Named Entity Recognition and Classification in Search Queries

Areej Mohammed Alasiry

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Computer Science and Information Systems

Birkbeck, University of London

United Kingdom

April 2015

Declaration

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

Areej Alasiry ______________________.

Abstract

Named Entity Recognition and Classification is the task of extracting from text, instances of different entity classes such as person, location, or company. This task has recently been applied to web search queries in order to better understand their semantics, where a search query consists of linguistic units that users submit to a search engine to convey their search need. Discovering and analysing the linguistic units comprising a search query enables search engines to reveal and meet users' search intents. As a result, recent research has concentrated on analysing the constituent units comprising search queries. However, since search queries are short, unstructured, and ambiguous, an approach to detect and classify named entities is presented in this thesis, in which queries are augmented with the text snippets of search results for search queries.

The thesis makes the following contributions:

1. A novel method for detecting candidate named entities in search queries, which utilises both query grammatical annotation and query segmentation.

2. A novel method to classify the detected candidate entities into a set of target entity classes, by using a seed expansion approach; the method presented exploits the representation of the sets of contextual clues surrounding the entities in the snippets as vectors in a common vector space.

3. An exploratory analysis of three main categories of search refiners: nouns, verbs, and adjectives, that users often incorporate in entity-centric queries in order to further refine the entity-related search results.

4. A taxonomy of named entities derived from a search engine query log.

By using a large commercial query log, experimental evidence is provided that the work presented herein is competitive with the existing research in the field of entity recognition and classification in search queries.

Dedication

To my wonderful mother, Aisha.

To the memory of my father, Mohammed.

Acknowledgements

There are no words that describe how grateful I am to the people who supported me during this journey.

My thanks to ...

My mother, your strength has always inspired me and I am eternally grateful for everything you did for me.
My sisters, Fatima, Salha, Sharifa, and Abeer, for your constant support and encouragement.
My brothers, Abdullah, Faisal, and Abdulaziz, for your words of wisdom.

My friends, Hana, Maitrei, and Manuella, through good times and bad times, you were there for me. Farida, Lama, and Shroug, thank you for being there whenever I needed a break.

King Abdullah bin Abdulaziz Al Saud (RIP), for giving me and thousands of other Saudi students the opportunity to continue our higher degree education abroad.

Prof. Poulovassilis, for your support and valuable feedback during my PhD.

Finally, my academic supervisor during MSc and PhD, Prof. Levene. Thank you for your support and guidance over the past years. I learned a lot from you and I will always be grateful to you.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Web: From a World of '*Strings*' to a World of '*Things*'

When the World Wide Web was first invented by Tim Berners-Lee in 1990, it provided a means for a digital information exchange among humans. At the time, the main problem was how to index and search the various sources of information and present the users with the most related resources that may satisfy their information needs. As a response, search engines were introduced and with the rapid improvement of their performance, they quickly evolved to become one of the first access points for the Web to discover and search information. In order to meet users' expectations to answer their search needs, search engines exploited different techniques such as page ranking via link analysis and phrase matching. Nevertheless, the Web has consistently grown since its beginning with approximately 4.48 billion pages [1], where information on nearly everything that humans could imagine would exist on the Web. Because of this high volume of information, new techniques are needed to harness and make sense of it. Berners-Lee introduced his vision towards a semantic web in 1999 [16], where data would be linked and made machine-readable via meta-data information. As a response, search engines today aim towards a better understanding of the Web's content on one hand, and understand the query keywords that users submit to search this content on the other.

One of the techniques through which search engines approach the vision of the semantic web is *Named Entity Recognition* (NER), which is the task of detecting and classifying entities mentioned in text into categories, such as, person, location, and organisation [56]. Through NER, search engines may make sense of the Web's contents as interconnected entities instead of strings of information to be retrieved for every keyword match in a search query. Over the past ten years, major search engines, including Google, Bing, and Yahoo! were granted patents

[1] http://www.worldwidewebsize.com/

where named entities played the key role in search algorithms. In 2005, Google was granted one of the first patents, where NER was introduced as a technique for query re-writing [106]. In 2007, Microsoft was granted a patent for augmenting 'blocks', i.e., passages of web pages with linguistic features that include mentions of named entities [71]. Following in 2009, Yahoo! was granted a patent for utilising named entities for search query disambiguation [76]. With Google's acquisition of the Metaweb in 2010, which was operating the Freebase directory, they released the Google Knowledge Graph in 2012. One of their objectives was to provide direct answers to users' search queries without the need for further navigation to other resources to satisfy their information need. More recently, in 2013, Microsoft was granted the patent for an 'entity-based search system' that is described to be able to detect named entities on the Web and organise search results related to a search query accordingly [64].

Given how search engines may leverage named entities, research efforts were directed toward mining named entities with the intention to create a comprehensive resource of human knowledge within the Web. For example, Google Knowledge Graph contains more than 500 million entities and more than 3.5 billion facts and relationships among these entities [2]. Although different techniques have been presented to mine named entities from the Web, recent studies have shown [8] that even a search log is rich with named entities. A search log is a repository maintained by search engines to record users' search activities along with their submitted search queries. The advantage of mining named entities from search logs is that they contain mentions of the named entities as expressed from users' perspectives, and the additional keywords they are associated with, to satisfy user's information need. Nevertheless, named entity recognition and classification is a challenging task given the linguistic nature of search queries.

In this thesis, a novel approach for named entity recognition and classification in search queries is presented. By utilising query enrichment, the approach attempts to address the challenges imposed by the specific structure of English language search queries. The following section formally defines the problem of NER in search queries and the techniques presented herein to address this problem.

1.2 Problem Definition and Contributions

Previous work in query NER [44, 49, 80], utilised solely the search query string to identify named entities, which relies either on the surface form with which the keyword is spelled or on

[2]http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html

the surrounding keywords in queries shared by named entities. For example, given the query 'Celine Dion bio', if the user submits the query following the English language spelling rules, such as capitalising proper nouns, then 'Celine Dion' may be detected as a named entity. On the other hand, 'bio' can be used as a contextual clue to identify other named entities of the class Person in search queries whose last keyword is 'bio'. However, users typically express their information need through only a few keywords, so queries often do not have enough context to identify or classify a named entity. Furthermore, the form through which query keywords are spelled, such as capitalisation, cannot be used reliably because users often do not follow the orthographic rules of spelling when submitting their search queries. In addition, search queries are highly ambiguous as are the contextual clues within these queries, and resolving this ambiguity is not trivial. Finally, the Web is consistently changing as are search queries, so relying on supervised NER models tailored for specific search queries may also not be sufficient. The challenges imposed by the nature of search queries are discussed in greater detail in Chapter 2 by reflecting on previous approaches to understanding queries in general as well as to recognising and classifying named entities in search queries.

Although previous research efforts with NER in search queries reported high performance scores [44, 49, 80], there remains a problem when the query consists of nothing other than the named entity string and the form of its keywords does not provide clues as to whether the search keywords are part of a named entity.

The research within the context of NER in search queries raises the following questions:

- How to enrich search queries in order to overcome their brevity and lack of grammatical structure in order to identify and classify named entities?
- In the case where a search query consists of more than one named entity, how to identify the boundaries of these named entities?
- Given that the search keywords which may refer to a named entity are identified, is it possible to classify them to the target entity classes using contextual features that are domain-independent, grammar-independent, and adaptable to the dynamic nature of the Web and search queries?
- Previous approaches have utilised the remainder of the query after excluding the named entity string as a contextual clue of its class. However, what is the percentage of search queries that contain no contextual clues of the named entity they contain; how often do users further specify their entity-centric queries by adding additional keywords related to

the named entity; and what is the proportion of search queries that include more than one named entity?

- Given that a named entity in a search query is identified and classified to a target entity class, what are the most common linguistic components that users associate with that class of named entities; how can these components be leveraged to infer a user's search intent?

- Given that the class to which a named entity appearing in a query is identified, how can the search intent behind such a query can be identified?

By addressing the above questions, the thesis makes the following contributions:

Enrichment of queries with snippets for NER: Each query is enriched with the snippets of the top search results related to the query. A snippet of the web page is the deciding factor as to whether the page is related to a user's information need underlying the submitted query. Therefore, with only a few sentences, a snippet should tell the user what the page is about and provide context within which the search keywords occur. Due to their impact on a user's search experience, search engines generate the snippets based on scoring the sentences in the originating web page against the query [58, 99]. Therefore, they often provide enough contextual clues to identify and classify named entities. Enriching queries with search results derived from the Web also makes the approach adaptable to the dynamic nature of search queries. Although query enrichment with search results has been used in other techniques of query understanding (see Chapter 2), it has not been used before for the task of named entity recognition and classification.

Grammatical annotation and probabilistic segmentation for candidate named entity detection: Each query is processed to identify candidate named entities, i.e., any sequence of keywords which may refer to a proper name. In order to achieve this, each keyword in the query is annotated with the most frequent grammatical features given to the same word in the snippets. Furthermore, in order to set the boundaries of named entities, query segmentation is employed to find the most probable group of keywords referring to a single named entity. Last, using a small set of handcrafted rules, which exploits the grammatical features as well as the boundaries set by query segmentation, candidate named entities are identified.

Bag-of-Context-Words (BoCW) expansion for named entity classification: This approach exploits the representation of the sets of contextual clues surrounding the entities in the snippets as BoCW vectors in a common vector space. For every target entity class, a single BoCW (referred to as a *Class Vector*) is iteratively expanded as new named entities from queries are tagged with the target class. New named entities can only be tagged when they share enough context words with the Class Vector. Using a Bag-of-Words, rather than an exact query context (i.e., the remainder of a query after excluding the target named entity string) is much more flexible, since it does not impose a specific sequence or number of words to be matched in order to find new named entities and, accordingly, weight can be attached to each context word. Furthermore, using snippets rather than just the query string resolves the problem of query named entity detection when the query consists of just the named entity string without any contextual clues that could be used to reveal its class.

Exploratory analysis of the linguistic components comprising an entity-centric search query: Using the grammatically annotated search queries and the identified named entity, an exploratory analysis is conducted to analyse the linguistic components contained in the query. The analysis focuses on three main categories of search refiners — nouns, verbs, and adjectives — which users often incorporate in their entity-centric queries, in order to further refine search results related to the named entity.

Defining search intent via detected named entities: A taxonomy of named entities derived from a query log is presented where each entity is defined by reflecting on the possible search intents that each class of entities may reveal about the query.

Each of these contributions is discussed in greater details within the following chapters of this thesis. On overview of the thesis structure is described in the following section.

1.3 Thesis Overview

The thesis is organised as follows: Chapter 2 presents an overview of related work, highlighting four main tasks that were introduced by researchers as an attempt to better understand search queries. Chapter 3 presents the NER model for mining named entities in search queries by first detecting candidate named entities. In Chapter 4, the classification of candidate named entities through BoCW expansion is discussed. Following this, an analysis of refining keywords that users associate with named entities in search queries is presented in Chapter 5. A taxonomy of named entities derived from a search query log is defined in Chapter 6. Last, conclusions drawn from the work along with directions for future work are presented in Chapter 7.

Chapter 2

Related Work

2.1 Overview

In order to reveal the intent behind search queries and meet users' expectations, one of the objectives of search engines is to understand search queries. This chapter depicts the research efforts made towards that objective, from search intent discovery to named entity detection and classification in search queries. First, a background review of search queries and search log analysis is presented in Section 2.2. Section 2.3 presents the first research attempts toward search intent understanding and discovery. Following that, rather than directly inferring the search intent behind a query, the research shifted toward techniques that analyse and identify the linguistic composites that constitute search queries via the following four tasks of query understanding: (i) query topical classification, which is briefly reviewed in Section 2.4, since it is one of the first attempts to semantically label search queries, (ii) query grammatical annotation, discussed in Section 2.5, (iii) query segmentation, described in Section 2.6, and (iv) named entity recognition and classification in search queries, reviewed in Section 2.7. Finally, Section 2.8 presents a discussion of the main observations drawn from previous work that motivated the named entity mining approach proposed in this thesis for search queries.

2.2 Background Review of Search Queries

A *search query* is the set of keywords that a user submits to a search engine in order to satisfy a search need. Shneiderman et al. [94] defined information need as "the perceived need for information that leads to someone using an information retrieval system in the first place". When a user submits a query to a search engine, the web pages related to the query are presented in the form of snippets. Each *snippet* consists of the web page title, summary of the web page with a few sentences selected from the page that highlights the query keywords, and the page URL. Within a *search session*, a user reviews the results, and either navigates to the target web page, checks more than one web page, or submits another search query. The set of search queries within a session can stem from the same information need, and each is rephrased by the user to refine the search results. Alternatively, the search need is shifted, and thus the user submits a different search query within a session.

Search engines record user search activities within a search session in a *search log* [50]. In general, three types of information are recorded: (i) user information, e.g., IP address, (ii) query information such as ID, query string, and timestamp, and (iii) search session information such as the clicked URL, timestamp, and rank of the clicked URL.

Search query logs are valuable resources for search engines, because they provide important insights into users' interactions with the retrieval system, and thus can be exploited to improve their search experience. One of the earliest analytical studies of search query logs was conducted by Jansen et al. [52] in 1998 with 51,473 queries posted by 18,113 users to the Excite web service. In 1999, Silverstein et al. [95] analysed a larger-scale query log, which consisted of one billion queries. The advantage of using a large-scale query log is that it ensures a diversity of search queries that cannot be affected by a specific time period or trending topics. Some followup efforts targeted search log analysis in terms of query length, query duplication, and sessions, such as the work of Spink et al. [97, 96], Baeza-Yates et al. [3], and Bendersky and Croft [11]. A summary of the main findings is now presented.

The frequency distribution of queries and query keywords follows a power law distribution as the search log analysis conducted by [3] reveals. A power law distribution implies that there are few highly frequent search queries and keywords, whereas a large percentage of queries submitted to the search engine are uncommon and appear in the query log with low frequencies; for example, in [3], 50% of queries are unique, and similarly, 63% of the queries occur only once in the study performed by Silverstein et al. [95].

In addition, looking at the length of search queries, which is the number of keywords users submit to search engines in order to convey their search information need, the majority of search queries are extremely short. The average number of keywords range between 2.3 to 2.6 [95, 96] and Bendersky et al. [11] reported that 90% of search queries consist of, at most, four keywords. Moreover, Silverstein et al.'s analysis [95] shows that 77% of the search sessions include only one query, whereas only 4.5 % have more than three search queries.

Lastly, search engines receive a high volume of search queries; for example, worldwide, Google receives over 40,000 search queries every second, which is equivalent to 3.5 billion per day and 1.2 trillion per year[1]. Furthermore, search queries are highly diverse; Beitzel et al. reported that there is a considerable proportion of search queries categorised by topics (i.e., business, entertainment, and other) whose relative frequency fluctuates over time [9], and from 1997 to 2001, Spink et al.'s [96] analysis of search queries revealed that there is a consistent shift in the search topics. More recently, looking at the top ten trending searches submitted to Google over the years 2012 to 2014 (see Table 2.1), one can see how they vary in terms of search topics [2].

Table 2.1: Top 10 trending searches submitted to Google from 2012 to 2014

	2012	2013	2014
1	Whitney Houston	Nelson Mandela	Robin Williams
2	Gangnam Style	Paul Walker	World Cup
3	Hurricane Sandy	iPhone 5s	Ebola
4	iPad 3	Cory Monteith	Malaysia Airlines
5	Diablo 3	Harlem Shake	ALS Ice Bucket Challenge
6	Kate Middleton	Boston Marathon	Flappy Bird
7	Olympics 2012	Royal Baby	Conchita Wurst
8	Amanda Todd	Samsung Galaxy S4	ISIS
9	Michael Clarke Duncan	PlayStation 4	Frozen
10	BBB12	North Korea	Sochi Olympics

As a result, one can summarise the main challenges posed by the nature of search queries: first, search queries are extremely short, and thereby there is insufficient context to identify and meet a searcher's information need. Second, because of their brevity, search queries are highly ambiguous, e.g., the search query 'eagles' may refer to a type of birds, the American football team, or the rock band. Third, search queries are consistently evolving; therefore, predicting search trends and users' information needs is not an easy task.

Search logs are recorded at the server-side and they do not directly reveal users' information need, users' satisfaction with the search results, or the search experience itself; however, they can be used to estimate and predict users' search behaviour and search trends. Therefore, understanding search queries has became one of the main objectives in the field of search log analysis. The objective behind query understanding is to identify the search intent behind a search query, and thereby improve the ranking and presentation of relevant results [54]. The following review presents five tasks introduced in the literature as an attempt towards a better understanding of search queries:

[1]http://www.internetlivestats.com/google-search-statistics/
[2]http://www.google.co.uk/trends/

Intent Discovery: Aims to analyse and detect the search intent behind submitted search queries, as one of the earliest attempts to gain a better understanding of search queries.

Query Classification: Aims to classify search queries based on search topic, such as 'Business', 'Fashion', or 'News'.

Query Linguistic Structure: Aims to analyse and identify the grammatical features of the keywords that constitute the search query.

Query Segmentation: Given a single query, the objective is to divide it into the most probable phrases, where each phrase refers to a single concept.

Named Entity Recognition and Classification: Aim to identify named entities within the search queries, as well as classify them into target named entity classes.

Previous work is discussed below in order to address the following questions: regardless of the task objective, how did previous techniques overcome the challenges imposed by the nature of queries? Is it sufficient to utilise off-the-shelf tools that target the same task, but different textual genres, e.g., long, grammatically structured, and domain-specific text? What were the main outcomes of their experimental evaluations and how can their findings be employed to design a named entity recognition system specifically tailored for search queries?

2.3 Search Intent Discovery

One of the earliest papers to identify the search goal behind a search query was introduced by Broder in [21]. Broder stated that search intent, within the context of the Web, can be more than simply an informational need. Accordingly, Broder categorises search query goals into three types: 'Informational', 'Navigational', and 'Transactional'. Queries with navigational intent are those where the user's aim is to navigate to a specific web site or web page. Navigational queries are closely related to the type of web searches referred to as *known item searches*, where a user knows that the particular resource she or he is searching for exists in the Web [63, 79]. On the other hand, transactional intent involves 'web-mediated' activities other than simply reading, which is the case of queries with an informational search intent.

Later, Rose and Levinson derived in [91], a web search goal taxonomy that is similar to Broder's [21] at its high level categorisation, with the categories 'Informational' and 'Navigational'. However, the intent 'Transactional' is replaced with the category 'Resource'. Furthermore, the informational goal is further specified to include 'Directed' (which includes 'Closed' and ‘Open’), ‘Undirected’, ‘Advice’, ‘Locate’, and ‘List’. Similarly, ‘Resource’ is extended to ‘Download’, ‘Entertainment’, ‘Interact’, and ‘Obtain’.

In order to estimate the prevalence of each category of search intents, Broder [21] conducted a survey where users explicitly specified the need behind their search using a simple questionnaire. The implication of using a pop-up window survey is that it is highly unlikely that users will complete the questionnaire. Broder’s survey had only 10% response ratio over the course of approximately four months [21]. Alternatively, Rose and Levinson [91] utilised, in addition to the query, the click-through data associated with each query in order to infer the search intent. Using randomly selected queries, the authors labelled each query with a possible search intent.

Subsequent research presented techniques to automatically identify search goals using supervised learning techniques. Kang and Kim [57] utilised a set of heuristics to infer whether a query is navigational or informational. These heuristics include the distribution of query keywords in pages sampled from the Web, the part-of-speech of the query keywords, and the average rate with which query keywords appear in the anchor text of web pages. For instance, if the query keywords appear more often in the homepage of a web site, it is likely to be navigational, or if the query contains proper names, it is more likely to be navigational.

On the other hand, Lee et al. [62] utilised statistics from past click behaviours to infer the search goal. For example, by exploiting the click frequency distribution of a query, if the majority of the users who submitted the same query clicked a single link, the query is assumed to be navigational. In other words, the frequency distribution of clicks per query would be skewed towards the first rank of the distribution, whereas the click distribution of an informational query is almost flat because the majority of users with an informational need would click more than one result. In addition, the anchor-link distribution is utilised: when the majority of the destination links whose anchor text includes the query keywords point to the same web page, the query is assumed to be navigational; otherwise, it is assumed to be informational.

In contrast, Jansen et al. [51] used a sample of queries randomly chosen from seven search engines to manually derive attributes for each intent in the taxonomy. The attributes are based on the query string, query length, and user click behaviour. For example, if the query contains a company or person name, the query length is less than three, and the user visited the first search result, then the query is likely to be navigational. On the other hand, queries with movies, songs, or those that contain the keywords, ‘obtaining’, ‘download’, or ‘interact’ are more likely to be transactional.

To summarise, previous efforts to infer the intent behind a search query used the following features:

1. *Collection-based Statistics:* Statistics drawn from the web pages in which the query keyword occurred.

2. *Query Log-based Statistics:* Users' past behaviour, i.e., when users submit similar queries and have similar click behaviour, they might have similar search needs.

3. *Query Features:* The query string itself, e.g., query length or query keywords.

Because search intent is inherently subjective, latent, and changes constantly, a search engine's ability to identify the exact search intent is a controversial issue. Moreover, queries are extremely short, and relying only on a small number of query keywords in order to determine whether a query is informational, navigational, or transactional is often not sufficient. For example, not every query bearing the name of a company or person is navigational; for instance, in the query 'Tom Cruise age', it is likely that the user needs a direct answer to such query, rather than to navigate to an authoritative web page with the actor's profile. On the other hand, analysing the query log in order to learn from a user's past click behaviour, or the keyword distribution in the Web in order to derive some heuristics [57, 62], does not necessarily reflect the user's perception of the search, nor the information need. Consequently, as a departure from the task of revealing search intent, research has focused on labeling the query and its constituents with semantic concepts in order to gain a better understanding of search queries, as discussed in the following sections.

2.4 Topical Classification

In 2005, the ACM Conference on Knowledge and Data Discovery (KDD) held a search query categorisation competition to classify search queries into a two-level taxonomy that consisted of 67 topical categories, such as News, Business, and Entertainment [68]. The objective was to automatically classify 800,000 queries with no training data, requiring each query to be tagged with a maximum of five categories. In the competition, different approaches were presented; however, the classification techniques of Shen et al. [93] and Vogel et al. [100] achieved the first and second highest evaluation scores, respectively, among all participants. Both of the approaches relied on *query enrichment* via the Web, i.e., using web pages related to each query to predict the categories to which the query might belong.

In [93], a search result related to a query included a title, snippet, and topic label provided by the web directory to which the search query was submitted. Two types of classifiers were utilised: one set was used to map a topic label of a result to one or more of the KDD topics, and a statistical classifier was used to classify the results related to the query to the KDD taxonomy. Using both classification results, a final classification decision was made for each query. On the other hand, [100] relied solely on the categories assigned to each result related to the query to derive a classification decision by mapping the web directory categories to those of the KDD taxonomy.

In the same spirit, Broder et al. [22] classified search queries based on the results related to the query. However, a larger scale taxonomy that consisted of 6,000 topics, was used, and a document classifier was trained using 150 query examples per topic annotated by human editors. Then, a voting method was applied to assign the most probable topic to each query.

On the other hand, instead of using external resources, Beitzel et al. [10] relied on the query log to derive the topical category of a query. Using Resnik's model of *selectional preferences* [87], Beitzel et al. measured the association score between query keywords and a target topical category using a large set of manually classified queries.

Other approaches targeted specific types of queries, such as the work of Dai et al. [29], where the researchers focused on commercial and non-commercial search queries, whereas Li et al. [65] evaluated their approach specifically on queries related to products and jobs. Dai et al. relied on a manually annotated training dataset to train a classifier to identify whether a query is commercial [29]. On the other hand, [65] used an automatically generated training dataset to train a classifier. In particular, the dataset was generated using a click-graph constructed from the queries and the corresponding clicked URLs, and the technique started with a few labelled query seeds to propagate label information to unlabelled queries. In the same spirit, in order to minimise human effort to annotate search queries, Hu et al. [47] utilised Wikipedia to represent a query search domain, where each Wikipedia domain represented a category, and the corresponding articles that appeared within the domain were used to derive features of each topic.

As stated previously, because of the brevity of search queries, several approaches resorted to augmenting them with additional related information via various media. Nevertheless, in an online scenario, where a query needs to be classified when submitted by a user, these approaches are impractical given the volume of queries that search engines process every second. Therefore, approaches such as that of [10] relied on the query log to classify a search query. On the other hand, training a classifier requires a labelled dataset, which is a costly task because it requires human judgment. Moreover, given the fact that the Web content and queries are evolving constantly, relying on periodically trained models is also not sufficient.

2.5 Query Linguistics

Focusing on English-language queries, in this section, the linguistic structure of search queries is discussed from the following perspectives: (i) the orthographic features of the keywords that constitute the search query, (ii) their Part-of-Speech, and (iii) the syntax in which queries are formed. Orthography is the surface form with which a keyword is spelled, such as capitalisation, punctuation, or spelling, and Part-of-Speech is the grammatical category to which a keyword belongs, e.g., noun, verb, or adjective. On the other hand, the syntax depicts the order with which the query keywords are arranged.

Looking at the task of Part-of-Speech tagging, previous work on search log analysis recognised the advantage of annotating search queries grammatically. For example, Zukerman and Raskutti [107] utilised the query keywords' Part-of-Speech along with the keywords' corresponding entries in WordNet [40] to derive new synonyms. The resulting paraphrases of the query were exploited to improve the quality and coverage of the retrieved results.

Because of the potential advantages of identifying the linguistic composites of a search query, Barr et al. [8] presented a statistical analysis of the lexical properties of search keywords. The major findings of their work revealed that there is a fundamental difference between the distribution of the grammatical categories of English-language search queries and that of structured and published English-language corpora. In particular, the study showed that the keywords that belong to the noun category constitute 71% of the sample, out of which there were 40% proper nouns. In addition, users do not often employ function keywords such as 'from' or 'of', and often keywords are not syntactically structured. Furthermore, a majority of search query keywords did not follow the orthographic rules of capitalisation, and therefore a majority of proper nouns were not capitalised. Consequently, when running off-the-shelf Part-of-Speech taggers, namely the Brill tagger and the Stanford tagger that were trained over a structured English-language corpus, the reported performance was low.

Because of the low performance that off-the-shelf natural language processing tools scored when executed over search queries, Bendersky et al. [11] presented a probabilistic approach that exploited documents related to a query for annotation purposes (query enrichment). The "structural annotations" with which the queries are annotated include: orthography in terms of 'capital-case initial' or 'lower-case initial', and the grammatical category in terms of 'noun', 'verb', or 'other'. The probability of each query keyword's annotation given a retrieved document, or part of the document, related to the query is smoothed by the probability of the same keyword given a larger corpus. Their experiment showed that enriching a search query with the search results related to that query outperformed the structural annotation given only the query string. In a subsequent work of the authors, using the independent annotations assigned to each query keyword (orthography, and Part-of-Speech), an annotation model was trained to leverage the dependencies between keywords' annotations to improve the overall structural annotation of a query [14].

More recently, Ganchev et al. [41] presented a model for instantaneous Part-of-Speech tagging of search queries when submitted to a search engine. Similarly to [13, 14], a subset of queries were augmented with the snippets of the web pages related to the query. However, a supervised Part-of-Speech tagger trained over news articles was used to Part-of-Speech tag the snippets. Then, the Part-of-Speech labels were transferred to the query keyword to retrain the Part-of-Speech tagger. This resulted in a Part-of-Speech tagger that could be used for online annotation.

Because of the lack of context in search queries, query enrichment via search results proved to be a solution to linguistically annotate search queries and, in turn, it can be used to better understand search queries. Whereas the research into structural annotation of queries focused on the lexical and grammatical features of the individual search keywords, query segmentation groups them into semantically related concepts, as presented in the next section.

2.6 Query Segmentation

Query segmentation is the process of finding the sequence of keywords within the query that refer to a single concept. For a query of n keywords, there are 2^{n-1} possible ways to segment the query, i.e., between every two keywords, a decision for segmentation is made. For example, the query 'New York weather' can be segmented into one of the following ways, assuming the keywords are in the right order, i.e., from left to right:

- `[New York weather]`.
- `[New] [York weather]`.
- `[New York] [weather]`.

`- [New] [York] [weather].`

Retrieval systems can leverage query segmentation in order to improve retrieval accuracy, where the relatedness of a document to a query shifts from word-based (e.g., ‘new’ and ‘york’) to phrase-based (e.g., ‘new york’). Several works in the literature acknowledged the positive impact of employing keyword dependency measures to improve the accuracy of information retrieval systems [4, 42, 75].

Query segmentation approaches can generally be categorised into the following two major approaches:

To break or not to break Given a position i between two keywords, the segmentation model makes a decision as to whether to segment.

Segmentation as a whole Find the optimal segmentation by scoring the individual segments that constitute the query. Then, the overall segmentation score is measured by augmenting the segments’ scores.

One of the earliest works that targeted search query segmentation was that by Risvik et al. [89], where a measure, referred to as *connexity*, was introduced. This measure assigns a score for each segment based on its frequency within a corpus as well as the Mutual Information between the keywords that fall within that segment. Although their work was the first to introduce a measure for segmentation, the approach was not evaluated experimentally.

Another technique introduced by Jones et al. [55] utilised the Mutual Information of every two consecutive keywords. If the Mutual Information score was below a threshold, a break is made; otherwise, the two keywords are assumed to belong to the same segment. The drawback of utilising Mutual Information is that it only estimates the correlation between two consecutive keywords, rather than an entire phrase within a segment that may contain more than two keywords.

Bergsma and Wang [15] presented a supervised learning approach for query segmentation. The segmentation model utilised a set of features derived from the keywords surrounding the position in which the segmentation decision was to be made. The set of features included lexical features, such as keyword string (e.g., keyword = “the”), Part-of-Speech, break position; and corpus-based features, such as keyword raw count, and co-occurrence count of the surrounding keywords extracted from the Web or query log. Later, Bendersky et al. [12] presented a two-stage model for query segmentation. First, a query was segmented based on noun phrase detection and, in the second stage, the detected noun phrases were segmented further into finer-grained segments. A shallow syntactic parser trained over the Wall Street Journal corpus was utilised at the first stage, whereas in the second stage the parser was trained using pre-segmented noun phrases that exploited features similar to those used in [15]. As discussed previously, because search queries evolve rapidly, relying on a supervised model for query segmentation may not be practical, and it requires a large set of manually segmented queries. Therefore, subsequent research on query segmentation resorted to unsupervised approaches as presented in the following paragraphs.

