in our software, but not acknowledged in the list above.

D.2 Source code

D.2.1 myDataSharer and visualRSS

When applied to our myDS and vRSS software, CLOC (Appendix B.1) produced an *indigenous* count of approximately 172,000 lines of Java and other source code. Nevertheless, the author believes that an exact count is not possible because of comments, disabled code, open-source, third-party products written in JavaScript, e.g. TinyMCE, HTML or CSS, as well as code for experimentation and testing. Furthermore, the Android OS client (Appendix C) written by Shema [354] for vRSS is not included. Thus, given these criteria, the author considers an approximate figure of 100,000 lines to be more realistic.

D.2.2 Principal open-source, third-party products

Approximate counts of lines of source code for the principal open-source, third-party products used in our software are listed in Table D.1. Except where specifically stated, the counts were provided by Black Duck Open Hub (Appendix B.1) on 30 Jan 2016. For the case studies for our first RSS-mining paradigm, we employed Google Charts to visualise data: to the best of our knowledge, no count of lines of code is available for that product.

Product	**Approximate line count**	**Comments**
Apache Tomcat	1,129,200	
Lucene	532,400	Excludes customisations used in case study three (Section 11.6.1), and may also include components of Lucene not used in myDS and vRSS.
MySQL	2,776,200	Includes MySQL database, MySQL Workbench and other administrative tools.
Quartz Scheduler	53,400	
Rome	18,300	
SentiStrength	5,000	Provided by Professor M. Thelwall, i.e. SentiStrength's (Appendix B.1) writer, on 30 Jan 2016 via email.
Weka	700,000	Count may not include all code of classifiers used in case study three (Section 10.5).
Total	5,214,500	Excludes author's own software.

Table D.1: Approximate counts of lines of source code of principal open-source, third-party products used in myDataSharer and visualRSS: cf. *all case studies*.

Bibliography

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1995.

[2] Academia. `https://www.academia.edu/`, 2015.

[3] G. Adam, C. Bouras, and V. Poulopoulos. Utilizing RSS Feeds for crawling the Web. In *Proceedings of the Fourth International Conference on Internet and Web Applications and Services (ICIW 2009)*, pages 211–216, held in Venice/Mestre, Italy, May 2009.

[4] B. Adams, D. Phung, and S. Venkatesh. Eventscapes: Visualizing Events over time with Emotive Facets. In *Proceedings of the 19th ACM International Conference on Multimedia (MM'11)*, pages 1477–1480, held in Scottsdale, AZ, USA, 2011.

[5] D. A. Adeniyi, Z. Wei, and Y. Yongquan. Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. *Applied Computing and Informatics*, 12(1):90–108, 2016.

[6] R. Adhikari and P. K. Agrawal. An Introductory Study on Time Series Modeling and Forecasting. *Computing Research Repository*, abs/1302.6613, 2013.

[7] Adobe Acrobat Reader. `https://get.adobe.com/uk/reader/`, 2015.

[8] S. Agarwal, A. Singhal, and P. Bedi. Classification of RSS Feed News Items Using Ontology. In *Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (ISDA 2012)*, pages 491–496, held in Kochi, Kerala, India, Nov 2012.

[9] C. C. Aggarwal, editor. *Data Classification, Algorithms and Applications*. Chapman and Hall & CRC Press, Boca Raton, FL, USA, 2014.

[10] C. C. Aggarwal and P. S. Yu. Mining Large Itemsets for Association Rules. *IEEE Data Engineering Bulletin*, 21(1):23–31, 1998.

[11] C. C. Aggarwal and CX. Zhai. A SURVEY OF TEXT CLASSIFICATION ALGORITHMS. In C. C. Aggarwal and CX. Zhai, editors, *Mining Text Data*, pages 163–222. 2012.

[12] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing Time-Oriented Data - A Systematic View. *Computers & Graphics*, 31(3):401–409, Jun 2007.

[13] C. Albrecht-Buehler, B. Watson, and D. A. Shamma. TextPool: Visualizing Live Text Streams. In *Proceedings of the 10th IEEE Symposium on Information Visualization (InfoVis 2004))*, held in Austin, TX, USA, Oct 2004.

[14] M. S. Ali, M. P. Consens, and F. Rizzolo. Visualizing Structural Patterns in Web Collections. In *Proceedings of the 16th International Conference on World Wide Web (WWW2007)*, pages 1333–1334, held in Banff, AB, Canada, Apr 2007.

[15] M. H. Alomari, H. Abusaimeh, S. Shahin, and R. Joudeh. Postat: A Cross-Platform, RSS-based Advertising and Event Notification System for Educational Institutions. *Procedia - Social and Behavioral Sciences*, 73:120–127, Aug - Sep 2012. Proceedings of the 2nd International Conference on Integrated Information (IC-ININFO 2012).

[16] Amazon. `http://www.amazon.co.uk/`, 2015.

[17] Amazon Prime. `https://www.amazon.co.uk/gp/prime/pipeline/landing`, 2016.

[18] M. Anastopoulos and T. Romberg. Referenzarchitekturen für Web-Applikationen. Fraunhofer IESE/FZI Karlsruhe, 2001. (German language).

[19] A. Anjomshoaa, A. Min Tjoa, and H. Rezaiee. A Lightweight Data Integration Architecture Based on Semantic Annotated RSS Feeds. In R. Meersman, T. Dillon, and P. Herrero, editors, *Proceedings of On the Move to Meaningful Internet Systems: OTM 2011 Workshops*, pages 170–177, held in Hersonissos, Crete, Greece, Oct 2011.

[20] I. Antonellis, C. Bouras, and V. Poulopoulos. Personalized News Categorization Through Scalable Text Classification. In X. Zhou, J. Li, H. T. Shen, M. Kitsuregawa, and Y. Zhang, editors, *Proceedings of the 8th Asia-Pacific Web Conference on Frontiers of WWW Research and Development*, pages 391–401, held in Harbin, Heilongjiang, China, 2006.

[21] L. Augustyniak, T. Kajdanowicz, P. Szymański, W. Tuligłowicz, P. Kazienko, R. Alhajj, and B. Szymanski. Simpler is Better? Lexicon-based Ensemble Sentiment Classification Beats Supervised Methods. In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 924–29, Aug 2014.

[22] D. Ayers and A. Watt. *Beginning RSS and Atom Programming.* Wiley Publishing, Inc., Indianapolis, Indiana, USA, 2008.

[23] S. Baker. Why Do People Use Online Social Networking? lovetoknow. `http://socialnetworking.lovetoknow.com/Why_Do_People_Use_Online_Social_Networking`, 2016.

[24] Bandcamp. `https://bandcamp.com/`, 2015.

[25] S. Banerjee, K. Ramanathan, and A. Gupta. Clustering short texts using Wikipedia. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*, pages 787–788, held in Amsterdam, The Netherlands, 2007.

[26] E. Banos, I. Katakis, N. Bassiliades, G. Tsoumakas, and I. Vlahavas. PersoNews: A Personalized News Reader Enhanced by Machine Learning and Semantic Filtering. In *Proceedings of the 2006 Confederated International Conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASE - Volume Part I (ODBASE'06/OTM'06)*, pages 975–982, held in Montpellier, France, 2006.

[27] A. Barth. HTTP State Management Mechanism. `http://tools.ietf.org/html/rfc6265`, 2011.

[28] BazQux. `https://bazqux.com/`, 2015.

[29] Bebo. `http://www.bebo.com/`, 2015.

[30] T. E. Bell and T. A. Thayer. Software Requirements: Are They Really a Problem? In *Proceedings of the 2nd International Conference on Software Engineering (ICSE '76)*, pages 61–68, held in San Francisco, CA, USA, 1976.

[31] A. Bellaachia. Performance Evaluation, Data Mining. School of Engineering and Applied Science, Department of Computer Science, The George Washington University, Washington, DC, USA. `http://www.seas.gwu.edu/~bell/csci243/lectures/performance.pdf`, 2015.

[32] J. Benson, D. Crist, and P. Lafleur. Agent-based Visualization of Streaming Text. In *Proceedings of the IEEE Information Visualization Conference*, held in Columbus, OH, USA, 2008.

[33] T. Berners-Lee. Information Management: A Proposal. Technical report, CERN, Mar 1989.

[34] T. Berners-Lee, R. Fielding, U. C. Irvine, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. `https://www.ietf.org/rfc/rfc2396.txt`, 1998.

[35] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[36] S. V. Bharathi, J. Kalaimathi, and A. Geetha. Emotion Classification of RSS News Feeds. `https://www.academia.edu/6985862/Emotion_Classification_of_RSS_News_feeds`, 2015.

[37] BibSonomy. `http://www.bibsonomy.org/`, 2015.

[38] Bing Maps. `http://microsoft.com/maps/`, 2015.

[39] Blackboard. `http://www.blackboard.com/`, 2015.

[40] A. Blekas, J. Garofalakis, and V. Stefanis. Use of RSS feeds for Content Adaptation in Mobile Web Browsing. In *Proceedings of the 2006 International Cross-disciplinary Workshop on Web Accessibility (W4A): Building the Mobile Web: Rediscovering Accessibility*, pages 79–85, held in Edinburgh, Scotland, UK, 2006.

[41] Blogger. `http://www.blogger.com/`, 2015.

