Mining Named Entities from Search Engine Query Logs

Areej Alasiry
Birkbeck, University of London
United Kingdom, London
areej@dcs.bbk.ac.uk

Mark Levene
Birkbeck, University of London
United Kingdom, London
mark@dcs.bbk.ac.uk

Alexandra Poulovassilis
Birkbeck, University of London
United Kingdom, London
ap@dcs.bbk.ac.uk

ABSTRACT

We present a seed expansion based approach to classify named entities in web search queries. Previous approaches to this classification problem relied on contextual clues in the form of keywords surrounding a named entity in the query. Here we propose an alternative approach in the form of a *Bag-of-Context-Words* (BoCW) that is used to represent the context words as they appear in the snippets of the top search results for the query. This is particularly useful in the case where the query consists of only the named entity without any context words, since in the previous approaches no context is discovered. In order to construct the BoCW, we employ a novel algorithm, which iteratively expands a *Class Vector* that is created through expansion by gradually aggregating the BoCWs of similar named entities appearing in other queries. We provide comprehensive experimental evidence using a commercial query log showing that our approach is competitive with existing approaches.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval —*Query formulation*
Keywords: named entity recognition, query logs.

1. INTRODUCTION

Named Entity Recognition (NER) is the task of extracting from text entity instances falling into different categories, such as Person, Location, or Film. In the literature, many approaches have been proposed for NER in text in order to create repositories of named entities [10, 16]. More recently, in the context of web usage, search query logs have been used since they include named entities as expressed from users' perspectives. A search query comprises a list of keywords that a user submits to a search engine to express a certain information need. Beyond the sequence of words, queries often contain references to names of entities, e.g. a person's name such as in 'nelson mandela biography' or a film name such as in 'the hobbit'. As search queries are short, and ambiguous, classical NER approaches cannot be applied to them without modification; recent research such as that in [14, 17] has been proposed to meet these challenges.

We now briefly outline the contributions of this paper to the problem of NER in queries. First, we present a seed expansion approach that utilises a *Bag-of-Context-Words (BoCW)* to represent the context of query NEs. The context words are collected from the snippets returned from a search engine's results list for that query. During the expansion a single BoCW (which we refer to as a *Class Vector*) is iteratively expanded as new NEs from queries are tagged with a target NE class. New NEs can only be tagged when they share enough words with the class vector, i.e., the extracted BoCW vectors for the query NEs are *similar* to the class vector. Using a Bag-of-Words as we do here, rather than a pre-defined pattern or an exact query context (i.e., the remainder of a query after excluding the target NE string) [12, 17], is much more flexible, since it does not impose a specific sequence or number of words to be matched in order to find new NEs, and also allows us to attach a weight to each context word. Furthermore, using snippets rather than just the query allows us to resolve the problem of query named entity detection when the query consists of just the NE string without any contextual clues that could be used to reveal the class of the target NE. We found that this type of query comprises approximately 27% of all queries in the query log. Finally, through experimental evidence, we found that the recall scored by our expansion approach can be further improved. This is achieved by gradually decreasing the similarity threshold between the BoCW of the candidate NE and the Class Vector as the expansion proceeds.

Second, our proposed expansion approach starts with automatically discovered fertile seed instances over a small uniform random sample of the query named entities. A fertile seed is a NE instance whose BoCW expansion will lead to the classification of other NEs belonging to the target class.

Third, given a target query NE to be classified, we found that it can occur in one of three types of queries: (i) a *Focused Query* (FQ), (ii) a *Very Focused Query* (VFQ), or (iii) an *UnFocused Query* (UFQ). This categorisation of queries is based on observing whether the target NE occurs solely in the query (FQ), with extra words that give further details of that NE (VFQ), or with one or more other named entities (UFQ). We found that the quality of the constructed Bag-of-Context-Words differs given the type of query in which the NE occurs.

We evaluated our approach of query named entity recognition (QNER) for six target NE classes, to assess: (1) the precision and recall that can be achieved by our expansion

IDEAS'14, July 07-09 2014, Porto, Portugal

http://dx.doi.org/10.1145/2628194.2628224

approach over an annotated subset of the query log, and (2) the usefulness of the created Class Vector at the end of the expansion in classifying NEs appearing in the full query log. Measuring precision and recall can only be achieved when "gold standard" lists of instances existing in the dataset are identified beforehand. Therefore, we employed DBpedia [4] to find the possible class of each candidate NE in the query log. The candidate NEs were identified in a previous stage of this work [1, 2], where we used query segmentation and part-of-speech tagging to identify candidate NEs. In addition, to assess the quality of the created Class Vector, we measured the similarity of each candidate named entity BoCW to the Class Vector and we manually checked the top 250 NEs to measure precision@K for each of the NE classes. Moreover, for comparative assessment, we implemented Paşca's approach [17] as a baseline for seed-based expansion approaches for QNER, and we compared its performance against our approach using our annotated dataset.

The rest of the paper is organised as follows. In Section 2, we discuss related work in the area of NER in search queries. In Section 3, we define the problem of QNER, followed by the categorisation of NE queries in Section 4. In Section 5 we present our seed expansion approach to classify candidate NEs in queries. Experimental evaluation using our approach on a commercial query log is described in Sections 6 and 7. In Section 8, we provide a comparative assessment of our approach to that of Paşca's seed based expansion approach for QNER. Finally, in Section 9 we draw conclusions from our work and present directions for future work.

2. RELATED WORK

Early work in NER was usually performed with handcrafted rules [11], or with supervised learning techniques, where an annotated corpus (e.g., CoNLL-2003 consisting of news articles) was used as a training dataset to detect NEs [15]. In the case where an annotated corpus is not available, weakly supervised approaches such as seed expansion may be applied. Such an approach starts with a set of seed instances per class, and then the patterns in which these seeds appeared are extracted and used to find new instances belonging to the same class due to sharing similar patterns. The way the pattern, also referred to as *context*, is defined depends on the type of the data on which a particular approach is applied. NER approaches such as those of Collins and Singer [6] and Rilloff and Shoen [19] rely on the grammatical structure of the sentences in which the NE appeared in order to create a pattern.

When applying NER to open-domain documents such as on the Web, the patterns need to be more general and corpus-derived. Therefore, Paşca et al. [18] and Etzoni et al. [10] used patterns that were mined from the data set, by using fact seeds such as a person's name and a date, in order to find patterns representing the binary relation of a person and his/her date of birth. In [18], a generalised pattern for a seed, e.g. (Person, Year) is a generalised form of the sentence in which it appeared, such as (Prefix: *StartOfSent*, Infix: *CL4 CL8 CL22 born*, Postfix: *in CL17*), where CL# is a class label of distributionally similar words, e.g., CL4:{*is, was, does*}, and CL26:{*entrepreneur, illustrator, artist,...*}.

On the other hand, QNER approaches as in [12, 17] define a pattern to be simply the remainder of the query after excluding the seed string. For example, '# games' is the context extracted from query 'final fantasy games' to detect NEs in the class Video Games. We observed that the same context may appear in queries that do not contain NEs of the target class. For example, the same context appears in the query 'free online games', where 'free online' is not a NE, or in the query 'PS3 games', where 'PS3' is a NE yet not of the class Video Game. We also have queries such as 'The Hunger Games', where the context is part of a NE that is in the class Film. In addition, if the query contains only the NE string, there is no information from which to derive a context that can be used to classify that NE.