Both Tan and Peng [98] and Li et al. [66] presented a probabilistic model for query segmentation. Using a language model based on concepts, rather than on words built over a web corpus, the probability of a query segmentation that consisted of n segments $s_1, s_2, ..., s_n$, was estimated by $P = \prod_{i=1}^{n} P(s_i)$. The probability of each segment s was estimated from the language model. In order to improve the segmentation accuracy, [98] assigned higher weights to those segments that appeared with higher frequency as a title of a Wikipedia article. Rather than using a web corpus to derive the probabilities of the segments, and without the need for additional sources of knowledge (e.g., Wikipedia), [66] utilised the documents clicked by the users during their search sessions when submitting their queries (i.e., click-through data) to estimate the probability of a possible query segmentation. To introduce an interpolated model, the probability of a query segmentation was estimated further by augmenting its probability given the click-though data, with the probability given a web N-gram model, namely, the Microsoft N-gram model introduced in [102].

Instead of relying on a web corpus, Brenes et al. [20] utilised the snippets of the top-n results related to a query to perform segmentation instantly; that is, when the query is submitted to the search engine, segmentation is performed. The score of each segment is estimated using word association statistical measures, such as Mutual Information, Phi, and Loglike. On the other hand, Mishra et al. [77] used query logs to find the most significant query segments used later for unsupervised query segmentation.

As a departure from complicated segmentation models that either require training or parameters estimation, Hagen et al. [45] presented a naïve query segmentation technique. The score of every segment was simply based on the raw frequency of the n-grams that constituted the segment, and the length of the segment. Thereby, the score of a segment s is given by $|s|^{|s|}.count(s)$ (in order to tip the scale of segmentation toward the longest possible match for each segment, the factor $|s|^{|s|}$ is used). Similarly to [98], Hagen et al. [46] subsequently utilised Wikipedia to normalise the weight of the segments in order to ensure that those segments that appeared as Wikipedia article titles would not be segmented.

Similarly to other tasks of query understanding, the approaches to query segmentation vary from supervised learning approaches to unsupervised approaches. On the one hand, supervised approaches require extensive training with manually labelled queries; the disadvantage is the prohibitive cost of manual labelling, and the dynamic nature of search queries that the supervised learning techniques might not adapt to. On the other hand, unsupervised approaches resort to external resources in order to derive an estimation of query segmentation, such as the Web, query log, or Wikipedia. The appealing side of unsupervised approaches utilising n-grams periodically updated from the Web is that they are more adaptable to the evolving nature of search queries.

One vital outcome of query linguistics research is that proper nouns represent a large percentage of query phrases, and therefore search query logs are rich sources of named entities. Furthermore, using Wikipedia, a source of human-built knowledge, for query segmentation improved performance scores, which implies that search queries often refer to things and concepts. Query grammatical annotation and query segmentation annotate the structure of search queries at the keyword and segment levels. On the other hand, named entity recognition in search queries assigns semantic concepts to these segments in order to perceive the intended search need behind a query. In the following section, named entity recognition and classification within the context of search queries is discussed.

2.7 Named Entity Recognition

Named Entity Recognition (NER) was first coined in 1993 as one of the subtasks for information extraction in the Sixth Message Understanding Conference (MUC-6) [43]. The goal was to develop practical and domain-independent techniques in order to detect named entities with high accuracy. The task specifically involved the automatic extraction of the names of people, organisations, and geographical locations, referred to as ENAMEX tags, in addition to numeric expressions (NUMEX), which include time, currency, and percentages. Given the target domain of research, subsequent work introduced more specific fine-grained classes of entities, such as ‘politician’, ‘actor’, ‘city’, ‘country’, ‘disease names’, and ‘drugs’.

Early research in named entity recognition and classification presented different approaches and techniques [78]; nevertheless, typically, the introduced approaches often aim at NER in properly structured news articles. Moreover, structured text often contains consistent patterns and context features that can be used. Hence, when applied within the context of the Web, NER techniques were adapted to meet the following two challenges:

- *Diversity of the text structural genre*: web documents can originate from encyclopedia entries (e.g., Wikipedia), news, blogs, or forums that can be properly structured natural language, or unstructured with missing function words, slang, and acronyms.

- *Diversity of the domain of the text*: web documents can cover different domains or topics, such as business, fashion, entertainment, and others.

Accordingly, NER models that employ hand-crafted rules such as [92], or supervised approaches that rely on a large annotated dataset, such as Decision Trees [27], Hidden Markov Models [17], or Conditional Random Fields [74], are not practical within the context of the Web. Therefore, researchers resort for semi-supervised and unsupervised techniques to NER in the Web.

Bootstrapping is one of the semi-supervised approaches to NER that inspired the work of named entity detection in the Web. Starting with a small set of seeds for each target class, the corpus is scanned to find sentences in which the seed instances occur. Then, these sentences are parsed to find contextual clues that can be used to induce new instances that share the induced contextual clues. The corpus is scanned again to find new sentences, where the newly induced instances appear. Next, the new sentences are parsed in order to derive new contextual clues. The algorithm alternates between the two tasks, extracting instances and extracting contextual clues, until a termination condition is reached.

Yarowsky [104] first used the algorithm for sense disambiguation, and then Collins and Singer [25] adapted the approach for the task of named entity recognition. Given the linguistic structure of the target text, approaches to NER utilise different techniques to induce contextual clues. For example, Collins and Singer [25] used the grammatical structure of the sentence to induce a pattern, e.g., the named entity occurs in a sentence that is a complement to a proposition. In addition, Riloff [88] utilised a variation of Bootstrapping to induce extraction patterns with the aid of some heuristics, for example "`<subj> passive-verb`", that were applied to news related to terror attacks.

Clearly, approaches such as [25, 88] aim to identify named entities in structured text. In order to address the nature of the text in the Web, Paşca et al. [84] defined the pattern to be a generalised form of a sentence via classes of distributionally similar words, which is a technique introduced by Lin for grouping similar words [69]. For example, a generalised pattern to mine Person-BornIn-Year facts is "*Prefix* :`[among CL6 ...]` *Infix* : `[CL4 born on 00 CL3]` *Suffix* : `[in CL10]`", given the following classes of distributionally similar words:

- `CL3 = {March, October, April, Mar, Aug., February, Jul, Nov., ...}`
- `CL4 = {is, was, has, does, could}`
- `CL6 = {You, Lawmakers, Everyone, Nobody, Participants, ...}`
- `CL10 = {Pennsylvania, Denver, Oxford, Marquette, Hartford, ...}`

More complex systems that integrate various models to identify named entities in the Web, as well as relationships among these named entities (i.e., information extraction), have also been presented, for example, KnowItAll by Etzioni et al. [37, 36], and the Open Information Extraction paradigm introduced by Banko et al. [5]. A general trait of these systems is that they rely on shallow linguistic patterns to induce new named entities. The heuristics exploited are domain independent and loosely structured to meet the structure of text in the Web.

As discussed previously, one of the main objectives of mining named entities from the Web is to improve Web search by identifying the facts that exist in the Web that might satisfy a user information need [84]. Consequently, identifying named entities in search queries has become one of the recent tasks for obtaining a better understanding of search. Furthermore, the linguistic analysis conducted by Barr et al. [8] revealed that there is a large percentage of proper names in queries, thus making search logs a rich source of named entities. In addition to the fact that search queries are multi-domain and often they are grammatically unstructured, they are also extremely short. Accordingly, the research presented new techniques for NER in search queries, as discussed in the following subsections.

2.7.1 Techniques and Methodologies

In [80], Paşca utilised a variation of the Bootstrapping algorithm for mining named entities from a search log. Starting with a set of named entity seeds for each target entity class, the query log is scanned to find those queries that contain one of the seeds' strings. Matched queries are processed to extract a query pattern, which is the remainder of the query after excluding the named entity string. For every target entity class, the set of all query patterns are aggregated into a single vector referred to as a reference-search-signature vector. Then, the patterns are used to find new instances that share the same pattern. For every new instance, a search-signature vector is extracted that consists of all the query patterns in which the instance appeared. The weight given to each component (i.e., query pattern) of the vector equals the frequency of that query in the query log. Finally, using the Jensen-Shannon divergence [61], the similarity between every instance's search-signature vector and the target class's reference-search-signature vector is measured to quantify its membership to the target entity class.

In contrast to the deterministic approach presented by [80], Guo et al. [44], Xu et al. [103], and Pantel et al. [83] utilised a probabilistic approach where every extracted named entity might belong to more than one class. Their approach utilises a weakly-supervised variation of the topic modeling technique Latent Dirichlet Allocation (LDA) [19]. The target entity classes correspond to the topics, and the query pattern corresponds to the words of the document in LDA. Accordingly, the problem of classifying a named entity e that appears in a query pattern t to a class c is given by:

$$Pr(e, t, c) = Pr(e)Pr(c|e)Pr(t|c).$$

In order to build and train the topic model, [44, 83, 103] employed Paşca's variation of the Bootstrapping technique used in [80] to create a training dataset of the form (e_i, t_i). The training dataset is used to learn the topic model, and from this the probabilities $Pr(c|e)$ and $Pr(t|c)$ of each seed entity were obtained. After that, the query log was scanned again, to find new named entities that shared the same context as the seeds. The probability of each new named entity $Pr(c|e)$ was estimated using the model trained over the seeds with a fixed $Pr(t|c)$. The probability $Pr(e)$ was estimated using the frequency of the queries containing the named entity in the log.

In contrast, Jain and Pennacchiotti [48, 49] presented an unsupervised approach for NER in search queries. The approach was based on the heuristic that users copy and paste their search queries when submitting them to search engines, and thereby the surface form of the keyword is preserved, i.e., the capitalisation of keywords. Accordingly, the assumption made was that a named entity is any consecutive sequence of capitalised keywords. After that, similar named entities were identified using a clustering algorithm, namely Clustering By Committee (CBC) [82].

More recently, Eiselt and Figueroa [35] presented a supervised NER approach using 80,000 manually labeled search queries to train a two-step Conditional Random Field model (CRF). In the first step (CRF-S1), the query keywords were classified as to whether they were part of a named entity, and in the second step (CRF-S2), the named entity was classified to one of 29 predefined classes.

Similar to other query understanding tasks, named entity recognition in search queries varies from supervised [35], to fully unsupervised approaches [49]. On the one hand, supervised approaches require a large dataset of manually annotated search queries, and, on the other hand, unsupervised techniques, via clustering named entities, introduce the issue of labelling clusters. However, weakly-supervised approaches, such as [80, 44], exhibit the strengths of both, where human judgment is in the form of a few seeds that represent each of the target entity classes, and then the techniques operate automatically without human interaction. Furthermore, weakly supervised techniques are most adaptable to the dynamic nature of search queries, which makes them more practical than supervised techniques.

Regardless of the technique employed, the objective behind detecting named entities in search queries was either to mine named entities and create a knowledge-base of them, such as [80, 83, 103], or train a model for online detection of named entities in search queries submitted to search engines [44, 35]. In the following subsection, NER techniques for search queries are compared based on the feature space employed for named entity detection and classification.

2.7.2 Feature space

In general, the features utilised for named entity detection and classification were either derived from the query string, or from the click-through data, as presented in the following paragraphs.

Although queries are short (on average two to three keywords), techniques such as [80, 44] exploited the keywords that constituted a search query in order to identify whether a named entity existed, as well as the class to which it belonged. As described previously, given a seed named entity that belongs to a target named entity class, the search log is scanned to find contextual clues (the remainder of the query after excluding the matched named entity string). For example, consider the seed 'Wind of Change' in the named entity class Songs. When scanning the search log, the same instance may appear in the following queries: 'Wind of Change lyrics', 'Wind of Change chords', and 'Wind of Change mp3', and therefore the contextual clues used for detecting and classifying the named entities in the class Songs are '`# lyrics`', '`# chords`', and '`# mp3`', where '`#`' denotes the position of the entity in the query. This Bootstrapping mechanism is used by Paşca [80] to harvest named entities, and it was used by Guo et al. [44] to create the training dataset. Nevertheless, the nature of search queries impose the following challenges to the task of NER.

Because of their brevity, search queries are highly ambiguous. For example, although humans often associate the keyword 'lyrics' with songs, the same contextual clue appears in the query 'Beyonce lyrics', where the preceding is a named entity, but it is not of the class

Songs, or the query 'Music and Lyrics', where the contextual clue is part of a named entity that refers to the name of a movie, as well as the query 'songs lyrics', where the preceding is not a named entity. This can also be observed among different contextual clues, e.g., '`# side effects`' used to extract drug names [80], where the same context appears in the query 'trailer of Side Effects', in which the contextual clue is a named entity that refers to a film. Table 2.2 lists some examples of query contextual clues for the classes Video Game and Book.

Table 2.2: Sample of contextual clues and corresponding named entity classes

Query pattern: '# games'		Query pattern: '# book'	
Query	Class	Query	Class
Uncharted games	Video game	*Harry Potter* book	Book
The Hunger Games	Film	*The Notebook*	Film
free games	*Not NE*	*Kelley Blue Book*	Company

Another issue that arises when using the query string to extract contextual clues of named entities is inaccurate detection of the named entity boundaries. For example, using the contextual clue '`# lyrics`' to extract a song name from the query 'James Morrison Beautiful Life lyrics' would lead to inaccurate detection of the entity boundaries.

Finally, when a query consists of nothing but the named entity string, there is no evidence of the class to which the named entity should be classified. As presented in Chapter 4 of this thesis, at least one third of search queries do not contain any contextual clues, and thereby, using the approach in [44, 80], the named entities cannot be detected or classified.

In order to further validate the classification of a named entity in a search query and its boundaries, Du et al. [33] utilised the set of queries in a search session. The approach makes use of two features: a class feature that looks into the class of the named entity that occurs in the previous query, and an overlap feature that considers the overlapping words in both the current and previous queries within the same session. The approach was evaluated using two classification techniques, topic modeling by Guo et al. in [44], and Conditional Random Fields by [60]. However, the validation approach depends on the fact that a user may submit more than one query related to a single named entity within a search session, and that is not often the case. As presented in Section 2.2, the analysis presented by Silverstein et al. [95] revealed that only 4.5 % of the search sessions have more than three search queries, whereas 77% of the sessions include only one query.

Jain and Pennacchiotti [48, 49] utilised the surface form with which search keywords are spelt as a feature to detect a named entity. Each extracted named entity is validated using two confidence scores based on the following: (1) how often a named entity appears capitalised within the Web, and (2) how often the named entity appears as a stand-alone query (i.e., with no context keywords) in the search log. The higher the scores, the more likely the detected query keywords are indeed a named entity. Although the confidence scores may validate the detected named entity, relying on capitalisation leads to missing many potential named entities submitted to search engines regardless of the orthographic rules of spelling. Similarly the detection and classification model of Eiselt and Figueroa [35] relied on features drawn solely from the query string, such as current term, previous term, following term, bi-gram, word shape, and position in the query and length. As discussed in Section 2.2, the study of search queries' linguistic structure of Barr et al. [8] showed that users often submit search queries without considering the orthographic rules of spelling or the proper syntax in which keywords are submitted.

Rather than using the query string alone, the clicked URL associated with each query in a query log was used as a feature for named entity classification. For example, [83, 103] used both query contextual clues and the host of the corresponding clicked URL to train the topic model. Considering the named entity 'Harry Potter', the extracted features can be of the form ('`# book`', '`www.amazon.com`'), or ('`# trailer`', `www.youtube.com`). Although this approach bridges the problem of missing context, the Web resources are not often exclusive for target entity classes, e.g., the host `www.imdb.com` includes both movies and actors, writers, or directors.

To summarise, there are four main problems when relying solely on a search query string to identify and classify named entities:

1. Ambiguity of the contextual keywords.
2. Inaccurate recognition of the boundaries of named entities.
3. Lack of syntactic and surface form features that can be exploited for named entity detection.
4. Lack of contextual keywords when the query consists of only the named entity string.

2.7.3 Evaluation Measures

In previous work on named entity recognition and classification in search queries researchers have used one or more of the evaluation measures listed in Table 2.3. Precision measures the percentage of correctly detected/classified named entities (i.e., true positives TP) given all those identified by the system as named entities (i.e., true and false positives $TP + FP$), whereas recall measures the percentage of correctly detected/classified named entities given the set of all target named entities in the target corpus (i.e., true positives and false negatives $TP + FN$). On the other hand, precision@K measures the precision achieved by a model over the top-K named entity extractions ordered based on an extraction-scoring schema.

Table 2.3: Evaluation measures of NER in search queries

$Precision$	$\frac{TP}{TP+FP}$
$Recall$	$\frac{TP}{TP+FN}$
$Precision@K$	$\frac{\sum_{i=1}^{K} tp_i}{K}$
F_1	$2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

For example, Paşca [80] ordered the extracted named entities based on their Jensen-Shannon divergence scores, and the top-250 were checked manually at $K = 1, ..., 250$. On the other hand, in addition to precision@K, Guo et al. [44] tested their trained topic modeling system on over 12 million queries that resulted in 400,000 extractions, out of which 400 were sampled and checked as *correct* or *incorrect* to measure the precision of the top three most probable classes assigned for each extraction. Xu et al. [103] also used precision@K; however, instead of marking partially detected named entities as either *correct* (in the case of lean evaluation) or *incorrect* (in the case of strict evaluation), the factor 0.5 was used to count partially detected named entities, and the factor 1 was used to count correctly detected named entities. Nevertheless, the work of [44, 80, 103] did not measure recall, and therefore the system's coverage for NER was not estimated.

In contrast, Jain and Pennacchiotti [48, 49] estimated their system coverage with Wikipedia; however, 61% of the named entities that were correct were not found in Wikipedia. Eiselt and Figueroa [35] exploited their manually labelled queries (consisting of 82,723 queries) to perform 10-fold cross validation, and the performance was evaluated using the measure F_1 (see Table 2.3).

The evaluation measures listed in Table 2.3 were defined in this section in terms of NER evaluation. Within the context of information retrieval, precision computes the proportion of retrieved documents that are relevant to a query, whereas recall computes the proportion of relevant documents that are retrieved to a query. Because these measures are computed using an unordered set of documents, these measures were extended and new ones were defined for evaluating a ranked retrieval system, e.g., Cumulative Gain (CG) and Discount Cumulative

Gain (DCG) [72]. Evaluation measures that compute the quality of a system's ranking are outside the scope of this thesis.

2.7.4 What is Next?

Identifying named entities in search queries allowed for other directions of research. For example, Paşca used also search logs to mine attributes of named entities in target entity classes [81]. The approach started with a small set of extraction patterns used to extract pairs of instances and their attributes from a search log, for example, the extraction pattern '`the A of I`', where A is an attribute of instance I. Subsequently, these pairs are filtered from noise and ambiguous attributes.

Moreover, with the discovery of named entities in search queries, the perspective of search intent varied. Yin and Shah [105] defined an intent phrase to be the keywords that users associate with a named entity, and the objective of their work was to create a taxonomy of intent phrases for a set of target named entity classes, such as 'song lyrics', 'released album', and 'concert' which were all identified as possible intent phrases of the class 'Singers'.

More recently, Cheung and Li [23] presented a clustering approach to find clusters of similar queries from which intent patterns are derived, for example, inducing the pattern ('`[Movie] showtime`') for the cluster of all queries that includes a movie name followed by the keyword 'showtime'. Moreover, Li et al. [67] presented an approach to identify synonymous query intent templates (i.e., intent phrases), and group them into a cluster for a set of general attribute intents. For instance, for the intent attribute '`[Person] birthday`', the query intent templates '`when [Person] is born]`' and '`[Person] birthday`' are grouped into a single cluster.

On the other hand, Lin et al. [70] presented the notion of actions on entities that is a finer specification of search intents. The objective of their work was to infer an action given the query string and the clicked URL, e.g., inferring the action 'get contact information' for the query 'Hobart corp' with the clicked URL 'hobartcorp.com/contact-Us/'.

In addition to NER over search queries, an emerging research area is the recognition of named entities in tweets. Similar to search queries, tweets are user-generated text that are short, where each may contain a maximum of 140 characters, and they may not be grammatically structured. In addition, a tweet often contains emoticons, abbreviations, and hashtags. As an attempt to address these challenges, different experimental studies were introduced in literature such as in [31, 90].

2.8 Discussion

In this chapter, an overview of the major techniques of web search query understanding was presented. The first attempt to understand search queries was to infer the search intent by categorising it into either informational, navigational, or transactional. Because search intent is latent, dynamic, and inherently subjective, researchers resorted to labeling the queries and their components with semantic concepts without the need for an exact inference. Accordingly, query topical classification, structural annotation, segmentation, and named entity recognition dominated the research trends in the last decade.

Regardless of the target task of query understanding, because of the nature of queries, i.e., brevity, ambiguity, and lack of grammatical structure, previous work exploited query enrichment. Query enrichment was attained either via related search results, related knowledge base entries (e.g., Wikipedia) or via click-through data recorded in a query log.

Given the observations collected from previous research efforts, in this thesis the named entity recognition and classification task in search queries is conducted as follows:

- In order to minimise search query ambiguity, lack of context and grammatical features, each query is enriched with the snippets of search results related to that query.

- The strengths of both query grammatical annotation and query segmentation are combined in order to identify candidate named entities, as presented in Chapter 3.

- For named entity classification, instead of relying on a general form of the context or its grammatical structure, a Bag-of-Words approach is used to infer the class to which a named entity belongs, as presented in Chapter 4.

- Since the keywords associated with a named entity in a query may reveal the intent behind an entity-centric query [105], in Chapter 5, an exploratory analysis of three types of search keywords, namely, nouns, verbs, and adjectives, is presented.

- Finally, previous efforts for NER in search queries focused on testing the proposed approaches on a set of predetermined named entity classes with no general study of the types of named entities that users would generally use in search queries. In Chapter 6, the distribution of the detected named entity classes is investigated, and a taxonomy of named entities is derived, along with the possible search intent behind each class of entity-centric queries.

Chapter 3

Query Named Entity Recognition QNER

3.1 Overview

This chapter introduces the Query Named Entity Recognition (QNER) approach, which utilises query segmentation and grammatical annotation to identify candidate named entities in search queries. The chapter is organised as follows. First, the methodology to QNER is presented in Section 3.2, where the three main phases of named entity recognition in search queries are defined: (i) *grammatical annotation*, (ii) *probabilistic segmentation*, and (iii) *candidate named entity recognition*. Following that, a description of the experimental setting used to assess the performance of QNER is presented in Section 3.3. Section 3.4 discusses the experiment results derived from the QNER evaluation at query-level, as well as those from the entity-level evaluation. A discussion of the main findings and results is presented in Section 3.5.

3.2 Methodology

A *candidate named entity* is any sequence of keywords in a query that may refer to the name of an entity regardless of the class to which it might belong. For example, the underlined keywords in the following search queries refer to candidate named entities, ‘Hillary Clinton bio’, ‘The Hobbit’, and ‘Pizza Hut London’. The approach proposed for QNER consists of the following three phases:

1. *Grammatical annotation* of queries to identify the grammatical features of the keywords that constitute the query.
2. *Query segmentation* to identify the boundaries of the query segments, where the named entity is contained.

3. *Recognition of candidate named entities.* Using the grammatical features identified in the first phase and the boundaries set in the second phase, a small set of hand-crafted rules were defined to find the sequence of keywords that refer to a named entity.

3.2.1 Grammatical Annotation of Queries

In the first phase of QNER, each keyword in the query is annotated with two types of features: Part-of-Speech (POS) and Orthography (ORTH). POS is the grammatical category to which a keyword belongs, such as common noun, proper noun, comparative adjective, and past participle. ORTH depicts the surface form with which the keywords are spelled, such as capital case initial, lower case initial, or all capitalised letters.

Because queries are generally short, consisting of two to four words on average [11, 95, 96], there may not be sufficient context to identify the grammatical categories of the keywords that constitute the queries. Furthermore, based on the study conducted by Barr et al. [8], a majority of the queries that contained proper nouns were not capitalised, a proper noun being the grammatical category that primarily refers to unique entities in the English language (see Section 2.5). For example, consider the two queries 'apple springs texas' and 'blue earth river', where users did not follow the orthographic rules of spelling when submitting these queries. Grammatically annotating the queries using the Stanford NLP POS tagger [1] results in tagging 'blue' as an adjective and 'texas' as a verb, whereas the rest are all annotated as common nouns; however, 'springs' and 'blue' are proper nouns and they are part of the named entities 'Apple Springs' (a community in Texas, USA) and the river 'Blue Earth', respectively.

In order to overcome search query brevity, and the lack of orthographic clues, each query is enriched with the snippets of the top ranked web pages related to the target query. Augmenting queries with web pages related to the queries in order to reveal their grammatical structure was used in the work of Bendersky's in [13, 14] (Section 2.5). In a similar manner, QNER utilises the top-n web pages related to a query; however, to avoid the cost of parsing full web pages, only the snippets of the web pages are exploited. Moreover, users make a decision on whether a search result is related to their information need based on the context words that surround their search keywords in the snippets. Therefore, in the snippets, the grammatical features of the search keywords are preserved and can be transferred to search queries for annotation.

In this thesis, QNER adopts a naïve approach to query grammatical annotation, where two assumptions are made. The first assumption is that each keyword is annotated independently

[1] nlp.stanford.edu/software/tagger.shtml

of the other query keywords that surround it, and of the grammatical annotations of the keywords that surround it. The second assumption is that each keyword is annotated with the most common annotation given to the same word in the snippets set. Accordingly, query grammatical annotation is defined as follows. The top-n snippets related to a query Q, that consists of the keywords $w_1, w_2, ..., w_n$, are grammatically annotated so that each keyword is assigned a POS and an ORTH feature. As a result, every keyword w_i, such that $w_i \in Q$, has a unique set of possible grammatical annotations $G = \{\gamma_{i1}, \gamma_{i2}, ..., \gamma_{im}\}$ that is assigned to each occurrence of w_i in the snippet set. Every possible grammatical annotation $\gamma_{ij} \in G$ is assigned a score $S(\gamma_{ij}) = freq(w_{i\gamma_j})$, that is the number of times w_i is annotated with γ_j. Then, the query keyword w_i is annotated with the grammatical feature γ^* that has the highest score, that is,

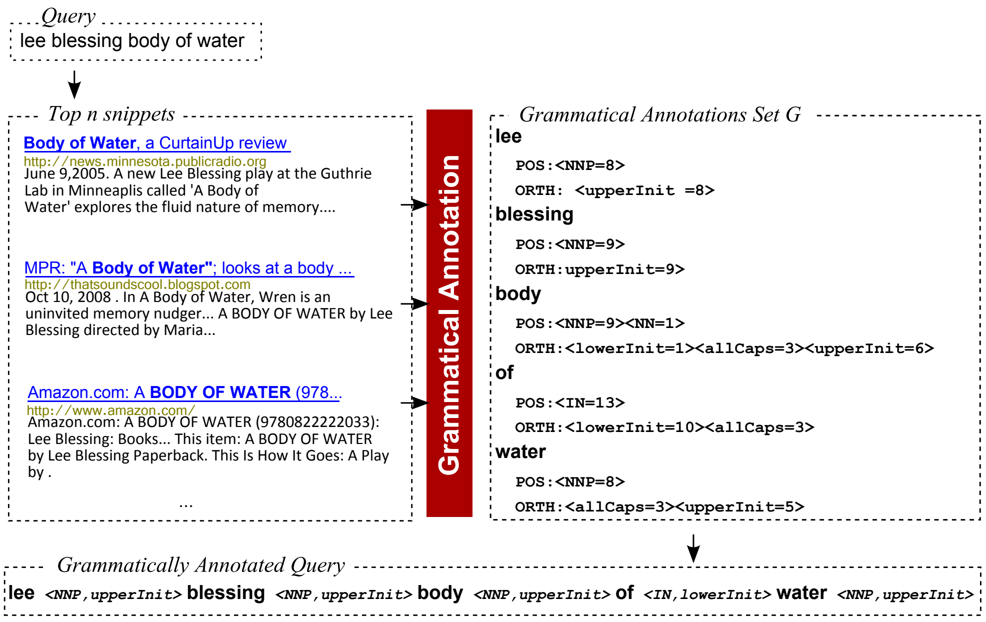
$$\gamma^* \leftarrow \arg\max_{\gamma \in G} S(\gamma) \tag{3.1}$$

For example, consider the query 'lee blessing body of water' in Figure 3.1. Each keyword in the query is annotated with the most frequent POS and ORTH features with which the same word is tagged in the snippets set. For instance, the word 'body' has two possible POS features in the snippets set, $\gamma_1 = NNP$ (proper noun) and $\gamma_2 = NN$ (common noun). Because the score of the POS feature *NNP* is higher than that of *NN*, 'body' is tagged as a proper noun. Similarly, the same keyword appears in the snippets set with three different surface forms: lower initial (`lowerInit`), capital initial (`upperInit`), and all capitalised (`allCaps`). However, the most frequent ORTH feature is capital initial, and thus, according to Equation 3.1, 'body' is annotated with the most frequent annotation (`upperInit`).

3.2.2 Query Segmentation

Query segmentation is the process of finding the most probable sequence of keywords in the query that constitutes a phrase. For example, segmenting the query 'University of London Russell Square' results in the phrases '[University of London]' and '[Russell Square]'. In NER, query segmentation can be used to set the boundaries of the segments in the query where a named entity might occur.

Here, the query segmentation is defined as follows. For a query Q that consists of the keywords $w_1, w_2, ..., w_n$, the set of all possible segmentations is denoted by $S(Q) = \{S_1, ..., S_m\}$, where $m \leq 2^{n-1}$. In other words, there are 2^{n-1} ways to segment a query that consists of n keywords, where between every two keywords, a segmentation decision might be made. Each


Query
lee blessing body of water
Top n snippets
Grammatical Annotation
Grammatical Annotations Set G
Grammatically Annotated Query

Body of Water, a CurtainUp review
http://news.minnesota.publicradio.org
June 9,2005. A new Lee Blessing play at the Guthrie Lab in Minneapolis called 'A Body of Water' explores the fluid nature of memory....