[42] BlogLines. `http://www.bloglines.com/`, 2014.

[43] J. G. Blumler and E. Katz. *The Uses of Mass Communications: Current Perspectives on Gratifications Research (SAGE Series in Communication Research)*. Sage Publications Inc., Beverly Hills, CA, USA, Feb 1975.

[44] S. Bolasco, A. Canzonetti, F. M. Capo, F. della Ratta-Rinaldi, and B. K. Singh. Understanding Text Mining: A Pragmatic Approach. In S. Sirmakessis, editor, *Knowledge Mining*, volume 185 of *Studies in Fuzziness and Soft Computing*, pages 31–50. 2005.

[45] L. C. Borges, V. M. Marques, and J. Bernardino. Comparison of Data Mining Techniques and Tools for Data Classification. In *Proceedings of the International C* Conference on Computer Science and Software Engineering (C3S2E13)*, pages 113–116, held in Porto, Portugal, 2013.

[46] S. Boriah, V. Chandola, and V. Kumar. Similarity Measures for Categorical Data: A Comparative Evaluation. In *Proceedings of the SIAM International Conference on Data Mining (SDM08)*, pages 243–254, held in Atlanta, GA, USA, Apr 2008.

[47] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT '92)*, pages 144–152, held in Pittsburgh, PA, USA, 1992.

[48] S. Bossa, G. Fiumara, and A. Provetti. A Lightweight Architecture for RSS Polling of Arbitrary Web sources. In *Proceedings of the 7th WOA 2006 Workshop, From Objects to Agents (Dagli Oggetti Agli Agenti)*, held in Catania, Italy, Sep 2006.

[49] C. Bouras, V. Poulopoulos, and V. Tsogkas. PeRSSonal's core functionality evaluation: Enhancing text labeling through personalized summaries. *Data & Knowledge Engineering*, 64(1):330–345, Jan 2008.

[50] D. M. Boyd and N. B. Ellison. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[51] BrandRepublic. `http://www.brandrepublic.com/`, 2016.

[52] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yeargeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). `http://www.w3.org/TR/REC-xml/`, 2008.

[53] F. P. Bretherton and P. T. Singley. Metadata: a User's View. In *Proceedings of the Seventh International Working Conference on Scientific and Statistical Database Management*, pages 166–174, Sep 1994.

[54] A. Buche, M. B. Chandak, and A. Zadgaonkar. Opinion Mining and Analysis: A survey. *Computing Research Repository*, abs/1307.3336, 2013.

[55] BuiltWith. `http://trends.builtwith.com/feeds/RSS/`, 2016.

[56] BuySellAds. `https://www.buysellads.com/`, 2016.

[57] BuzzFeed. `http://www.buzzfeed.com/`, 2015.

[58] E. Cambria, B. Schuller, Y. Xia, and C. Havasi. New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems*, 28(2):15–21, Mar 2013.

[59] CanvasJS. `http://www.canvasjs.com/`, 2015.

[60] L. F. Capretz. A Brief History of the Object-oriented Approach. *SIGSOFT Software Engineering Notes*, 28(2):6–15, Mar 2003.

[61] R. Cattell. Scalable SQL and NoSQL Data Stores. *SIGMOD Record*, 39(4):12–27, 2010.

[62] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, San Francisco, CA, USA, 2002.

[63] D. D. Chamberlin and R. F. Boyce. SEQUEL: A Structured English Query Language. In *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control (SIGFIDET '74)*, pages 249–264, held in Ann Arbor, MI, USA, 1974.

[64] S. W Chan and R. Mordani. Java Servlet Specification, Version 3.1. `http://download.oracle.com/otndocs/jcp/servlet-3_1-fr-eval-spec/index.html`, 2013.

[65] C-C. Chang and C-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Interactive Intelligent Systems*, 2(3):27:1–27:27, May 2011.

[66] P. P-S. Chen. The Entity-relationship Model–Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, Mar 1976.

[67] T. Chen, W-L. Han, H-D. Wang, Y-X. Zhou, B. Xu, and B-Y. Zang. CONTENT RECOMMENDATION SYSTEM BASED ON PRIVATE DYNAMIC USER PROFILE. In *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*, volume 4, pages 2112–2118, held in Hong Kong, China, Aug 2007.

[68] Y-F R. Chen, G. Di Fabbrizio, D. Gibbon, S. Jora, B. Renger, and B. Wei. Geotracker: Geospatial and Temporal RSS Navigation. In *Proceedings of the 16th International Conference on World Wide Web (WWW2007)*, pages 41–50, held in Banff, AB, Canada, Apr 2007.

[69] D. Chmielewski and G. Hu. A Distributed Platform for Archiving and Retrieving RSS Feeds. In *Proceedings of the 4th Annual ACIS International Conference on Computer and Information Science (ICIS 2005)*, pages 215–220, held on Jeju Island, South Korea, Jul 2005.

[70] K-M. Chung. JavaServer Pages Specification, Version 2.3 Maintenance Release 3. `http://download.oracle.com/otndocs/jcp/jsp-2_3-mrel2-spec/`, 2013.

[71] J. Churchill. rss4j. `http://sourceforge.net/projects/rss4j/`, 2015.

[72] M. O. Cingiz and B. Diri. Content Mining of Microblogs. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 835–838, held at Kadir Has University, Istanbul, Turkey, Aug 2012.

[73] CiteSeer. `http://citeseerx.ist.psu.edu/`, 2015.

[74] P. Coad and E. Yourdon. *Object-oriented Analysis*. Yourdon Press, Upper Saddle River, NJ, USA, 2nd edition, 1991.

[75] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, Jun 1970.

[76] E. F. Codd. Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems*, 4(4):397–434, Dec 1979.

[77] S. J. Cold. Using Really Simple Syndication (RSS) to Enhance Student Research. *ACM SIGITE Newsletter*, 3(1):6–9, Jan 2006.

[78] A. Collomb, C. Costea, D. Joyeux, O. Hasan, and L. Brunie. A Study and Comparison of Sentiment Analysis Methods for Reputation Evaluation. Technical Report RR-LIRIS-2014-002, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, Mar 2014.

[79] Coursera. `http://www.coursera.org`, 2015.

[80] J. Creus, B. Amann, N. Travers, and D. Vodislav. RoSeS: A Continuous Query Processor for Large-scale RSS Filtering and Aggregation. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM 2011)*, pages 2549–2552, held in Glasgow, Scotland, UK, 2011.

[81] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2010.

[82] D. H. Crocker. RFC822: Standard for ARPA Internet Text Messages. `http://www.w3.org/Protocols/rfc822/`, 1982.

[83] DabbleDB. `http://dabbledb.com/`, 2010.

[84] R. L. Daft and R. H. Lengel. Information Richness: A New Approach to Managerial Behavior and Organizational Design. *Research in Organizational Behaviour*, 6:191–233, 1984.

[85] M. Daiyan, S. K. Tiwari, M. Kumar, and M. Aftab Alam. A Literature Review on Opinion Mining and Sentiment Analysis. *International Journal of Emerging Technology and Advanced Engineering*, 5(4):262–280, 2015.

[86] M. Danani. Multi-format data upload and display components for myDataSharer community platform. Master's thesis, DCSIS, Birkbeck, University of London, 2008.

[87] M. Darabi, H. Adeli, and N. Tabrizi. Automatic multi-label categorization of news feeds. In *Proceedings of The 2012 World Congress in Computer Science, Computer Engineering, and Applied Computing (DMIN'12)*, held in Las Vegas, NV, USA, Jul 2012.

[88] C. J. Date. *An Introduction to Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 6th edition, 1995.

[89] I. de la Torre-Díez, S. Álvaro-Muñoz, M. López-Coronado, and J. Rodrigues. Development and performance evaluation of a new RSS tool for a Web-based system: RSS_PROYECT. *Journal of Network and Computer Applications*, 36(1):255–61, Jan 2013.

[90] C. De Maio, G. Fenza, M. Gaeta, V. Loia, F. Orciuoli, and S. Senatore. RSS-based e-learning recommendations exploiting fuzzy FCA for Knowledge Modeling. *Applied Soft Computing*, 12(1):113–124, Jan 2012.

[91] Delicious. `https://delicious.com/`, 2015.

[92] L. DeMichiel and W. Shannon. Java Platform, Enterprise Edition (Java EE) Specification, v7. `http://download.oracle.com/otn-pub/jcp/java_ee-7-mrel-eval-spec/JavaEE_Platform_Spec.pdf`, 2015.

[93] Denmag. 10 Popular Java Machine Learning Tools & Libraries. Data Science Central. `http://www.datasciencecentral.com/profiles/blogs/10-popular-java-machine-learning-tools-libraries`, 2015.

[94] Y. Deshpande, S. Murugesan, A. Ginige, S. Hansen, D. Schwabe, M. Gaedke, and B. White. WEB ENGINEERING. *Journal of Web Engineering*, 1(1):3–17, Oct 2002.

[95] Disqus. `http://disqus.com/`, 2015.

[96] S. Drazin and M. Montag. Decision Tree Analysis using Weka. Machine Learning Project II, University of Miami, 2012.

[97] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan. Making Data Structures Persistent. *Journal of Computer and System Sciences*, 38(1):86–124, Feb 1989.

[98] Drop Box. `https://www.dropbox.com`, 2015.

[99] Drupal. `https://www.drupal.com/`, 2015.