Instead of just using the current user query, Du et al.[8] used the full search session to identify and classify named entities. Two search session features were used, namely, 'class features', the class of the named entity appearing in the previous query in the session, and the 'overlap feature' of the words between the previous and the current query. Similarly to Guo's approach [12], they used only the query's string to identify named entities.

Another approach presented by Jain and Pennacchiotti [14], makes use of Capitalisation to identify NEs in queries. Although confidence scores are assigned to each extracted NE based on its presence in a web corpus, relying on Capitalisation will often miss many potential NEs in the query log that were not typed with capital letters by the user.

3. PROBLEM DEFINITION

Given a set of candidate NEs extracted from each query in a search engine query log, the objective is to classify each candidate NE into a target NE class. In order to achieve that, clues surrounding a NE occurrence can be used to identify the class to which a NE belongs. In the case of search engine queries, a pattern can be defined as the remainder of a query after excluding the NE. For example, the query template '# trailer' can be used to identify NEs in the class Film. However, there are three main challenges posed by the nature of queries: (i) they are very *short*, consisting of two to three words on average, (ii) they are *unstructured*, as users do not follow grammatical rules such as capitalisation when submitting queries to search engines, and (iii) they are *ambiguous*, e.g. 'trailer' follows TV shows and Video Games as well as Films. When defining a pattern to be the remainder of the query, two main problems emerge. First, setting the boundaries of a NE, and secondly, the ambiguity of the query context. Table 1 presents examples of queries consisting of named entities, where a typical context such as '# book' usually follows a name of a Book, but it can also occur as part of a NE, for example 'The Note Book' is a Film name and 'Kelly Blue Book' refers to a Company's name.

Here, we use snippets of the results related to the target NE query in order to resolve the ambiguity of the query context. Rather than just using the queries themselves, we utilise the snippets of the top-n results of the query in which the NE appeared. Although the snippets for a query with a NE often do not follow a generalised pattern and are not grammatically structured, they often share some common words. Therefore, rather than using a fixed set of patterns or a generalised form of a sentence as a pattern, we use the Bag-of-Words approach to represent the contexts of candidate NEs. Representing the contexts of NEs as vectors in a common vector space provides additional flexibility to identify the similarity scores between instances belonging to the same NE class.

Table 1: Sample of query contexts and corresponding NE classess

Query template: '# lyrics'		Query template: '# games'		Query template: '# book'	
Query	NE Class	Query	NE Class	Query	NE Class
Beautiful Life lyrics	Song	**Uncharted** games	Video game	**Harry Potter** book	Book
James Morrison lyrics	Person	**The Hunger Games**	Film	**The Note Book**	Film
Music and Lyrics	Film	free games	*Not NE*	**Kelley Blue Book**	Company

4. THREE TYPES OF NE QUERIES

Given a target query NE, analysing the snippets related to that query we found that queries can be categorised into three types:

(1) Focused Queries (FQ): queries consisting of only a single candidate NE. We found that the snippets set consists of sentences that provide clear indication of the NE class, and are generally more exclusive to the target NE class. For example, the query 'Nicolas Cage' has the following snippets sets of search results:

"Nicolas Cage, Actor: Adaptation. The son of comparative literature professor ...and dancerchoreographer..." (IMDB).

"Nicolas Cage (born Nicolas Kim Coppola; January 7, 1964) is an American actor, producer and director, having appeared in over 60 films including Raising ..." (Wikipedia).

"The following is the complete filmography of American actor, director, and producer Nicolas Cage ..."(Wikipedia).

The snippets contains words with high term frequencies such as 'actor' and 'filmography' that are specifically shared by Actors, and words like 'born' as well as 'son', that are generally shared by the NEs in the class Person.

(2) Very Focused Queries (VFQ): queries having a single NE and one or more query context words. For example, the following very focused queries and the corresponding snippets contain the NE Nicolas Cage:

'nicolas cage pics': *"29 January 2011... 414 pictures of Nicolas Cage. Recent images. Hot! View the latest Nicolas Cage photos. Large gallery of Nicolas Cage pics. Movie posters."(IMDB).*

'nicolas cage interview': *"Nicolas Cage has a reputation for excess ... failure to publicise it during the interview" (The Guardian).*

In the snippets set, words like 'pictures', 'view', 'pics', 'interviews', 'news' are the dominating ones, with higher term frequency. Other examples of VFQs are 'nicolas cage films', 'nicolas cage wife', and 'how old is nicolas cage'.

(3) Unfocused Queries (UFQ): queries containing more than one candidate NE that may or may not belong to the same NE class. For example, the unfocused query 'next nicolas cage' containing the named entities 'nicolas cage' (actor) and 'next' (movie) has the snippets:

"Directed by Lee Tamahori. With Nicolas Cage, Julianne Moore, Jessica Biel, Thomas Kretschmann. A Las Vegas magician who can see into the future is ... "(IMDB).

"Next is a 2007 American science-fiction action thriller film directed by Lee Tamahori and stars Nicolas Cage, Julianne Moore and Jessica Biel. The film's original ..." (Wikipedia).

The snippets contain words related to the movie 'next' such as 'review', 'fiction', 'movie', as well as words such as other actor names starring in that movie. Other examples of UFQs are 'kevin bacon and nicolas cage', 'nicolas cage in world trade center', and 'the ghost rider with nicolas cage'.

In this paper, in order to validate the effectiveness of using snippets versus queries to classify NEs, we will focus on FQs and VFQs. We leave the classification of candidate NEs in UFQs for future work.

5. QUERY NE CLASSIFICATION

In this section we first define BoCW and following that we present our approach to query NE classification.

5.1 Bag of Context Words

Each query in the log is processed to extract a *NE instance* and a *Bag-of-Context-Words (BoCW)*. A NE instance is the string representing the candidate NE that is extracted from a search query; since the same NE may occur in more than one query we refer to each occurrence of the NE as a NE instance. In the snippets set, each mention of the NE instance may be preceded or followed by words that can provide some evidence of the NE class; we refer to these as *context words*. The collection of context words surrounding a NE instance in the snippets set is used to create a BoCW. A BoCW is a vector with one component for each context word, together with a weight which is the term frequency of that word in the snippets set. For a word to be considered a context word it should satisfy the following conditions:

1. It is not a member of a stop words list consisting of extremely common English words such as 'is', 'are', 'an' and 'a'.
2. Its part of speech is either a noun or a verb. For example, adjectives are excluded, since they can be common attributes of many named entities regardless of their class.
3. After excluding words in the conditions 1 and 2, the context word should fall within a predefined window size of n words around the NE occurrence.
4. It appears within the sentence boundaries in which the NE occurred.

Table 2 shows examples of the NE instances and the corresponding extracted BoCWs. When extracting context words, a match of the NE may be full or partial. For example, for the NE 'Britney Spears', the context words in the sentence '*Britney Jean Spears* (born December 2, 1981) is an American recording artist and entertainer' were extracted even though an exact full match was not found.

5.2 Classification of NEs

Our algorithm for query NE classification is based on the assumption that NEs belonging to the same NE class have similar BoCWs. Starting with a set of seeds for each target NE class, the algorithm iterates through three main phases: *Instance Matching*, *Class Vector Aggregation*, and *BoCW Matching*. Each phase is executed as follows:

Instance Matching: In this phase, the list of NE instances are scanned to find those whose string exactly matches a seed instance. Every matched NE instance is labelled with the target class. In the first iteration, the list of instances is scanned to find those matching a seed, while in later iterations, instances that are induced by the third phase (BoCW Matching) are used to find matching NEs instances.