MPR: "A Body of Water"; looks at a body ...
http://thatsoundscool.blogspot.com
Oct 10, 2008 . In A Body of Water, Wren is an uninvited memory nudger... A BODY OF WATER by Lee Blessing directed by Maria...

Amazon.com: A BODY OF WATER (978...
http://www.amazon.com/
Amazon.com: A BODY OF WATER (9780822222033): Lee Blessing: Books... This item: A BODY OF WATER by Lee Blessing Paperback. This Is How It Goes: A Play by .

...

```
lee
  POS:<NNP=8>
  ORTH: <upperInit =8>
blessing
  POS:<NNP=9>
  ORTH:upperInit=9>
body
  POS:<NNP=9><NN=1>
  ORTH:<lowerInit=1><allCaps=3><upperInit=6>
of
  POS:<IN=13>
  ORTH:<lowerInit=10><allCaps=3>
water
  POS:<NNP=8>
  ORTH:<allCaps=3><upperInit=5>
```

lee <NNP,upperInit> blessing <NNP,upperInit> body <NNP,upperInit> of <IN,lowerInit> water <NNP,upperInit>

Figure 3.1: Illustration of query grammatical annotation

segmentation $S_i \in \mathbf{S(Q)}$ consists of one or more segments, and each segment s_{ij} is a sequence of query keywords that obeys the original order as when the user submitted the query. The best segmentation S^{β} is defined as the most probable one over all $S_i \in \mathbf{S(Q)}$. The probability of each S_i is calculated by multiplying the probabilities of the individual segments (s_j) that constitute the segmentation S_i:

$$Pr(S_i) = \prod_{s_{ij} \in S_i} Pr(s_{ij}). \tag{3.2}$$

The probability of a segment s_{ij} is estimated using a *local* n-gram model $(M_{snippet})$ created from the set of the top-n snippets related to Q. This probability is computed as the ratio of that segment given all the possible n-grams (N) in the snippets set, which have the same number of grams as s_{ij}. Because there are some segments of the query that may not appear in the snippets set, *additive smoothing* is performed, where a number α is added to the count of segment s_{ij} before normalising it into a probability. Furthermore, the n-gram count is augmented by the total number of unique words in the vocabulary V from the same snippets set related to Q, thus giving rise to:

$$Pr(s_{ij}) = \frac{count(s_{ij}) + \alpha}{N + (V * \alpha)}. \tag{3.3}$$

Using Jelinek-Mercer smoothing [53], the probability of the segment s_{ij} is further smoothed

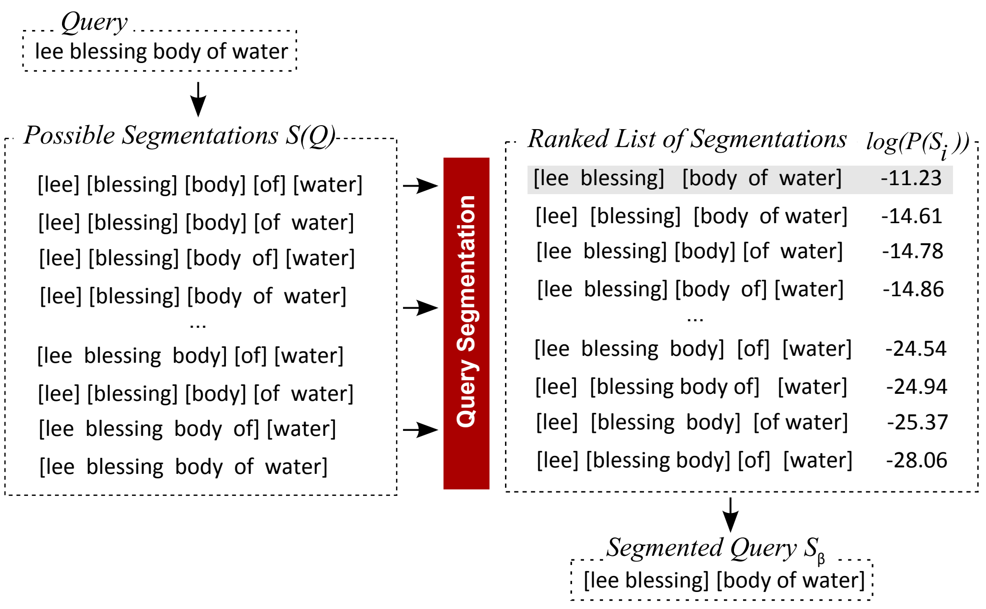

Query
lee blessing body of water
Possible Segmentations S(Q)
[lee] [blessing] [body] [of] [water]
[lee] [blessing] [body] [of water]
[lee] [blessing] [body of] [water]
[lee] [blessing] [body of water]
...
[lee blessing body] [of] [water]
[lee] [blessing] [body] [of water]
[lee blessing body of] [water]
[lee blessing body of water]
Query Segmentation
Ranked List of Segmentations
log(P(S_i))
[lee blessing] [body of water] -11.23
[lee] [blessing] [body of water] -14.61
[lee blessing] [body] [of water] -14.78
[lee blessing] [body of] [water] -14.86
...
[lee blessing body] [of] [water] -24.54
[lee] [blessing body of] [water] -24.94
[lee] [blessing body] [of water] -25.37
[lee] [blessing body] [of] [water] -28.06
Segmented Query S_β
[lee blessing] [body of water]

Figure 3.2: Illustration of query segmentation using logarithmic probability

by the probability of the same segment given a *web* n-gram model (M_{web}) using an empirically set parameter λ between zero and one to obtain:

$$Pr(s_{ij}) = \lambda Pr(s_{ij}|M_{snippet}) + (1 - \lambda)Pr(s_{ij}|M_{Web}). \tag{3.4}$$

The justification for using a mixture between a local n-gram model created from the snippets set related to the target query, and a global n-gram model created from the Web is as follows. Consider a query that consists of a single named entity and one or more additional keywords that refine the search results of the target entity, such as ‘The Hobbit trailer’. When segmenting the query using the snippets set solely, the probability of having both ‘trailer’ and ‘The Hobbit’ is high given that the top snippets of the web pages include mentions of both the entity ‘The Hobbit’ and the keyword ‘trailer’. However, when utilising a Web based n-gram model, the probability of a segment is measured given a larger number of additional keywords that might be associated with the named entity, such as ‘cast’, ‘release date’, or ‘cinema’. On the other hand, looking at the query ‘New Look’, which consists of a single named entity that refers to a brand name, its constituting keywords ‘new’ and ‘look’ appear independently far more often than jointly as a mention of the brand name. Therefore, using the snippets, the query is segmented in light of the search results in which the two keywords often appear as a single phrase.

Figure 3.2 shows the set of all possible segmentations, $S(Q)$, of the query ‘Lee Blessing Body of Water'. Using the logarithm of the probabilities, the query is segmented to '[lee blessing]' and '[body of water]'. The log of probabilities is often used when computing language model probabilities to avoid numerical overflow and to facilitate computation [56].

3.2.3 Detection of Candidate Named Entities

Given that search queries are grammatically annotated and segmented, candidate named entities can be identified using a small set of hand-crafted rules. The following definitions for candidate named entities in queries have been derived by examining a sample of grammatically annotated and segmented queries.

(A) General Entities: A sequence of proper nouns contained in a segment, such as 'United Kingdom', 'Harrison Ford', or 'Pizza Hut'.

(B) Entities with Connectors: One of the connectors '&', 'of', 'for', or 'de' followed and preceded by proper nouns and contained in a segment. Some examples are 'University of Wisconsin', 'Black & Decker', 'Battle for Manila', and 'Anthony de Mello'.

(C) Entities with Numbers: A sequence of proper nouns that include a keyword whose type is a number, and that are contained in a single segment, such as 'Microsoft Office Professional *2003*', or a keyword whose POS is a number and whose type is a word or alphanumeric with a capital initial, or all capitalised letters such as '*First* Citizens Bank', and 'Fit4less'.

(D) Entities based on Orthographic Features: A sequence of keywords whose orthographic feature is mixed letters such as 'eBay'. In addition, a keyword whose POS is an adjective but its initial letter is capitalised, along with one or more keywords whose POS annotation is a proper noun contained within a single query segment. Some example queries are 'Big Bang Theory' (TV series), where 'big' is an adjective.

Each of the definitions above can be preceded by the determiner 'the', for example, 'The Hobbit', if it appears within the same query segment. Furthermore, any keyword that falls in one of the definitions above can be followed by apostrophe and an 's' (''s'), such as 'The Complete Idiot's Guide' (Book series).

Because the objective is to identify the search keywords that might refer to a named entity, the set of rules defined is small and is not class specific, and thereby, does not require domain knowledge. Therefore, the heuristic employed for QNER is that a candidate named entity is *any sequence of search keywords that obeys the rules and falls within the boundaries set by the probabilistic segmentation of queries.*

In the following sections, the experimental settings employed to evaluate the QNER approach are presented, followed by the performance scores achieved via different variants of evaluation settings.

3.3 Experimental Settings

Table 3.1 summarises the settings employed for experimentally evaluating the proposed approach for QNER. These settings are described in detail in the following paragraphs.

Table 3.1: Summary of experimental setting.

Query Log	MSN query log sampled in 2006, $total = 14,921,285$ queries.
Query Enrichment	*Google Web Search API* used to retrieve top-n snippets, where $n = 8$.
Query Segmentation	For each query, the probability of each segment is estimated from *local n-grams* drawn from the top-8 snippets using Eqn. 3.3, where $\alpha = 0.0001$. Then, the probability of the segment is smoothed with the probability of the same segment given *Bing Web n-gram model* [102] using Eqn. 3.4, where $\lambda = 0.6$. Lastly, the probability of each possible segmentation of the query is estimated using Eqn. 3.2.
Query Grammatical Annotation	Top-8 snippets annotated using GATE toolkit [28]. Then, annotations were transferred to search keywords using Eqn. 3.1.
Candidate Named Entity Recognition	Using Java Annotation Patterns Engine (JAPE), eight rules were created (see Appendix B).
Samples of Randomly Selected Queries	*Validation sample = 1000* queries to set α and λ, and to validate the rules defined in Sec. 3.2.3. *Evaluation sample = 1000* queries to report performance.

Query Log: The proposed approach to QNER was applied to an MSN query log that was sampled over several weeks in 2006, and that consisted of approximately 14.9 million queries. Each query record in the query log consisted of a session ID, timestamp, query string, and per query search result click-through. The query log is pre-processed as follows. First, queries were transformed to lower-case letters in order to eliminate duplicates, e.g., the queries 'google', 'GOOGLE', and 'Google' were transformed to 'google' and all duplicates were excluded. Then, using a simple regular expression pattern, URLs were detected and also eliminated. Finally, noise, which includes all queries consisting of a sequence of symbols, were detected and excluded. Table 3.2, presents general statistics of the MSN query log.

Table 3.2: MSN query log general statistics

	Frequency	%
Unique Queries	6,623,959	44.393
Duplicates	7,380,143	49.461
URL	912,202	6.113
Noise	4,981	0.033
Total	14,921,285	

The spelling of each query was checked and corrected using the Yahoo Spelling Suggestion API [2]. In addition, for each query, the top eight snippets were retrieved and stored using Google's Web Search API [3]. Only eight snippets were used, because it was the maximum number of results that could be retrieved per API request. From the pre-processed queries, a validation sample of 1,000 queries was randomly selected in order to validate the segmentation parameters and the rules defined in Section 3.2.3. Moreover, another sample that consisted of 1,000 random queries was selected in order to report the performance of the QNER approach, as described in the following section. The justification behind using 1,000 to be the size of the samples is described in Section A.1.

Query Grammatical Annotations: The snippets set for each query was annotated grammatically using the GATE toolkit [28], where each snippet was processed through a pipeline that consisted of a sentence splitter, a tokeniser, and a POS tagger. The tokeniser assigns to each token its orthographic feature, as well as the type of keyword, such as number, or word. Then, using the grammatically annotated snippets related to a query, each search keyword is assigned a POS and an ORTH feature whose score is the highest using Equation 3.1 page 3.1. Refer to Section A.4 for more details of how GATE was utilised to process the snippets set.

[2]Yahoo! spelling suggestions API services were deprecated.
[3]Google Web Search API services were deprecated.

Query Segmentation: For each query, the probability of each segment is estimated using local n-grams drawn from the snippets set using Equation 3.3, with $\alpha = 0.0001$. Next, the probability of each segment is smoothed by the probability of the same segment obtained from the Bing web n-gram model [102] using Equation 3.4, with $\lambda = 0.6$. Segmentation parameters, i.e., λ and α, were empirically set using the validation sample. Furthermore, because the Bing web n-gram model contains a maximum of 5-grams, the probabilities of the segments whose order is greater than five are estimated using the chain rule along with the Markov assumption (refer to Section A.2 for details).

Candidate Named Entity Recognition: Finally, the candidate named entities were extracted using eight hand-crafted rules designed to model the definition of query candidate named entities presented in Subsection 3.2.3. The rules were modeled using Java Annotation Patterns Engine (JAPE), which is a pattern matching language developed by GATE [28]. JAPE provides a finite state transducer that utilises regular expressions over annotations. The set of rules are listed in Appendix B.

Table 3.3 presents examples of grammatically annotated and segmented queries, and the corresponding extracted candidate named entities (selected from the query log after running QNER). This provides an illustration of how query segmentation and grammatical annotation were used to identify candidate named entities in queries. When there is a consecutive sequence of proper nouns in a query, such as 'abc dallas' in Table 3.3, segmentation sets the most probable boundaries for the candidate named entities in the query.

3.4 Evaluation and Analysis of Results

Previous approaches to named entity recognition in queries were evaluated manually either by checking a random sample of the extracted entities [49], or by checking the top-k extracted entities ranked based on a specific classification score [80, 44] without referring to their context in search queries (see Section 2.7.3). Nevertheless, the following issues may arise: first, the extracted named entities are evaluated out of context. For example, 'safari' is an entity in the query 'download safari', whereas it is not in the query 'safari trips'. Second, when choosing the evaluation sample out of the extracted named entities, there is no indication of the percentage of the *False Negatives* (i.e., the named entities that were incorrectly missed by the approach). Third, because the extraction is evaluated out of context, there is no reference to how accurate the named entity detection is, i.e., whether the boundaries of the extracted named entities

Table 3.3: Examples of extracted candidate named entities using query grammatical annotation and segmentation. `POS: (NNP: proper noun, NN: common noun, CC:conjunction, CD: cardinal number). ORTH: (upperInit: Capital Initial, lowerInit: lower case initial, AllCaps: all capitalised)`

Query	**Grammatical annotation**	**Segmentation**	**Candidate NE**
leon final fantasy	leon `<NNP,upperInit>` final `<NNP,upperInit>` fantasy `<NNP,upperInit>`	[leon] [final fantasy]	leon final fantasy
abc dallas	abc `<NNP,upperInit>` dallas `<NNP,upperInit>`	[abc] [dallas]	abc dallas
antique cars chevrolet	antique `<NN,lowerInit>` cars `<NN,lowerInit>` chevrolet `<NNP,upperInit>`	[antique cars] [chevrolet]	chevrolet
bmw owner blogspot	bmw `<NNP,AllCap>` owner `<NN,lowerInit>` blogspot `<NN,lowerInit>`	[bmw owner] [blogspot]	bmw
backup ipod	backup `<NN,lowerInit>` ipod `<NN,MixedCap>`	[backup] [ipod]	ipod
ward & glass and oklahoma	ward `<NNP,upperInit>` & `<CC,NA>` glass `<NN,upperInit>` and `<CC,lowerInit>` oklahoma `<NNP,upperInit>`	[ward & glass] [and] [oklahoma]	ward & glass oklahoma
wizard of oz film photos	wizard `<NNP,upperInit>` of `<IN,lowerInit>` oz `<NNP,upperInit>` film `<NN,lowerInit>` photos `<NN,lowerInit>`	[wizard of oz] [film] [photos]	wizard of oz
warcraft 3	warcraft `<NNP,upperInit>` 3 `<CD,N/A>`	[warcraft 3]	warcraft 3

are set correctly, for example, extracting 'London' as an entity from the query 'University of London'. To overcome these issues, the QNER algorithm presented in this thesis is evaluated from two perspectives.

Query-Level Evaluation: A random sample from the query log is taken in order to evaluate the extraction of candidate named entities given their representation in the queries from which they were extracted.

Entity-Level Evaluation: Each individual extracted named entity is automatically checked as to whether it is correct using DBpedia as a source of named entities [18]. The knowledge base of the English version of DBpedia describes 3.64 million things which include names of people, movies, places, or organisation. For more details refer to Section A.3.

In both of the evaluation approaches, the performance measures precision, recall, and F_1 were used (see Chapter 2 Table 2.3 page 35). In the following subsections, the True and False Positives (TP, FP), and True and False Negatives (TN, FN) are defined for both evaluation perspectives, along with the evaluation scores achieved.

3.4.1 Query-Level Evaluation

Evaluating QNER at the query level provides insight into the accuracy of the extracted named entities boundaries, the frequency of missed named entities, and the accuracy of the entities extractions given their context in the queries from which they were extracted. Therefore, the metrics of evaluation TP, FP, FN, and TN are defined as follows:

TP: The query has one or more named entities, and all were detected correctly.

TN: The query has no named entities, and none were detected.

FP: One or more query keywords were incorrectly tagged by the rules as named entities.

FN: One or more query named entities were (incorrectly) missed by QNER.

The evaluation sample, which consisted of 1,000 randomly selected queries, was checked manually and each query was annotated as TP, FP, TN, and FN. Table 3.4 presents the precision, recall, and F_1 achieved by QNER. Two evaluation settings were used: *Evaluation I*, where the named entity detection is manually annotated as correct regardless of its boundary, e.g., extracting 'Sony PS3 news'; and a stricter version, *Evaluation II*, where the named entity boundary must be accurate, e.g., extracting 'Sony PS3'.

Table 3.4: Query-level QNER evaluation

	Precision	Recall	F_1
Evaluation I	0.794	0.973	0.874
Evaluation II	0.643	0.966	0.772

Examining the sample, it was found that the percentage of correctly annotated queries, i.e, the true positives ($TP/total$), represent 67.7% in Evaluation I, and 54.9% in Evaluation II. On the other hand, the percentage of true negatives ($TN/total$), which includes queries with no named entities and none detected, is 12.9% in both evaluations. This results in a total accuracy ($(TP + TN)/total$) of 80.6% and 67.8% for Evaluation I and Evaluation II, respectively.

The main boundary errors encountered in the evaluation include cases where only part of the actual named entity is detected, e.g., extracting 'britney' as a candidate named entity from the query 'britney spears pictures' or the case where the extraction crosses the boundary of the actual entity, such as extracting 'britney spears pictures' as a candidate named entity. These types of errors occur because of incorrect Part-of-Speech tagging of the query keywords, such as tagging 'spears' as a common noun or 'pictures' as a proper noun. In addition, for queries with more than one named entity, the commonly observed error was that of detecting a sequence of named entities that appeared in a query as a single named entity, meaning that the boundaries set by query segmentation were incorrect. All boundary related errors were counted as FPs in Evaluation II, thus resulting in a decrease in the precision from 79.4% in Evaluation I to 64.3%.

Furthermore, for both evaluations, when a query keyword that is a common noun appeared more often in the snippets set as a proper noun, the keyword was extracted as a candidate named entity, e.g., the word 'architect' was part-of-speech tagged in the snippets set as a proper noun more often than as a common noun, thus resulting in its detection as a candidate named entity from the segmented query '[daniel nyante] [ny] [architect]' in addition to the named entities 'daniel nyante' and 'ny'. Similarly, the keywords 'bio' when associated with a person name, or 'trailer' when associated with a movie name, if they appeared more often as proper nouns in the snippets set, they often were detected as named entities.

The percentage of queries with a named entity missed by the rules was only 1.9% ($FN/Total$). Named entities, such as movies and songs that included keywords that were not necessarily proper nouns were missed by the rules (defined in Subsection 3.2.3). For example, the determiner 'a' and the preposition 'with' were not defined in the rules, thus in the queries 'dancing with the stars' (TV show), and 'a rose for emily' (Book), the named entities were inaccurately detected, i.e., 'Rose for Emily', instead of 'A Rose for Emily', as well as detecting 'dancing' and 'the stars' as two distinct named entities. Other examples of named entities include the movie 'Thank you for Smoking', the company name 'Bed Bath and Beyond', and the novel 'A Tale of Two Cities'. The defined candidate entity recognition rules failed to detect these named entities, or set their boundaries incorrectly. Nevertheless, the recall scored is 97.3% and 96.6% for Evaluation I and 96.6% in Evaluation II, respectively, thus revealing that the large percentage of entities in the query log follow the definition of entities in these rules.

Looking specifically at how query grammatical annotation and segmentation affect the performance of QNER, it was found that the performance decreases when the query keywords have different string representation(s) in the snippets set from the one used by the user to submit the query. One of the cases that often appeared is when the query keyword is abbreviated and it appears in the snippets set in the full form, or vice versa, for example, 'NY', instead 'New York'. Another case is when a keyword of the query submitted by the user has more than one token, and the same word appeared in the snippets set as a single token, such as 'JCrew' versus 'J Crew' or 'J.Crew', as well as 'J tek' and 'Jtek', or 'the note book' versus 'the notebook'.

3.4.2 Entity-Level Evaluation

From the evaluation sample that consisted of 1,000 search queries, 1,172 candidate named entities were extracted. The entity-level evaluation is conducted using the following two phases:

Phase 1 (*Manual Evaluation*): Each candidate named entity was examined manually and labelled as either:

(i) An *Accurate Named Entity Extraction* (A-NE) in the case where the candidate named entity is accurately extracted by QNER.

(ii) An *Inaccurate Named Entity Extraction* (I-NE) in the case where the candidate named entity is extracted, but its boundaries were incorrect, as described in the previous section.

(iii) *Not a Named Entity* (N-NE) in the case where the named entity extraction is incorrect.

Phase 2 (*Automatic Evaluation using DBpedia*): Each named entity extraction was automatically checked against DBpedia and labelled as either:

(i) An *Exact Match* (EM) when a DBpedia result is found whose label is identical to the extracted named entity string.

(ii) A *Partial Match* (PM) when a DBpedia result is found whose label is part of the extracted named entity, or vice versa, e.g., the entity extracted is 'Rhodes & Rhodes', which is a company name, and the most similar DBpedia result label is 'Rhodes', which is a location name.

(iii) A *No Match* (NM) when there are no DBpedia results related to the extracted entity.

Using the manual evaluation and DBpedia automatic evaluation, the performance scores achieved by QNER can be presented as shown in Table 3.5.

Table 3.5: Manual evaluation versus DBpedia results (%)

		Automatic Evaluation			
		Exact Match	Partial Match	No Match	Total
Manual Evaluation	Accurate NE	24.317	17.235	22.184	**63.736**
	Inaccurate NE	1.792	5.546	5.546	**12.884**
	Not NE	8.191	11.433	3.754	**23.378**
	Total	**34.300**	**34.214**	**31.484**	**100**

From the 1,172 extracted candidate named entities, 34.30% of the instances had a DBpedia result whose label exactly matched the string of the named entity extracted by QNER. Examining the entities whose strings exactly matched DBpedia results (EM), 24.3% of these entities were accurately extracted named entities according to the manual evaluation. On the other hand, 8.2% of these entities were actually not named entities, but DBpedia results were found whose labels exactly matched the incorrectly extracted entities. For example, ‘battles’ was extracted incorrectly as a candidate named entity from the segmented query ‘[nebraska] [indian battles]’. Although ‘battles’ is not a named entity, an exactly matching DBpedia result was found that referred to a band name. Furthermore, 1.79% of the inaccurately extracted entities were matched to a DBpedia result, such as ‘rush’, which was inaccurate named entity extraction from the query ‘rush henrietta ny schools’. The ‘rush’ entry in DBpedia refers to a band, whereas in the query ‘rush henrietta’, ‘rush’ refers to RushHenrietta Senior High School in New York. Table 3.6 shows more examples of each manual evaluation case and the corresponding exactly matched DBpedia result.

Table 3.6: Named entity examples that Exactly Matched (EM) DBpedia results

	Query	Extracted NE	Matched DBpedia Result
A-NE	[vivien cardone]	vivien cardone	vivien cardone (actress)
	[edgar cayce] [history]	edgar cayce	Edgar Cayce (person)
	[american idol]	american idol	american idol (TV show)
I-NE	[your mine and ours]	ours	Ours (band)
	[j alexander meinard]	j alexander	J. Alexander (model)
	[aim conversations]	aim	Aim (musician)
N-NE	[nebraska] [indian battles]	battles	Battles (band)
	[cheap] [tickets from okc to atl]	tickets	Tickets (film)
	[recovery disk downloads]	recovery	Recovery (Eminem album)

Looking at candidate named entities whose string partially matched a DBpedia result, 17.23% were accurate named entity extractions. This was often observed when the user submitted a part of the entity name or an acronym of the named entity, rather than the full name. For example, from the segmented query ‘[lexus dealers] [and] [ma]’, ‘ma’ was extracted as a candidate named entity (in addition to Lexus); however, the entry in DBpedia was Massachusetts, instead of MA. Other examples include the extracted candidate named entity ‘powerpoint’, instead of Microsoft Powerpoint, or the candidate named entity ‘shakespeare’, instead of William Shakespeare. On the other hand, 5.54% of partially matched named entities (PM) were inaccurate named entity extractions. In this case, either the extracted candidate named entity and the partially matched DBpedia result referred to the same entity, or they referred to different named entities. For example, ‘pittsburgh post’ was inaccurately extracted from the query 'pittsburgh post gazette', and the partially matched DBpedia result was Pittsburgh Post-Gazette, where both referred to the same named entity. On the other hand, from the query 'garza and garza', 'garza' was inaccurately extracted from the query that referred to a company's name, and the partially matched DBpedia result was Hector Garza, who is an athlete. Table 3.7 lists more examples of extracted named entities whose string partially matched a DBpedia result.

Table 3.7: Named entity examples that Partially Matched (PM) DBpedia results

	Segmented Query	Extracted NE	Matched DBpedia Result
A-NE	[sarbanes]	sarbanes	Paul Sarbanes (politician)
	[animated backgrounds for powerpoint]	powerpoint	Microsoft PowerPoint (software)
	[bear fools gladly] [shakespeare]	shakespeare	William Shakespeare (artist)
I-NE	[papa john's] [coupons] [gainesville]	john's	John Lennon (musician)
	[sunsetter awnings]	sunsetter	SunSetter Awnings (company)
	[bryce] [tech]	bryce	Quentin Bryce (governor)
N-NE	[rhodes & rhodes] [surveyors]	surveyors	The Surveyors (film)
	[ferries]	ferries	BC Ferries (company)
	[idaho rv] [reservations]	reservations	No Reservations (film)

Lastly, looking at candidate named entities not found in DBpedia, 22.2% were accurately extracted named entities. These entities often referred to small businesses and people on Facebook, or have personal web pages, which are not recorded in DBpedia, such as 'johnny esposito' and 'shawnee studios'. For more examples of named entities extracted by QNER and with no matching DBpedia results, refer to Table 3.8.

Table 3.8: Named entity examples with No Matching (NM) DBpedia results

	Query	Extracted NE
A-NE	[david bennes barkley] [biography]	david bennes barkley
	[rose & womble]	rose & womble
	[intellicast]	intellicast
I-NE	[hummel bros.] [inc.] [new haven ct]	hummel bros
	[match point trailer]	match point trailer
	[halo 2 toys]	halo 2 toys
N-NE	[sweatshirt wholesale]	sweatshirt
	[remove] [mass email]	mass email
	[valley specialized trucking companies]	valley specialized trucking

The results presented in Table 3.5 can be exploited (i) to validate the usefulness of DBpedia for evaluating the detected named entities, and (ii) to extrapolate the performance of QNER when executed over a large set of search queries, as discussed in the following paragraphs.

DBpedia for NER Evaluation

In order to validate the use of DBpedia as a source for QNER evaluation using the results presented in Table 3.5, the manually checked sample is assumed to be a gold standard list to which the DBpedia results are compared. To achieve this, the following definitions of True and False Positives (TP, FP), and True and False Negatives (TN, FN) were made:

TP: A candidate named entity extraction by the system that is a named entity according to the manual evaluation, and has a corresponding DBpedia result.

TN: A candidate named entity extraction that is not a named entity, and has no DBpedia result.

FP: A candidate named entity extraction that is not a named entity, and has a DBpedia result.

FN: A candidate named entity extraction that is a named entity, and has no DBpedia result.

In the evaluation, each examined instance is True or False according to the manual evaluation, and the same instance is labelled as Positive when DBpedia has a result related to it; otherwise, it is labelled Negative. Similar to query-level evaluation, two settings were used: *Evaluation I* is conducted regardless of the boundaries of the candidate named entity extracted and how similar it is to the DBpedia result label, whereas in *Evaluation II*, the extracted named entity should be accurate and should match the DBpedia result label exactly. Accordingly, TP, TN, FP, and FN were counted for each evaluation, as indicated in Table 3.9.

Table 3.9: Breakdown of Evaluation I and Evaluation II

		Automatic		
		EM	**PM**	**NM**
Manual	**A-NE**	TP	TP	FN
	I-NE	TP	TP	FN
	N-NE	FP	FP	TN

Evaluation I

		Automatic		
		EM	**PM**	**NM**
Manual	**A-NE**	TP	FN	FN
	I-NE	FP	TN	TN
	N-NE	FP	TN	TN

Evaluation II

Table 3.10 presents the precision, recall, and F_1 measure of the DBpedia evaluation. In Evaluation I, it was found that 71.4% of the related DBpedia results labeled (EM, PM) were similar to the candidate named entities accurately or inaccurately detected by QNER. In Evaluation II, an instance was considered correct (TP) only when the DBpedia result label exactly matched the accurately extracted instance, thus resulting in a precision of 70.9%. On the other hand, recall, which is the fraction of the extracted named entities found in DBpedia, was 63.8%, where I-NE extraction and PM DBpedia results were considered correct; otherwise, the recall dropped to 38.6% when only those whose label exactly matched the DBpedia result were considered correct.