[100] D. Duan, W. Qian, S. Pan, L. Shi, and C. Lin. VISA: A Visual Sentiment Analysis System. In *Proceedings of the 5th International Symposium on Visual Information Communication and Interaction (VINCI '12)*, pages 22–28, held in Hangzhou, Zheijiang, China, 2012.

[101] Ebay. `http://www.ebay.com/`, 2015.

[102] E. Ellis. Social Data vs Social Media. Gnip. `http://blog.gnip.com/social-data-vs-social-media-2/`, 2013.

[103] Encyclopædia Britannica. `http://www.britannica.com/`, 2015.

[104] Enron Corpus. `http://www.cs.cmu.edu/~./enron/`, 2016.

[105] European Media Monitor. `http://emm.newsbrief.eu/`, 2016.

[106] Facebook. `http://www.facebook.com/`, 2015.

[107] Facebook Messenger. `https://www.messenger.com/`, 2015.

[108] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37, 1996.

[109] Feedbin. `https://feedbin.com/`, 2015.

[110] FeedBlitz. `http://www.feedblitz.com`, 2015.

[111] FeedBurner. `https://accounts.google.com/servicelogin?service=feedburner`, 2015.

[112] FeedForAll. `http://www.feedforall.com/`, 2015.

[113] Feedjira. `http://feedjira.com/`, 2015.

[114] Feedly. `https://feedly.com`, 2015.

[115] FeedsAPI. `http://www.feedsapi.com/`, 2015.

[116] R Feldman. Sentiment Analysis Tutorial. In *TF4 Sentiment Mining from User Generated Content, 23rd International Joint Conference on Artificial Intelligence*, held in Beijing, China, Aug 2013.

[117] R. Feldman. Techniques and Applications for Sentiment Analysis. *Communications of the ACM*, 56(4):82–89, Apr 2013.

[118] R. Feldman and I. Dagan. Knowledge Discovery in Textual Databases (KDT). In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 112–117, held in Montreal, Québec, Canada, Aug 1995.

[119] Fever. `http://feedafever.com/`, 2015.

[120] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, CA, USA, 2000.

[121] Firefox. `https://www.mozilla.org/firefox/`, 2015.

[122] D. Fisher, A. Hoff, G. Robertson, and M. Hurst. Narratives: A Visualization to Track Narrative Events as they Develop. In *Proceedings of the 2008 IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 115–122, Oct 2008.

[123] R. A. Fisher. THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. *Annals of Eugenics*, 7(7):179–188, 1936.

[124] G. Fiumara, M. La Rosa, and T. Pimpo. (X)querying RSS/Atom Feeds Extracted from News Web Sites: a Cocoon-based Portal. In *Proceedings of the Workshop Between Ontologies and Folksonomies (BOF-2007)*, held in East Lansing, MI, USA, Jun 2007.

[125] FiveFilters. `http://fivefilters.org/`, 2015.

[126] Flickr. `http://www.flickr.com`, 2015.

[127] Flipboard. `https://flipboard.com`, 2015.

[128] FourSquare. `http://foursquare.com/`, 2015.

[129] E. Frank and R. R. Bouckaert. Naive Bayes for Text Classification with Unbalanced Classes. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2006)*, pages 503–510, held in Berlin, Germany, 2006.

[130] FreshRSS. `http://freshrss.org/`, 2015.

[131] J. R. L. Froget, A. B. Baghestan, and Y. S. Asfaranjan. A Uses and Gratification Perspective on Social Media Usage and Online Marketing. *Middle-East Journal of Scientific Research*, 14(1):134–145, 2013.

[132] FusionCharts. `http://www.fusioncharts.com/`, 2015.

[133] A. Gallion. Applying the Uses and Gratifications Theory To Social Networking Sites: A Review of Related Literature. `https://www.academia.edu/1077670/Applying_the_Uses_and_Gratifications_Theory_to_Social_Networking_Sites_A_Review_of_Related_Literature`, 2015.

[134] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[135] I. Garcia and Y-K. Ng. Eliminating Redundant and Less-Informative RSS News Articles Based on Word Similarity and a Fuzzy Equivalence Relation. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '06)*, pages 465–473, 2006.

[136] GeoRSS. `http://www.georss.org`, 2015.

[137] F. Getahun and R. Chbeir. RSS query algebra: Towards a better news management. *Information Sciences*, 237:313–342, Jul 2013.

[138] F. Getahun, J. Tekli, R. Chbeir, M. Viviani, and K. Yetongnon. Relating RSS News/Items. In *Proceedings of the 9th International Conference on Web Engineering (ICWE 2009)*, pages 442–452, held in San Sebastián, Spain, Jun 2009.

[139] G. Giannopoulos, I. Weber, A. Jaimes, and T. Sellis. Diversifying User Comments on News Articles. In *Proceedings of the 13th International Conference on Web Information Systems Engineering (WISE2012)*, pages 100–113, 2012.

[140] K. E. Gill. Blogging, RSS and the Information Landscape: A Look At Online News. In *Proceedings of the 14th International World Wide Web Conference (WWW2005) 2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamic*, held in Chiba, Japan, 2005.

[141] A. Ginige and S. Murugesan. Web Engineering: A Methodology for Developing Scalable, Maintainable Web Applications. *Cutter IT Journal*, 14(7):24–35, Jul 2001.

[142] F. Gioachin, R. Shankesi, M. J. May, C. A. Gunter, and W. Shin. Emergency Alerts as RSS Feeds with Interdomain Authorization. In *Proceedings of the Second International Conference on Internet Monitoring and Protection (ICIMP 2007)*, held in San Jose, Silicon Valley, CA, USA, Jul 2007.

[143] N. Glance, M. Hurst, and T. Tomokiyo. BlogPulse: Automated Trend Discovery for Weblogs. In *Proceedings of the 13th International World Wide Web Conference (WWW2004): Workshop on Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, pages 1–8, held in New York, NY, USA, May 2004.

[144] R. J. Glotzbach, J. L. Mohler, and J. E. Radwan. RSS As a Course Information Delivery Method. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '07) Educators Program*, held in San Diego, CA, USA, 2007.

[145] A. Goldberg and D. Robson. *Smalltalk-80: The Language and Its Implementation.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983.

[146] H. Gomes, M. de Castro Neto, and R. Henriques. Text Mining: Sentiment analysis on news classification. In *Proceedings of 8th Iberian Conference on Information Systems and Technologies (CISTI 2013)*, pages 1–6, Jun 2013. (Portuguese language).

[147] P. Gonçalves, M. Araújo, F. Benevenuto, and M. Cha. Comparing and Combining Sentiment Analysis Methods. In *Proceedings of the First ACM Conference on Online Social Networks (COSN'13)*, pages 27–38, held in Boston, MA, USA, 2013.

[148] Google. `http://www.google.com`, 2015.

[149] Google+. `https://plus.google.com/`, 2015.

[150] Google AdSense. `https://www.google.com/adsense/`, 2016.

[151] Google Feed API. `https://developers.google.com/feed/`, 2015.

[152] Google Maps. `http://maps.google.com/maps`, 2015.

[153] Google News. `http://news.google.com/`, 2015.

[154] Google Reader. `http://www.google.com/reader/`, 2015.

[155] Google Scholar. `https://scholar.google.com/`, 2015.

[156] C. Goopta. Six of the Best Open Source Data Mining Tools. The New Stack. `http://thenewstack.io/six-of-the-best-open-source-data-mining-tools/`, 2014.

[157] F. Goossen, W. IJntema, F. Frasincar, F. Hogenboom, and U. Kaymak. News Personalization Using the CF-IDF Semantic Recommender. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics (WIMS '11)*, pages 10:1–10:12, held in Sogndal, Norway, 2011.

[158] J. Gosling and H. McGilton. The Java Language Environment: A White Paper. Technical report, Sun Microsystems Computer Company, Mountain View, CA, USA, Oct 1995.

[159] M. Govindarajan and M. Romina. A Survey of Classification Methods and Applications for Sentiment Analysis. *International Journal of Engineering & Science*, 2(12):11–15, 2013.

[160] GraphViz. `http://www.graphviz.org`, 2009.

[161] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information Diffusion Through Blogspace. In *Proceedings of the 13th International Conference on World Wide Web (WWW2004)*, pages 491–501, held in New York, NY, USA, May 2004.

[162] A. Gulli. AG's corpus of news articles. Department of Computer Science, University of Pisa, Italy. `http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`, 2016.

[163] V. Gupta and G. S. Lehal. A Survey of Text Mining Techniques and Applications. *Journal of Emerging Technologies in Web Intelligence*, 1(1):60–76, Aug 2009.

[164] T. Haerder and A. Reuter. Principles of Transaction-oriented Database Recovery. *ACM Computing Surveys*, 15(4):287–317, Dec 1983.

[165] B. Hammersley. *Developing Feeds with RSS and Atom*. O'Reilly, 1st edition, 2005.

[166] H. Han, T. Noro, and T. Tokuda. An Automatic Web News Article Contents Extraction System Based on RSS Feeds. *Journal of Web Engineering*, 8(3):268–284, 2009.

[167] D. Harriott. Query, Analysis & Visualisation interface for community-based, data-sharing application myDataSharer. Master's thesis, DCSIS, Birkbeck, University of London, 2008.