Table 2: Sample of NE instances and their corresponding Bag-of-Context-Words

Class	NE Instance	BoCW
Location	barcelona	(guide,4), (information,3), (city,3), (travel,3), (catalonia,2), (featuring,2), (attractions,2), (airport,2), (maps,1), (source,1), (environment,1), (priorities,1), (situated,1), (tourism,1), (activities,1), (do,1), (kilometres,1), (conservation,1), (flights,1), (time,1), (hotels,1), (spain,1), (capital,1),...
Film	nanny mcphee	(movie,3), (trailers,2), (returns,2), (critic,1), (starring,1), (widescreen,1), (amazon,1), (tomatoes,1), (added,1), (check,1), (reviews,1), (film,1), (bang,1), (role,1), (thompson,1), (fantasy,1), (wanted,1), (appears,1), (clips,1), (pictures,1),(queue,1), (edition,1)...
Organisation	acxiom	(company,2), (corporation,2), (learn,2), (services,2), (culture,1), (industry,1), (stimulating,1), (coverage,1), (screening,1), (business,1), (agency,1), (data,1), (marketing,1), (employment,1), (information,1)...

Class Vector Aggregation: For every labelled instance in the previous phase, the corresponding BoCW is extracted and used to create a *Class Vector*. A Class Vector is the aggregation of BoCWs of all NE instances labelled in previous iterations. Each component in the class vector corresponds to a context word together with a weight, which is set to the *instance frequency* (if) of that word. The instance frequency is defined as the number of instances labelled in previous iterations sharing that context word. In every iteration, the Class Vector is updated by aggregating the BoCW of newly labelled instances.

BoCW Matching: The BoCWs of unlabelled instances are scanned to find those whose similarity to the class vector is greater than or equal to a pre-defined threshold. All the NE instances whose BoCW similarity score passes the threshold proceed to the Instance Matching phase to be labelled with the target class.

We refer to each iteration over these phases as a *Classification Round*. The algorithm repeats these phases until no new instances are labelled or a maximum number of classification rounds is reached. The similarity between each BoCW and a class vector is measured using *cosine similarity*, defined as follows:

$$CosineSim(I, C) = \frac{V(I) \cdot V(C)}{\parallel V(I) \parallel \parallel V(C) \parallel}.$$

Here, $V(I)$ is the BoCW of an instance with each component being a context word, and the corresponding *term frequency* (tf) of that term in the snippets set, and $V(C)$ is the Class Vector with each component being the *instance frequency* (if) of a context word, that is the frequency of previously labelled instances sharing that word.

Instance Frequency or Aggregated Term Frequency. In the Class Vector aggregation phase, through preliminary experiments, we found that defining the weights of words in the Class Vector as an instance frequency rather than the aggregation of the term frequencies of the words in the BoCWs of previously labelled instances leads to accurate expansion, due to the following reasoning. In the case where a context word appears with a high term frequency but very few instances share that word, there is not sufficient evidence that the NE belongs to the target class. However, the NE is more likely to belong to the target class when the context word is shared by many NE instances. In other words, if a context word has low tf in the BoCW of the instance, but it is shared by many instances i.e. has high if in the Class Vector, then it should be classified to the target class. Aggregating the tf of the labelled instances rather than using the if of the labelled instances would lead to biasing the class vector toward non-relevant context words that happen to appear with high tf in few instances' BoCWs.

6. EXPERIMENTAL SETTING

In this section, we briefly describe the approach we implemented in order to detect candidate NEs in queries. Following that, the query log and our approach to discover fertile seeds in order to initiate our seed expansion approach are presented.

6.1 Detecting Queries Candidate NEs

Prior to classifying NEs, we need to detect the terms in the query that may refer to candidate named entities. We assume that a *query candidate NE* is either: (1) a sequence of proper nouns, (2) the conjunction '&' or preposition 'of' followed and preceded by proper nouns, or (3) a sequence of proper nouns that include numbers, such as 'Microsoft Office Professional 2003'. (In the English language, a proper noun is the grammatical category of words referring to unique entities, and is spelt with capital initials.)

Each query is *grammatically annotated*, where each word is assigned two features: Part-of-Speech (POS), that is the grammatical category of a word such as noun, adjective, or verb, and Orthography (ORTH), which is the feature that captures the string representation of a word, such as capital initial, all capital, or mixed letter cases. Moreover, it is possible that a single query consists of more than one named entity, such as 'mcdonalds london', and using the definition above we would detect the two distinct named entities, 'mcdonalds' and 'london', as a single NE. Therefore, to deal with such cases, each query is segmented to find the most probable segments constituting the query. In particular, *query segmentation* will help in setting the boundaries of the detected candidate NE.

The approach applied to detect the candidate NE using grammar annotation and query segmentation had been presented in previous work [1, 2]. Briefly, each word in the query is assigned the most likely grammatical annotation of the same word occurring in the snippets set related to that query. This is similar to Bendersky's approach in [3], but instead of using full web documents the set of snippets from the query search results is used. The advantage of using snippets is that they usually provide enough context from web pages from the query answer set, avoiding the cost of parsing complete web documents.

Similarly to grammar annotation, the snippets set for the query is used to find the most probable segments of the query. Many approaches to query segmentation have been proposed in literature [5, 13, 21]. The approach applied here is that each possible segment of the query is assigned a probability given its frequency in the snippets set. This is similar to the work of Brenes et al. in [5], but instead of using just the snippet set we use a *web* n-gram model to smooth the probability of the segments using an empirically set parameter. Table 3 shows examples of the extracted

Table 3: Examples of extracted candidate NEs using query grammatical annotation and segmentation. POS: (NNP: proper noun, NN: common noun, PP: Proposition). ORTH: (Cap: Capital Initial, s: small letters, AllCaps: all capitalised)

Query	Grammatical annotation	Segmentation	Candidate NE
abc dallas	$\text{abc}_{(NNP,Cap)}$ $\text{dallas}_{(NNP,Cap)}$	[abc] [dallas]	abc dallas
antique cars chevrolet	$\text{antique}_{(NN,s)}$ $\text{cars}_{(NN,s)}$ $\text{chevrolet}_{(NNP,Cap)}$	[antique cars] [chevrolet]	chevrolet
bmw owner blogspot	$\text{bmw}_{(NNP,AllCap)}$ $\text{owner}_{(NN,s)}$ $\text{blogspot}_{(NN,s)}$	[bmw owner] [blogspot]	bmw
dale earnhardt jr girlfriend	$\text{dale}_{(NNP,Cap)}$ $\text{earnhardt}_{(NNP,Cap)}$ $\text{jr}_{(NNP,Cap)}$ $\text{girlfriend}_{(NN,(s)}$	[dale earnhardt jr] [girlfriend]	dale earnhardt jr
allegheny college of pennsylvania	$\text{allegheny}_{(NNP,Cap)}$ $\text{college}_{(NNP,Cap)}$ $\text{of}_{(PP,s)}$ $\text{pennsylvania}_{(NNP,Cap)}$	[allegheny college] [of] [Pennsylvania]	allegheny college pennsylvania

candidate NEs from queries that have been grammatically annotated and segmented. The accuracy achieved using our method is 81% ignoring the accuracy of NE boundaries, and 68% if accurate NE boundaries are required [1, 2].