Table 3.10: Evaluation of DBpedia as a source for QNER validation

	Precision	**Recall**	F_1
Evaluation I	0.714	0.638	0.674
Evaluation II	0.709	0.386	0.496

To summarise, there are two main issues when evaluating a NER system using a knowledge base, which consists of listings of classified named entities:

- The knowledge base may not contain the named entity. Jain and Pennacchiotti in [49] reported that 61% of their named entity extractions were correct, although there were no records in Wikipedia that matched the extraction.

- The knowledge base contains an entry whose string matched an incorrectly extracted named entity.

Performance Extrapolation

Given that the sample is representative of the query log, the performance of QNER approach can be estimated when it is tested on a larger query set using Table 3.5. For example, the percentage of accurate named entity extractions with exact matching DBpedia results is approximately 71%. This percentage is the result of dividing the percentage of accurately extracted named entities with a matching DBpedia result by the total of all named entities found in DBpedia (24.317 divided by 34.3). Out of all the extractions that had an exact match in DBpedia, the estimation that approximately 24% are Not NEs can be made by dividing 8.191 by 34.3, where 8.191 is the percentage of incorrectly extracted named entities found in DBpedia As observed from the sample, this was because of the ambiguity of some named entities. For example, the common noun 'meatballs' was incorrectly extracted as a candidate named entity from the query 'gif or picture of spaghetti and meatballs'. Although it was labeled as *Not NE* in the manual evaluation, there was a DBpedia result whose label exactly matched it when referring to a Canadian comedy film. Furthermore, it can be assumed that approximately 70% of candidate named entity extractions with no DBpedia results were accurately extracted named entities (22.184 divided by 31.484). In the sample, the named entities that fall in this category included non-celebrities, facility names such as restaurants and hotels, and small companies and businesses.

Using a larger uniform random sample of 100,000 search queries, the QNER approach resulted in the extraction of 110,113 candidate named entities. Checking each automatically against DBpedia, it was found that the percentages of the DBpedia evaluation cases (EM, PM, and NM) were 34.967%, 33.800%, and 31.231%, respectively. These are close to the results found in the evaluation sample that consisted of 1,000 queries, as shown on the last line of Table 3.5. This provides further evidence that the above estimates are representative of the query set.

3.5 Discussion

In this chapter, a novel approach for recognising candidate named entities in search queries was presented. The main contribution of this approach is that it utilises grammatical annotation and probabilistic segmentation to detect candidate named entities in search queries.

Because queries consist of only few keywords, the snippets of the top web pages related to the queries were employed to enrich search keywords. The advantage of utilising a page snippet rather than the full page is three-fold: (i) the cost of parsing full web pages to eliminate HTML tags and other web page contents not related to the user search query is avoided, (ii) the surface form with which search keywords should be spelled according to the orthographic rules of spelling in English is preserved, and (iii) the snippets are often composed of short phrases with sufficient context surrounding the query keywords so that users can understand them. Accordingly, the grammatical features of search keywords can be transferred from the snippets to the search query.

Query grammatical annotation identifies the keywords that may refer to a candidate named entity, whereas query segmentation was used to set the boundaries of the query segments in which the named entity might occur. QNER employed a small set of rules that were derived specifically for the linguistic structure of queries, which leveraged the grammatical features and segmentation boundaries in order to identify candidate named entities in search queries.

In contrast, previous approaches to identify named entities in search queries have relied on one of the following two heuristics to identify an initial set of potential named entities: (i) the remainder of the query is a candidate named entity if it matches a query pattern [44, 80, 103], and (ii) a sequence of capitalised keywords in a search query is a candidate named entity [49].

Using the first heuristic, a small set of seeds, i.e., named entities selected for each target entity class, is used and the query log is scanned to find query patterns in which the seeds occurred. A query pattern is simply the remainder of the query after excluding the matched entity string, for example, '# lyrics' is a query pattern extracted from the query 'Echo lyrics' using the seed entity 'Echo' (a song). This pattern is used to find other named entities that belong to the entity class Song from the queries, e.g., 'Listen lyrics'. However, the same pattern applies to the following queries:

- 'brad paisley dolly parton lyrics', where the phrase preceding 'lyrics' is an inaccurate extraction of the named entities 'Brad Paisley' and 'Dolly Parton', and the techniques may result in incorrect classification, because both entities refer to singer names, instead of a song name.
- 'songs lyrics', where the phrase preceding 'lyrics' is not a named entity.

The other major problem when using this heuristic is when a query consists of nothing but the named entity string, and there is no context to be matched, e.g., the queries 'celine dion', 'united kingdom', and 'frozen'.

On the other hand, using the second heuristic to identify named entities may result in missing many potential named entities submitted to a search engine with no regard to the orthographic rules of spelling in the English language, i.e., capitalisation.

QNER addresses these problems by first annotating each query keyword with the most common POS given to the same word in snippets of the top-n web pages related to the target query. The POS and ORTH features assigned were used to reveal whether a search keyword might be or belong to a named entity. Furthermore, setting the boundaries through query segmentation often results in accurate extraction of named entities, rather than assuming that the remainder of the query, after excluding the query pattern, is a named entity.

Although query grammatical annotation and query segmentation are among the major problems that recent research in query log analysis has targeted, none of the previous approaches utilised both of the techniques to identify named entities in search queries. Furthermore, previous approaches evaluated the proposed techniques by manually checking the extracted named entities without referring to the queries in which the named entities appeared. By contrast, QNER approach presented herein has been evaluated both at query and entity levels in order to obtain an accurate estimate of its performance.

Chapter 4

Query Named Entity Classification QNEC

4.1 Overview

In this chapter, the vector expansion approach for classifying detected candidate named entities in search queries to a set of target entity classes is presented. The chapter is organised as follows: first, three types of entity-centric queries are defined and presented in Section 4.2. Section 4.3 presents the proposed vector expansion approach to classify entities in search queries. The evaluation and the experimental setting used to evaluate the approach is presented in Section 4.4. Next, the vector expansion method is assessed using two experiments: performance assessment, which is presented in Section 4.5, and comparative assessment, which is presented in Section 4.6. Finally, the chapter is concluded with a discussion of the main findings in Section 4.7.

4.2 Three Types of Entity-Centric Queries

An entity-centric query is a query that includes the mention of at least one candidate named entity. In the literature, different categorisations of web search queries have been proposed. In the context of search log analysis, search queries were categorised based on information need [21] or from the semantic search perspective, as in [86]. On the other hand, in this work, given a target candidate named entity to be labeled with a target entity class, a search query can fall into one of the following categories:

Focused Query (FQ): an entity-centric query that consists solely of the target candidate named entity string. For example, the following queries are all focused queries:

- 'Nicolas Cage' (Actor).
- 'The Big Bang Theory' (TV series).

- ‘Better Business Bureau’ (Organisation).

Very Focused Query (VFQ): an entity-centric query that consists of a single candidate named entity and one or more query keywords. The additional keywords are often employed by users to further refine the search results related to the target named entity. For example, if the target named entity is ‘Nicolas Cage’, the following are variations of VFQs:

- ‘list of movies with Nicolas Cage’.
- ‘Nicolas Cage 2006’.
- ‘Nicolas Cage films’.
- ‘Nicolas Cage pics’.

UnFocused Query (UFQ): an entity-centric query that consists of one or more named entities in addition to the target named entity that may or may not belong to the same entity class. For instance, when the target candidate named entity detected is ‘Nicolas Cage’, the following are variations of UFQs that contain the mention of the actor name and other detected candidate named entities.

- ‘*Kevin Bacon* and Nicolas Cage’.
- ‘Nicolas Cage *Emmy*’.
- ‘*The Ghost Rider* with Nicolas Cage’.

As discussed in Chapter 2, previous approaches to named entity detection in search queries, such as [44, 80], relied on the context surrounding the target named entity to reveal its class. Given the query categorisations defined above, this can only be applied to VFQs and UFQs, where queries contain additional keywords in addition to the target named entity string; however, this is not applicable to FQs. In Section 4.5 the classification of named entities in search queries is evaluated given the type of queries in which the named entity occurred.

4.3 Vector Expansion for Entity Classification

To address the problem of context sparsity (e.g., the case of FQs) and ambiguity (e.g., ‘trailer’ may precede a Film or a Video Game) in entity-centric queries imposed when relying solely on the query string to classify named entities, the snippets set related to each query is used. Although the snippets for a query with a named entity often do not follow a generalised pattern and are not fully grammatically structured, they often share some common words. Therefore, a *Bag-of-Words* approach is used to represent the contextual clues of candidate named entities. Representing the clues of entities as vectors in a common vector space provides flexibility for identifying the similarity scores between entities that belong to the same named entity class.

4.3.1 Bag-of-Context-Words (BoCW)

The vector expansion approach operates over two features derived for each candidate named entity extracted from entity-centric queries: a *named entity instance* (NE) and a *Bag-of-Context-Words* (BoCW).

Because the same named entity might occur in more than one query in a query log, a NE is used to refer to each occurrence of that named entity. Furthermore, the top-n snippets related to each entity-centric query are processed to derive a BoCW vector, with one component for each *context word*, together with a weight set to the term frequency of that word in the snippets set. For a word to be selected as a context word in the snippets set, it should satisfy the following conditions:

1. It is not a member of a stopwords list that consists of extremely common English words, such as 'is', 'are', 'an', and 'a'.
2. Its Part-of-Speech is either a noun or a verb.
3. After excluding stop words (condition 1) and words whose Part-of-Speech is neither a verb nor a noun (condition 2), the context word should fall within a predefined window size of k words around the named entity occurrence.
4. It appears within the sentence boundaries in which the named entity occurred.

The justification behind using only nouns and verbs as contextual clues and excluding other words is as follows: generally, a noun is, as defined in the Oxford dictionary, 'a word used to identify any of a class of people, places, or things', and a verb is 'a word used to describe an action, state, or occurrence'. When examining a set of snippets related to a query bearing a named entity, often the set of common nouns and verbs that surround a named entity are strong indicators of its class. For example, consider the following different snippets of web pages related to an entity X:

(i) X (born July 3, 1962) is an American actor and filmmaker...

Table 4.1: A sample of NEs and the corresponding derived BoCWs

Class	NE	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)...

(ii) X is an American corporation headquartered in Redmond, Washington, that develops, manufactures, licenses, supports...

(iii) X is a 1939 American epic historical romance film adapted from Margaret Mitchell's novel ...

In snippet (i), the nouns 'actor' and 'filmmaker' indicate that X is a Person, whereas the noun 'corporation' in (ii) reveals that X belongs to the class of companies. Moreover, the noun 'film' in (iii) shows that X refers to a movie name. Likewise, the verb 'born' is often associated with Person entities, whereas 'develops', 'manufactures', and 'licenses' are associated with companies. On the other hand, the same adjective 'American' appeared in the three snippets regardless of the named entity class. Adjectives often refer to properties and qualities with which different named entities can be described, such as white, black, good, and bad. Therefore, given a BoCW representation of contextual clues, adjectives were excluded.

Table 4.1 presents examples of extracted BoCWs for a sample of NEs. The extracted BoCW derived for each entity instance, collectively, provides an indication of the class to which the entity instance might belong.

4.3.2 Vector Expansion

The proposed approach to QNEC is based on the assumption that NEs that belong to the same named entity class have similar BoCWs. Given a target entity class, the objective is first to find a few instances whose BoCW vector contains context words that are distinctive and descriptive of the entity class. This initial set of named entity instances is referred to as seed entities. Then, new entities that belong to the target entity class can be discovered if they share sufficient context words with the seeds' BoCW vectors. Aggregating the BoCWs of the newly discovered entities with the seeds' vectors increases the weights of the context words that are common among the discovered entities. This process is repeated over several iterations, where in each iteration the weights are updated by aggregating the similar BoCWs into a single vector. The gradual aggregation of similar entities' BoCW vectors is referred to as *vector expansion*.

More specifically, vector expansion for QNEC is defined as follows. Starting with a set of seeds for each target entity class, the expansion iterates through three main phases: *Instance Matching*, *Vector Aggregation*, and *BoCW Matching*. Each iteration of the expansion through these phases is called a *classification round*. Algorithm 1 summarises the vector expansion method.

The vector expansion iterates through these three phases until no new NEs are tagged or a maximum number of classification rounds is reached. The execution of each phase is performed as follows:

Instance Matching: in this phase (see Algorithm 1, lines 8 to 18), the list of extracted NEs is scanned to find those instances whose string exactly matches one of the seed entities. Every matched entity instance is tagged with the target entity class. In the first classification round, the list of NEs is scanned to find those matching a seed, whereas in later classification rounds, instances that are induced by the third phase (BoCW Matching) are used to find matching named entity instances.

Vector Aggregation: the corresponding BoCWs of the instances tagged in the previous phase are extracted and aggregated into a single vector called the *Class Vector* (see Algorithm 1, lines 19 to 24). Each component of the Class Vector corresponds to a context word together with a weight that is set to the *instance frequency* of that word. The instance frequency is defined as the number of instances tagged in the previous classification rounds that share that context word.

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 vector similarity threshold (see Algorithm 1, lines 25 to 31). All the instances whose BoCW similarity score to the Class Vector passes the threshold proceeds to the phase *Instance Matching* to be labelled with the target named entity class.

The similarity between each instance's BoCW and the iteratively expanded Class Vector is measured using *cosine similarity*. The *cosine* measure is commonly used in text mining and information retrieval specifically for sparse vectors [73], which is the case with the instances'

Algorithm 1 Vector Expansion for QNEC

```
Input: Repository, $E$, of pairs of named entity instance $e$ and corresponding BoCW
           vector $v$,
       Set of seeds $S$,
       Target entity class $\mathcal{C}$.
Output: Class Vector $V_\mathcal{C}$, set of labelled entity pairs $E_\mathcal{C}$
Variable: $I$ = Set of entity instances to be matched
          $B$ = Set of BoCW vectors to be aggregated
          $\mathfrak{t}$ = Vectors similarity threshold

1: initialise $I \leftarrow S$, $V_\mathcal{C} \leftarrow \varnothing$, $B \leftarrow \varnothing$
2: repeat
3:     INSTANCE MATCHING( )
4:     VECTOR AGGREGATION( )
5:     BOCW MATCHING( )
6: until $I = \varnothing$ or maximum number of classification rounds is reached
7: return $(V_\mathcal{C}, E_\mathcal{C})$

8:  INSTANCE MATCHING( )
9:     for all $i \in I$ do
10:        for all $< e, v > \in E$ do
11:            if $(e \equiv i)$ then
12:                $B = B \cup v$
13:                $E_\mathcal{C} = E_\mathcal{C} \cup < e, v >$
14:            end if
15:        end for
16:    end for
17:    $I \leftarrow \varnothing$
18: end

19:  VECTOR AGGREGATION( )
20:    for all $v \in B$ do
21:        aggregate $v$ to class vector $V_\mathcal{C}$
22:    end for
23:    $B \leftarrow \varnothing$
24: end

25:  BOCW MATCHING( )
26:    for all $< e, v > \in E$ do
27:        if ($sim(v,V_\mathcal{C}) \geq \mathfrak{t}$ and $< e, v > \notin E_\mathcal{C}$) then
28:            $I = I \cup e$
29:        end if
30:    end for
31: end
```

context word vectors. This measure is defined as follows:

$$Cosine(I, C) = \frac{V(I) \cdot V(C)}{\| V(I) \| \| V(C) \|}. \tag{4.1}$$

Here, $V(I)$ is the BoCW of an instance with each component being a context word, and the corresponding *term frequency* of that term in the snippets set, and $V(C)$ is the Class Vector with each component being the *instance frequency* of a context word, which is set to the frequency of instances labelled in previous classification rounds that share that word.

The theoretical justification behind using instance frequency rather than aggregating the term frequencies of the context words is as follows. 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 named entity 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 term frequency in the BoCW vector, but it is shared by many instances, i.e., has high instance frequency in the Class Vector, this is a strong indication that the instance belongs to the target class. Aggregating the term frequencies of the labelled instances rather than using the instance frequency of the labelled instances would lead to biasing the Class Vector towards those non-relevant context words that may appear with high term frequency in few instances' BoCW vectors.

4.3.3 Fertile Seeds Discovery

In order to run the expansion, a set of fertile seeds per target entity class should be selected. Given the vector expansion approach presented in this thesis, a fertile seed is a named entity instance whose BoCW expansion is more likely to lead to the classification of other named entities that belong to the same class.

As reported by Vyas et al. in [101], the performance of seed expansion approaches is highly affected by the choice of seeds. Vyas et al. reported that the quality of the set of seeds is based on the following attributes: (1) ambiguity of the seeds; (2) prototypicality of the seeds, where human editors select typical seed instances of a class that may not necessarily be fertile and may introduce noise during the expansion; and (3) coverage of the seed set chosen. Accordingly, a measure of seed fertility has to be defined that takes into consideration these attributes before seed selection.

In this work, a seed fertility is estimated through precision and recall scored by running the vector expansion using that seed over a sample from the target dataset. The precision scored by expanding a seed reveals whether it is ambiguous, and the recall scored by expanding the seed reveals its coverage. In other words, the lower the precision, the higher is the seed ambiguity, as its expansion leads to tagging many false positives (i.e., entities that do not belong to the target entity class). On the other hand, the higher the recall, i.e., the proportion of correctly classified entities, the higher is the coverage of the expansion using that seed.

Formally, given a *gold list* $\mathcal{G}_{\mathcal{C}}$, which consists of all the named entities that belong to a target named entity class $\mathcal{C}$, for every instance $i \in \mathcal{G}_{\mathcal{C}}$, the vector expansion is executed over the sample using i as a seed. A list of labelled entities is produced as a result of the expansion using seed i. Every labelled entity is automatically checked against the gold list and marked as true positive (tp) if it belongs to the gold list and false positive (fp) otherwise. Furthermore, every instance in the gold list that was not labeled during the expansion is marked as false negative (fn). Accordingly, the fertile seed discovery is executed as follows:

1. Measure the performance of the expansion for every instance i in the gold list using:

$$Precision(i) = \frac{tp_i}{tp_i + fp_i}.$$

$$Recall(i) = \frac{tp_i}{tp_i + fn_i}.$$

2. All instances whose $tp = 0$ are excluded, and the list of instances is sorted by precision in a descending order.
3. Then, the accumulative precision and recall are recorded as follows:

$$Precision_{acc}(i) = \frac{TP_{acc}(i)}{TP_{acc}(i) + FP_{acc}(i)}.$$

$$Recall_{acc}(i) = \frac{TP_{acc}(i)}{TP_{acc}(i) + FN_{acc}(i)}.$$

   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, because there might be an overlap between the set of classified instances via the expansion of the different seeds.
4. For every instance i, the change of accumulative precision and recall is measured 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. Finally, given tolerable minimum performance scores precision $P_{\min}$ and recall $R_{\min}$, and change in accumulated precision $\Delta P_{\min}$, instance i is a fertile seed if it satisfies the following conditions:

   (i) $\Delta Recall_{acc}(i) > 0$;

   (ii) $\Delta Precision_{acc}(i) \geq \Delta P_{\min}$;

   (iii) $Precision(i) \geq P_{min}$;

   (iv) $Recall(i) \geq R_{min}$.

Once the initial set of fertile seeds is created, any seed whose list of classified instances was already detected by other fertile seeds is excluded. This results in a ranked list of fertile seed instances ordered by decreasing precision. Accordingly, this list can be used to select the fertile seeds, which can be used to initiate the vector expansion algorithm.

4.3.4 Expansion Modes

In the first classification round, the vector aggregation phase of the vector expansion algorithm can be executed in either of the following ways:

Single Class Vector (SCV): when the expansion starts, the BoCWs of the instances that match the seeds are aggregated to create a single class vector and then the expansion proceeds. Throughout the classification rounds of the expansion, a single class vector is maintained until the expansion terminates.

Multiple Class Vectors (MCV): when the expansion starts, each seed is used to create a separate sub-class vector. In other words, the BoCW of each seed is expanded separately as new instances are tagged. In this mode of expansion, each sub-class vector can be executed either simultaneously or sequentially, yet independently, for every seed.

Figure 4.1 presents an illustration of both of the expansion modes SCV and MCV and how the performance of each may vary.

The motivation for these two modes of expansion stems from the observation made by Vyas et al. in [101], where they state that the choice of seeds can bias the expansion towards a specific subclass in a *concept's semantics space* of a target named entity class. Although NE instances belong to a single entity class, such as Person, Company, or Location, each class has instances that fall into different subclasses. For example, looking at instances in the

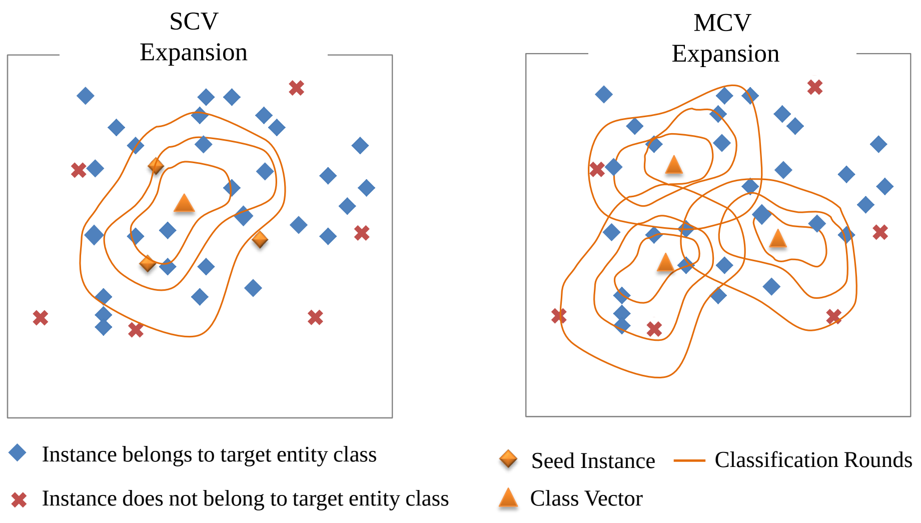

SCV
Expansion
MCV
Expansion
Instance belongs to target entity class
Instance does not belong to target entity class
Seed Instance
Classification Rounds
Class Vector

Figure 4.1: Illustration of expansion modes

entity class Person, these may belong to the subclasses Actors, Singers, Politicians, Athletes, or Historical figures, and would have different context words surrounding them in the snippets sets, and therefore different BoCWs. Assuming that the chosen seeds cover the semantic space of the target entity class, in MCV each seed is expanded to tag those instances that fall in the corresponding semantic subclass, thus resulting in a higher recall; however, this does not ensure a higher precision because each seed's expansion might introduce false positives, i.e., misclassified instances. On the other hand, in SCV, if the frequency of instances that belong to a particular subclass is higher than others within the target dataset, the expansion would be biased towards that particular subclass because the weight of its context words is higher, thus resulting in a lower recall. However, the fact that the Class Vector starts by tagging many instances that share those context words in the first classification rounds ensures accurate tagging, and thereby results in higher precision. In order to validate the effect of each expansion mode, expansion performance is assessed via the two expansion modes, as presented in Section 4.5.

4.4 Experimental Settings

In this section, the experiment setting utilised to evaluate the Vector Expansion algorithm is presented. Table 4.2 summarises the setting employed for evaluation, which is described in detail in the following subsections.

Table 4.2: Summary of experiment settings for QNEC evaluation.

Query Log	
	Running the QNER approach presented in Chapter 3 over unique search queries resulted in a total of $1,925,074$ unique candidate named entities.
Vector Expansion	
	BoCW
	A context word should fall within a window size of k words around a NE mention in a snippet, where $k = 3$.
	Fertile seed discovery parameters (see Subsection 4.3.3)
	$\Delta P_{min} = -0.01$
	$P_{min} = 0.70$
	$R_{min} = 0.01$
	Vector similarity threshold (Algorithm 1)
	$\mathsf{t} = 0.40$
Target Entity Classes	
	1. Person. / 4. Organisation.
	2. Location. / 5. Software.
	3. Educational Organisation. / 6. Film.
Experiments	
	Performance Assessment presented in Section 4.5.
	Comparative assessment to a baseline approach presented in Section 4.6.

Dataset

From the MSN query log, approximately 6.2 million candidate entities were extracted using the QNER approach presented in Chapter 3. The number of entity-centric queries, i.e., search queries that include at least one named entity, is 4.6 million unique queries. Table 4.3 presents a breakdown of the percentages of entity-centric query types defined in Section 4.2.

Table 4.3: Statistics of candidate named entities extracted from MSN query log and the corresponding percentages of the query types in which the entities occurred

	Full Log	FQ	VFQ	UFQ
Query	4,624,501	30.70%	37.67%	31.63%
Entity	6,299,933	22.54%	27.65%	49.81%
Unique Entity	1,925,074	60.65%	26.79%	33.71%

The percentage of unique candidate named entities that appear as FQs, i.e., queries that consists of nothing but the named entity string, is 60.65%. On the other hand, entities that appear in VFQs and UFQs represent 26.79%, and 33.71% of the set of unique extracted entities, respectively. Looking at the percentages of unique entities, there is an overlap of approximately 21.15%, where entities may appear in two or all types of queries. When considering only those named entities that appear in FQs, it was found that 47% of these entities do not occur in VFQ or UFQ, and therefore there are no contextual clues in the query that can be used to identify these named entities. This leads to the conclusion that previous approaches that relied on contextual clues extracted from the query to identify the named entities, such as [44, 80], are not applicable to a considerable proportion of search queries.

BoCW Extraction

For every NE instance, a BoCW vector is extracted from the snippets of the web pages related to the query in which the NE occurs. As described in Subsection 4.3.1, the BoCW contains nouns and verbs surrounding the named entity mention within a sentence. Moreover, for a context word to be included in the BoCW vector, it should appear within a window size of k words surrounding the named entity occurrence after excluding stop words and other words whose part-of-speech is not noun or verb. The window size of words is set to three words (i.e., $k = 3$).

Gold Lists of Named Entities

In order to evaluate the performance of the Vector Expansion approach presented herein, each entity classified through the expansion should be checked as to whether it is annotated with the correct class label in order to measure the scored precision. Moreover, in the dataset, the set of all named entities that belong to the target named entity class should be known beforehand in order to measure the recall scores that can be achieved by the algorithm. Therefore, a *gold list* of named entities for each target entity class need to be created to measure precision and recall. Because of the prohibitive cost of annotating a dataset manually, each extracted candidate named entity is checked with DBpedia to create the gold lists. Furthermore, only those named entities that satisfy the following conditions are considered:

1. The extracted named entity string should exactly match a DBpedia result label.
2. The extracted named entity should be classified by the DBpedia ontology.
3. The extracted named entity should belong to a single DBpedia class.

As presented in Chapter 3, only 34.3% of the extracted named entities were found in DBpedia with an exactly matching result label (refer to Table 3.5, Chapter 3). Moreover, when excluding all named entities that may fall into more than one entity class in the DBpedia ontology, i.e., ambiguous named entities, a total of 104,000 unique entities satisfy the previous conditions. These named entities appear in a total of 886,439 unique search queries. Although a large number of named entities were excluded, the inclusion of these entities may affect the evaluation. This specifically occurs when a NE instance appears in the gold list of more than one target named entity class. Therefore, this dataset is used to estimate the performance in terms of precision and recall.

Target Named Entity Classes

The classes used to evaluate QNEC through the vector expansion approach are:

1. 'Person', which represents 27.26% of the candidate NEs found in DBpedia, and it includes instances whose DBpedia ontology classes are Artist, Actor, Person, Athlete, and Journalist.
2. 'Location', which represents 19.90% of NEs found in DBpedia and it includes Administrative Region, City, Place, Village, and Location.
3. 'Organisation', which represents 12.80% of the NEs.
4. 'Educational Organisation', which represents 6.86% of the NEs and it refers to names of schools, colleges, and universities.
5. 'Film', which represents 2.01% of the NEs.
6. 'Software', which represents 1.24% of the NEs and it includes names of video games and software programs.

These named entity classes were chosen because they were the most frequent entity classes observed over the dataset. Moreover, the target entity classes represent both coarse-grained classes (Location, Person, and Organisation) and fine grained classes (Educational Organisation, Film, and Software).

Experiments

In order to assess the performance of the expansion proposed in this thesis, the following two experiments were performed:

Performance Assessment: the performance of the vector expansion approach is assessed for each identified type of entity-centric queries, i.e., FQs, VFQs, and UFQs, described in Section 4.2.

Comparative Assessment: using the named entity instances extracted from FQs and VFQs, the expansion is evaluated to show how the performance varies when the context words are extracted from the snippets or from the query string. Furthermore, Paça's approach in [80] was implemented to compare the vector expansion approach proposed in this thesis to an expansion baseline approach of named entity recognition and classification within the context of web search queries.

4.5 Performance Assessment

The set of unique candidate named entities was divided into a training and a test dataset. The training dataset is used to find potential fertile seeds and to set the vector similarity threshold, whereas the test dataset is used for assessment. Table 4.4 lists the frequency of named entities instances and corresponding queries in both the training and test datasets for each query type. The training and test datasets are of equal size, i.e., both represent 50% of the DBpedia validated named entities.

Table 4.4: Performance assessment dataset query and entity instances statistics

Query type	Full Dataset		Training Dataset		Test Dataset	
	NE #	query #	NE #	query #	NE #	query #
FQ	79,318	79,318	39,571	39,571	39,747	39,747
VFQ	41,385	292,509	20,611	150,476	20,774	149,031
UFQ	50,005	506,847	25,023	257,209	24,982	249,638
Total	104,000	886,439	52,000	447,618	52,000	438,416

4.5.1 Fertile Seed Selection

In order to find the fertile seeds using the approach presented in Section 4.3.3, an initial vector similarity threshold needs to be set. Therefore, a pilot experiment was conducted, where the expansion was executed using random seeds selected from the gold lists of the training dataset. The experiment revealed that the expansion has an acceptable performance when the similarity threshold is set to 0.4. This initial threshold was common among all the target named entity classes.