[168] B. Hartmann, L. Wu, K. Collins, and S. R. Klemmer. Programming by a Sample: Rapidly Creating Web Applications with d.mix. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, pages 241–250, held in Newport, RI, USA, Oct 2007.

[169] O. A-H. Hassan, T. Al-Rousan, A. A. Taleb, and A. Maaita. An efficient and scalable ranking technique for mashups involving RSS data sources. *Journal of Network and Computer Applications*, 39:179–190, 2014.

[170] S. Havre, B. Hetzler, and L. Nowell. ThemeRiver: Visualizing Theme Changes over Time. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2000)*, pages 115–123, held in Salt Lake City, UT, USA, 2000.

[171] M. A. Hearst. Untangling Text Data Mining. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics (ACL'99)*, pages 3–10, held in College Park, University of Maryland, MD, USA, 1999.

[172] M. A. Hearst. *Search user interfaces*. Cambridge University Press, Cambridge, NY, USA, 2009.

[173] P. Hennig, P. Berger, C. Lehmann, A. Mascher, and C. Meinel. Accelerate the detection of trends by using sentiment analysis within the blogosphere. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 503–508, held in Beijing, China, Aug 2014.

[174] R. Hiscott. How the Feed Changed the Way We Consume Content. Mashable. `http://mashable.com/2014/01/20/feed-history-infographic/`, 2014.

[175] History of RSS. RSS Specifications. `http://www.rss-specifications.com/history-rss.htm`, 2012.

[176] Z. Hmedeh, H. Kourdounakis, V. Christophides, C. Du Mouza, M. Scholl, and N. Travers. Indexes Analysis for Matching Subscriptions in RSS feeds. In *Proceedings of Bases de Données Avancées, BDA 2012*, pages 1–20, held in Clermont-Ferrand, France, Oct 2012.

[177] Z. Hmedeh, N. Vouzoukidou, N. Travers, V. Christophides, C. du Mouza, and M. Scholl. Characterizing Web Syndication Behavior and Content. In *Proceedings of the 12th International Conference on Web Information System Engineering (WISE2011)*, pages 29–42, held in Sydney, NSW, Australia, Oct 2011.

[178] T. K. Ho. Random Decision Forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR'95)*, pages 278–282, held in Montreal, Québec, Canada, Aug 1995.

[179] G. Hochard, Z. Lacroix, J. Creus, and B. Amann. A Semantic Map of RSS Feeds to Support Discovery. In *Proceedings of the Third International Conference on Resource Discovery (RED'10)*, pages 122–133, 2012.

[180] R. Horincar, B. Amann, and T. Artières. Best-Effort Refresh Strategies for Content-Based RSS Feed Aggregation. In L. Chen, P. Triantafillou, and T. Suel, editors, *Web Information Systems Engineering (WISE2010)*, volume 6488, pages 262–270. 2010.

[181] A. Hotho, A. Nürnberger, and G. Paaß. A Brief Survey of Text Mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20(1):19–62, May 2005.

[182] V. Hristidis, O. Valdivia, M. Vlachos, and P. S. Yu. Information Discovery Across Multiple Streams. *Information Sciences*, 179(19):3268–3285, 2009.

[183] I-C. Hsu. Personalized web feeds based on ontology technologies. *Information Systems Frontiers*, 15(3):465–479, 2013.

[184] L-F. Hsu. Mining on Terms Extraction from Web News. In *Proceedings of the Second International Conference on Computational Collective Intelligence: Technologies and Applications (ICCCI 2010)*, pages 188–194, held in Kaohsiung, Taiwan, 2010.

[185] C-L. Hu and C-K. Chou. RSS Watchdog: An Instant Event Monitor on Real Online News Streams. In *Proceedings of the 18th ACM Conference on Information and*

Knowledge Management (CIKM 2009), pages 2097–2098, held in Hong Kong, China, 2009.

[186] Y. Hu, S. Lim, and C. Rizos. Delivering GNSS Data Over the Internet Using RSS for Post-processing Applications. International Symposium on GPS/GNSS, held in Hong Kong, China, Dec 2005.

[187] A. Huang. Similarity Measures for Text Document Clustering. In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZC-SRSC2008)*, pages 49–56, held in Christchurch, New Zealand, Apr 2008.

[188] A. Hubmann-Haidvogel, A. Scharl, and A. Weichselbraun. Multiple coordinated views for searching and navigating web content repositories. *Information Sciences: Special Section: Web Search*, 179(12):1813–1821, 2009.

[189] Huffington Post. `http://www.huffingtonpost.co.uk/`, 2015.

[190] Huginn. `https://github.com/cantino/huginn/`, 2015.

[191] J. Hurtado. Automated System for Improving RSS Feeds Data Quality. *Computing Research Repository*, abs/1504.01433, 2015.

[192] IBM Builders reference room. John Backus. `http://www-03.ibm.com/ibm/history/exhibits/builders/builders_backus.html`, 2015.

[193] iCloud. `https://www.icloud.com`, 2015.

[194] iGooglePortal. `http://www.igoogleportal.com/`, 2015.

[195] ImageBam. `https://www.imagebam.com/`, 2015.

[196] IMDb. `http://www.imdb.com/`, 2015.

[197] Imgur. `https://www.imgur.com/`, 2015.

[198] InoReader. `https://www.inoreader.com/`, 2015.

[199] Instagram. `http://www.instagram.com/`, 2015.

[200] Instant RSS Search. `http://ctrlq.org/rss/`, 2015.

[201] Internet Archive. `https://archive.org/`, 2015.

[202] ISO/IEC 14977:1996. Information technology -- Syntactic metalanguage -- Extended BNF. `http://www.iso.org/iso/catalogue_detail.htm?csnumber=26153`, 1996.

[203] ISO/IEC 9075-1:2011. Information technology -- Database languages -- SQL -- Part 1: Framework (SQL/Framework). `http://www.iso.org/iso/catalogue_detail.htm?csnumber=53681`, 2011.

[204] A. N. Jebaseeli and E. Kirubakaran. A Survey on Sentiment Analysis of (Product) Reviews. *International Journal of Computer Applications*, 47(11):36–39, Jun 2012.

[205] C. Ji and J. Zhou. A Study on Recommendation Features for an RSS Reader. In *Proceedings of the 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 193–198, held in Huangshan, Anhui, China, Oct 2010.

[206] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning (ECML '98)*, pages 137–142, 1998.

[207] B. Johnson and B. Shneiderman. Tree-maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. In *Proceedings of the 2nd Conference on Visualization (VIS '91)*, pages 284–291, held in San Diego, CA, USA, 1991.

[208] Joomla. `http://www.joomla.org/`, 2015.

[209] G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger. *Web Engineering*. John Wiley and Sons, 2003.

[210] Kimono. `https://www.kimonolabs.com/`, 2015.

[211] N. Kittiphattanabawon and T. Theeramunkong. Relation Discovery from Thai News Articles Using Association Rule Mining. In *Proceedings of the Pacific Asia Workshop on Intelligence and Security Informatics (PAISI '09)*, pages 118–129, held in Bangkok, Thailand, 2009.

[212] R. Kohavi and F. Provost. Glossary of Terms. *Machine Learning*, 30(2-3):271–274, Feb 1998.

[213] M. Krstajić, F. Mansmann, A. Stoffel, M. Atkinson, and D.A. Keim. Processing Online News Streams for Large-Scale Semantic Analysis. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW)*, pages 215–220, Mar 2010.

[214] M. Krstajić, M. Najm-Araghi, F. Mansmann, and D. A. Keim. Incremental Visual Text Analytics of News Story Development. In P. K. Wong, D. L. Kao, M. C. Hao, C. Chen, R. Kosara, M. A. Livingston, Park. J, and I. Roberts, editors, *Proceedings of the Conference on Visualization and Data Analysis (VDA '12)*, 2012.

[215] O. Kucuktunc, B. B. Cambazoglu, I. Weber, and H. Ferhatosmanoglu. A Large-Scale Sentiment Analysis for Yahoo! Answers. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM '12)*, pages 633–642, held in Seattle, WA, USA, 2012.

[216] P. Kuzniewicz. 21 Essential Data Visualization Tools. KDnuggets. `http://www.kdnuggets.com/2015/05/21-essential-data-visualization-tools.html/`, 2015.

[217] R. Lambiotte, M. Ausloos, and M. Thelwall. Word statistics in Blogs and RSS feeds: Towards empirical universal evidence. *Journal of Informetrics*, 1:277–286, 2007.

[218] Y-F. Lan and Y-S. Sie. Using RSS to support mobile learning based on media richness theory. *Computers & Education*, 55(2):723–732, 2010.

[219] D. Lane. Mass Communications Context, Uses And Gratifications Theory. University of Kentucky, Lexington, KY, USA. `http://www.uky.edu/~drlane/capstone/mass/uses.htm`, 2001.

[220] P. R. Leary, D. P. Remsen, C. N. Norton, D. J. Patterson, and I. N. Sarkar. uBioRSS: Tracking taxonomic literature using RSS. *Bioinformatics*, 23(11):1434–1436, 2007.

[221] HK. Lee, H. Y. Kim, and H-K Lee. News package service based on TV-Anytime metadata gathered from RSS. In *Proceedings of the IEEE International Symposium on Consumer Electronics (ISCE 2007)*, pages 1–6, held in Irving, TX, USA, Jun 2007.