6.2 Query Log

The candidate NE method was implemented using a 2006 MSN query log consisting of approximately 5.5 million unique queries. The spelling of each query was checked and corrected using Yahoo API provided through the Spelling Suggestion YQL table (this service has been deprecated). Using Google Custom Search API, the top eight snippets, that is the maximum number of results that can be retrieved per request, were collected for each query. Applying the query candidate NE extraction approach resulted in the extraction of 6,299,933 candidate named entities from 4,624,501 queries. Approximately 82% of queries contain NEs. The evaluation and results analysis of that approach are discussed in detail in [1, 2]. Out of the total number of extracted candidate NEs, 1,925,074 are unique NEs.

Each unique candidate NE was checked with DBpedia (`http://dbpedia.org/`) and only those instances whose string exactly matches a DBpedia result and is classified by the DBpedia ontology are used to create the *gold standard lists*. In addition, to avoid biased evaluation, we stipulated that each candidate NE should not be ambiguous, i.e. each candidate NE in the gold standard list should belong only to one class in the DBpedia ontology. Furthermore, for comparative evaluation purposes, we only used instances that appeared in both FQ and VFQs, resulting in 27,505 candidate NEs appearing in 295,025 VFQs. Therefore, for each extracted candidate NE, the following BoCWs were created:

SBoCW, created from the snippets set related to the FQ of that NE.

QBoCW, created from the remainder of the VFQs after excluding the NE string.

S&Q-BoCW, consisting of the combination of query context words extracted from the VFQs as well as context words extracted from the snippets set of the FQ.

The BoCW for each NE instance is extracted from the snippets by setting the context window size to be up to three words surrounding a NE occurrence on each side within the sentence boundaries. We divided the DBpedia validated instances into two sets selected randomly: a training data set and a test data set. Since a seed expansion approach was to be tested, the training dataset and the test dataset were of the same size, that is 50% of instances each.

6.3 Target Named Entity Classes

The classes used to evaluate our approach are 'Person' (representing 27.26% of the candidate NEs) including instances whose DBpedia ontology labels are Artist, Actor, Person, Athlete, and Journalist, and the class 'Location' (19.90%), including Administrative Region, City, Place, Village, and Location. Furthermore, to evaluate the classification method using more specific NE classes, we also used the classes 'Organisation' (12.80%), 'Educational Institution' (6.86%), 'Film' (2.01%), and 'Software' (1.24%).

6.4 Fertile Seeds Detection

Vyas et al. [20] reported on three main factors that effect the performance of seed expansion approaches: (1) Ambiguity of the seeds; (2) Prototypicality of seeds selection, where human editors select typical seed instances of a class that are not necessarily fertile and may introduce noise in the expansion; and (3) Coverage of the seeds set chosen.

Considering these factors, and using the training data set for each query type, our proposed fertile seeds discovery experiment is carried out as follows:

1. We run the algorithm for every instance i in the gold list, and the corresponding performance using that instance as a seed is measured by

$$Precision(i) = tp_i/(tp_i + fp_i)$$

$$Recall(i) = tp_i/(tp_i + fn_i)$$

2. All instances whose $tp = 0$ are excluded and the list of instances are sorted by precision in descending order.

3. We run the algorithm again using each instance in the sorted list as a seed, and the accumulative precision and recall are recorded as follows:

$$Precision_{acc}(i) = (TP_{acc}(i))/(TP_{acc}(i) + FP_{acc}(i))$$

$$Recall_{acc}(i) = (TP_{acc}(i))/(TP_{acc}(i) + FN_{acc}(i))$$

Here, $TP_{acc}(i) = \Sigma_{j=1}^{i} tp_j$, $FP_{acc}(i) = \Sigma_{j=1}^{i} fp_j$, and $FN_{acc}(i)\Sigma_{j=1}^{i} fn_j$. The tp, fp, and fn are reported over unique instances, since there will be an overlap between the set of classified instances using each seed.

Table 4: Percentages of fertile seeds selected for each target NE class

Class	Gold list	Dataset	Seeds	%	Class	Gold list	Dataset	Seeds	%
Person	3572	SBoCW	74	2.07%	Educational	537	SBoCW	16	2.98%
		QBoCW	117	3.28%	Institution		QBoCW	7	1.30%
		Q&S-BoCW	57	1.60%			Q&S-BoCW	11	2.05%
Location	3356	SBoCW	81	2.41%	Film	381	SBoCW	4	1.05%
		QBoCW	241	7.18%			QBoCW	0	0.00%
		Q&S-BoCW	68	2.03%			Q&S-BoCW	3	0.79%
Organisation	2283	SBoCW	10	0.44%	Software	241	SBoCW	7	2.90%
		QBoCW	25	1.10%			QBoCW	10	4.15%
		Q&S-BoCW	4	0.18%			Q&S-BoCW	4	1.66%

4. For every instance i, we measure the change of accumulative precision and recall before and after using instance i as a seed.

$$\Delta Precision_{acc}(i) = Precision_{acc}(i) - Precision_{acc}(i-1)$$

$$\Delta Recall_{acc}(i) = Recall_{acc}(i) - Recall_{acc}(i-1)$$

5. Instance i is a fertile seed if it satisfies the following conditions:

   (i) $\Delta Recall_{acc}(i) > 0$ (i.e. the addition of the seed improved the overall coverage);

   (ii) $\Delta Precision_{acc}(i) \geq -0.01$ (i.e. the addition of the seed did not decrease the overall precision by more than 1%);

   (iii) $Precision(i) \geq 0.70$ (i.e. the precision of the algorithm using that seed should at least be 70%); and

   (iv) $Recall(i) \geq 0.01$.(i.e. the recall of the algorithm using that seed should at least be 1%).

Once the initial set of fertile seeds is created, we exclude any seed whose list of classified instances, resulting from the expansion of that seed, where already detected by other fertile seeds. Table 4 shows the percentage of fertile seeds detected per target NE class. However, we found that for the class Film, fertile seeds could not be detected for QBoCW, and therefore instances of that class were not classified.

The threshold for the BoCW similarity to the class vector was set to 0.4. This was based on preliminary experiments using the training dataset, where 10 threshold settings were tested from 0.1 to 0.9. We found that the algorithm has an acceptable performance when the threshold is set between 0.3 and 0.5, with a minimum precision of 70%. Therefore, we chose a common threshold of 0.4 to carry out the performance evaluation experiments that are presented in the following section.

7. EXPERIMENTAL EVALUATION

The performance achieved by our approach was measured by applying the expansion over the test dataset with the aid of the fertile seeds selected from the training dataset. As a result of the fertile seed detection experiment over the training dataset, two BoCW vectors are extracted: (i) $BoCW^{FS}$ that is the BoCW of the NE instance whose string matched a fertile seed in the training dataset, and (ii) $ClassVector^{FS}$ that is the class vector created over the training dataset by the expansion of the matched fertile seed BoCW. Accordingly, the seed expansion approach proposed is evaluated using the following variants:

SCV-A, where the $BoCW^{FS}$s of the seeds are first aggregated to a Single Class Vector, and then the algorithm proceeds.

SCV-B, where the $ClassVector^{FS}$s of the seeds are first aggregated to a Single Class Vector, and then the algorithm proceeds.

In the case where the same instance is tagged by more than one fertile seed in the training dataset, the weights of the context words of that instance will be counted only once in the aggregation. In light of this, the class vector, using this setting, can be defined as the aggregation of all unique instances' BoCWs tagged by expansion using the fertile seeds over the training dataset.

Table 5 reports the performance for each target NE class, BoCW type, and expansion setting. Using the gold standard list of each NE class, the true positives, false negatives, and false positives are automatically recorded by scanning the set of unique extractions per setting to obtain the precision and recall.