Over the training dataset, the vector expansion is executed once for each instance of the gold list of the target entity class, as described in Section 4.3.3. This resulted in a list of instances for each target entity class ranked by decreasing precision. Table 4.5 presents the percentage of the fertile seeds selected based on the following conditions:

- $\Delta Recall_{acc}(i) > 0$ (i.e., the addition of the seed improved the overall coverage);
- $\Delta Precision_{acc}(i) \geq -0.01$ (i.e., the addition of the seed did not decrease the overall precision by more than 1%);
- $Precision(i) \geq 0.70$ (i.e., the precision achieved by the expansion using that seed should be at least 70%); and
- $Recall(i) \geq 0.01$. (i.e., the recall achieved by the expansion using that seed should be at least 1%).

Table 4.5: Percentages of fertile seeds found in the gold list of each target entity class

Entity Class	Query Type	Gold list	Fertile Seeds	%
Person	FQ	11,902	255	2.14%
	VFQ	5,111	88	1.72%
	UFQ	5,927	25	0.42%
Location	FQ	6,603	156	2.36%
	VFQ	4,688	201	4.29%
	UFQ	6,444	260	4.03%
Edu Org	FQ	3,249	130	4.00%
	VFQ	669	4	0.60%
	UFQ	1,747	73	4.18%
Org	FQ	5,969	52	2.14%
	VFQ	2,958	29	1.72%
	UFQ	3,242	17	0.42%
Film	FQ	874	18	2.36%
	VFQ	597	7	4.29%
	UFQ	608	1	4.03%
Software	FQ	509	12	4.00%
	VFQ	392	11	0.60%
	UFQ	331	12	4.18%

4.5.2 Setting Vector Similarity Threshold

In order to validate the vector similarity threshold, each fertile seed discovered for each target entity class is used to initiate the vector expansion over the training dataset using ten different settings: 0.1, 0.2, ..., 1.0. Then, the expansion performance is measured using the final accumulative rates, i.e., TP_{acc}, FP_{acc}, and FN_{acc}, defined in Section 4.3.3. These rates were used to produce the Receiver Operator Characteristic (ROC) curves presented in Figure 4.2, and the Precision-Recall (PR) curves shown in Figure 4.3.

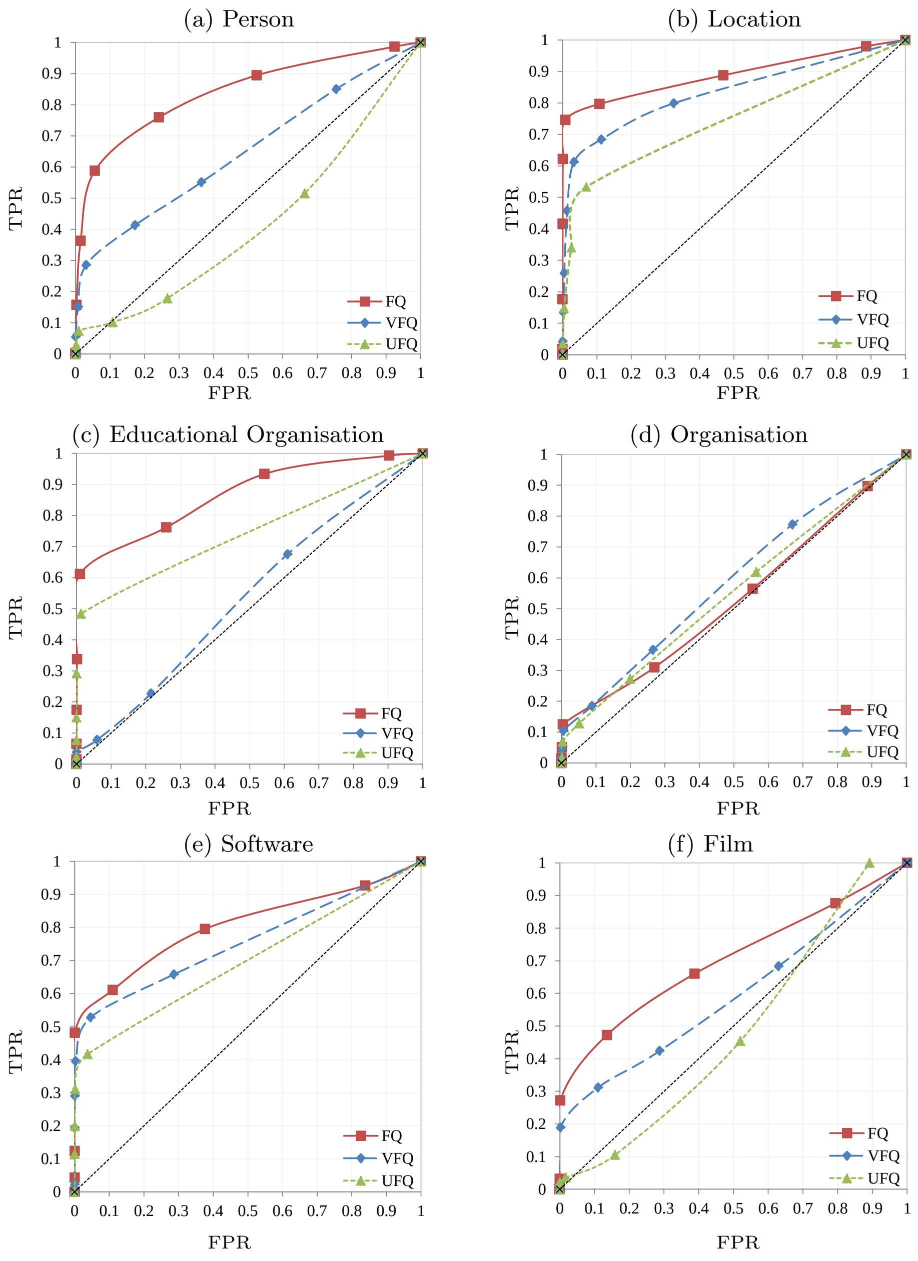


Figure 4.2: ROC curves generated by running the expansion using ten different vector similarity measures over the training dataset

Analysing the induced ROC curves, the area under the curve of the expansion over FQ is the largest when compared to the remaining query types over all the target entity classes, with the exception of the class Organisation, where the curves for all query types are almost the same. Furthermore, the expansion performance over UFQ instances is lower than that scored over VFQs.

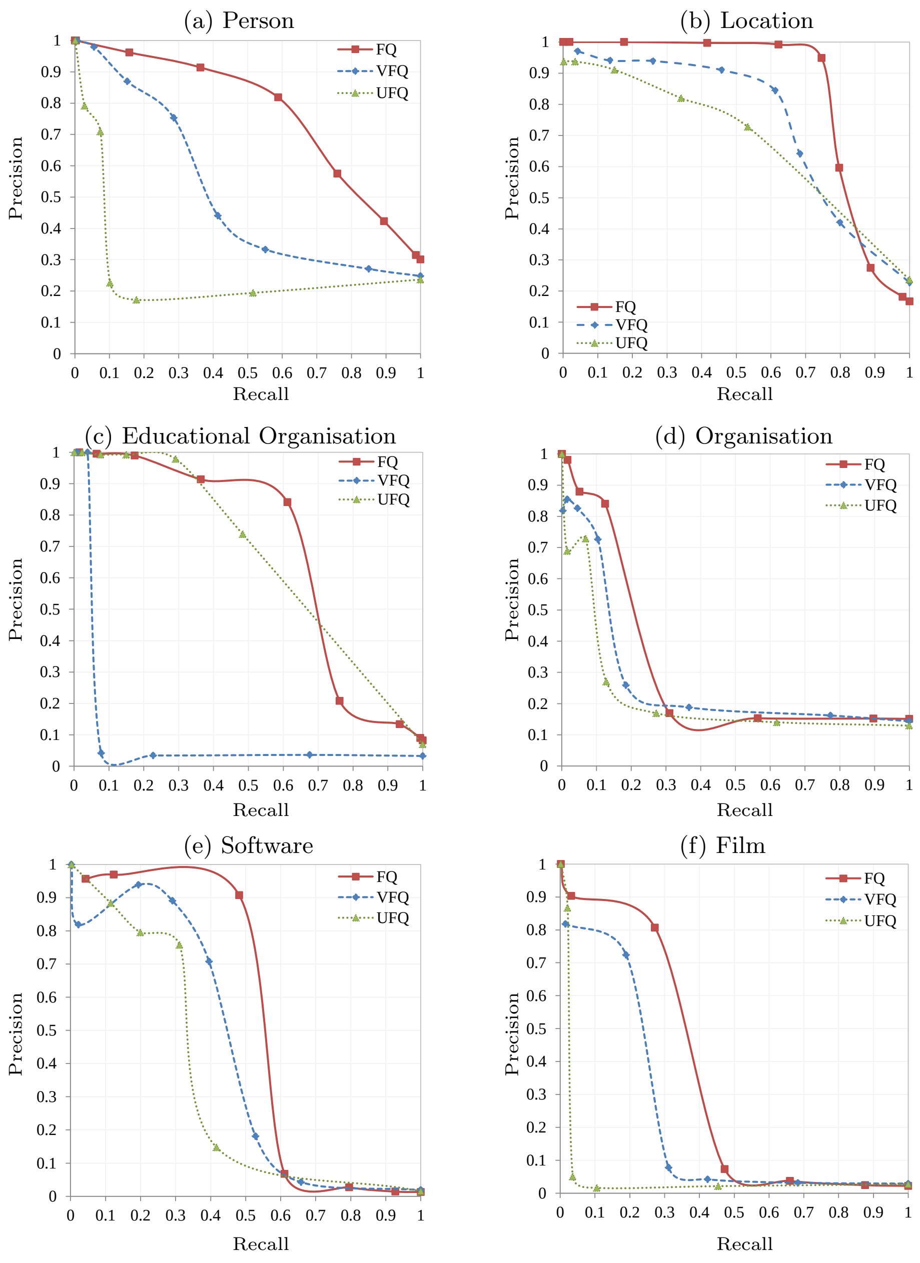


Figure 4.3: Precision-Recall curves generated by running the expansion using ten different vector similarity measures over the training dataset

In the ROC graphs, the true positives rate (TPR) is plotted against the false positives rate (FPR), where TPR is given by $tp/(tp + fn)$ (i.e., recall) and FPR is given by $fp/(fp + tn)$. On the other hand, PR curves plot precision against the recall [72, 73]. In a ROC graph, the further the curve is to the top left corner of the graph, the better is the performance. For PR curves, it is the opposite, i.e., the performance is best when the curve climbs toward the top right corner of the graph. Plotting the expansion performance using PR and ROC curves helps in find the threshold where the expansion performs best, and also visualise the overall performance using different settings of the vector similarity threshold.

Although ROC graphs are often used in machine learning for evaluating and comparing the algorithms' performance, PR graphs are more informative of the performance of the vector expansion for the QNEC task because of the skewed class distribution. Davis and Goadrich [30], and Fawcett [39] discussed how the PR and ROC curves induced by algorithms performance varies given the class distribution of a dataset. For a target entity class, if the number of all negative instances (i.e., instances that do not belong to the target class) greatly exceeds the number of positive instances, FPR is not affected greatly as the number of false positives increases. On the other hand, because the precision measures the proportion of the true positive instances against all the instances tagged by the expansion (i.e the positive instances), it will be affected greatly as the number of false positives increases.

Therefore, in this work, the threshold was set based on the PR curves produced for each target entity class. The expansion has an acceptable performance when the threshold is set between 0.3 and 0.5, with a minimum precision of 70%. Consequently, a common threshold of 0.4 was chosen to execute the assessment experiment presented in the following sections.

4.5.3 Performance Assessment Using Precision & Recall

Because the fertile seeds were selected from the training dataset and the vector similarity threshold is also set using the training dataset, the expansion approach is evaluated over the test dataset.

In addition to the fertile seeds discovered, a Class Vector ($V_{\mathcal{C}}$) is created for each seed by expanding its BoCW over the training dataset. For the remainder of this chapter, $v(s)$ is used to denote the BoCW vector associated with a seed s selected from the training dataset and $V_{\mathcal{C}}(s)$ is used to denote the Class Vector created by expanding the BoCW of seed s over the training dataset. In addition to the discovered seed, $V_{\mathcal{C}}(s)$ can be utilised to classify entities in the test dataset. Therefore, the Vector Expansion algorithm is modified slightly so that the expansion starts with the Vector Aggregation phase. This is achieved using the following two Class Vector initialisation settings:

Setting A: the Class Vector is initialised with $v(s)$.

Setting B the Class Vector is initialised with $V_{\mathcal{C}}(s)$.

Using the initialisation settings A and B, the expansion is assessed for each of the two expansion modes SCV and MCV (see Subsection 4.3.4) as follows:

SCV-A: the $v(s)$ of all the seeds are first aggregated to a single Class Vector, and then the expansion proceeds.

SCV-B: the $V_{\mathcal{C}}(s)$ of the seeds are first aggregated to a single Class Vector, and then the expansion proceeds. In the case where the same NE instance in the training dataset is tagged by the expansion of more than one fertile seed, the weights of the context words constituting the BoCW of that instance are counted only once in the aggregation. In light of this, using this setting, the Class Vector can be defined as the aggregation of the BoCWs of all unique instances tagged by the expansion of each fertile seed's BoCW over the training dataset.

MCV-A: for each discovered fertile seed, the vector expansion approach is executed. During each expansion, the Class Vector is initialised with the $v(s)$ of that seed, and then the expansion proceeds.

MCV-B: for each discovered fertile seed, the expansion is executed by initialising the Class Vector with the $V_{\mathcal{C}}(s)$ of that seed, and then the expansion proceeds. In MCV, the expansion is executed for each fertile seed independently, and the performance is reported on the final set of distinct NE extractions.

For each target named entity class, query type, and mode of expansion (using both Class Vector initialisation settings), the expansion performance is measured using precision and recall. This is performed automatically with the aid of the gold list of each target entity class found in the test dataset. Accordingly, the true positives, false positives, and false negatives are recorded automatically by scanning the set of unique extractions per setting to obtain the precision and recall scores presented in Table 4.6.

Table 4.6: Precision (P) and Recall (R) scored by the vector expansion for each target entity class

Entity	Query	SCV-A		SCV-B		MCV-A		MCV-B		Micro		Macro	
Class	Type	P	R	P	R	P	R	P	R	P	R	P	R
Person	FQ	0.9994	0.1461	0.9902	0.1178	0.5889	0.5698	0.8227	0.3961	0.7174	0.3075	0.8503	0.3075
	VFQ	0.8728	0.1250	0.6667	0.0004	0.6292	0.3096	0.7823	0.1504	0.7069	0.1463	0.7377	0.1463
	UFQ	0.8701	0.0344	-	-	0.6348	0.0705	0.4338	0.0163	0.6442	0.0403	0.6462	0.0404
Location	FQ	0.9967	0.4154	0.9981	0.4061	0.9507	0.7277	0.9972	0.4414	0.9798	0.4976	0.9857	0.4976
	VFQ	0.9216	0.0203	0.8889	0.0052	0.8501	0.5724	0.8605	0.0320	0.8531	0.1575	0.8803	0.1575
	UFQ	0.8430	0.0158	1.0000	0.0005	0.6542	0.4892	0.7054	0.0735	0.6646	0.1448	0.8006	0.1448
Org	FQ	0.9683	0.0502	0.9624	0.0421	0.3186	0.1473	0.9056	0.0852	0.4982	0.0812	0.7887	0.0812
	VFQ	0.6776	0.0417	0.6667	0.0007	0.4746	0.1037	0.4608	0.0316	0.5082	0.0444	0.5699	0.0444
	UFQ	0.5531	0.0309	-	-	0.3745	0.0804	0.2719	0.0281	0.3728	0.0348	0.3998	0.0348
Edu Org	FQ	0.9848	0.3444	0.9882	0.3400	0.3566	0.5471	0.9873	0.4370	0.6247	0.4171	0.8292	0.4171
	VFQ	1.0000	0.0044	1.0000	0.0015	0.2636	0.0422	0.3137	0.0233	0.2970	0.0178	0.6443	0.0178
	UFQ	0.8450	0.3115	-	-	0.5113	0.4517	0.4131	0.2044	0.5462	0.2419	0.4424	0.2419
Film	FQ	0.9478	0.1173	0.9658	0.1216	0.1433	0.2228	0.8878	0.1959	0.3247	0.1644	0.7362	0.1644
	VFQ	0.7200	0.0550	-	-	0.1297	0.1804	0.2727	0.0505	0.1730	0.0953	0.3741	0.0953
	UFQ	0.0212	0.0125	0.0212	0.0125	0.0212	0.0125	0.0212	0.0125	0.0212	0.0125	0.0212	0.0125
Software	FQ	0.9803	0.3098	0.9745	0.3181	0.8849	0.4636	0.9645	0.3389	0.9425	0.3576	0.9511	0.3576
	VFQ	0.8611	0.3444	0.5067	0.1056	0.8849	0.4636	0.9645	0.3389	0.8563	0.3258	0.8043	0.3131
	UFQ	0.7802	0.2132	-	-	0.2195	0.2913	0.8438	0.1622	0.3719	0.1667	0.6145	0.2222
Micro	FQ	0.9926	0.2100	0.9913	0.1944	0.5637	0.5020	0.8974	0.3388				
	VFQ	0.8403	0.0710	0.6147	0.0049	0.6500	0.3377	0.7197	0.0844				
	UFQ	0.6212	0.0556	0.0260	0.0009	0.5538	0.2585	0.4265	0.0587				
Macro	FQ	0.9795	0.2305	0.9799	0.2243	0.5405	0.4464	0.9275	0.3157				
	VFQ	0.8422	0.0985	0.7458	0.0227	0.5387	0.2786	0.6091	0.1044				
	UFQ	0.6521	0.1031	0.3404	0.0032	0.4026	0.2326	0.4482	0.0828				

How Query Type Affects Vector Expansion

Analysing the average precision and recall over all the expansion variants in Table 4.6 shows that vector expansion achieves a macro-average precision that varies between a minimum of 73% (Film class) and a maximum of 98% (Location class) over NEs extracted from FQs, as well as a macro-average recall that varies between a minimum of 8.1% (Organisation class) and a maximum of 49.7% (Location class).

Comparing the performance to that achieved when executing the expansion over entities extracted from VFQs, the macro-average precision decreases by 10% and 12% over the classes Location and Person, respectively, whereas it drops by 20%-37% over the remaining entity classes. Furthermore, the macro-average recall decreases by at least 4% over the class Organisation, and at most by 40% over the class Educational Organisation.

When executing the approach over named entities extracted from UFQs, the performance in terms of both macro-average precision and recall is lower than that achieved over VFQ named entities, except for instances that belong to the class Educational Organisation, where the macro-averaged recall is higher than VFQ entities by 23%. Looking closely at the results, the NE instances that belong to the Educational Organisation class are often paired with the name of the city in which they reside, and therefore, the snippets set would still have those context words that describe a university, school, or college.

When analysing the results by looking at the macro-precision and macro-recall averaged over all target entity classes, the performance of the approach over instances extracted from VFQs and UFQs is outperformed by the expansion performance when executed to classify named entities that appear in the form of FQs. The macro-averaged class precision achieved over FQ named entities is greater than 90% among all the expansion settings, except for the case where the setting MCV-A is used, where it decreases to 54%. Moreover, the average class recall of FQ named entities is at least 10% higher than VFQ named entities, as well as 10% to 24% higher than named entities in UFQs.

The performance scores of the vector expansion approach vary given the query type from which the named entities are extracted. For example, the candidate named entity ‘Nicolas Cage’ appeared in an FQ, VFQ, and UFQ. The BoCW vector extracted for each instance of this named entity is presented in Table 4.7.

In the FQ, the BoCW contains words with high term frequencies, such as ‘actor’, ‘producer’, and ‘filmography’, that are specifically shared by Actors, in addition to words like ‘born’, ‘age’, and ‘biography’ that are generally shared by entities in the class Person. On the other hand,

Table 4.7: The BoCW components vary given the query type in which the named entity 'Nicolas Cage' appeared

Query	Type	BoCW
'Nicolas Cage'	FQ	((actor,7), (producer,2), (video,2), (com,2), (make,1), (view,1), (filmography,1), (nephew,1), (bio,1), (photos,1), (picture,1), (nepotism,1), (adaptation,1), (director,1), (career,1), (losing,1), (born,1),...)
'Nicolas Cage pics'	VFQ	((pictures,9), (news,4), (view,3), (pics,3), (photos,3), (sgy,3), (gallery,3), (wallpapers,2), (jpg,2), (videos,2), (picsearch,1), (profile,1), (find,1), (com,1), (mouse,1), ...)
'Next Nicolas Cage'	UFQ	((jessica,3), (biel,3), (weirdness,2), (film,1), (review,1), (starring,1), (fiction,1), (moore,1), (movie,1)...)

when the vector created from the snippets related to the VFQ of the entity, words such as 'pictures', 'view', 'pics', and 'gallery' are the dominating ones with higher term frequency unlike the context words extracted for the same entity from the snippets set of the FQ. Finally, looking at the UFQ that consists of the two entities 'Nicolas Cage' and 'Next', the extracted BoCW contains words related to the movie 'Next', such as 'review', 'fiction', 'movie', and words such as names of other actors starring in that movie.

Table 4.8 presents the final top ten context words from the Class Vectors created during the SCV-A expansion. When comparing the lists of context words, it is evident that those extracted from FQ snippets are more descriptive of the entity class. For instance, words such as 'born', 'biography', and 'actress' give a clear indication that the entity belongs to the class Person. Such is also the case for the classes Film and Organisation, where the most frequent context words shared by the NE instances are 'film'and 'inc' respectively.

In addition, when looking at the weights (i.e., instance frequency) of the context words of the class Educational Organisation under VFQ, the weights are very low when compared to the other lists. This results in a very low recall with 100% precision (see Table 4.6 for the class Educational Organisation under the SCV-A setting). On the other hand, in the case of UFQ under the class Film, the weights are the highest for the context words 'state', 'located', and 'city', thus showing a semantic drift during the expansion, and therefore, drops of precision to 2% and recall to 1%.

Table 4.8: Top ten context words in the Class Vector created at the end of the expansion using SCV-A

Person

FQ		VFQ		UFQ	
born	1740	com	3059	com	1202
photos	1626	pictures	2931	has	1103
news	1465	photos	2717	find	920
biography	1214	find	2692	photos	757
american	965	has	2683	news	747
actress	931	news	2140	get	737
pictures	897	get	1946	information	711
view	747	have	1830	pictures	672
com	716	view	1790	video	662
actor	637	videos	1696	state	557

Location

FQ		VFQ		UFQ	
news	2277	state	9164	state	17421
city	2021	find	6261	city	11923
county	1987	information	5127	located	11591
sports	1987	city	3680	find	10806
everything	1976	search	2789	county	10107
community	1688	com	2761	area	8491
forum	1533	department	2672	university	7678
registered	1530	including	2355	information	7632
posts	1528	home	2284	tx	7132
users	1524	guide	2255	texas	6704

Organisation

FQ		VFQ		UFQ	
inc	279	find	5490	tx	7904
company	266	state	4783	city	7799
profile	189	com	3952	texas	6352
information	141	information	3404	state	5319
get	116	online	2753	find	5215
view	101	get	2548	located	5099
type	92	city	2237	area	4519
compare	92	car	2184	reviews	3898
chart	90	used	2105	county	3746
linkedin	78	buy	1970	information	3361

Educational Organisation

FQ		VFQ		UFQ	
located	1209	official	15	state	17866
find	1150	site	12	located	10480
student	1095	partner	12	city	10347
test	1094	school	10	find	9875
scores	1093	wolverines	8	county	9386
alumni	614	team	8	information	7235
members	446	football	7	university	7092
schools	272	home	7	area	6148
students	234	department	7	department	6041
district	189	college	7	has	5824

Film

FQ		VFQ		UFQ	
film	118	get	320	state	18559
com	81	com	315	located	14188
added	81	movie	290	find	14060
queue	80	find	248	city	13412
amazon	75	news	247	county	11163
movie	68	offers	241	area	10555
comedy	60	online	232	information	9132
american	58	movies	194	university	8584
trailer	52	music	185	tx	7703
reviews	48	email	168	center	7220

Software

FQ		VFQ		UFQ	
ign	157	season	739	profile	2236
resource	156	find	601	state	1477
screenshots	155	com	598	codes	1335
trailers	154	game	589	com	1319
game	125	download	503	leading	1299
video	71	windows	468	dc	1289
gamefaqs	69	news	454	entertainment	1143
playstation	49	cheats	446	layouts	1136
games	33	collection	441	graphics	962
com	31	show	440	destination	952

How Expansion Mode Affects Performance

Examining the results with respect to the expansion mode, when the Class Vector is first initialised with either the aggregated fertile seeds' BoCWs (SCV-A), or initialised with the Class Vectors created by expanding the fertile seeds over the training dataset (SCV-B and MCV-B), the macro precision averaged over the entity classes is higher than 90%, with a macro-average recall between 22% and 31%. On the other hand, when using the BoCWs of the seeds solely to initialise the Class Vector (MCV-A), the precision decreases to approximately 54% over FQ named entities, whereas the macro-average recall increases to approximately 44%.

Analysing the Class Vectors created during the SCV expansion mode, it was found that the common context words among instances that belong to a single named entity class are higher than others and results in an accurate tagging of named entity instances at the beginning of the expansion, thus resulting in higher precision. On the other hand, this biases the expansion towards a specific subclass in the semantic space of the target named entity class.

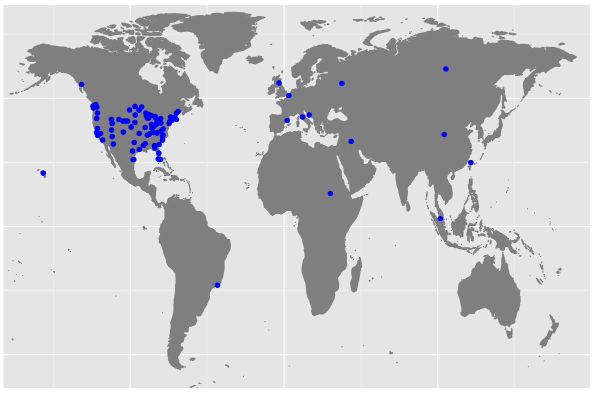
To illustrate the affect of each expansion mode, consider initialising the expansion with the three seeds 'Seattle', 'Iraq', and 'Edinburgh' using SCV and MCV expansion modes. Figure 4.4 plots the first 100 extractions when executing SCV and MCV expansion over a world map, along with the first ten instances extracted during the expansion on the right side of the figure[1]. It is evident that the SCV (Figure 4.4 (a)) mode biased the tagging toward location names in the USA. However, the MCV expansion mode (Figure 4.4 (b)) diversified the tagging to include names of countries, and cities other than location names in the USA.

MCV expansion is versatile with respect to tagging instances within different sub-semantic spaces within the target entity class. However, this is only observed during the first few classification rounds of the expansion. As the expansion proceeds and instances of a dominating semantic space are tagged, the Class Vector eventually becomes biased toward that semantic space. This is because the dataset is biased towards that specific semantic space within the target class. For the case of Location instances, a majority of the queries in the log include mentions of USA states, and therefore the instances tagged are USA states.

When using the BoCW of the fertile seeds ($v(s)$) in combination with the MCV expansion mode (MCV-A), the likelihood of semantic drift is higher. Accordingly, although recall scores are higher, the precision scores are between 14% and 35%, with exception of the classes Location and Software (see Table 4.6 for FQs). However, using the initialisation setting B, where the $V_C(s)$ of the seeds is utilised, the recall scores are improved when compared with SCV. For

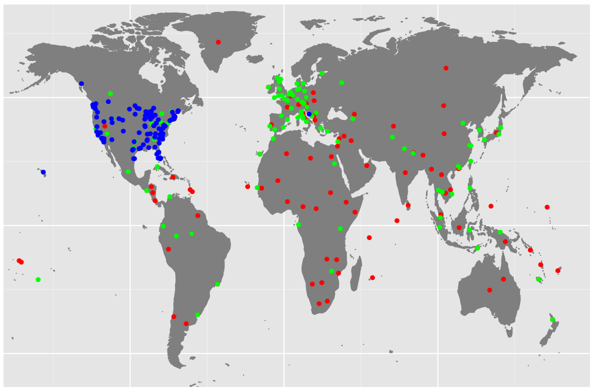
[1]Created using the R tool with the aid of the ggmap package

(a) SCV Expansion



Visalia California
Rogers Arkansas
San Antonio
Salem Massachusetts
Louisville Kentucky
Indianapolis
Peabody Massachusetts
Orange California
China
Fort Worth Texas
Bruges

(b) MCV Expansion



Seattle ●	Iraq ●	Edinburgh ●
Salem Massachusetts	Sudan	Aberdeen
Natchez Mississippi	Somalia	Havana
Fort Worth Texas	Russia	Bruges
Split Croatia	Haiti	Southampton
St. Ignace Michigan	Syria	Barcelona
San Antonio	Libya	Montevideo
Redmond Oregon	Zambia	Kuala Lumpur
Ocean shores Washington	Indonesia	Lisbon
Charlottesville Virginia	Cameroon	Philadelphia
Cambridge Massachusetts	Senegal	Copenhagen

Figure 4.4: The expansion using seeds Seattle, Iraq, and Edinburgh over the test.

instance, for the class Person, the recall improved by 25% for FQs and 15% for VFQs over instances that belong to the Person class while maintaining a precision greater than 78% (see Table 4.6 under MCV-B).

The tradeoff between SCV and MCV is that the first ensures high precision extractions at the expense of lower recall, whereas the latter detects many candidate named entities, thus resulting in a higher recall, but at the expense of lower precision.