[222] T. P. Lee, A. A. A. Ghani, H. Ibrahim, and R. Atan. Coalescence of XML-based Really Simple Syndication (RSS) Aggregator for Blogosphere. In *Proceedings of*

the 7th International Conference on Advances in Mobile Computing and Multimedia (MoMM 2009), pages 530–534, held in Kuala Lumpur, Malaysia, 2009.

[223] T. P. Lee, A.A.A. Ghani, and C. Y. Huang. Survey on application tools of Really Simple Syndication (RSS): A case study at Klang Valley. In *Proceedings of the International Symposium on Information Technology (ITSim 2008)*, volume 3, pages 1–8, Aug 2008.

[224] M. Levene. *An Introduction to Search Engines and Web Navigation.* Pearson Education Limited, Harlow, Essex, England, UK, 1st edition, 2006.

[225] D. Lewis. Reuters-21578 Text Categorization Test Collection. `http://www.daviddlewis.com/resources/testcollections/reuters21578/`, 2015.

[226] X. Li, J. Yan, Z. Deng, L. Ji, W. Fan, B. Zhang, and Z. Chen. A Novel Clustering-based RSS Aggregator. In *Proceedings of the 16th International Conference on World Wide Web (WWW2007)*, pages 1309–1310, held in Banff, AB, Canada, 2007.

[227] LinkedIn. `https://www.linkedin.com/`, 2015.

[228] LinkedIn Pulse. `https://www.pulse.me/`, 2015.

[229] Listal. `https://www.listal.com/`, 2015.

[230] B. Liu. *Sentiment Analysis and Opinion Mining.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.

[231] B. Liu, H. Han, T. Noro, and T. Tokuda. Personal News RSS Feeds Generation Using Existing News Feeds. In *Proceedings of the 9th International Conference on Web Engineering (ICWE 2009)*, pages 419–433, held in San Sebastián, Spain, Jun 2009.

[232] H. Liu. *Spring 4 for Developing Enterprise Applications: An End-to-End Approach.* CreateSpace Independent Publishing Platform, 2014.

[233] H. Liu, V. Ramasubramanian, and E. G. Sirer. Client Behavior and Feed Characteristics of RSS, a Publish-Subscribe System for Web Micronews. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*, pages 3–3, held in Berkeley, CA, USA, 2005.

[234] LiveJournal. `http://www.livejournal.com/`, 2015.

[235] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text Classification Using String Kernels. *Journal of Machine Learning Research*, 2:419–444, Mar 2002.

[236] H. O. D. Longe and F. Salami. A Text Classifier Model for Categorizing Feed Contents Consumed by a Web Aggregator. *International Journal of Advanced Computer Science and Application*, 5(9):95–100, 2014.

[237] Lovestruck. `http://www.lovestruck.com`, 2016.

[238] M. Luenendonk. Top Programming Languages used in Web Development. Cleverism. `https://www.cleverism.com/programming-languages-web-development/`, 2015.

[239] D. Luo, J. Yang, M. Krstajić, W. Ribarsky, and D. A. Keim. EventRiver: Visually Exploring Text Collections with Temporal References. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):93–105, Jan 2012.

[240] D. Ma. Use of RSS feeds to push online content to users. *Decision Support Systems*, 54(1):740–749, 2012.

[241] S. Machlis. Chart and image gallery: 30+ free tools for data visualization and analysis. Computerworld. `http://www.computerworld.com/article/2506820/business-intelligence/business-intelligence-chart-and-image-gallery-30-free-tools-for-data-visualization-and-analysis.html`, 2016.

[242] M. Makpangou, B. Ngom, and S. Ndiaye. Friticores: A RSS Feed Monitoring and Dissemination System. In *Proceedings of AFRICOMM 2012 - Fourth International IEEE EAI Conference on e-Infrastructure and e-Services for Developing Countries*, held in Yaounde, Cameroon, Nov 2012.

[243] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[244] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[245] Many Eyes. `http://many-eyes.com`, 2009.

[246] S. Maple. The Ultimate Java Web Frameworks Comparison: Spring MVC, Grails, Vaadin, GWT, Wicket, Play, Struts and JSF. ZeroTurnaround.

http://zeroturnaround.com/rebellabs/the-curious-coders-java-web-frameworks-comparison-spring-mvc-grails-vaadin-gwt-wicket-play-struts-and-jsf/1/, 2013.

[247] J. Martin. *Information Engineering, Planning & Analysis: Book 2*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[248] N. Martin. Data Mining - Overview, Data Warehousing and Data Mining. DCSIS, Birkbeck, University of London, England, UK. http://www.dcs.bbk.ac.uk/study-with-us/modules/data-warehousing-and-data-mining, 2006.

[249] Mashable. http://mashable.com/category/web-apps/, 2015.

[250] Match. http://www.match.com/, 2015.

[251] H. Mayr. *Web Engineering* as a Specialization of *Software Engineering*: Differences in Project Management Education. *Journal on Systemics, Cybernetics and Informatics*, 3(5):84–91, 2005.

[252] M. McCandless, E. Hatcher, and O. Gospodnetić. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT, USA, 2010.

[253] W. Medhat, A. Hassan, and H. Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, 2014.

[254] Media RSS. RSS Advisory Board. http://www.rssboard.org/media-rss, 2015.

[255] Medical Information System. http://medisys.newsbrief.eu/, 2016.

[256] Mendeley. https://www.mendeley.com/, 2015.

[257] A. Messina and M. Montagnuolo. Content-Based RSS and Broadcast News Streams Aggregation and Retrieval. In *Proceedings of the Third IEEE International Conference on Digital Information Management (ICDIM 2008)*, pages 93–98, held in London, England, UK, Nov 2008.

[258] R.S. Michalski and R.L. Chilausky. Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2):125–161, 1980.

[259] Microsoft Excel. `https://products.office.com/excel`, 2015.

[260] Microsoft Office. `https://www.office.com/start/default.aspx`, 2015.

[261] Microsoft OneDrive. `https://onedrive.live.com`, 2015.

[262] Microsoft Outlook. `https://products.office.com/outlook`, 2015.

[263] Microsoft Windows. `http://windows.microsoft.com/`, 2015.

[264] Microsoft Windows RSS Platform. `https://msdn.microsoft.com/en-us/library/ms684701(v=vs.85).aspx`, 2015.

[265] Microsoft Word. `https://products.office.com/word`, 2015.

[266] R. Mikut and M. Reischl. Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(5):431–443, 2011.

[267] J. S. Modha, G. S. Pandi, and S. J. Modha. Automatic Sentiment Analysis for Unstructured Data. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(12):91–97, 2013.

[268] A. Møller and M. I. Schwartzbach. *An Introduction to XML and Web Technologies*. Pearson Education Limited, Harlow, Essex, England, UK, 2006.

[269] Moodle. `https://moodle.org/`, 2015.

[270] S. Mukherjee and P. Bhattacharyya. Sentiment Analysis, A Literature Survey. *Computing Research Repository*, abs/1304.4520, 2013.

[271] J. Murach and A. Steelman. *Murach's Java Servlets and JSPs*. Mike Murach and Associates Inc, 2nd edition, 2008.

[272] MySpace. `http://www.myspace.com/`, 2015.

[273] M. Naaman, J. Boase, and C-H. Lai. Is It Really About Me? Message Content in Social Awareness Streams. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (CSCW '10)*, pages 189–192, held in Savannah, GA, USA, 2010.

[274] Z. Nanli, Z. Ping, L. Weiguo, and C. Meng. Sentiment analysis: A literature review. In *Proceedings of the 2012 IEEE International International Symposium on Management of Technology (ISMOT'2012)*, pages 572–576, held in Hangzhou, Zhejiang, China, Nov 2012.

[275] National Information Standards Organization. Understanding Metadata. `http://www.niso.org/standards/resources/UnderstandingMetadata.pdf`, 2004.

[276] Netflix. `https://www.netflix.com/`, 2016.

[277] Netvibes. `http://www.netvibes.com/`, 2015.

[278] NewsBlur. `https://www.newsblur.com/`, 2015.

[279] NewzCrawler. `http://www.newzcrawler.com/`, 2015.

[280] F. A. Nielsen. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. *Computing Research Repository*, abs/1103.2903, 2011.

[281] K. Nørmark. Programming Paradigms, Functional Programming in Scheme. Department of Computer Science, Aalborg University, Aalborg, Denmark. `http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html`, 2013.

[282] Object Management Group. Documents Associated With OMG Unified Modeling Language (OMG), Version 2.5. `http://www.omg.org/spec/UML/2.5/`, 2015.

[283] Office of Communications. The Communications Market Report: United Kingdom, The UK is now a "smartphone society". `http://stakeholders.ofcom.org.uk/market-data-research/market-data/communications-market-reports/cmr15/uk/`, 2015.

[284] Open Office. `http://www.openoffice.org/`, 2015.

[285] Open University. `http://www.open.ac.uk/`, 2015.

[286] Oracle JDBC FAQ. `http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-faq-090281.html`, 2014.

[287] O'Reilly Media. `http://www.oreilly.com/`, 2015.

[288] M. O'Shea and M. Levene. Mining and visualising information from RSS feeds: a case study. *International Journal of Web Information Systems*, 7(2):105–129, 2011.