Table 5 shows that our approach achieves an average precision between 91% and 100% over NEs extracted from FQs where the SBoCW is used to represent context of NEs, as well as an average recall between 9.65% and 41%, with the exception of the class Organisation where the average precision drops to 62% with recall of 0.9%. As for the case where the BoCW was extracted from VFQs (i.e. VFQ-QBoCW), the classification average precision decreased by 4.6% to reach 94.95% over the class Location and it drops by 25% for the class Educational Institution. On the other hand, for the class Organisation, the performance of the expansion using QBoCW scored a precision of 79%, higher than SBoCW by 17%, with the lowest scored average recall of 2%. Analysing the false positives of the class Organisation, we found that most of the instances that were automatically recorded as false positives were clothing brands that can be classified either as a Company's name or a Person's name, such as 'Micheal Kors', 'Christian Louboutin', and 'Jeffrey Campbell'. For the remaining classes, the average precision was approximately 80% except for the class Film, for which no fertile seeds were detected.

Analysing the results achieved when using S&Q-BoCW, we found that the results were approximately the same after the addition of query context words to the SBoCW for each NE except for the class Film where the average precision decreased to 87.37% and the average recall increased by only 4.6%. Furthermore, we used FQ-SBoCW+VFQ-QBoCW to denote the evaluation where precision and recall were measured using the unique instances extracted by both SBoCW and QBoCW. In this setting, we found that the recall increased by only 4%, while the precision decreased by approximately 2% among all the NE classes.

Looking at the average performance over the target NE classes (see Table 5), the average precision and recall scored by the expansion over VFQ instances is outperformed by

Table 5: Expansion performance using BoCW

class	dataset	SCV-A		SCV-B		Average	
		Precision	Recall	Precision	Recall	Precision	Recall
Person	FQ-SBoCW	0.9941	0.1432	0.8616	0.1387	0.9279	0.1410
	VFQ-QBoCW	0.8472	0.0516	0.7590	0.0710	0.8031	0.0613
	S&Q-BoCW	0.9936	0.1311	0.8558	0.1305	0.9247	0.1308
	FQ-SBoCW + VFQ-QBoCW	0.9423	0.1658	0.8115	0.1917	0.8769	0.1787
Location	FQ-SBoCW	0.9958	0.4229	0.9956	0.4106	0.9957	0.4168
	VFQ-QBoCW	0.9244	0.1505	0.9747	0.0463	0.9495	0.0984
	S&Q-BoCW	0.9963	0.4064	0.9962	0.3974	0.9963	0.4019
	FQ-SBoCW + VFQ-QBoCW	0.9732	0.4803	0.9938	0.4347	0.9835	0.4575
Organisation	FQ-SBoCW	0.6216	0.0104	0.6250	0.0090	0.6233	0.0097
	VFQ-QBoCW	0.7903	0.0221	0.7966	0.0212	0.7935	0.0217
	S&Q-BoCW	0.6129	0.0086	0.6774	0.0095	0.6452	0.0090
	FQ-SBoCW + VFQ-QBoCW	0.7097	0.0298	0.7209	0.0280	0.7153	0.0289
Educational Institution	FQ-SBoCW	1.0000	0.2314	1.0000	0.1989	1.0000	0.2151
	VFQ-QBoCW	0.6000	0.0229	0.9118	0.0593	0.7559	0.0411
	S&Q-BoCW	0.9904	0.1969	0.9901	0.1912	0.9902	0.1941
	FQ-SBoCW + VFQ-QBoCW	0.9412	0.2447	0.9766	0.2390	0.9589	0.2419
Film	FQ-SBoCW	0.9091	0.0930	0.9149	0.1000	0.9120	0.0965
	VFQ-QBoCW	-	-	-	-	-	-
	S&Q-BoCW	0.8361	0.1186	0.9114	0.1674	0.8737	0.1430
	FQ-SBoCW + VFQ-QBoCW	0.9091	0.0930	0.9149	0.1000	0.9120	0.0965
Software	FQ-SBoCW	0.9886	0.3466	0.9882	0.3347	0.9884	0.3406
	VFQ-QBoCW	0.8228	0.2590	0.7848	0.2470	0.8038	0.2530
	S&Q-BoCW	0.9778	0.3506	0.9785	0.3625	0.9781	0.3566
	FQ-SBoCW + VFQ-QBoCW	0.8881	0.4741	0.8636	0.4542	0.8758	0.4641
Average	FQ-SBoCW	0.9182	0.2079	0.8976	0.1986		
	VFQ-QBoCW	0.6641	0.0843	0.7045	0.0741		
	S&Q-BoCW	0.9012	0.2020	0.9016	0.2098		
	FQ-SBoCW + VFQ-QBoCW	0.8939	0.2480	0.8802	0.2413		

the performance achieved over FQ instances. The average precision scored for FQs was between 90.82% and 98.76% using the two expansion settings with an average recall of 20% using setting SCV-A and 19% for SCV-B. On the other hand, the maximum average recall scored for VFQs was 8% using setting SCV-A and 7% using SCV-B.

Analysing the class vectors created during the expansion over FQ instances, the frequency of common context words among instances belonging to a single NE class is higher than other context words. This results in an accurate classification of NEs at the beginning of the expansion, and therefore higher precision. On the other hand, it biases the expansion towards a specific subclass in the semantic space of the target NE class. For example, analysing the class vector created by expanding the seeds of the class Person, we found that majority of NEs extracted were actors as apposed to, e.g. singers or athletes.

Regarding VFQs instances, the chance of semantic drift during the expansion is higher, thus decreasing the performance scores. Semantic drift occurs after some classification rounds when the seed expansion approach starts tagging instances that do not belong to the target NE class [7]. This is due to the noise introduced to the class vector by the context words extracted from VFQs that are not necessarily exclusive to the target NE class.

Finally, when aggregating the fertile seeds' $BoCW^{FS}$s (SCV-A) or the $ClassVector^{FS}$s (SCV-B) to initialise the class vector, it is possible that the class vector is too big to be compared to any other instances' BoCWs. This leads to tagging few instances (i.e. low recall) with precision that is greater than 95%. Therefore, as an attempt to improve the recall of the expansion to classify named entities in FQs, in the following subsection we provide experimental results showing that the recall can be increased while maintaining a precision greater than 70%.

7.1 Gradual Threshold Decrement

In this experiment we gradually decreased the threshold after each expansion is terminated. Thus, starting with the initial threshold 0.4, the expansion proceeds until there are not any instances whose BoCW's similarity to the class vector exceeds or equals 0.4. Then, we decrease the threshold by 0.01 to become 0.39, and continue the expansion process by scanning the instances' BoCWs to find those whose similarity to the class vector exceeds or equals the new threshold, and so forth. This process is repeated until the threshold value reaches 0.2. For every threshold decrement, we measured the precision and recall scored for each of the target NE classes, and for each dataset, i.e. SBoCW, QBoCW, and S&Q-BoCW. Figure 1 plots the precision and recall charts for each of the NE classes using the expansion settings SCV-A and SCV-B.

Table 6 reports the maximum recall scored before the precision drops to lower than 70%. Using the expansion setting SCV-A over SBoCW, although the precision of the classes Person and Software decreased to approximately 75%, the recall was improved with every threshold decrement and the total recall was increased by 36%, and 39%, respectively. On the other hand, using the same expansion setting, running the gradual threshold decrement over the class Location, improved the recall to reach a maximum of 67.8%, while maintaining a precision higher than 90%. When using the SCV-B expansion setting, the performance was approximately similar to that achieved using SCV-A for the classes Person and Location. However, the performance was improved using this setting for the class Software, to reach a recall of 67.7% with a precision of 82%. For the classes Educational Institution and Film, the recall was only improved by a range between 3% and 10%, while the precision decreased by not more than 13% using both expansion settings.