Improving SCV Expansion Mode Coverage

As presented previously, although SCV expansion mode results in higher precision, the recall scores achieved are lower than those scored with MCV expansion mode. When using $V_{\mathcal{C}}(s)$ to initialise the Class Vector before the expansion (SCV-B), it is possible that the resulting Class Vector is too big to be compared with any other instances' BoCWs. This leads to either tagging very few instances with 100% precision (e.g., Location NEs that appear in UFQs), or not tagging any candidate NE instances, i.e., zero precision and recall (e.g., Person NEs that appear in UFQs).

In order to improve the recall of the SCV expansion mode, the vector similarity threshold is gradually decreased after each expansion round is terminated. Thus, starting with the initial threshold 0.4, the expansion proceeds until there are no named entity instances whose BoCW similarity to the Class Vector passes the threshold. Then, the threshold is decreased by 0.01 to become 0.39, and the expansion process is continued to find new entities whose BoCW similarity to the Class Vector passes the new threshold. This process is repeated until the threshold reaches a minimum of 0.2. For every threshold decrement, the performance is measured using precision and recall. This experiment is performed for each target named entity class and each query type. Figure 4.5 shows the PR curves generated for each target entity class and query type.

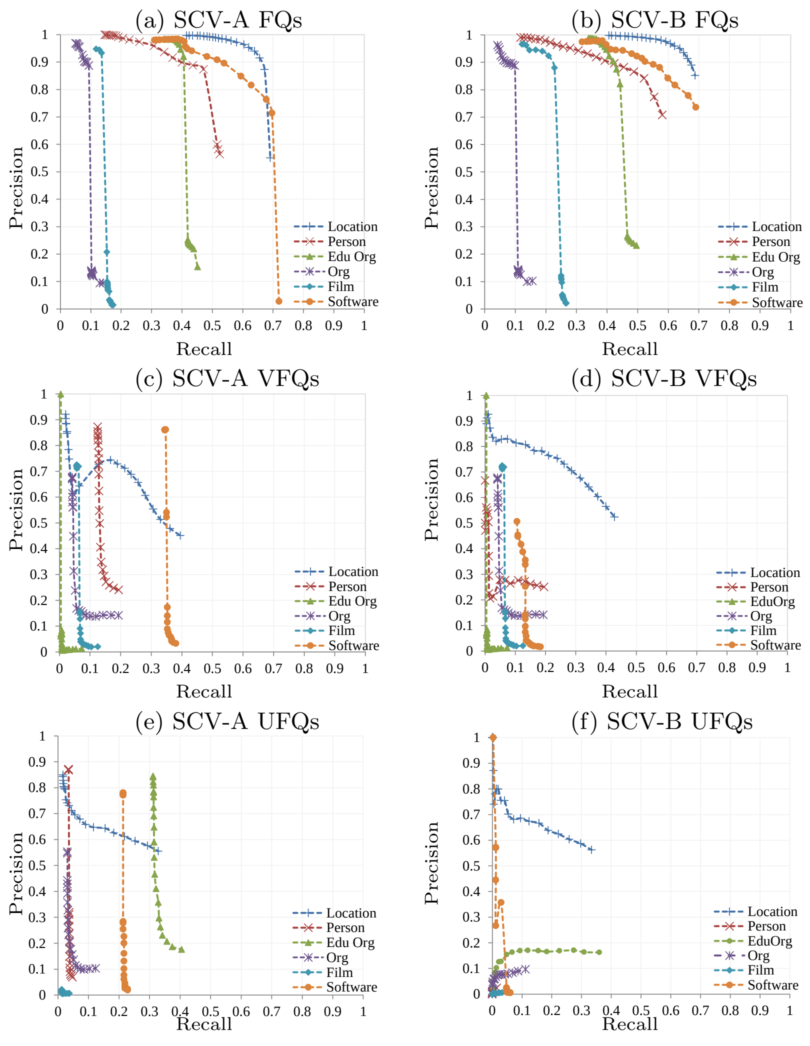


Figure 4.5: PR curves generated by gradually decreasing threshold as the expansion is executed

Table 4.9: The highest Recall (R) scored during gradual threshold decrement before Precision (P) drops to lower than 70%

Entity	Query	SCV-A			SCV-A		
Class	Type	Threshold	P	R	Threshold	P	R
Person	FQ	0.23	0.8737	0.4711	0.20	0.7076	0.5805
	VFQ	0.32	0.7225	0.1287	0.4	0.6667	0.0004
	UFQ	0.34	0.8675	0.0347	0.4	-	-
Location	FQ	0.21	0.8732	0.6723	0.20	0.8521	0.6878
	VFQ	0.27	0.7120	0.2137	0.25	0.7052	0.2859
	UFQ	0.3	0.7122	0.0445	0.29	0.7016	0.0540
Edu Org	FQ	0.28	0.9205	0.4054	0.26	0.8212	0.4426
	VFQ	0.4	1.0000	0.0044	0.4	1.0000	0.0044
	UFQ	0.33	0.7238	0.3133	0.4	-	-
Org	FQ	0.30	0.8877	0.0936	0.29	0.8877	0.0989
	VFQ	0.4	0.6776	0.0417	0.4	0.6776	0.0417
	UFQ	0.4	0.5531	0.0309	0.4	-	-
Film	FQ	0.38	0.9333	0.1356	0.33	0.8797	0.2282
	VFQ	0.36	0.7193	0.0627	0.36	0.7193	0.0627
	UFQ	0.4	0.0212	0.0125	0.4	-	-
Software	FQ	0.21	0.7143	0.6965	0.20	0.7361	0.6902
	VFQ	0.33	0.8621	0.3472	0.4	0.5067	0.1056
	UFQ	0.36	0.7717	0.2132	0.39	1.0000	0.0030

As shown in Figure 4.5 (a) and (b), as the expansion over FQ instances is executed using gradual threshold decrement, the PR curves of the classes Person, Location, and Software decline slowly toward the top right corner of the graphs, thus causing recall to increase while maintaining a precision greater than 70%. On the other hand, the curves of the remaining classes decrease more steeply before the precision falls to lower than 20%. For the other query types, VFQ and UFQ, the PR curves slump sharply without improving the recall scores, with the exception of the class Location, where recall improves steadily; however, precision decreases rapidly. Table 4.9 reports the maximum recall scored before the precision drops to lower than 70% for each of the target entity classes and query types.

Comparing the initialisation settings, when using the Class Vector created over the training dataset (SCV-B) rather than using the seeds' BoCWs (SCV-A), the precision scores are stabilised as the expansion operates through gradual similarity decrement. This is observed among all entity classes whose instances are extracted from FQs, and therefore, recall scores improve by at least 5% (Organisation class) and at most 46% (Person class).

Although the gradual vector similarity threshold improves recall, it is at the cost of reduced precision. Furthermore, the recall scores improve only when the expansion operates over instances found in the form of FQs, but not in the case of VFQs and UFQs,

Seeds Fertility

To investigate whether the fertile seeds selected over the training dataset are also fertile over the test dataset, Table 4.10 lists the percentage of seeds whose scored precision is at least 70% ($P \geq 0.70$), and percentage of those that score lower than 70% ($P < 0.70$). Furthermore, the table shows the proportion of the seeds that scored a recall less than 1% ($R < 0.01$), between 1% and 10% ($0.01 \leq R < 0.10$), and at least 10% ($R \geq 0.10$).

Table 4.10: Fertility of the discovered seeds in terms of Precision and Recall

Entity	Query	Percentage of Seeds					
Class	Type	$tp = 0$	$P \geq 70$	$P < 0.70$	$R < 0.01$	$0.01 \leq R < 0.10$	$R \geq 0.10$
	FQ	15.69	72.55	11.76	3.53	44.71	36.08
Person	VFQ	10.23	78.41	11.36	12.50	19.32	57.95
	UFQ	8.57	80.00	11.43	14.29	77.14	-
	FQ	5.13	94.87	-	-	-	94.87
Location	VFQ	5.47	93.53	1.00	9.95	71.14	13.93
	UFQ	3.46	84.23	12.31	11.54	84.23	0.77
	FQ	13.85	83.08	3.08	4.62	22.31	60.00
EduOrg	VFQ	-	50.00	50.00	25.00	75.00	-
	UFQ	-	84.93	10.96	2.74	8.22	84.93
	FQ	17.31	78.85	3.85	5.77	78.85	-
Org	VFQ	6.90	13.79	79.31	20.69	72.41	-
	UFQ	11.76	29.41	58.82	29.41	58.82	-
	FQ	27.78	55.56	16.67	-	11.11	61.11
Film	VFQ	28.57	-	71.43	-	57.14	14.29
	UFQ	-	100.00	-	-	100.00	-
	FQ	25.00	66.67	8.33	-	-	75.00
Software	VFQ	-	81.82	18.18	-	-	100.00
	UFQ	-	75.00	25.00	-	25.00	75.00

Analysing the results in Table 4.10, at least 50% of every set of seeds discovered from the training dataset for each target entity class and query type is fertile when initialising the expansion over the test dataset with a precision greater than 70%, with the exception of the class Organisation, where only 13% of the discovered seeds over VFQs and 29% of those found in UFQs scored a precision of at least 70%. Furthermore, for the class Film, where the number of discovered seeds were very low, i.e., seven seed instances from FQs and one seed instance in UFQ, the seeds scored a precision less than 70% when expanded over the test dataset.

When examining the recall scored by the discovered seeds, the majority of the seeds achieved a recall greater than or equal to 1% of the gold list instances. Furthermore, at least 13% of the discovered seeds score at least 10% recall when initialising the expansion over the test dataset. This reveals that if the seeds appear to be fertile over a sample of the dataset, it is highly likely they will be fertile over the test dataset.

In this section, the vector expansion approach proposed in this thesis was evaluated to analyse how the performance may vary given the query type from which the named entity is extracted. The approach performs best in mining named entities that appear in FQs, where the named entity has no contextual clues in the query string that can be used for classification. On the other hand, previous approaches for NER in search queries presented in [44, 80] were not applicable to this type of queries. Moreover, in this section, the performance was analysed as well to investigate how the different expansion settings, i.e., SCV and MCV, may affect the performance of the expansion for each of the target entity classes. Although the vector expansion using SCV mode scores higher precision, the recall scored was lower than that achieved using MSV expansion mode.

In the following section, rather than using the subset of named entities validated with DBpedia, the usefulness of the Class Vector is evaluated over all unique named entities extracted from the full log regardless of the query type from which the named entity is extracted.

4.5.4 Performance Assessment Using Precision@K

In this setting of the evaluation, the objective is to assess the usefulness of the Class Vector created for each target entity class over the training dataset in classifying entities that appear in the full query log regardless of the query type.

As presented in the previous subsection, the quality of the Class Vectors created over those entities extracted from FQs is better than those derived from VFQ and UFQ (Table 4.8). Therefore, the Class Vectors created by expanding the fertile seeds' BoCWs over the training dataset are used. Simply, each instance in the full log, regardless of the query from which it was extracted, is tagged with the target entity class if the similarity of its BoCW to the Class Vector is greater than or equal the threshold of 0.4. Furthermore, rather than using the full set of seeds, only the top ten discovered fertile seeds are used, i.e., those that achieved highest precision in the fertile seeds discovery experiment (see Subsection 4.3.3). The instances are ordered by decreasing similarity to find the top 250 extractions which are checked manually to pinpoint the precision@K achieved over the target entity classes (see Table 4.11).

When applying MCV, multiple sub-class vectors are used (each corresponding to an expanded fertile seed BoCW over the training dataset) in order to find the similarity of each NE instance, and those that achieve a cosine similarity greater than or equal to 0.4 are tagged with the target entity class. It is possible for the same instance to be tagged by more than one sub-class vector, and in that case an instance is assigned the highest similarity score. Similarly, the instances are ordered by decreasing similarity scores, and the top 250 are checked manually

Table 4.11: SCV precision@K

Class	@50	@100	@150	@200	@250	Average
Location	1.00	1.00	1.00	1.00	1.00	1.00
Person	1.00	0.96	0.94	0.95	0.92	0.95
Edu Org	1.00	1.00	1.00	1.00	1.00	1.00
Org	0.94	0.95	0.96	0.97	0.96	0.96
Film	0.98	0.96	0.95	0.95	0.94	0.96
Software	0.94	0.95	0.95	0.97	0.96	0.95

Table 4.12: MCV precision@K

Class	@50	@100	@150	@ 200	@ 250	Average
Location	1.00	1.00	1.00	1.00	1.00	1.00
Person	1.00	0.96	0.94	0.94	0.92	0.95
Edu Org	1.00	1.00	1.00	1.00	1.00	1.00
Org	0.94	0.97	0.97	0.96	0.96	0.96
Film	0.98	0.99	0.97	0.97	0.95	0.97
Software	0.94	0.96	0.96	0.96	0.97	0.96

to pinpoint the precision@K (see Table 4.12).

Using the same performance measure used by [80], it was found that proposed vector expansion approach to QNEC achieves an average precision@K greater than 95%. Therefore, the results achieved are comparable to the approaches proposed in the literature for named entity recognition in queries [80].

The false positives observed in the top 250 named entities fall under one of the following cases:

(i) *Misclassified named entity.* Named entities extracted from a single UFQ are usually misclassified to each other's class. In a query such as 'Patriot Mel Gibson', it was found that the actor 'Mel Gibson' was incorrectly classified as a Film, because the BoCW extracted is similar to the one extracted for the NE 'Patriot'. Furthermore, the snippets for such a query describe a movie and the actors starring in that movie with words such as 'directed', 'starring', and 'film' rather than 'born' and 'biogrophy', which usually identify instances that belong to the class Person.

(ii) *Not a named entity.* In a VFQ such as 'Patriot trailer', it is possible that 'trailer' and 'Patriot' are both POS tagged as proper nouns, because the word 'trailer' often appears as a heading of the snippets for such a query (i.e., capitalised). Therefore, the QNER approach presented in Chapter 3 results in tagging 'trailer' as part of the named entity. Similarly to the previous case, this leads to incorrectly classifying the word 'trailer' as a named entity of the class Film.

4.6 Comparative Assessment

The vector expansion approach, presented in this thesis, is next compared to previous approaches in the following aspects:

1. *Queries versus snippets*, where the objective is to compare how the performance of the vector expansion for QNEC varies when contextual clues are extracted from the query string or the snippets set related to that query.
2. *Baseline versus vector expansion*, where the approach presented in [80] is implemented on the same dataset and the performance is compared using the measures precision and recall as well as precision@K.

4.6.1 Query Versus Snippet for QNEC

As discussed in Chapter 2, previous approaches to entity recognition and classification in search queries relied solely on the query to identify the class to which it belongs. In contrast, this work relies on query enrichment with the snippets of the top-n related results. Moreover, the work presented here exploits BoCW representation of the clues that surround a named entity occurrence, unlike previous approaches where the remainder of the query after excluding the named entity string was used to extract the contextual clues.

For comparative assessment purposes, the named entity instances that appear in both FQs and VFQs are used. Instances extracted from UFQs were excluded to validate the extent to which the context extracted for the remainder of a single named entity query can reveal its class. Accordingly, processing the same dataset defined in Section 4.5, Table 4.4, results in a total of 27,505 named entity instances that appear in 295,025 VFQs. For each NE instance, the following types of BoCWs vectors representations were used:

SBoCW: created from the snippets set related to the FQ of that named entity.

QBoCW: created from the remainder of the VFQs after excluding the NE string. The context words extracted from the remainder of the query should satisfy the same conditions applied to the snippets listed in Subsection 4.3.1.

S&Q-BoCW: consists of the combination of context words extracted from the VFQs, and the context words extracted from the snippets set of the FQs.

The set of instances is divided into a training dataset, which is used to identify the fertile seeds, and a test dataset to evaluate the performance of the expansion using the identified seeds. Table 4.13 presents the frequency and percentage of seeds found for each type of the BoCW representations and for each target named entity class.

The percentage of instances found fertile in the training dataset are similar to those discovered in the performance assessment experiment presented in Section 4.5, Table 4.5. The only exception is the class Film: when the QBoCW vector representation was used, fertile seeds could not be detected.

Table 4.13: Percentage of fertile seeds selected for each target named entity class

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

Using the fertile seeds selected per vector representation, the vector expansion approach is executed for each target entity class over the test dataset. Table 4.14 presents the precision and recall scored as a result of the expansion.

Over all the expansion modes SCV and MCV (Table 4.14, last column), the vector expansion approach achieves a macro-average precision between 86% and 99% over named entities extracted from FQs, when the SBoCW is used to represent the context of the named entities, as well as a macro-average recall between 11% and 49%, with the exception of the class Organisation, where the macro-average precision drops to 67% with a macro-averaged recall of 1.9%.

Analysing the false positives of the class Organisation, most instances that were recorded automatically as false positives were clothing brands that can be classified either as a Company's name or a Person's name, such as 'Michael Kors', 'Christian Louboutin', and 'Jeffrey Campbell'.

As for the case when the BoCW is extracted from the remainder of every VFQ after excluding the NE string (i.e., QBoCW), the expansion macro-average precision decreases by 12.6% to reach 75.5% over the class Educational Organisation, and it drops by 22% for the class Software. Furthermore, although the scores of the expansion over Organisation instances are lower than 70% using both vector representations QBoCW and SBoCW, for the class Organisation, the macro-average recall of the expansion that uses QBoCW is 5%, which is higher than the score achieved via SBoCW by only 3%. For the remaining classes, the macro-average precision ranges between 72% and 82%, with the exception of the class Film, for which no fertile seeds are detected.

When comparing the results of the expansion performance using the SBoCW representation with the QBoCW representation, the macro-precision scores using SBoCW are greater than QBoCW by at least 11% and at most 22% over all the target entity classes, with the exception of the Organisation and Film classes. In terms of macro-averaged recall, the scores of the expansion when using the SBoCW representation are slightly higher than those achieved when using the QBoCW representation over the classes Person and Software (3% and 6%, respectively). On the other hand, looking at the classes Location and Educational Organisation, the macro-averaged recall scored when using SBoCW is significantly higher than that of QBoCW by 42% and 20%, respectively.

Analysing the macro-precision and macro-recall scores of the expansion when using S&Q-BoCW, the results are approximately the same after the addition of the query context words to the SBoCW, with the exception of the class Educational Organisation. Looking at the expansion mode MCV-A, when using a BoCW that includes contextual clues from both the FQ snippets and remainder of the VFQs, the expansion precision increased from 70% (SBoCW) and 75% (QBoCW) to 97% over Educational Organisation instances, and thereby increasing the macro-averaged precision from 88% (SBoCW) and 75% (QBoCW) to reach 98%, while maintaining a macro average recall greater than 20%.

S+Q-BoCW is used to denote the evaluation where precision and recall are measured using the unique instances extracted though both vector representations, SBoCW and QBoCW. The objective here is to quantify the overlap of true positives and false positives induced by one vector representation and not the other, and vice versa. In this setting, when comparing the performance with that achieved using the SBoCW representation, the macro-average recall scored is improved by at least 5% and at most 13.6% over the classes Location, Person, Educational Organisation, and Software. This reveals that using QBoCW leads to tagging, from 5% to 13.6%, of instances that were not tagged using SBoCW. However, the improvement of the macro-average recall is at the cost of lower precision. The false positives induced using the QBoCW representation are higher than those induced using the SBoCW representation, thus causing the macro-average precision to decrease by at least 3% and at most 14%. On the other hand, comparing QBoCW to S+Q-BoCW reveals that the true positives induced by SBoCW improve both the macro-average precision and recall among the classes Location, Person, Educational Organisation, and Software. More specifically, precision improved by at least 14% and at most 28% and recall improved by at least 15% and at most 28%.

Looking at the macro-average performance over all the target entity classes (see Table 4.14, last row), the average precision and recall scored by the expansion when using the QBoCW representation is outperformed by the performance achieved over the SBoCW representation of the instances' context vectors over all modes of expansion. Similarly to the results scores discussed in Section 4.5, over both snippet and query context representations (i.e SBoCW and QBoCW), the SCV expansion mode scores higher macro-average precision than the MCV mode at the cost of lower recall.

Table 4.15 lists the top ten context words from the Class Vector created as a result of the expansion using SCV-A setting over each of the BoCW vector representations. Although the Class Vectors created using QBoCW contain descriptive clues for most of the target entity classes, the instances frequencies of the top context words are low when compared to that of SBoCW vector representation. This leads to the conclusion that the contextual clues that surround a named entity string in the search queries are often not shared by many named entities that fall in the same named entity class.

Table 4.14: Precision (P) and Recall (R) scored by the vector expansion for each target entity class

Entity	Query	SCV-A		SCV-B		MCV-A		MCV-B		Micro		Macro	
Class	Type	P	R	P	R	P	R	P	R	P	R	P	R
Person	SBoCW	0.9941	0.1432	0.8616	0.1387	0.7902	0.4299	0.8315	0.3048	0.7905	0.4319	0.8694	0.2542
	QBoCW	0.8472	0.0516	0.7590	0.0710	0.5863	0.5086	0.7246	0.2819	0.5863	0.5106	0.7293	0.2283
	S&Q-BoCW	0.9936	0.1311	0.8558	0.1305	0.8057	0.3953	0.8252	0.3062	0.8064	0.3992	0.8701	0.2408
	S+Q-BoCW	0.9423	0.1658	0.8115	0.1917	0.6153	0.6693	0.7447	0.4844	0.6153	0.6718	0.7784	0.3778
Location	SBoCW	0.9958	0.4229	0.9956	0.4106	0.9859	0.6915	0.9946	0.4428	0.9859	0.6915	0.9930	0.4920
	QBoCW	0.9244	0.1505	0.9747	0.0463	0.5623	0.6386	0.8409	0.3524	0.5625	0.6395	0.8255	0.2969
	S&Q-BoCW	0.9963	0.4064	0.9962	0.3974	0.9849	0.6672	0.9947	0.4518	0.9849	0.6681	0.9930	0.4807
	S+Q-BoCW	0.9732	0.4803	0.9938	0.4347	0.6204	0.8165	0.8962	0.5783	0.6204	0.8168	0.8709	0.5774
Org	SBoCW	0.6216	0.0104	0.6250	0.0090	0.6667	0.0262	0.7802	0.0320	0.7238	0.0343	0.6734	0.0194
	QBoCW	0.7903	0.0221	0.7966	0.0212	0.2740	0.0862	0.7143	0.0744	0.2803	0.0898	0.6438	0.0510
	S&Q-BoCW	0.6129	0.0086	0.6774	0.0095	0.7419	0.0207	0.7656	0.0221	0.7901	0.0289	0.6995	0.0152
	S+Q-BoCW	0.7097	0.0298	0.7209	0.0280	0.3037	0.1046	0.7219	0.0983	0.3224	0.1150	0.6140	0.0652
Edu Org	SBoCW	1.0000	0.2314	1.0000	0.1989	0.7054	0.3480	0.8243	0.3499	0.7164	0.3767	0.8824	0.2820
	QBoCW	0.6000	0.0229	0.9118	0.0593	0.7529	0.1224	0.7586	0.1262	0.7212	0.1434	0.7558	0.0827
	S&Q-BoCW	0.9904	0.1969	0.9901	0.1912	0.9728	0.2734	0.9801	0.2830	0.9709	0.3193	0.9834	0.2361
	S+Q-BoCW	0.9412	0.2447	0.9766	0.2390	0.6962	0.4207	0.7908	0.4264	0.6939	0.4551	0.8512	0.3327
Software	SBoCW	0.9886	0.3466	0.9882	0.3347	0.9899	0.3904	0.9886	0.3466	0.9900	0.3944	0.9889	0.3546
	QBoCW	0.8228	0.2590	0.7848	0.2470	0.7266	0.4024	0.7333	0.2629	0.7214	0.4024	0.7669	0.2928
	S&Q-BoCW	0.9778	0.3506	0.9785	0.3625	0.9794	0.3785	0.9890	0.3586	0.9796	0.3825	0.9812	0.3625
	S+Q-BoCW	0.8881	0.4741	0.8636	0.4542	0.7833	0.5618	0.8264	0.4741	0.7790	0.5618	0.8404	0.4910
Film	SBoCW	0.9091	0.0930	0.9149	0.1000	0.9091	0.1163	0.8529	0.1349	0.8611	0.1442	0.8965	0.1110
	QBoCW	-	-	-	-	-	-	-	-	-	-	-	-
	S&Q-BoCW	0.8361	0.1186	0.9114	0.1674	0.8525	0.1209	0.9048	0.1767	0.8571	0.2093	0.8762	0.1459
	S+Q-BoCW	0.9091	0.0930	0.9149	0.1000	0.9091	0.1163	0.8529	0.1349	0.8611	0.1442	0.8965	0.1110
Micro	SBoCW	0.9874	0.2124	0.9539	0.2049	0.8853	0.4093	0.9086	0.2869				
	QBoCW	0.8814	0.0787	0.8248	0.0530	0.5692	0.4162	0.7765	0.2399				
	S&Q-BoCW	0.9844	0.2019	0.9526	0.2010	0.9076	0.3845	0.9177	0.2868				
	S+Q-BoCW	0.9499	0.2467	0.9196	0.2400	0.6207	0.5570	0.8129	0.4138				
Macro	SBoCW	0.9182	0.2079	0.8976	0.1986	0.8412	0.3337	0.8787	0.2685				
	QBoCW	0.6641	0.0843	0.7045	0.0741	0.4837	0.2930	0.6286	0.1830				
	S&Q-BoCW	0.9012	0.2020	0.9016	0.2098	0.8895	0.3093	0.9099	0.2664				
	S+Q-BoCW	0.8939	0.2480	0.8802	0.2413	0.6547	0.4482	0.8055	0.3661				

Table 4.15: Top ten context words in the Class Vector created at the end of the expansion that uses SCV-A

Person					
SBoCW		QBoCW		S&Q-BoCW	
photos	515	pics	240	photos	477
born	490	pictures	70	born	445
news	480	photos	41	news	438
com	333	bio	31	pictures	316
actress	309	biography	28	actress	311
biography	295	movies	23	com	309
gossip	286	books	21	biography	295
american	267	porn	19	gossip	268
check	267	picture	17	check	250
take	259	lyrics	17	take	246

Location					
SBoCW		QBoCW		S&Q-BoCW	
city	1215	hotels	429	city	1157
news	1102	estate	399	news	1098
county	1015	sale	202	county	996
sports	877	homes	190	sports	859
everything	851	newspaper	133	everything	832
registered	810	weather	111	registered	816
forum	810	rent	91	forum	815
posts	809	news	88	posts	814
users	804	restaurants	88	users	809
vacations	624	car	86	hotels	655

Organisation					
SBoCW		QBoCW		S&Q-BoCW	
shop	42	store	72	shop	34
find	40	stores	23	stores	32
women	39	department	22	women	31
shopstyle	38	coupon	21	find	31
stores	37	online	20	fashion	31
sale	37	coupons	20	shopstyle	30
fashion	35	clothing	17	sale	29
com	18	locations	14	com	15
store	11	employment	10	shoes	12
online	10	clothes	10	clothing	12

Educational Organisation					
SBoCW		QBoCW		S&Q-BoCW	
located	130	class	22	located	112
find	107	graduating	5	find	104
student	102	rings	3	student	104
scores	100	action	3	scores	102
test	99	campus	2	test	101
alumni	65	faculty	1	alumni	62
members	38	reunion	1	members	35
founded	34	grades	1	students	24
institution	29	yearbook	1	schools	24
schools	25	sons	1	district	23

Film					
SBoCW		QBoCW		S&Q-BoCW	
com	42			queue	55
amazon	41	-	-	added	55
film	38	-	-	film	47
american	27	-	-	movie	46
added	23	-	-	american	38
queue	22	-	-	views	33
trailer	20	-	-	com	30
comedy	18	-	-	amazon	27
reviews	16	-	-	comedy	24
movie	15	-	-	trailer	24

Software					
SBoCW		QBoCW		S&Q-BoCW	
trailers	94	cheats	66	trailers	93
resource	94	codes	52	resource	93
ign	94	cheat	43	ign	93
screenshots	93	game	29	screenshots	92
game	69	walkthrough	15	game	75
gamefaqs	48	pc	12	cheats	48
has	47	games	11	video	46
video	41	download	10	gamefaqs	42
playstation	32	tips	10	has	40
games	23	strategy	9	codes	35

Gradual Threshold Decrement in SCV Expansion

In the same spirit, in order to investigate the likelihood of improving the recall scored by SCV, the vector similarity threshold is decreased gradually by 0.01 over a number of expansion rounds to reach 0.2. The accumulative precision and recall scores are plotted in PR graphs as presented in Figure 4.6. This experiment is executed using each of the vector representations SBoCW, QBoCW, and S&Q-BoCW.