[289] M. O'Shea and M. Levene. visualRSS: a Platform to Mine and Visualise Social Data from RSS Feeds. In *Proceedings of the 12th International Conference on Web Engineering (ICWE 2012) 4th International Workshop on Lightweight Integration on the Web (ComposableWeb)*, pages 121–133, held in Berlin, Germany, Jul 2012.

[290] Page2RSS. `http://page2rss.com/`, 2015.

[291] I. Paik and H. Mizugai. Recommendation System Using Weighted TF-IDF and Naïve Bayes Classifiers on RSS Contents. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 14(6):631–637, 2010.

[292] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[293] P. Patel and S. G. Desai. A Comparative Study on Data Mining Tools. *International Journal of Advanced Trends in Computer Science and Engineering*, 4(2), 2015.

[294] K. Pearson. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, Jan 1895.

[295] M. S. Pera and Y-K. Ng. Synthesizing Correlated RSS News Articles Based on a Fuzzy Equivalence Relation. *International Journal of Web Information Systems*, 5(1):77–109, 2009.

[296] M. S. Pera and Y-K. Ng. Using Maximal Spanning Trees and Word Similarity to Generate Hierarchical Clusters of Non-Redundant RSS News Articles. *Journal of Intelligent Information Systems*, 39(2):513–534, 2012.

[297] D. Petrova-Antonova and R. Simov. jQRSS: A jQuery plugin for RSS and Atom feeds Parsing. In *Proceedings of the 12th International Conference on Computer Systems and Technologies (CompSysTech'11)*, pages 641–646, held in Vienna, Austria, 2011.

[298] X-H. Phan, L-M. Nguyen, and S. Horiguchi. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. In *Proceedings of the 17th International Conference on World Wide Web (WWW2008)*, pages 91–100, held in Beijing, China, Apr 2008.

[299] M. Pilgrim. What is RSS. `http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html`, 2002.

[300] M. Pilgrim. The myth of RSS compatibility. `http://web.archive.org/web/20110726001954/http://diveintomark.org/archives/2004/02/04/incompatible-rss`, 2004.

[301] W. A. Pinheiro, T. de S. Rodrigues, M.A.R. da Silva, M. A. N. da Silva, M. C. O. Silva, G. Xexéo, and J. M. de Souza. Autonomic RSS: Discarding Irrelevant News.

In *Proceedings of the Fifth International Conference on Autonomic and Autonomous Systems (ICAS 2009)*, pages 148–153, Apr 2009.

[302] Pinterest. `http://www.pinterest.com`, 2015.

[303] R. K. Pon, A. F. Cárdenas, D. Buttler, and T. Critchlow. Tracking Multiple Topics for Finding Interesting Articles. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-07)*, pages 560–569, held in San Jose, CA, USA, 2007.

[304] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[305] R. Prabowo and M. Thelwall. A comparison of feature selection methods for an evolving RSS feed corpus. *Information Processing & Management*, 42(6):1491–1512, 2006.

[306] R. Prabowo, M. Thelwall, and M. Alexandrov. Generating overview timelines for major events in an RSS corpus. *Journal of Informetrics*, 1(2):131–144, 2007.

[307] L. Preechaveerakul and W. Kaewnopparat. A Novel Approach: Secure Information Notifying System Using RSS Technology. In *Proceedings of the International Conference on Future Networks*, pages 95–99, held in Bangkok, Thailand, Mar 2009.

[308] R. Pressman and D. Lowe. *Web Engineering: A Practitioner's Approach*. McGraw-Hill, New York, NY, USA, 1st edition, 2009.

[309] R. S. Pressman. *Software Engineering: A Practitioner's Approach, European Adaptation*. McGraw-Hill, Europe, Maidenhead, Berkshire, England, UK, 3rd edition, 1992. Adapted by D. Ince.

[310] Python RSS Code. `http://wiki.python.org/moin/RssLibraries`, 2015.

[311] L. Qingcheng and L. Youmeng. Extracting Content from Web Pages Based on RSS. In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering (CSSE '08)*, volume 5, pages 218–221, held in Wuhan, Hubei, China, 2008.

[312] Quartz News. `http://qz.com/`, 2015.

[313] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA, 1993.

[314] Quora. `http://www.quora.com/`, 2015.

[315] C. M. Rahman, F. A. Sohel, P. Naushad, and S. M. Kamruzzaman. Text Classification using the Concept of Association Rule of Data Mining. *Computing Research Repository*, abs/1009.4582, 2010.

[316] S. Rajput and A. Arora. Designing Spam Model - Classification Analysis using Decision Trees. *International Journal of Computer Applications*, 75(10):6–12, Aug 2013.

[317] Y. Ramamohan, K. Vasantharao, C. K. Chakravarti, and A.S.K. Ratnam. A Study of Data Mining Tools in Knowledge Discovery Process. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(3):191–194, 2012.

[318] RDF Rich Site Summary (RSS). Cover Pages. `http://xml.coverpages.org/rss.html`, 2007.

[319] Reddit. `https://www.reddit.com`, 2015.

[320] J. W. Reed, Yu Jiao, T. E. Potok, B. A. Klump, M. T. Elmore, and A. R. Hurson. TF-ICF: A New Term Weighting Scheme for Clustering Dynamic Data Streams. In *Proceedings of the 5th International Conference on Machine Learning and Applications (ICMLA '06)*, ICMLA'06, pages 258–263, held in Orlando, FL, USA, 2006.

[321] Reedah. `https://www.reedah.com/`, 2015.

[322] T. Reenskaug. Models-views-controllers. Technical report, Xerox Palo Alto Research Center, CA, USA, Dec 1979.

[323] A. H. Renear, S. Sacchi, and K. M. Wickett. Definitions of Dataset in the Scientific and Technical Literature. In *Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem (ASIS&T '10)*, volume 47, pages 81:1–81:4, held in Pittsburgh, PA, USA, 2010.

[324] J. D. M. Rennie and R. Rifkin. Improving Multiclass Text Classification with the Support Vector Machine. Technical report, Artificial Intelligence Laboratory, Massachussetts Institute of Technology, MA, USA, Oct 2001.

[325] ResearchGate. `http://www.researchgate.net/`, 2015.

[326] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[327] R. Roesler. Relational RSS Clustering Techniques, Machine Learning. Computer Science, Stanford University, Stanford, CA, USA. `http://www.stanford.edu/class/cs229/proj2009/Roesler.pdf`, 2009.

[328] C. Rohrdantz, M. C. Hao, U. Dayal, L-E. Haug, and D. A. Keim. Feature-Based Visual Sentiment Analysis of Text Document Streams. *ACM Transactions on Intelligent Systems and Technology, Special Issue on Intelligent Visual Interfaces for Text Analysis*, 3(2):26:1–26:25, Feb 2012.

[329] L. Rokach and O. Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2008.

[330] W. Romsaiyud. RSS Offline. `https://www.researchgate.net/publication/228418759_RSS_Offline`, 2016.

[331] W. W. Royce. Managing the Development of Large Software Systems: Concepts and Techniques. In *Proceedings of the 9th International Conference on Software Engineering (ICSE '87)*, pages 328–338, Monterey, CA, USA, 1987.

[332] RSS 2.0 Specification. RSS Advisory Board. `http://www.rssboard.org/rss-specification`, 2009.

[333] RSS 2.0 Specification Roadmap. RSS Advisory Board. `http://www.rssboard.org/rss-specification#roadmap`, 2009.

[334] RSS Advisory Board. `http://www.rssboard.org/`, 2015.

[335] RSS Reader. `http://www.rss-readers.org/`, 2015.

[336] RSS Wizard. `http://rss-wizard.en.softonic.com/`, 2015.

[337] RSS4Medics. `http://www.rss4medics.com`, 2015.

[338] RSS4Twitter. `http://rss4twitter.appspot.com/`, 2016.

[339] RSSMicro. `http://www.rssmicro.com/`, 2015.

[340] RSSOwl. `http://www.rssowl.org/`, 2015.

[341] Sage. `http://sagerss.com/`, 2015.

[342] S. Saha, A. Sajjanhar, G. Shang, R. Dew, and Y. Zhao. Delivering Categorized News Items Using RSS Feeds and Web Services. In *Proceedings of the IEEE 10th International Conference on Computer and Information Technology (CIT 2010)*, pages 698–702, Jun 2010.

[343] A. Sajjanhar and Y. Zhao. Web Service to Deliver Filtered RSS Items to a Mobile Application. In *Proceedings of the Seventh ChinaGrid Annual Conference (ChinaGrid 2012)*, pages 128–133, 2012.

[344] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, USA, 1983.

[345] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, Nov 1975.

[346] J. J. Samper, P. A. Castillo, L. Araujo, J. J. Merelo, Ó. Cordón, and F. Tricas. NectaRSS, an intelligent RSS feed reader. *Journal of Network and Computer Applications*, 31(4):793–806, Nov 2008.

[347] A. Scharl, A. Hubmann-Haidvogel, M. Sabou, A. Weichselbraun, and H-P. Lang. From Web Intelligence to Knowledge Co-Creation – A Platform to Analyze and Support Stakeholder Communication. *IEEE Internet Computing*, 17(5):21–29, 2013.

[348] Scoop.It! `http://www.scoop.it//`, 2013.

[349] F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47, Mar 2002.