Analysing the performance over QBoCW, we found that

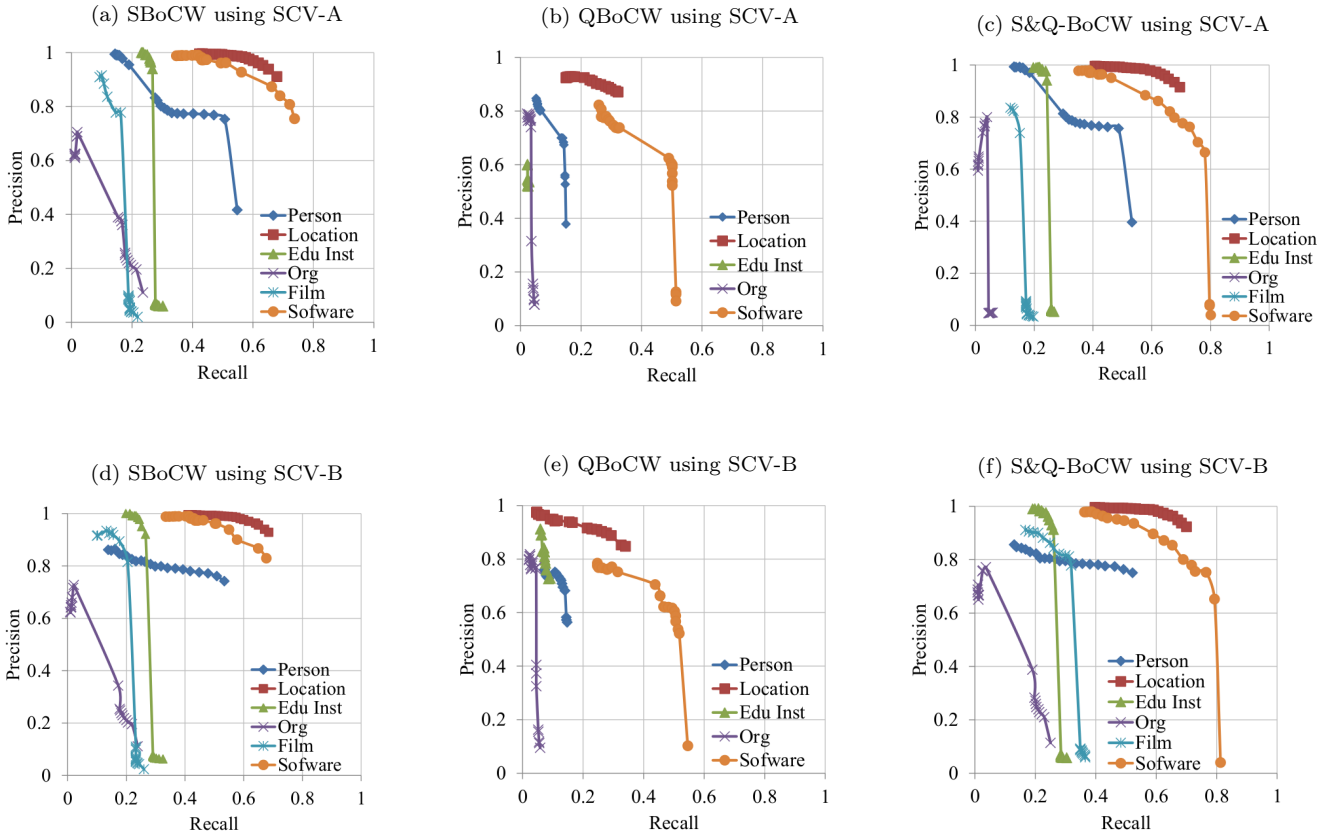


Figure 1: Decreasing threshold precision and recall charts

the recall of the expansion was only improved by a small percentage, between 1.3% and 5.9% for the classes Person, Educational Institution and Organisation using both expansion settings. On the other hand, the recall was increased from 15% to 32% (using SCV-A) and from 4% to 33% (using SCV-B) for the class Location, while the precision remained higher than 84%. In addition, the recall over the class Software was improved by 19.1%, while the precision decreased from 78% to 70% at threshold 0.32. However, when comparing SBoCW against QBoCW, we can see that the performance of QBoCW was outperformed by SBoCW in terms of precision and recall, with the exception of the class Organisation. Although the performance was lower than that achieved by QBoCW, analysing the results we found that the precision over the class Organisation using both expansion settings increased from 62% to 70% at the threshold 0.35 before it drops to 38% (see Figure 1a and 1d). The reason behind this is that as the threshold decreased it allowed for more instances belonging to this class to be tagged during the expansion which added more context words to the Class Vector that are exclusive to the Organisation class.

Finally, when using context words extracted from both the remainder of the VFQ and the snippets set of the FQ (i.e. S&Q-BoCW), the performance of the expansion with the gradual threshold decrement was approximately similar to that achieved when using SBoCW, except for the class Film. Analysing the results, we found that the recall was further improved to reach 31% using SCV-B with a precision of 77.8%.

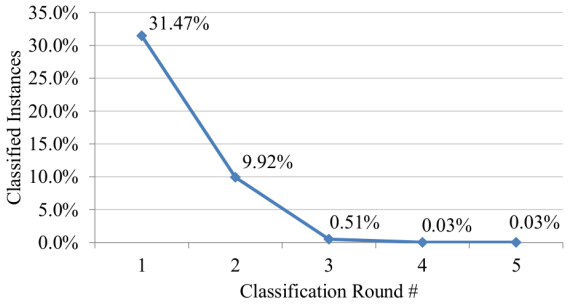
8. COMPARATIVE ASSESSMENT

As we discussed in Section 2, previous approaches to QNER used a query template to identify and classify NEs in queries. A query template is the remainder of the query after excluding a NE string. For comparative assessment, in this section we compare our QNER approach to the approach presented in [17]. In particular, this approach is similar to our approach in terms of objective (i.e. mining NEs from query log) and methodology (i.e. a seed-based expansion approach). However, our approach differs in the following aspects:

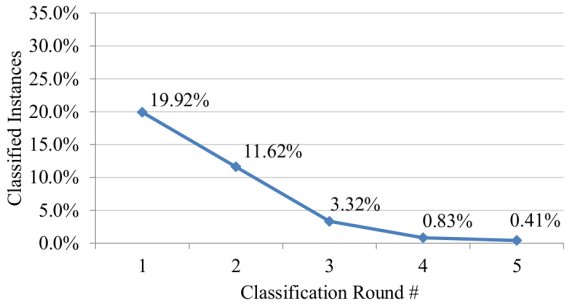
Context Definition: In our approach we define the context as a BoCW surrounding the target query NE in the snippets set related to that query. The BoCW is flexible and does not impose a specific structure for the context. On the other hand, [17] define the context as a query template that is used to find candidate NEs in the query log sharing the exact same template. Although this can be effective, it can be either too specific or ambiguous. For example, using the instance 'Celine Dion' as a seed to induce query templates for the class Singer, a query template can be '# my heart will go on lyrics', which is a very specific template to a single instance in the class Singer. In addition, other distinct templates for 'Celine Dion' would be '# biography', '# biography book', and '# biography movie' and each one of them could be used to extract new instances where it would have been sufficient to use the word 'biography'. Moreover, as we described in Section 3, a query template can be ambiguous which leads to extracting instances that do not belong to