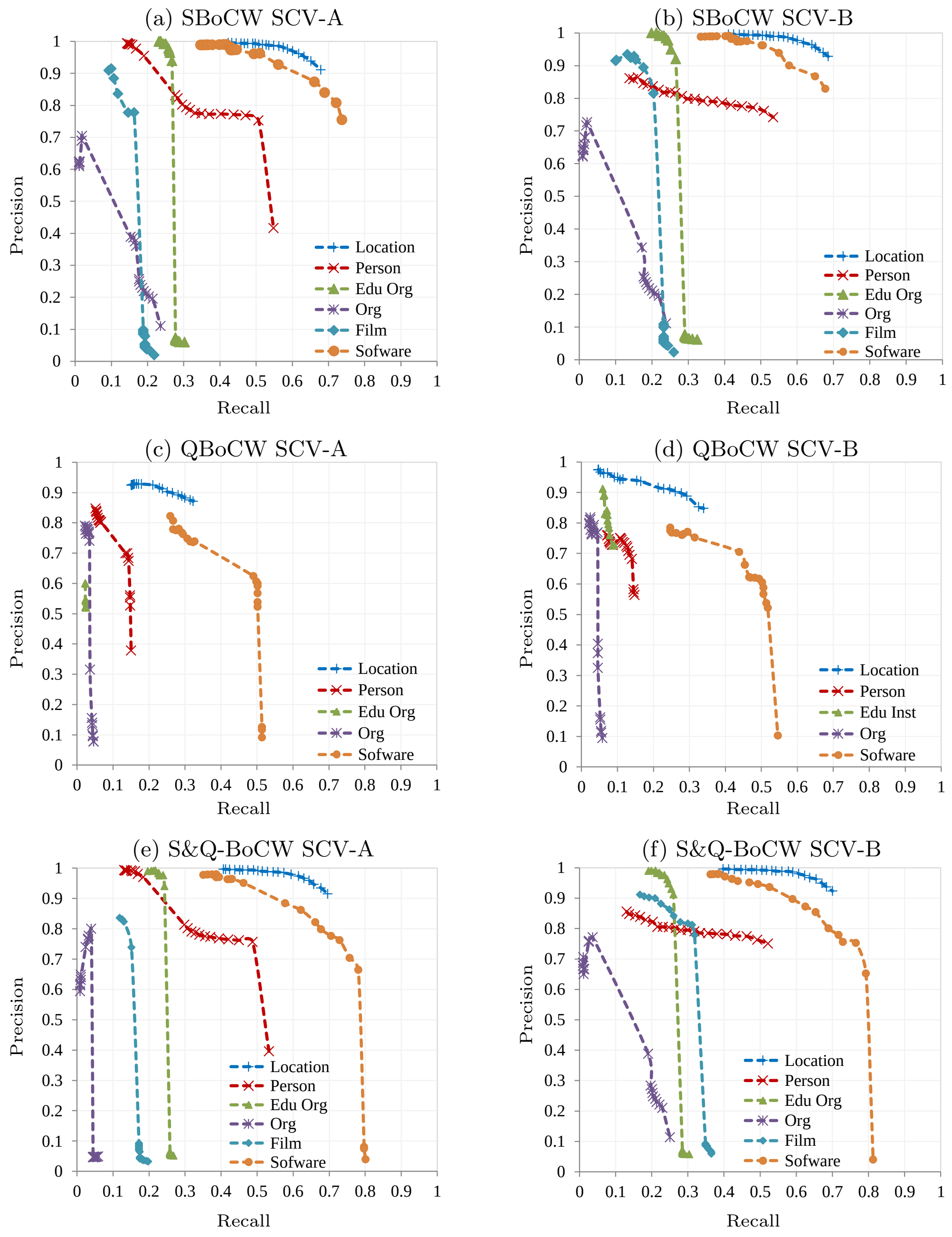


Figure 4.6: PR curves generated by gradually decreasing threshold as the expansion is executed

In general, the PR curves produced using the SBoCW vector representation (Figure 4.6 (a) and (b)) are similar to those produced by the gradual threshold experiment produced in Section 4.5 over FQs (Figure 4.5 (a) and (b)). On the other hand, when using the QBoCW representation of context vectors, the performance is better than the expansion when creating BoCW from the snippets set related to the VFQ. This can be observed over instances that belong to the class Location, Person, and Software. This shows that, in the case of VFQs, relying on the snippets to identify the class of entities may add noise, i.e., context words that are not necessarily exclusive to the target entity class, and thereby decreases the overall performance scores.

Table 4.16 reports the maximum recall scored before the precision drops lower than 70%. Using the expansion mode SCV-A over SBoCW, although the precision of the classes Person and Software decreased to approximately 75%, 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 recall to reach a maximum of 67.8%, while maintaining a precision higher than 90%. When using the SCV-B expansion mode, the performance is approximately similar to that achieved using SCV-A for the classes Person and Location. However, the performance was improved using this mode of expansion for the class Software to reach a recall of 67.7% with precision of 82%. For the classes Educational Organisation and Film, recall only improved by 3% and 10%, respectively, whereas precision decreased by no more than 13% using both expansion modes.

Analysing the performance over QBoCW, the expansion recall is improved only by a small percentage between 1.3% and 5.9% for the classes Person, Educational Organisation, and Organisation using both expansion modes. 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 precision remained higher than 84%. In addition, recall scored for the class Software was improved by 19.1%, whereas precision decreased from 78% to 70% at threshold 0.32. However, when comparing SBoCW against QBoCW, the performance of QBoCW is outperformed by SBoCW in terms of precision and recall, with the exception of the class Organisation. Although performance is lower than that achieved by QBoCW, precision scored for the class Organisation using both expansion modes increases from 62% to 70% at the threshold 0.35 before dropping to 38% (see Figure 4.6 (c) and (d)). The reason behind this is that, as the threshold decreases, more instances that belong to this class were tagged during the expansion, which adds 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 VFQs and the snippets set of the FQs (i.e., S&Q-BoCW), the expansion performance with gradual threshold decrement is approximately similar to that achieved when using SBoCW, with the exception of the class Film. By analysing the results, recall is further improved to reach a total of 31% using SCV-B with precision of 77.8%.

Table 4.16: Maximum performance scored by gradual threshold decrement before precision drops to less than 70%

Entity	Vector	SCV-A			SCV-B		
Class	Rep	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
Edu Org	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
Org	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
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

4.6.2 Comparison to Baseline Approach

In the previous section, the expansion approach was assessed comparatively in terms of BoCW representation in order to analyse how performance is affected when the contextual clues are extracted from snippets, as compared with extracting them from the query string. In this section, the vector expansion algorithm is assessed by comparing it to the baseline approach presented in [80]. In particular, [80] is chosen for comparative performance analysis because this approach is similar to the vector expansion approach in terms of objective, i.e., mining named entities from a query log. Moreover, Paşca's approach was employed by [44] to create the training dataset (see Chapter 2, Section 2.7). Table 4.17 summarises the differences between [80] and the vector expansion approach in terms of candidate named entity definition and the contextual clue used for classifying the named entity.

Table 4.17: Paşca's Approach [80] versus Vector Expansion

	Paşca's Expansion	Vector Expansion
Contextual Clue	The remainder of the query after excluding an entity's string forms a query pattern.	A BoCW that consists of the context words that surround the target query named entity in the snippets set related to that query.
Candidate NE	The remainder of the query after excluding the matched query pattern is a candidate NE.	A sequence of keywords whose snippets-validated grammatical features indicate a proper name contained within the boundaries set by query segmentation.

Briefly, Paşca's technique [80] involved the following steps: (1) for every target entity class, five seeds were selected; (2) the query log is scanned to find those queries whose keywords match the seeds' strings, and the corresponding query patterns are extracted; (3) a reference-search-signature vector is created that consists of all the seeds' query patterns whose weight is set to the frequency of the queries that matched the seed; (4) the query patterns are used to find new candidate named entities that share the same patterns as the seeds; (5) a search-signature vector is created for each candidate named entity that consists of all the query patterns in which the candidate named entity appears and the weight of each pattern is set to the query frequency; lastly, (6) every candidate named entity is assigned a score based on the Jensen-Shannon divergence [61] of its vector from the class reference-search-signature vector.

In the following subsections, the vector expansion approach presented in this thesis is compared with Paşca's technique [80] using the following evaluation methods:

1. Expansion evaluation using precision and recall.
2. For each target named entity class, precision@K is used to evaluate the usefulness of the vector created at the end of the expansion for classifying named entities in the query log.

Comparative Assessment Using Precision and Recall

In order to estimate the precision and recall for the approach of [80], the same dataset defined in Section 4.6.1 is used, where the VFQs in which the extracted candidate named entities occurred are used as they appear in the query log. Furthermore, the reference-search-signature vector is created using the training dataset, and performance is measured over the test dataset. Because [80] did not state the seeds selection technique, the approach is initiated using two sets of seeds: (1) ten randomly chosen seeds, where the approach is executed once for each of five sets of ten randomly chosen seeds and the highest performance scored is reported; and (2) the set of fertile seeds discovered for QBoCWs.

Table 4.18 reports precision and recall scores. When measuring the performance of this approach, each extraction was checked automatically against the gold standard list of the corresponding entity class to find an exact instance that matches the extracted named entity string. However, the following problem emerged during evaluation. The boundaries of the extracted candidate named entity may be incorrect, for example, extracting 'Celine Dion God Bless' as a Person named entity. Furthermore, the same named entity may occur in more than one distinct extraction, e.g., 'Celine Dion God Bless' and 'Celine Dion' and therefore non-exactly matching instances were counted as false positives. Therefore, the precision and recall scores are very low when compared with the vector expansion approach presented in this thesis.

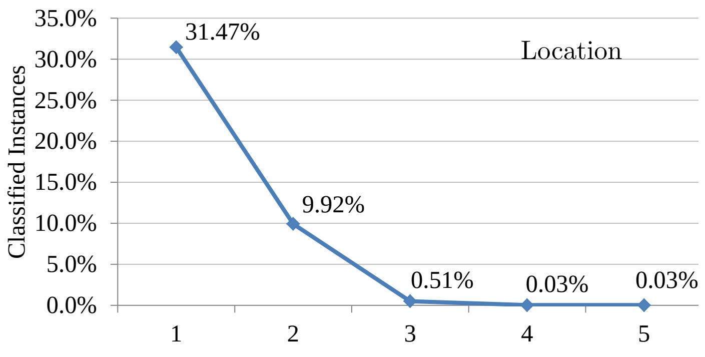
Table 4.18: Performance scored by Paşca's approach [80]

Entity	Randomly Chosen Seeds		VFQ Fertile Seeds	
Class	Precision	Recall	Precision	Recall
Person	0.4444	0.0169	0.54815	0.18776
Location	0.5803	0.3073	0.72774	0.17182
EduOrg	0.0853	0.1128	0.08611	0.08413
Org	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

Although the vector expansion approach utilises a single *Class Vector* per entity class (SCV expansion mode), instead of being static, which is the case of the *reference-search-signature vector* [80], the Class Vector is iteratively expanded as new instances are tagged through the expansion. Classifying named entities over more than one classification round may lead to tagging more instances that would not necessarily be classified in the first pass over the query log. For example, when analysing the expansion SCV-A for the classes Location and Software over the FQ instances, it can be seen that there were instances classified in the second and the following 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 4.7).

Comparative Assessment Using Precision@K

Here, the quality of the Class Vector, induced by the end of the vector expansion, is compared with the *reference-search-signature vector* induced by implementing Paşca's approach [80].



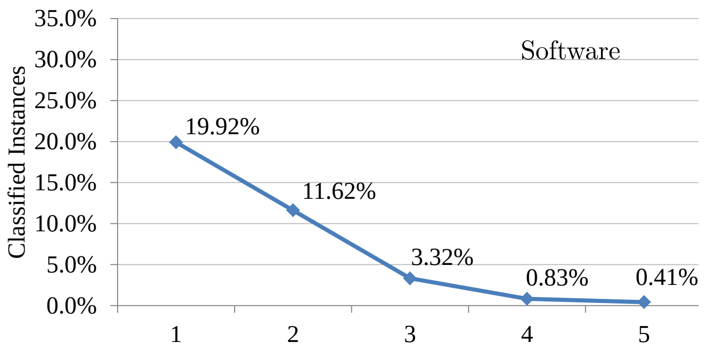


Figure 4.7: Percentage of instances classified per iteration during the SCV-A expansion

To conduct this experimental evaluation, for each candidate named entity extracted from the full query log, regardless of the query type from which it is extracted, two BoCW vectors were created:

SBoCW, which consists of all the context words that surround the named entity in the snippets set related to that query.

QBoCW, which consists of all the context words found in the query that surround the named entity string if it exists (because FQs do not contain any context words in the query).

Using the training dataset defined in Section 4.6.1, two Class Vectors are created:

$ClassVector^{SBoCW}$, that is, the aggregation of the Class Vectors created by expanding the BoCWs of the top ten fertile seeds over the training dataset whose instances' BoCWs are created from the snippets set related to the FQs in which the instances occurred; and

$ClassVector^{QBoCW}$, that is, the aggregation of the Class Vectors created by expanding the BoCWs of the top ten fertile seeds over the training dataset whose instances' BoCWs are created from the remainder of the VFQs in which the instances occurred.

After that, the similarity of each candidate named entity SBoCW to the $ClassVector^{SBoCW}$ is measured, and the similarity of each of the named entity QBoCW to the $ClassVector^{QBoCW}$ is measured as well. Then, the labelled entities are ordered by decreasing similarity scores, and the top 250 are checked manually as correct or incorrect to measure precision@K. Similarly, the top 250 candidate named entities induced by implementing Pasca's approach [80] are checked manually to measure precision@K.

Figure 4.8 plots the performance scored by each of these approaches. FQ-SBoCW is used to refer to the performance scored when instances are ranked by the similarity of their SBoCW to $ClassVector^{SBoCW}$, and VFQ-QBoCW is used to refer to the performance scored when instances are ranked by the similarity of their QBoCW to $ClassVector^{QBoCW}$. Furthermore, Query Template is used to refer to the precision@K scored when implementing the approach [80].

As the precision@K graphs show, the performance of FQ-SBoCW is better than the VFQ-QBoCW and Query Template approaches for most of the target entity classes, with the exception of the class Organisation, where it is outperformed by VFQ-QBoCW between ranks 1 and 33; at higher ranks, the precision of FQ-SBoCW increases above VFQ-QBoCW to reach 86%.

When comparing the precision@K scores of FQ-SBoCW to those presented in Subsection 4.5.4, the performance scores are lower than those scored for the classes Organisation, Film, and Software. Because the training dataset is smaller than that used in Section 4.5, the quality of the created Class Vector is not as good, and thereby the precision@K scored decreases.

Finally, Table 4.19 lists the average-class precision@K scores of SBoCW, QBoCW, and Query Template at the ranks 50, 100, 150, 200, and 250. Analysing the results using Eberhardt and Fligners test for equality of two proportions [34], the difference between the average-class precision@K of the SBoCW and the other two approaches is statistically significant at $\alpha < 0.01$ for all the ranks.

Table 4.19: Average-class precision@K scores, where $*$ denotes statistical significant differences with QBoCW and QueryTemplate using Eberhardt and Fligners test for equality of two proportions test $\alpha < 0.01$

	@50	@100	@150	@200	@250
SBoCW	0.957*	0.965*	0.958*	0.953*	0.953*
QBoCW	0.780	0.756	0.741	0.747	0.738
QueryTemplate	0.670	0.672	0.673	0.633	0.603

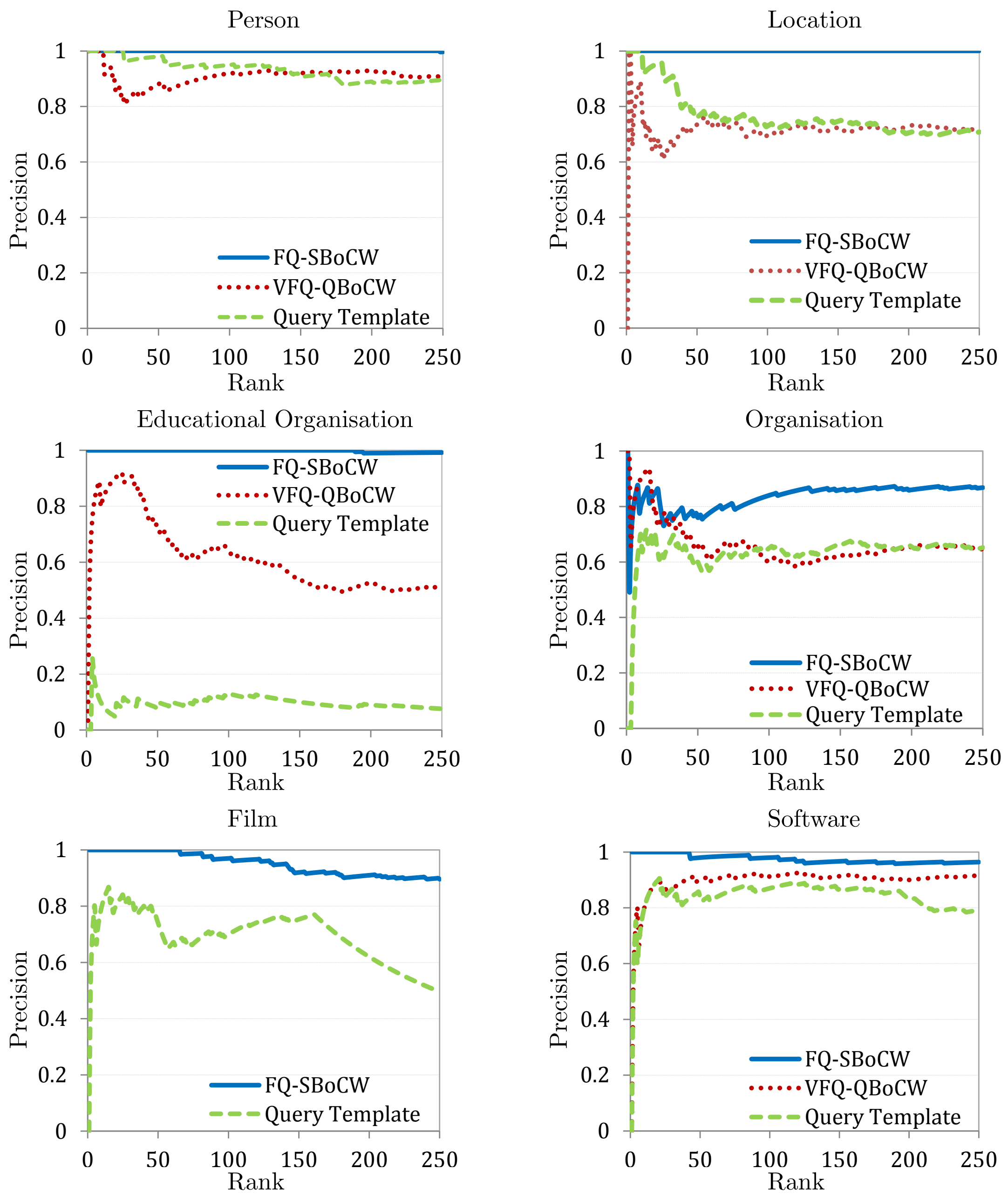


Figure 4.8: Comparative Assessment using precision@K

4.7 Discussion

In this chapter, we first introduced a categorisation of entity-centric search queries. Each query bearing a named entity mention can be categorised into three types: FQ, VFQ or UFQ. Then, a novel expansion approach for QNEC was presented. The approach utilises the snippets set related to queries in order to extract the contextual clues that surround named entities for identifying their class. Furthermore, the approach exploits a flexible representation of the contextual clues in the form of BoCW vectors in a common vector space. The BoCW of automatically selected seeds is expanded iteratively as new instances are tagged during expansion. Previous approaches to QNEC relied on the query context to identify the class to which a named entity belongs; however, this approach is only applicable to VFQs and UFQs, but not FQs.

The following is a summary of the findings that the QNEC approach presented in this chapter leads to:

- In the case of focused queries, using a flexible representation of the contextual clues in the form of BoCW can effectively classify named entities in the search queries using the vector expansion algorithm.

- Precision and recall metrics can be utilised to estimate seed fertility in order to initiate the expansion.

- The two expansion modes, SCV and MCV, affect the precision and recall scores that can be achieved by the vector expansion approach. Although SCV ensures higher precision during vector expansion, it biases the expansion towards a specific semantic space within the target entity class, thereby scoring lower recall. On the other hand, MCV scores a higher recall because each fertile seed is expanded independently of the other seeds, and thereby each seed expansion covers a larger portion of the semantic space of the target entity class; however, the improved recall is at the cost of lower precision. This is because the likelihood of semantic drift is higher when using the MCV expansion mode.

- In the case of VFQ, using the BoCW representations of the contextual clues extracted from queries rather than using query patterns, which is the approach used by [80], achieves better performance scores in terms of precision and recall as well as better precision@K.

Chapter 5

Search Refiners of Entity-Centric Queries: Nouns, Adjectives, and Verbs

5.1 Overview

This chapter presents an exploratory analysis of the linguistic units that constitute the remainder of VFQs (i.e., Very Focused Queries) after excluding the named entity string, denoted here by *search refiners.* In Section 5.2, the motivation behind mining search refiners from query logs is presented. Statistical analysis of the linguistic units that compose search queries in general, and the refining keywords of VFQs in particular, is presented in Sections 5.3 and 5.4. In order to analyse the identified search refiners, different ranking mechanisms were employed via the Hyperlink-Induced Topic Search (HITS) algorithm, as described in Section 5.5. With the aid of three ranking measures, a comparison of the different rankings of refiners induced by each setting of the HITS algorithm versus a rank of the refiners based on their frequency in the query log is presented in Section 5.6. Finally, in Section 5.7, an in-depth analysis of the types of refiners associated with named entity classes is presented, followed by a summary of the major findings and results presented in this chapter in Section 5.8.

5.2 Search Refiners of Entity-Centric Queries

Once named entities are identified and classified in search queries, analysing the keywords that surround entities helps search engines to further specify a user's search intent. In this thesis, *search refiners* refer to the additional search keywords that a user associates with an entity name to further specify the search results related to that entity. This type of query falls under the query category VFQs defined in Chapter 4, Section 4.2.

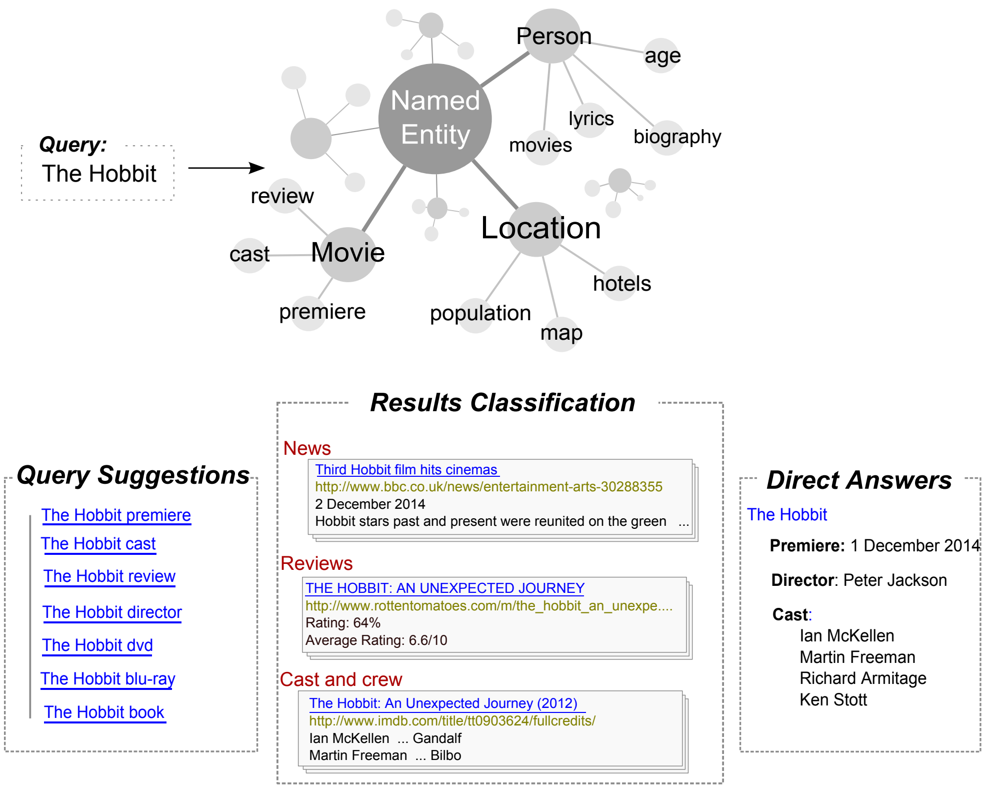
Search refiners have been investigated in previous works of query log analysis to serve different objectives. As presented in Chapter 2, besides exploiting it for named entity recognition and classification in search queries [33, 44, 80], in [85] Paşca and Durme presented an approach to mine the 'class attributes' of entities that belong to the same entity class from the query log. For example, '# lyrics', '# album', '# video clip' are all possible attributes of the entity class Singer that can be harvested from the log because they often follow a singer's name in the search queries. On the other hand, [105] used the remainder of a query to represent generic search intents of queries. These generic intents are used to build a taxonomy of intents related to a class of entities. Similarly, [23] used both the remainder of the query as well as the class label of the identified entity to introduce the notion of 'query intent template', e.g., '[Movie] showtimes in [City]', presenting an approach for identifying and clustering synonymous templates.

The motivation behind mining search refiners is to associate named entities with the common attributes that users include in their entity-centric queries. Figure 5.1 presents an illustration of the possible applications of mapping an entity-centric query to a taxonomy of entity classes along with their attributes. Mining search refiners related to a named entity class can be exploited as follows:

1. Suggest alternative queries to the query submitted by the user. Alternative queries are often referred to as query suggestions. Identifying the class to which a named entity belongs as well as the most common attributes associated with named entities that belong to that class can help in presenting useful query suggestions to users. For example, knowing that the named entity 'The Hobbit' may refer to an entity of the class Movie, and knowing that 'trailer', 'soundtrack', and 'review' are all attributes often associated with movie names in search queries, the query suggestions 'The Hobbit review' and 'The Hobbit soundtrack' can be presented to the user.

2. Results classification based on the possible attributes of an entity class. For example, in the case of an FQ, that is, a query that consists of nothing but the named entity string (see Chapter 4, Section 4.2), the set of results can be classified into subsets of those related to the 'cast', 'news' (e.g., the release date), or 'reviews' of the movie.

3. Reveal search intent behind search queries, and thereby present direct answers to users search need. For example, knowing that 'premiere' is an attribute of the class Movie, a search engine can present the exact date of the premiére of a movie without a user having



The Hobbit premiere
The Hobbit cast
The Hobbit review
The Hobbit director
The Hobbit dvd
The Hobbit blu-ray
The Hobbit book

News

Third Hobbit film hits cinemas
http://www.bbc.co.uk/news/entertainment-arts-30288355
2 December 2014
Hobbit stars past and present were reunited on the green ...

Reviews

THE HOBBIT: AN UNEXPECTED JOURNEY
http://www.rottentomatoes.com/m/the_hobbit_an_unexpe....
Rating: 64%
Average Rating: 6.6/10

Cast and crew

The Hobbit: An Unexpected Journey (2012)
http://www.imdb.com/title/tt0903624/fullcredits/
Ian McKellen ... Gandalf
Martin Freeman ... Bilbo

The Hobbit

Premiere: 1 December 2014

Director: Peter Jackson

Cast:
Ian McKellen
Martin Freeman
Richard Armitage
Ken Stott

Figure 5.1: Illustration of how mapping entity-centric queries to a named entity taxonomy extended with attributes can be used by search engines

to navigate to a specific web page to access that information.

In this chapter, an exploratory analysis of search refiners is presented as an attempt to address the following questions:

(i) What constitutes a search refiner?

(ii) How to identify and mine search refiners?

(iii) How to assign scores to the extracted refiners and rank them accordingly so they can fully be incorporated by search engines?

5.2.1 Search Refiners

In this work, the Query Grammatical Annotation approach presented in Chapter 3, Subsection 3.2.1 is exploited to reveal four major linguistic components that users often employ in their entity-centric queries:

Named Entity: search query keywords that refer to a specific entity within a target entity class, such as Person, Location, or Film.

Table 5.1: Examples of entity-centric queries associated with search refiners

Query	Query Template	Noun	Verb	Adjective
free Wolfenstein 3D download	'free # download'	-	download	free
good fishing spots at Pickwick Lake	'good fishing spots at #'	spots	fishing	good
buy a Honda Oddessy	'buy a #'	-	buy	-
cheap gas in Dallas	'cheap gas in #'	gas	-	cheap
Bad Boys official trailer	'# official trailer'	trailer	-	official

Noun: the search query keywords that identify a class of people, places, or things and whose grammatical category is a singular or plural common noun, such as 'biography', 'lyrics', and 'trailer'.

Adjective: the search query keywords whose grammatical category is an adjective and describe nouns' attributes, such as 'good', 'cheap', and 'free'.

Verb: the search query keywords whose grammatical category is a verb and describe, in addition to actions, a state or an occurrence, such as 'download', 'buy', or 'watch'.

Table 5.1 presents examples of VFQs along with query patterns [85] and the linguistic components identified. Query templates are very specific; however, when looking at the individual keywords that constitute the pattern, they often reveal how the results should be refined. For example, for the query 'free Wolfenstein 3D download', the search refiner 'download' provides a clear indication that the results need to be refined to include pages with downloadable contents of the Video Game, rather than 'review', 'buy', or 'play', and the refiner 'free' provides a clear indication to further refine results in order to exclude those with purchase option or price quotes.

In the following section, a statistical analysis of the linguistic structure in queries in general, and more specifically in single entity-centric queries, is presented.

5.3 Linguistic Components of Search Queries

For candidate named entity detection, each search query in the query log is grammatically annotated and segmented. Accordingly, in addition to annotating the query keywords as part of a named entity string, each keyword is assigned a POS, which is the English grammatical category to which a word belongs. Table 5.2 lists the total number of extracted named entities, nouns, verbs (regardless of their tense), and adjectives (regardless of their type) in the query log.

The first row of graphs in Figure 5.2 shows the frequency distribution of the linguistic units employed by users in their search queries to convey their search intent. The statistics are compiled from the full query log after excluding noise, URLs, and duplicates. Looking at the

Table 5.2: Frequency of entities, verbs, and adjectives detected in the query log

	Keyword Freq.	Query Freq.
Named Entity	6,299,933	4,624,501
Nouns	4,721,421	2,962,456
Adjectives	568,411	499,713
Verbs	669,688	610,389

distribution of named entities, one can observe only very few named entities that are highly common (i.e., named entities used in more than 10,000 unique search queries), whereas most of the named entities detected are used only in relatively few queries; the distribution is so extreme that almost 76% of named entities occur in only one query. This can be observed, as well, in the distribution of the linguistic units: nouns, adjectives, and verbs; that is: (i) there are very few common linguistic units, i.e units with query frequencies greater than 20,000 (nouns), 10,000 (adjectives), and 6,000 (verbs); (ii) there are many linguistic units that occur with low query frequencies; and (iii) there are extremely few linguistic units that occur in only one query.

The second row of graphs in Figure 5.2 shows the same plots on a log-log scale. The distribution in the doubly logarithmic axis plot is approximately linear, which is one of the characteristics of a power-law distribution [1]. This is expected because search query and query keywords frequencies are known to follow a power-law distribution [3].