[350] Selfoss. `http://selfoss.aditu.de/`, 2015.

[351] Semantria. `https://semantria.com/`, 2015.

[352] S. Sen, W. Geyer, M. Muller, M. Moore, B. Brownholtz, E. Wilcox, and D. R. Millen. FeedMe: A Collaborative Alert Filtering System. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work (CSCW '06)*, pages 89–98, held in Banff, AB, Canada, 2006.

[353] F. Shaikh and A. Rajawat. Approach for Developing Scientific News Aggregators Using ATOM Feeds. *International Journal of Electronics and Computer Science Engineering*, 1(4):2279–2284, 2012.

[354] Y. R. Shema. visualRSS Android Client. Master's thesis, DCSIS, Birkbeck, University of London, 2013.

[355] K. C. Sia, J. Cho, and H-K. Cho. Efficient Monitoring Algorithm for Fast News Alerts. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):950–961, Jul 2007.

[356] K.C. Sia, J. Cho, K. Hino, Y. Chi, S. Zhu, and B. L. Tseng. Monitoring RSS Feeds based on User Browsing Pattern. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM-07)*, pages 161–168, held in Boulder, CO, USA, Mar 2007.

[357] A. Silberstein, J. Terrace, B. F. Cooper, and R. Ramakrishnan. Feeding Frenzy: Selectively Materializing Users' Event Feeds. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD'10)*, pages 831–842, held in Indianapolis, IN, USA, 2010.

[358] A. Šilić and B. D. Bašić. Visualization of Text Streams: A Survey. In *Part II of Proceedings of the Knowledge-Based and Intelligent Information and Engineering Systems - 14th International Conference (KES 2010)*, pages 31–43, held in Cardiff, Wales, UK, Sep 2010.

[359] S. J. Simoff, M. H. Böhlen, and A. Mazeika. Visual Data Mining: An Introduction and Overview. In S. J. Simoff, M. H. Böhlen, and A. Mazeika, editors, *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*, pages 1–12. 2008.

[360] G. Singh and S. Sahu. Review on "Really Simple Syndication (RSS) Technology Tools". In *Proceedings of the IEEE International Conference on Computational Intelligence Communication Technology (ICICT-2015)*, pages 757–761, held in Ghaziabad, Uttar Pradesh, India, Feb 2015.

[361] Skype. `http://www.skype.com/`, 2015.

[362] K. Smitty. 9 Word Cloud Generators That Aren't Wordle. Edudemic. `http://www.edudemic.com/9-word-cloud-generators-that-arent-wordle/`, 2013.

[363] SnapChat. `http://www.snapchat.com/`, 2015.

[364] T. Snowsill, F. Nicart, M. Stefani, T. De Bie, and N. Cristianini. Finding surprising patterns in textual data streams. In *Proceedings of the 2nd International Workshop*

on Cognitive Information Processing (CIP), pages 405–410, held on Elba Island, Italy, Jun 2010.

[365] Y. E. Soelistio and M. R. S. Surendra. Simple Text Mining for Sentiment Analysis of Political Figure Using Naïve Bayes Classifier Method. *Computing Research Repository*, abs/1508.05163, 2015.

[366] H. Solanki. Comparative Study of Data Mining Tools and Analysis with Unified Data Mining Theory. *International Journal of Computer Applications*, 75(16):23–28, Aug 2013.

[367] I-Y. Song and K. Froehlich. Entity-relationship modeling. *IEEE Potentials*, 13(5):29–34, Dec 1994.

[368] SoundCloud. `https://soundcloud.com/`, 2015.

[369] K. Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[370] StackExchange. `http://stackexchange.com/`, 2015.

[371] Stanford Natural Language Processing Group. `http://nlp.stanford.edu/software/`, 2015.

[372] A. Statnikov, C. F. Aliferis, D. P. Hardin, and I. Guyon. *A Gentle Introduction to Support Vector Machines in Biomedicine: Case Studies*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1st edition, 2011.

[373] C. A. Steed, M. Drouhard, J. Beaver, J. Pyle, and P. L. Bogen. Matisse: A Visual Analytics System for Exploring Emotion Trends in Social Media Text Streams. In *Proceedings of the IEEE International Conference on Big Data (Big Data)*, pages 807–814, held in Santa Clara, CA, USA, Oct 2015.

[374] C. A. Steed, T. E. Potok, R. M. Patton, J. R. Goodall, C. Maness, and J. Senter. Interactive Visual Analysis of High Throughput Text Streams. In *2nd Workshop on Interactive Visual Text Analytics*, held in Seattle, WA, USA, Oct 2012.

[375] Storify. `https://storify.com/`, 2015.

[376] StumbleUpon. `https://www.stumbleupon.com/`, 2015.

[377] B. Suda. The 38 best tools for data visualization. Creative Bloq. `http://www.creativebloq.com/design-tools/data-visualization-712402?page=1`, 2016.

[378] Swivel. `http://www.swivel.com`, 2009.

[379] Syndic8. `http://www.syndic8.com/`, 2005.

[380] Tableau. `http://www.tableausoftware.com/`, 2011.

[381] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based Methods for Sentiment Analysis. *Computational Linguistics*, 37(2):267–307, Jun 2011.

[382] F. G. Taddesse, J. Tekli, R. Chbeir, M. Viviani, and K. Yetongnon. Semantic-based Merging of RSS Items. *World Wide Web*, 13(1-2):169–207, Mar 2010.

[383] TalkTalk TV Store. `https://www.talktalktvstore.co.uk/`, 2010.

[384] D. I. Tamir and J. P. Mitchell. Disclosing information about the self is intrinsically rewarding. *Proceedings of the National Academy of Sciences of the United States of America*, 109(21):8038–8043, 2012.

[385] Z. Tang and K. Ma. RSSCube: A Content Syndication and Recommendation Architecture. *International Journal of Data Base Theory and Application*, 7(4):237–248, 2014.

[386] TechCrunch. `http://www.techcrunch.com/`, 2015.

[387] Technorati. `http://www.technorati.com`, 2015.

[388] Techopedia. `http://www.techopedia.com/definition/27745/big-data`, 2015.

[389] J. Tee. Sharding in the Cloud. TheServerSide. `http://www.theserverside.com/feature/Sharding-in-the-Cloud`, 2011.

[390] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A New View on News. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS 2008)*, pages 18:1–18:10, held in Irvine, CA, USA, 2008.

[391] Z. Teng, Y. Liu, and F. Ren. Create Special Domain News Collections through Summarization and Classification. *IEEE Transactions on Electrical and Electronic Engineering*, 5:56–61, 2010.

[392] The Old Reader. `https://theoldreader.com/`, 2015.

[393] The PhD Programme. DCSIS, Birkbeck, University of London, London, UK. `http://www.dcs.bbk.ac.uk/research/pursuing-a-research-degree/the-phd-programme/`, 2016.

[394] M. Thelwall. Heart and Soul: Sentiment Strength Detection in the Social Web with SentiStrength. `http://sentistrength.wlv.ac.uk/documentation/SentiStrengthChapter.pdf`, 2013.

[395] M. Thelwall. Sentiment strength detection for the Social Web. `http://sentistrength.wlv.ac.uk/documentation/course.html`, 2013.

[396] M. Thelwall. SentiStrength User Manual. `http://sentistrength.wlv.ac.uk/documentation/SentiStrengthJavaManual.doc`, 2014.

[397] M. Thelwall, K. Buckley, and G. Paltoglou. Sentiment Strength Detection for the Social Web. *Journal of the Association for Information Science and Technology*, 63(1):163–173, 2012.

[398] M. Thelwall, R. Prabowo, and R. Fairclough. Are Raw RSS Feeds Suitable for Broad Issue Scanning? A Science Concern Case Study. *Journal of the Association for Information Science and Technology*, 57(12):1644–1654, 2006.

[399] Thunderbird. `https://www.mozilla.org/thunderbird/`, 2015.

[400] Tinder. `http://www.gotinder.com/`, 2015.

[401] Tiny Tiny RSS. `https://tt-rss.org/forum/`, 2015.

[402] W. H. Tok, S. Bressan, and M-L. Lee. Danaïdes: Continuous and Progressive Complex Queries on RSS Feeds. In R. Kotagiri, P. R. Krishna, M. Mohania, and E. Nantajeewarawat, editors, *Advances in Databases: Concepts, Systems and Applications*, volume 4443, pages 1115–1118. 2007.

[403] C. Trabelsi and S. B. Yahia. A Probabilistic Approach for Events Identification from Social Media RSS Feeds. In B. Hong, X. Meng, L. Chen, W. Winiwarter, and W. Song, editors, *Database Systems for Advanced Applications*, volume 7827, pages 139–152. 2013.

[404] Trackur. `http://www.trackur.com/`, 2015.

[405] Transport for London Corporate Archives Research Guides. Research Guide No 24: Harry Beck. `http://content.tfl.gov.uk/research-guide-no-24-harry-beck.pdf`, 2015.

[406] E. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, USA, 2nd edition, 2006.

[407] Tumblr. `http://www.tumblr.com/`, 2015.

[408] Twitter. `http://www.twitter.com`, 2015.

[409] Udacity. `http://www.udacity.com`, 2015.

[410] Udemy. `http://www.udemy.com`, 2015.

[411] Ukora News Search Service. `http://www.rsssearchhub.com/`, 2015.