Table 6: Maximum performance scored by gradual threshold decrement before precision drops to less the 70%

Class	Dataset	SCV-A			SCV-B		
		Threshold	Precision	Recall	Threshold	Precision	Recall
Person	SBoCW	0.21	0.7528	0.5066	0.2	0.7419	0.5340
	QBoCW	0.28	0.8034	0.0668	0.25	0.7080	0.1305
	S&Q-BoCW	0.21	0.7563	0.4875	0.2	0.7506	0.5218
Location	SBoCW	0.2	0.9113	0.6789	0.2	0.9280	0.6855
	QBoCW	0.2	0.8710	0.3226	0.2	0.8478	0.3397
	S&Q-BoCW	0.2	0.9151	0.6957	0.2	0.9235	0.7002
Organisation	SBoCW	0.33	0.7049	0.0194	0.3	0.7273	0.0217
	QBoCW	0.28	0.7404	0.0347	0.28	0.7692	0.0451
	S&Q-BoCW	0.3	0.8000	0.0397	0.3	0.7714	0.0365
Educational Institution	SBoCW	0.32	0.9396	0.2677	0.32	0.9205	0.2658
	QBoCW	0.4	0.6000	0.0229	0.2	0.7273	0.0918
	S&Q-BoCW	0.29	0.9407	0.2428	0.29	0.9128	0.2600
Film	SBoCW	0.35	0.7778	0.1628	0.33	0.8148	0.2047
	QBoCW	-	-	-	-	-	-
	S&Q-BoCW	0.37	0.7386	0.1512	0.3	0.7784	0.3186
Sofware	SBoCW	0.2	0.7551	0.7371	0.2	0.8293	0.6773
	QBoCW	0.3	0.7387	0.3267	0.32	0.7051	0.4382
	S&Q-BoCW	0.24	0.7037	0.7570	0.22	0.7529	0.7649



(a) Expansion over the class Location



(b) Expansion over the class Software

Figure 2: Percentage of instances classified per iteration during the expansion

the target class but share the same template.

Candidate NE Detection: In [17], the remainder of the query, excluding the exactly matched template, is considered to be a candidate NE. For example, if a query template was induced for the class Song, e.g. '# lyrics', in addition to the fact that it is ambiguous, based on the template all the preceding words will be considered a song name, which will lead to inaccurate extraction of the song *My Heart Will Go On* from the query 'celine dion my heart will go on lyrics'. However, in our approach, a candidate NE is any sequence of words in the query satisfying the conditions presented in Section 6 contained within the boundaries set by the query segmentation. Therefore, if the query was segmented first to the most probable segments, that is '[celine dion] [my heart will go on] [lyrics]', this will give an indication that the query consists of two segments preceding the template, and POS tagging would lead to the conclusion that the query consists of two NEs.

Candidate NE Classification: In [17], the approach starts with five seeds per target NE class, and then the query log is scanned to find all the query templates in which the seeds instances appear. After that, the query log is scanned again to extract a pool of candidate NEs. A candidate NE is the remainder of a query matching one of the seeds' query templates. The set of all the query templates associated with the seeds are used to create a *reference-search signature vector* per class. Then, in order to classify the NEs, a vector is created per candidate NE whose components are the set of all the query templates found in the query log associated with that candidate NE. Finally, each candidate NE is assigned a score based on the Jensen-Shannon divergence (JSD) between its vector and the reference search signature vector of the target class. Although our approach utilises a single *Class Vector* per NE class, instead of being static like the *reference-search signature vector* [17], the Class Vector is iteratively expanded as new instances are tagged through the expansion. Classifying NEs over the expansion, i.e. over more than one classification round, can lead to classifying additional instances that would not necessarily be classified in the first pass over the query log. For example, analysing the expansion SCV-A for the classes Location and Software over the FQ instances, we found that there were instances classified in the second and subsequent classification rounds, in addition to those classified in the first round, with a precision of 99% for the class Location and 98% for the class Software (see Figure 2).

In the following subsections, we will compare our approach to [17] using the following evaluation methods:

1. Expansion evaluation using precision and recall.
2. Evaluation of the usefulness of the vector created at the end of the expansion, for the target NE class, for classifying NEs in the query log using precision@K.

8.1 Comparison of Precision and Recall

In order to estimate the precision and recall for the approach of [17], we used the same dataset described in Sub-

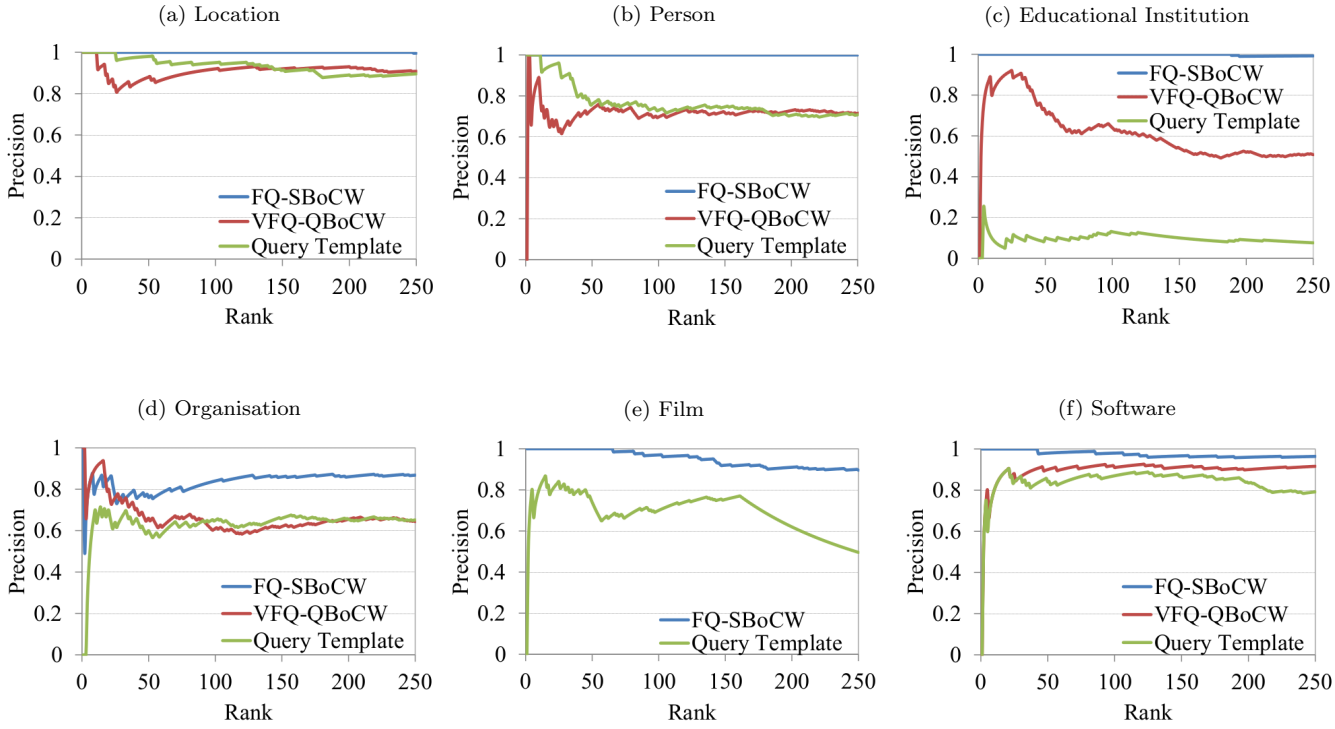


Figure 3: Performance of FQ-SBoCW versus VFQ-QBoCW and Query template approaches

section 6.2. Thereby, the set of all the VFQs in which the extracted NEs occurred were used as they appear in the query log. Furthermore, we created the reference-search signature vector using the training dataset and we evaluated the performance over the test dataset. The approach was initiated using two sets of seeds: (1) Ten randomly chosen seeds, where we ran the approach using five sets of randomly chosen seeds and we report the highest performance scored; and (2) The set of fertile seeds discovered for QBoCWs (see Table 4).

Table 7 reports on the performance scored using this approach when running over our dataset. When measuring the performance of this approach, each extraction was checked automatically against the gold standard list of the corresponding NE class to find an exact instance matching the extracted NE. However, the following problems emerged during the evaluation:

1. The query may have been misspelt, and thereby the extracted NE was not matching a gold standard list instance.
2. The boundaries of the extracted NE may have been incorrect, for example extracting 'celine dion god bless' as a Person NE. Furthermore, the same NE occurs in more than one distinct NE extractions, e.g. 'celine dion god bless' and 'celine dion'.

Therefore, the precision and recall scores were very low when compared to our approach to QNER. In the following subsection we further compare our approach that uses BoCW to the query template approach presented in [17] using the measure precision@K.

8.2 Comparison of Precision@K

Here, we evaluate the quality of the created class vector

Table 7: Performance scored by Paşca's approach [17]

Class	Randomly chosen Seeds		VFQ fertile seeds	
	Precision	Recall	Precision	Recall
Person	0.4444	0.0169	0.54815	0.18776
Location	0.5803	0.3073	0.72774	0.17182
Edu Inst	0.0853	0.1128	0.08611	0.08413
Organisation	0.3538	0.0207	0.42185	0.11322
Film	0.0822	0.0140	-	-
Software	0.2000	0.0040	0.11609	0.45418
Average	0.2910	0.0793	0.3800	0.2022

to classify NEs extracted from the query log using Precision@K. For each NE extracted from the log, two BoCW vectors are created: (1) SBoCW: consisting of all the context words surrounding the NE in the snippets set related to that query. (2) QBoCW: consisting of all the context words found in the query surrounding the NE string. After that, using the training dataset, two class vectors are created: $ClassVector^{SBoCW}$, that is the aggregation of the $ClassVector^{FS}$s of the top 10 fertile seeds created by expansion over the FQ training dataset; and $ClassVector^{QBoCW}$, that is the aggregation of the $ClassVector^{FS}$s of the top 10 fertile seeds created by expansion over the VFQ training dataset. After that, we measure the similarity of each named entity SBoCW to the $ClassVector^{SBoCW}$ and each of the named entity QBoCW to the $ClassVector^{QBoCW}$. Then, we order the NEs by decreasing similarity scores and the top 250 are checked manually as correct or incorrect to measure precision@K. Furthermore, using the query templates extracted from the seeds in the experiment described in Subsection 8.1, we extracted all the NEs from the full query log. After that, we ordered these NEs based on the JSD of their templates vector, extracted from the full log,

from the target NE class *reference-search signature vector*. Similarly, the top 250 NEs are checked manually to measure precision@K. Figure 3 plots the performance scored by each of these approaches. FQ-SBoCW is used to refer to the performance scored when we ranked instances by the similarity of their SBoCW to $ClassVector^{SBoCW}$, and VFQ-QBoCW is used to refer to the performance scored when we ranked instances by the similarity of their QBoCW to $ClassVector^{QBoCW}$. Moreover, we use Query Template to refer to the precision@K scored when implementing the approach of [17].

We see that the performance of FQ-SBoCW is better than the VFQ-QBoCW and Query Template approaches for most of the NE classes with the exception of the class Organisation, where it was outperformed by VFQ-QBoCW between ranks 1 and 33; at higher ranks the precision of FQ-SBoCW increases above VFQ-QBoCW to reach 86%.

Finally, analysing the results using Eberhardt and Fligner's test for equality of two proportions [9], we found the difference between the average-class precision@K of the FQ-SBoCW and the other two approaches (QBoCW and Query Template) is statistically significant at the alpha level 0.01.

9. CONCLUSIONS

We have presented a weakly supervised seed expansion approach to classify named entities appearing in search engine queries. The approach we have presented uses the top-n snippets related to the query to extract a Bag-of-Context-Words that is used to classify NEs in an accurate and more flexible manner than in previous approaches. Our algorithm addresses the problem of classifying focused queries (FQs), which comprise approximately 27% of the query log, that may not have any query context derived from other queries, whereby approaches such as [17] and [12] are not applicable. We have demonstrated our approach on six classes, including the most common classes Person and Location.

Following our results, which clearly show that the classification of FQs is by far the most accurate, we are currently working on optimising the approach to accurately classify NEs extracted from very focused (VFQ) and unfocused queries (UFQ).

10. REFERENCES

[1] A. Alasiry, M. Levene, and A. Poulovassilis. Detecting candidate named entities in search queries. In *Proc. of SIGIR*, pages 1049–1050, 2012.

[2] A. Alasiry, M. Levene, and A. Poulovassilis. Extraction and evaluation of candidate named entities in search engine queries. In *Proc. of WISE*, pages 483–496, 2012.

[3] M. Bendersky, W. B. Croft, and D. A. Smith. Joint annotation of search queries. In *Proc. of ACL*, pages 102–111, 2011.

[4] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 2009.

[5] D. Brenes, D. Gayo-Avello, and R. Garcia. On the fly query segmentation using snippets. In *Proc. of CERI*, 2010.

[6] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999.

[7] J. R. Curran, T. Murphy, and B. Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *Proc. of the PACLING*, 2007.

[8] J. Du, Z. Zhang, J. Yan, Y. Cui, and Z. Chen. Using search session context for named entity recognition in query. In *Proc. of SIGIR*, pages 765–766, 2010.

[9] K. R. Eberhardt and M. A. Fligner. A comparison of two tests for equality of two proportions. *The American Statistician*, 31, 1977.

[10] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 2008.

[11] R. Gaizauskas, T. Wakao, K. Humphreys, H. Cunningham, and Y. Wilks. University of Sheffield: Description of the LaSIE system as year = 1995. In *Proc. of MUC-6*.

[12] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *Proc. of SIGIR*, pages 267–274, 2009.

[13] M. Hagen, M. Potthast, B. Stein, and C. Bräutigam. Query segmentation revisited. In *Proc. of WWW*, pages 97–106, 2011.

[14] A. Jain and M. Pennacchiotti. Domain-independent entity extraction from web search query logs. In *Proc. of WWW*, pages 63–64, 2011.

[15] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proc. of Natural Language Learning at HLT-NAACL 2003*, 2003.

[16] M. Paşca. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *Proc. of WWW*, pages 101–110, 2007.

[17] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *Proc. of CIKM*, pages 683–690, 2007.

[18] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proc. of AAAI*, 2006.

[19] E. Riloff and J. Shoen. Automatically acquiring conceptual patterns without an annotated corpus. In *Proc. the Third Workshop on Very Large Corpora*, pages 148–161, 1995.

[20] V. Vyas, P. Pantel, and E. Crestan. Helping editors choose better seed sets for entity set expansion. In *Proc. of CIKM*, pages 683–690, 2009.

[21] X. Yu and H. Shi. Query segmentation using conditional random fields. In *Proc. of the First International Workshop on Keyword Search on Structured Data*, 2009.