[412] M. Van Kleek, P. André, D. Karger, and M. C. Schraefel. Personalized Experiences for End-User Programming on the Web. In *Proceedings of the End-User Programming Workshop (EUP) at Conference on Human Factors in Computing Systems (CHI'09)*, held in Boston, MA, USA, Apr 2009.

[413] M. Van Kleek, P. André, M. Perttunen, M. Bernstein, D. Karger, R. Miller, and M. C Schraefel. AtomsMasher: Personalised Context-Sensitive Automation for the Web. Technical report, MIT and University of Southampton, Mar 2008.

[414] M. Van Kleek, P. André, D. A. Smith, M. L. Wilson, M.l Bernstein, D. Karger, and M. C Schraefel. AtomsMasher: PeRSSonalized Information Delivery and Management on the Web. Technical report, MIT and University of Southampton, Nov 2007.

[415] F. Viégas and M. Wattenberg. TIMELINES: - Tag Clouds and the Case for Vernacular Visualization. *Interactions*, 15(4):49–52, 2008.

[416] F. M. Villoria, O. Díaz, and S. F. Anzuola. Powering RSS Aggregators with Ontologies: a case for the RSSOwl aggregator. In Y. Manolopoulos, J. Filipe, P. Constantopoulos, and J. Cordeiro, editors, *Proceedings of the Eighth International Conference on Enterprise Information Systems*, pages 197–200, held in Paphos, Cyprus, May 2006.

[417] Vimeo. `http://www.vimeo.com/`, 2015.

[418] G. Vinodhini and R. M. Chandrasekaran. Sentiment Analysis and Opinion Mining: A Survey. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(6):282–292, Jun 2012.

[419] F. Vis. A critical reflection on Big Data: Considering APIs, researchers and tools as data makers. *First Monday*, 18(10), 2013.

[420] Visual Complexity. `http://www.visualcomplexity.com/`, 2013.

[421] Voice RSS. `http://www.voicerss.org/`, 2015.

[422] W3C DATA ACTIVITY Building the Web of Data. `https://www.w3.org/2013/data/`, 2016.

[423] W3Schools. `http://www.w3schools.com/xml/xml_rss.asp`, 2016.

[424] A. H. Wahbeh, Q. A. Al-Radaideh, M. N. Al-Kabi, and E. M. Al-Shawakfa. A Comparison Study between Data Mining Tools over some Classification Methods. *International Journal of Advanced Computer Science and Applications, Special Issue on Artificial Intelligence*, 2(8):19–26, 2012.

[425] N. Walsh. A Technical Introduction to XML. `http://www.xml.com/pub/a/98/10/guide0.html?page=2`, 1998.

[426] C. Wang, Z. Xiao, Y. Liu, Y. Xu, A. Zhou, and K. Zhang. SentiView: Sentiment Analysis and Visualization for Internet Popular Topics. *IEEE Transactions on Human-Machine Systems*, 43(6):620–630, Nov 2013.

[427] F. Wanner, C. Rohrdantz, F. Mansmann, D. Oelke, and D. A. Keim. Visual Sentiment Analysis of RSS News Feeds Featuring the US Presidential Election in 2008. In *Proceedings of the IUI'09 Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW 2009)*, held on Sanibel Island, FL, USA, Feb 2009.

[428] F. Wanner, A. Stoffel, D. Jäckle, B. C. Kwon, A. Weiler, and D. A. Keim. State-of-the-Art Report of Visual Analysis for Event Detection in Text Data Streams. In *Proceedings of the Eurographics Conference on Visualization (EuroVis 2014)*, pages 125–139, held in Swansea, Wales, UK, Jun 2014.

[429] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2004.

[430] Web Master World RSS. `http://www.webmasterworld.com/rss_atom/`, 2015.

[431] WebLyzard. `https://www.weblyzard.com/`, 2016.

[432] Webopedia. `http://www.webopedia.com/`, 2015.

[433] D. E. Webster, W. Huang, D. Mundy, and P. Warren. Context-Orientated News Filtering for Web 2.0 and Beyond. In *Proceedings of the 15th International Conference on World Wide Web (WWW2006)*, pages 1001–1002, held in Edinburgh, Scotland, UK, 2006.

[434] K. Wegrzyn-Wolska and P. S. Szczepaniak. Classification of RSS-Formatted Documents Using Full Text Similarity Measures. In *Proceedings of the 5th International Conference on Web Engineering (ICWE 2005)*, pages 400–405, held in Sydney, NSW, Australia, Jul 2005.

[435] A. Weichselbraun, S. Gindl, and A. Scharl. Extracting and Grounding Context-Aware Sentiment Lexicons. *IEEE Intelligent Systems*, 28(2):39–46, Mar 2013.

[436] P. Weiner. LINEAR PATTERN MATCHING ALGORITHMS. In *Proceedings of the 14th Annual Symposium on Switching and Automata Theory (SWAT '73)*, pages 1–11, 1973.

[437] What is RSS? RSS Specifications. `http://www.rss-specifications.com/what-is-rss.htm`, 2011.

[438] WhatsApp. `http://www.whatsapp.com/`, 2015.

[439] A. Whiting and D. Williams. Why people use social media: a uses and gratifications approach. *Qualitative Market Research: An International Journal*, 16(4):362–369, 2013.

[440] Wikipedia. `http://www.wikipedia.org/`, 2015.

[441] Wikipedia. `http://en.wikipedia.org/wiki/Comparison_of_feed_aggregators`, 2015.

[442] M. Wilce. Mining Data Sets from Web Feeds. Master's thesis, DCSIS, Birkbeck, University of London, 2009.

[443] E. Wilde and M. Gaedke. Web Engineering Revisited. In *Proceedings of the 2008 International Conference on Visions of Computer Science, VoCS'08: BCS International Academic Conference*, pages 41–49, held in London, England, UK, Sep 2008.

[444] R. K. Wills. Efficient Sentiment Analysis of Feeds for Rapid User Information Gain. `https://www.cs.umd.edu/sites/default/files/scholarly_papers/Wills.pdf`, 2012.

[445] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing Contextual Polarity: An Exploration of Features for Phrase-level Sentiment Analysis. *Computational Linguistics*, 35(3):399–433, Sep 2009.

[446] I. H. Witten. Text mining. In M. P. Singh, editor, *The Practical Handbook of Internet Computing*, pages 14–1–14–22. Chapman and Hall & CRC Press, Boca Raton, FL, USA, 2004.

[447] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Francisco, CA, USA, 2nd edition, 2005.

[448] WizardRSS. `http://www.fastcomet.com/wizardrss-demo`, 2016.

[449] WordNet. `http://wordnet.princeton.edu/`, 2015.

[450] WordPress. `http://www.wordpress.com/`, 2015.

[451] W^{3}Techs. Usage of server-side programming languages for websites. `http://w3techs.com/technologies/overview/programming_language/all`, 2016.

[452] S. Wu. Click Prediction and Preference Ranking of RSS Feeds, Machine Learning. Computer Science, Stanford University, Stanford, CA, USA. `http://cs229.stanford.edu/proj2009/Wu.pdf`, 2009.

[453] XML Applications and Initiatives. Cover Pages. `http://xml.coverpages.org/xmlApplications.html`, 2005.

[454] XmlSlurper. `http://docs.groovy-lang.org/latest/html/api/groovy/util/XmlSlurper.html`, 2015.

[455] Yahoo. `http://www.yahoo.com`, 2015.

[456] Yahoo *Pipes*. `http://pipes.yahoo.com/pipes/`, 2012.

[457] Yahoo News. `https://uk.news.yahoo.com/world/`, 2015.

[458] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 42–49, held in Berkeley, CA, USA, 1999.

[459] Y. Yang and J. O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997)*, pages 412–420, held in Nashville, TN, USA, 1997.

[460] E. Yourdon. *Modern Structured Analysis*. Yourdon Press, Upper Saddle River, NJ, USA, 1989.

[461] YouTube. `http://www.youtube.com`, 2015.

[462] B. Yu. An evaluation of text classification methods for literary study. *Literary and Linguistic Computing*, 23(3):327–343, 2008.

[463] M. W. Yuan, P. Jiang, J. Zhu, and X. N. Wang. Sensing Semantics of RSS Feeds by Fuzzy Matchmaking. *Intelligent Information Management*, 2(2):110–119, 2010.

[464] Zapier. `https://zapier.com/`, 2015.

[465] Y. Zhou, X. Chen, and C. Wang. A Self-Organizing Search Engine for RSS Syndicated Web Contents. In *Proceedings of the 22nd International Conference on Data Engineering Workshops*, pages 52–61, 2006.

[466] Z. Zhou, X. Zhang, and M. Sanderson. Sentiment Analysis on Twitter through Topic-Based Lexicon Expansion. In H. Wang and M. A. Sharaf, editors, *Proceedings of Databases Theory and Applications - 25th Australasian Database Conference (ADC 2014)*, held in Brisbane, QLD, Australia, Jul 2014.

[467] X. Zhu, A. B. Goldberg, R. Brachman, and T. Dietterich. *Introduction to Semi-Supervised Learning*. Morgan & Claypool Publishers, 2009.

[468] M. M. Zloof. Query-by-Example: a data base language. *IBM Systems Journal*, 16(4):324–343, Dec 1977.