Mathematically, a power-law distribution, where y is the number of linguistic units employed in x unique search queries, holds when $y = Cx^{-\alpha}$. The power law distribution implies the linear form $\log y = -\alpha \log x + c$ that is a straight line with slope $-\alpha$ in the log-log scale [1]. In order to fit the distribution to a line and extract α, logarithmic binning is used to produce the last row of plots in Figure 5.2. This is achieved by arranging the frequency distribution into exponentially increasing bins [1], resulting in α being equal to 2.2 in the case of named entities, 1.8 for nouns and adjectives, and 1.67 in the case of verbs.

As it is evident in the frequency distribution, the vast majority of the identified linguistic components are extremely infrequent among unique queries, i.e., appear with query frequency less than three. Nevertheless, there is a considerable proportion of these components that are commonly employed by users to convey their search needs.

Here, the frequency distribution of nouns, adjectives, and verbs is compiled from the full set of unique queries, regardless of whether a query contains a named entity. In the following section, the grammatical categories of the keywords that surround a named entity mention in a query are analysed.

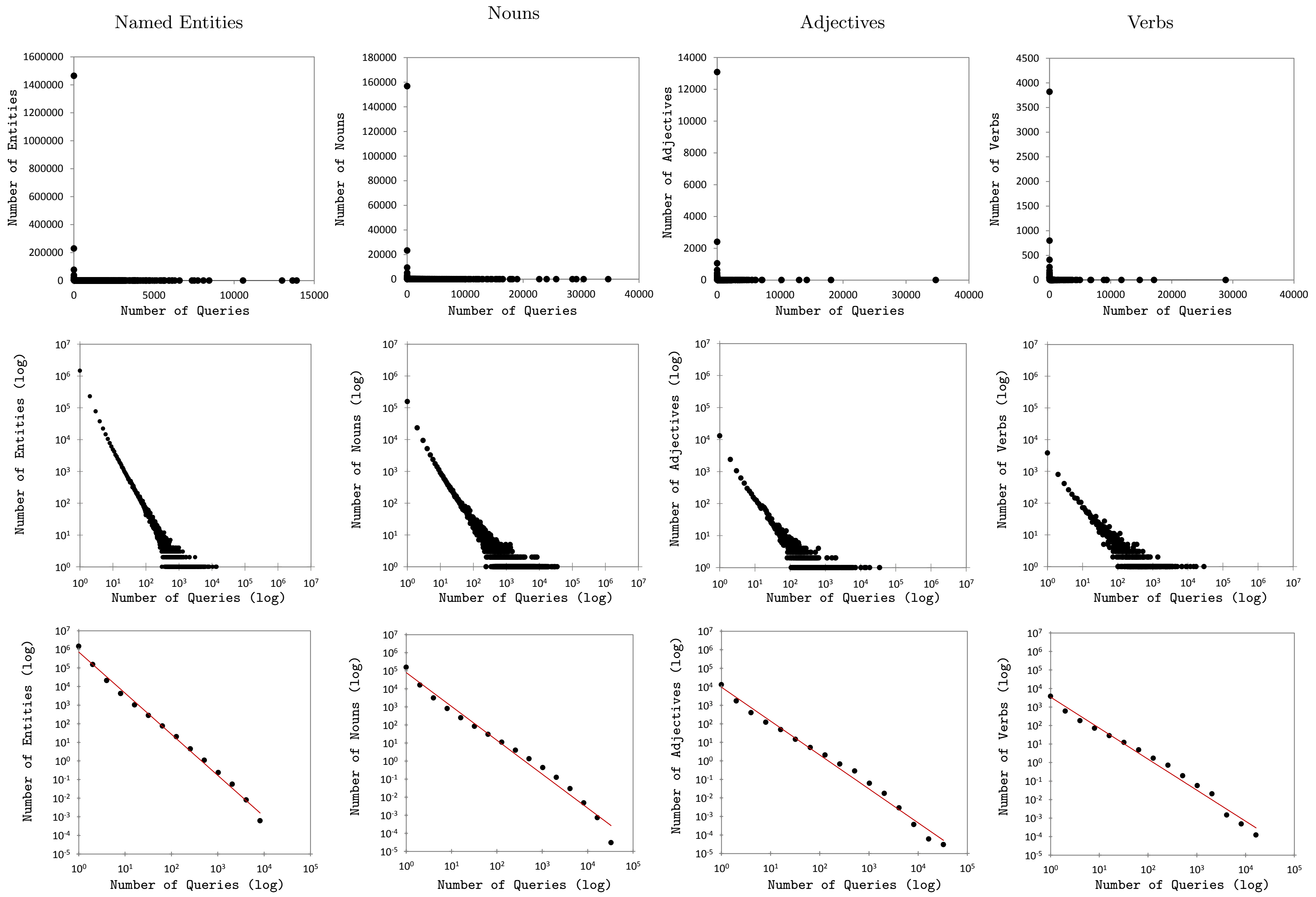


Figure 5.2: Frequency distribution of the linguistic components of unique queries in MSN query log after excluding noise, URLs, and duplicates

5.4 Refiners Associated with Named Entities in VFQs

The objective is to examine the type of keywords that users often employ to refine their entity-centric search queries. This statistical analysis is conducted to address the following questions:

A. How many keywords do users often employ to refine their entity-centric queries?

B. What are the most common keywords used to refine an entity-centric search query?

C. What are the grammatical categories of the keywords that users employ to refine their entity-centric queries?

Therefore, the dataset of VFQs employed for QNEC experimental evaluation is used (see Section 4.5). It consists of 41,385 named entities which appear in 292,509 different VFQs.

First, looking at the number of the keywords used to refine an entity-centric query, it can be seen that a majority of these queries are refined by one or two keywords (i.e., approximately 72%). Figure 5.3 plots the frequency of queries whose number of refining keywords ranges from 1 to 10 keywords. Overall, the most frequent number of keywords ranging from one to ten accounts for 99.87% of the VFQs, whereas 0.13% include long queries whose number of refiners reaches a maximum of 38 keywords. The pie charts in Figure 5.3 show the percentages of the grammatical categories that constitute the remainder of the query whose length ranges between one and five. Overall, the dominating grammatical category of the refining keywords is common nouns, followed by verbs and adjectives, when excluding numbers and prepositions. As the number of keywords increases, the percentage of prepositions, e.g., 'by', or subordinating conjunctions, e.g., 'before' (abbreviated by 'IN' in the figure), also increases.

Second, what are the most common keywords used to refine an entity-centric search query? Figure 5.4 presents the top 20 keywords that occur in VFQs that users associated with a named entity. One can observe that the list is dominated by words like determiners, e.g., 'the', or conjunctions, e.g., 'and'. In addition, at lower ranks, the keywords 'pictures', 'lyrics', and 'estate' also appear in the list. Furthermore, the frequency distribution of the refining keywords, as a whole, also follows a power-law distribution. As presented in Section 5.3, although there is a large percentage of rare search refiners, nevertheless, there is a considerable proportion of many low frequency refiners that are commonly used to refine entity-centric search queries.

Finally, Table 5.3 lists the frequency of the grammatical categories with which the refining keywords are tagged, along with the top five most occurring refining keywords. Looking at the grammatical categories of the search refiners, it can be seen that the dominating categories are

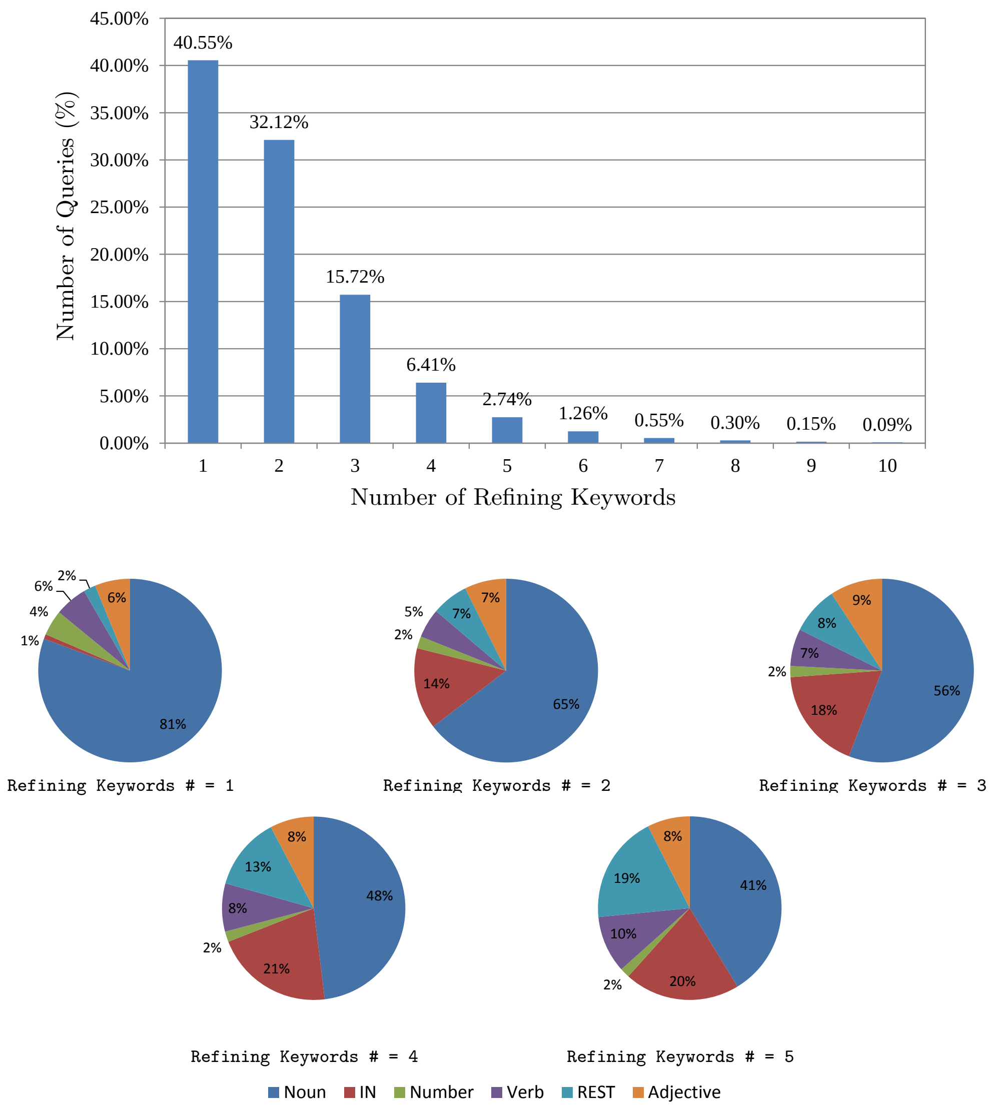


Figure 5.3: Query frequency against number of refining keywords utilised to refine results related to entities

common nouns, adjectives, and verbs (with the exception of prepositions and determiners).

The common nouns, including both singular and plural, account for almost 60% of VFQs. Nouns, as defined in Section 5.2, include words such as ‘sale’, ‘lyrics’, and ‘hotels’, and they are the most common grammatical category observed in queries. The word ‘com’ often follows entities to denote that the user wishes to access a web page related to the entity, for example, ‘Tom Cruise com’, or ‘eBay com wheels’ (the user did not include the punctuation ‘.’ when submitting the query). Furthermore, nouns in queries often denote the class of ‘things’ related to the entity, such as the refiner ‘lyrics’ for singers, albums, or songs, and ‘hotels’ or ‘estate’ associated with the names of locations.

Keyword	Query Freq.
in	30200
of	16138
for	10549
to	7647
the	6484
and	5177
on	3823
sale	3524
com	3260
lyrics	2980
is	2830
what	2688
estate	2671
how	2660
pictures	2383
free	2062
by	1977
from	1857
homes	1839
hotels	1795

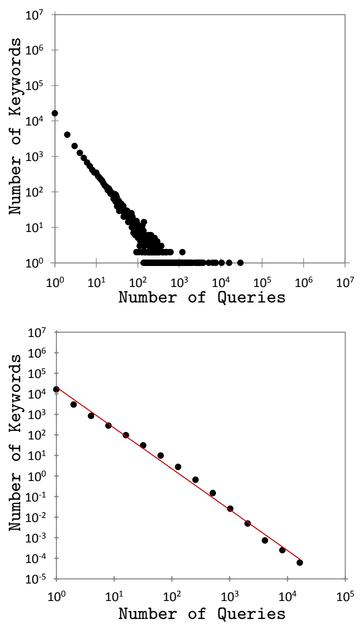


Figure 5.4: Frequency distribution of keyword refiners.

Adjectives, in their base form, reflect the attribute of the entity that the results should target, such as ‘cheap’, ‘free’, or ‘new’, in the case where the entity refers to a brand or a product name. On the other hand, when adjective refiners appear in comparative form, such as ‘better’, or superlative form, such as ‘cheapest’, it is because users often would like to access reviews related to a certain named entity.

Looking at verbs when used in base form, such as ‘buy’ in the query ‘buy PS3’, it can be seen that they often convey an *action*. When employed in the past or past participle form to refine entity-centric queries, they may convey a *state* such as ‘born’ or ‘killed’. On the other hand, verb refiners in present participle form may reflect *activities* related to named entities, such as ‘camping’ or ‘lodging’ in a location name.

Other grammatical categories, such as wh-determiners, or wh-pronouns, as well as adverbs, occur collectively in a small percentage of queries, and therefore, in this work, only nouns, adjectives, and verbs are considered as key refining keywords.

In this section, a statistical analysis of the length, frequency, and type of refining keywords is presented. In the following section, the refining keywords are ranked and analysed given the class of the entity with which they occurred.

Table 5.3: Frequency of the grammatical categories (POS) of refining keywords

POS	POS Description	Query Freq.	Top five refining keywords
NN	noun singular or mass	46.6940%	sale, com, estate, car, map.
IN	preposition or subordinating conjunction	14.0990%	in, of, for, on, by.
NNS	noun plural	12.9713%	lyrics, pictures, homes, parts, hotels.
JJ	adjective	7.1401%	free, real, new, rental, cheap.
VB	verb base form	2.5088%	buy, get, do, be, go.
CD	cardinal number	2.4375%	2006, one, 10, 100, 2005.
VBG	verb gerund or present participle	1.9925%	camping, living, lodging, driving, moving.
DT	determiner	1.7284%	the, all, no, an, that.
TO	literal ‘to’	1.5210%	to, na, how-to, toilet-to-tap, rent-to-own.
CC	coordinating conjunction	1.1003%	and, or, but, plus, versus.
VBN	verb past participle	1.0625%	used, made, wanted, born, killed.
VBZ	verb third person singular present	0.9075%	is, does, has, shows, escorts.
WRB	‘wh’ adverb	0.8268%	how, where, when, why, whenever.
VBP	verb non	0.7567%	do, are, find, have, ’m.
RB	adverb	0.7285%	not, n’t, back, up, now.
WP	‘wh’ pronoun	0.7072%	what, who, whats, whom, whos.
PRP	unknown	0.6762%	you, it, me, we, is.
VBD	verb past tense	0.6350%	was, did, lost, were, found.
PRP$	unknown, but probably possessive pronoun	0.4052%	my, your, his, her, our.
JJS	adjective superlative	0.3421%	best, largest, latest, most, cheapest.
MD	modal	0.2729%	can, will, may, ca, ’ll.
RP	particle	0.1167%	up, off, out, down, for.
FW	foreign word	0.1110%	la, pro, theatre, del, se.
JJR	adjective comparative	0.0862%	more, lower, cleaner, better, less.
WDT	‘wh’ determiner	0.0646%	that, which, whatever, what.
UH	interjection	0.0335%	oh, hello, tee, yes, wow.
EX	existential ‘there’: unstressed ‘there’	0.0278%	there, lyrics-there, theres,
RBS	adverb superlative	0.0230%	most, best, hardest, highest.
RBR	adverb comparative	0.0098%	less, closer, more, faster, longer.
LS	list item marker	0.0071%	first.
NNP	proper noun singular	0.0017%	recruits, shows, plays, state, start.
PDT	determiner	0.0015%	all.
SYM	symbol: technical symbols	0.0013%	es, e-mail, x-, e-card.
NNS—VBZ		0.0011%	matters.
VBG—NN		0.0008%	polishing.
WP$	possessive wh pronoun	0.0006%	whose.

5.5 Ranking of Search Refiners

In this work, Part-of-Speech tagging of search queries is exploited to mine the search refiners that users often employ in their entity-centric queries. In order for a search engine to utilise the harvested refiners, a ranking mechanism needs to be applied. For a target entity class, one may rely on the most frequent search refiners employed across users who submitted queries with entities that belong to the same class; however, most frequent refiners do not necessarily represent the search intent of users.

The click-through data collected in the query log for each search query can be utilised. Every click-though record in the log includes the URL of the web page that the user clicked during her/his search session. The web pages that users clicked can give an indication of the users' intent. Therefore, instead of relying solely on the frequency of search refiners in the query log to rank them, the users *search patterns* can be used. The *search pattern* of a query includes the class of the detected named entity, refining keyword, and clicked URL.

In particular, the approach to ranking search refiners in light of users' search patterns is based on the following heuristic: *there is a mutual reinforcement relationship between query search refiners of a target entity class and the clicked URLs.* Given a target entity class, a 'useful' (or 'relevant' or 'good') search refiner appears in those queries formulated by users who visited 'useful' resources (URLs) that satisfy their information need. A useful URL is one visited by many users who employed useful search refiners in their search queries. This assumption is inspired by the notion of 'authoritative' URL resources introduced by Kleinberg's HITS algorithm [59].

HITS is a link analysis method to rank documents based on the hyperlinks which connect them [59]. The algorithm assigns two scores for each web page: a hub score and an authority score based on the following assumption. A good authority web page is one to which many good hub web pages point, and a good hub web page is one that points to many good authoritative web pages. In the same spirit, for every target entity class, assuming that there are links between search refiners and the corresponding clicked URLs (denoted by a click graph), the set of refiners can be ranked according to their hub scores. Accordingly, in this work, the HITS algorithm is used to rank the search refiners detected for every target entity class, as presented in the following subsections.

5.5.1 Click Graph Construction

For each target entity class and search refiner category (Noun, Verb or Adjective), a bipartite graph $C = (V, E)$ is created. The graph consists of vertices V connected by edges E. The set of vertices V consists of two disjoints sets R and U, where the set R contains the target search refiners extracted from the query dataset (i.e., Nouns, Adjectives, or Verbs) and the set U contains the corresponding clicked URLs. An edge $e \in E$ connects a target refiner $r_i \in R$ to the corresponding clicked URL $u_i \in U$.

5.5.2 HITS Variants

To find the hub scores assigned to the refiners, and accordingly, the authority scores for the clicked URL, the following variations of the HITS algorithm are applied:

Non-Weighted HITS($H0$): using HITS as introduced by [59], each search refiner r is assigned an importance score denoted by $h(r)$, and each clicked URL u is assigned an importance score $a(u)$. Initially, the scores are set to one (i.e., $h(r) = a(u) = 1$) for all the vertices. Then, the HITS algorithm proceeds where $h(r)$ and $a(u)$ are iteratively updated until the scores converge using the formulas:

$$h(r) = \sum_{r \to u} a(u).$$

$$a(u) = \sum_{r \to u} h(r).$$

At the end of each iteration, the scores $h(r)$, and $a(u)$ are normalised.

Weighted Vertices ($H1$): in the HITS setting (H0), the scores are assigned based on the fact that a refiner r is used and the corresponding URL u is clicked regardless of how often users employed that refiner in their queries, or how often a URL is clicked when searching for a particular entity class. Therefore, in this setting, a weight is assigned to each vertex based on the click-through data, as follows. Every refiner $r \in R$ is assigned a weight W_r that equals the number of users that associated the same search refiner with the named entities that belong to the same class in their search queries. Furthermore, a weight W_u is assigned to each URL, and it is set to be the number of users who clicked that URL when searching for entities that belong to the same entity class. Accordingly, the HITS formulas are updated as follows:

$$h(r) = \sum_{r \to u} a(u) \cdot W_u$$

$$a(u) = \sum_{r \to u} h(r) \cdot W_r$$

Weighted Edges ($H2$): alternatively, in this setting, the weights are assigned to the edges rather than the vertices. Therefore, the importance scores are each weighted with W_ϵ, which is set to the number of users who included the refiner r in their queries and clicked the same URL u (ϵ refers to the edge that connects r to u) when searching for entities that belong to the same entity class.

$$h(r) = \sum_{r \to u} a(u) \cdot log(W_\epsilon + 1)$$

$$a(u) = \sum_{r \to u} h(r) \cdot log(W_\epsilon + 1)$$

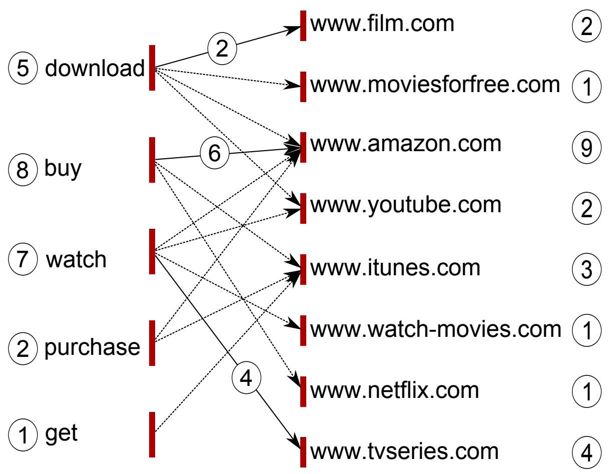
In this setting, the hub score of a refiner and the authority score of the corresponding clicked URL are first multiplied by the weight of the edge. Instead of using the weight of the edge as is, i.e., the frequency of users who employed refiner r and clicked URL u, the logarithm is utilised to dampen the weight. The reason behind this is as follows: consider the case where there is a dominating refiner and corresponding clicked URL whose edge weight is much higher than the weights of the other edges. This will drive up the hub and authority scores of the refiner and the corresponding clicked URL, and therefore will dominate the computation of the scores of other vertices in the graph.

Weighted Vertices & Edges ($H3$): finally, in this variation of the HITS algorithm, the importance scores are weighted with the vertices' and edges' weights defined previously.

$$h(r) = \sum_{r \to u} a(u) \cdot log(W_\epsilon + 1) \cdot W_u$$

$$a(u) = \sum_{r \to u} h(r) \cdot log(W_\epsilon + 1) \cdot W_r$$

To illustrate how the different variants of the HITS algorithm may affect the refiners' ranking, consider the click graph example in Figure 5.5 that consists of the entity refiners that belong to the class Film.



$H0$		$H1$		$H2$		$H3$	
watch	0.560	watch	0.541	buy	0.736	buy	0.823
download	0.560	download	0.493	watch	0.486	watch	0.368
buy	0.451	buy	0.488	download	0.361	purchase	0.309
purchase	0.391	purchase	0.472	purchase	0.293	download	0.303
get	0.129	get	0.061	get	0.075	get	0.029

Figure 5.5: Illustration of how the different settings of HITS varies the hub scores assigned to refiners

The non-weighted HITS ($H0$) ranks the refiners based on the click graph structure regardless of how often a specific refiner is used to convey a search need and regardless of how often a URL is clicked to satisfy that need. Therefore, 'download' and 'watch' are assigned the same hub scores because both have four edges, two of which are connected to highly scored authoritative URLs ('amazon' and then 'youtube'). Consequently, the ranking of the refiners is simply based on the authoritative URLs clicked by users to satisfy their search need.

Using $H1$, where the score of each URL is multiplied by the number of users who clicked that URL before measuring the score of a refiner, leads to ranking 'watch' higher than 'download'. This is because the URL 'tvseries' has a higher click rate than the URL 'film'. In other words, $h(watch) = (9 \times a(amazon)) + (2 \times a(youtube)) + (1 \times a(watch-movies)) + (4 \times a(tvseries))$ that results, at the last iteration, in the hub score of 'watch' being equal 0.541; however, $h(download) = (9 \times a(amazon)) + (2 \times a(youtube)) + (1 \times a(moviesforfree)) + (2 \times a(film))$ results in $h(download) = 0.493$.

In the HITS setting $H2$, the measure of ranking favours the weight of the edges, i.e., the more a specific URL is clicked with a specific refiner, the higher is the hub score of that refiner. Therefore, as shown in Figure 5.5, $H2$ ranks the refiner 'buy' at the top rank because the weight of its edge, 'amazon', has a weight of six (that is, dampened with the logarithm). In the figure, the dotted arrow denotes an edge with weight one, whereas a solid arrow denotes a weight higher than one with the weight shown over the arrow.

Finally, when the HITS setting that uses both the vertex weight and the edge weight ($H3$) is used, the rank for 'purchase' becomes higher than that for 'download', which is not the case when using the settings $H0$, $H1$, and $H2$. Although the click graph is small, and accordingly the difference in hub scores among refiners is small, the rankings vary for each setting.

In the following section, HITS is implemented on a larger-scale click graph for each search refiner type and target entity class.

5.5.3 Experimental Results

Using the same dataset upon which the statistics in Subsection 5.4 are drawn (consisting of 41,385 named entities that appear in 292,509 VFQs), a click graph is created for each search refiner type and target entity class, that is, Location, Person, Organisation, Educational Organisation, Film, and Software. This results in a total of 18 click graphs upon which the four different variants of the HITS algorithm are tested.

Before building the click graph, WordNet [40] is used to map the different forms of the words (e.g., 'logging', 'logged') to a base form (e.g., 'log'). Furthermore, every URL is processed to include only the website name rather than the full URL. For example, for the noun refiner 'bio' extracted from the query 'Nicolas Cage bio', the clicked URL becomes `http://en.wikipedia.org` that is processed to exclude the web page path `/wiki/Nicolas_Cage`. Table 5.4 lists the number of vertices that include the refining keywords and corresponding clicked URLs, as well as the number of edges that connect the refiners to the clicked URLs.

Tables 5.5, 5.6, and 5.7 present the top ten refining keywords of Nouns, Adjectives, and Verbs that resulted from running the HITS variations $H0$, $H1$, $H2$, and $H3$ over each of the click graphs. In addition, for comparison purposes, the top ten refiners for each entity class ranked by decreasing frequency are presented, and denoted by $Freq$.

There is no gold standard ranking of refiners to which the results are compared, and the objective here is not to find one; alternatively, a comparative analysis of how HITS variants can be utilised to reach different refiner rankings is presented. Although it can be seen that there are some common refiners between the HITS rankings and the ones based on query frequency, there are some differences in the individual refiners and their placements.

Consider the noun refiners 'sale', 'map', and 'hotel' of the Location class in Table 5.5. When using the HITS setting $H0$, which ranks the refiners based on the click graph structure

Table 5.4: Number of vertices (refiners and URLs) and connecting edges that constitute each click graph

	Class	Refiner #	URL #	Edge #
Noun	Person	3,132	22,547	34,589
	Location	6,607	79,585	114,715
	Org	4,139	29,657	43,872
	EduOrg	438	1,735	2,037
	Film	961	3,975	5,064
	Software	632	2,546	3,805
Adjective	Person	627	4,942	6,290
	Location	1,422	18,534	21,598
	Org	799	4,935	5,848
	EduOrg	74	137	137
	Film	166	512	574
	Software	100	448	489
Verb	Person	535	3,328	4,311
	Location	1,066	12,323	14,060
	Org	635	4,122	5,022
	EduOrg	56	134	138
	Film	155	508	556
	Software	104	623	715

regardless of the weights of the vertices (i.e., refiners and URLs), 'sale' is placed at the top rank, whereas 'map' and 'hotel' are placed in the fourth and fifth ranks, respectively. This reveals that 'sale' is connected to many highly-scored authoritative URLs that were visited by users who employed that refiner in their Location-centric queries to satisfy their information need; on the other hand, the refiner 'map' is placed at the top rank when using the weighted HITS variants $H1$, $H2$, and $H3$.

The refiner 'hotel' does not appear in the top ten noun refiners when using $H1$, although it is the second most frequently used noun refiner based on frequency (Freq). This reveals that when compared with, say, 'city', 'hotel' is connected to less authoritative URLs. In particular, when refining the Location-centric queries with 'city' the top most clicked URL is '`http://en.wikipedia.org`', whose weight is 1,191; however, in the case of the refiner 'hotel', the most clicked URL is '`http://www.wunderground.com`' with a weight of 566.

When employing $H2$ that depicts the frequency of users who employed the same refiner to convey their search need and clicked the same URL to satisfy that need, the rank of the refiner 'hotel' increases to second, and it becomes third when using $H3$ that utilises both the vertex weight and edge weights.

In addition to the different variants of HITS, the different rankings of noun, verb, and adjective refiners is affected by the following factors:

1. Search trend, i.e., search queries related to the most trending events or entities. In the

case of verbs, mentions of entities related to the class Person are often associated with verbs, such as 'married' as in the query 'Tom Welling married', or the verb 'die', such as 'Tiger Woods father died'.

2. Semantic space of the target entity class. For example, looking at the noun refiners of the class Person, it can be seen that the most highly ranked refiner is 'lyrics' to refine those results related to singers, such as 'Josh Turner lyrics', as apposed to, e.g., 'movie' that is often associated with actors, such as the query 'Tom Cruise movies', or 'book' that is often associated with authors, e.g., 'Wilbur Addison Smith books'.

As discussed in Section 5.4, although a majority of queries are refined by a single word, in some cases the full phrase that refine the entity-centric query is more informative, for example, 'real estate' as apposed to the adjective 'real' and the noun refiner 'estate'.

Finally, by examining the false positives of search refiners, it can be seen that they often search keywords that are part of the named entity. For example, in the query 'Going to California lyrics', where the detected named entity is 'California', the keyword 'going' is mistakenly detected as a verb refiner, although it is part of the named entity. Such is the case of the verb 'ask', which is part of the named entity 'Ask Jeeves' that refers to a website name.

In the case of noun refiners, consider the refiner 'flag'. It occurs in many queries with a location name with the intent to view the flag of a country, such as 'Hong Kong's flag', 'Venezuela flag', 'flag of Hungary'; however, the same refiner occurs in the query 'Six Flags San Antonio', where 'flags' is part of the named entity Six Flags, which is a theme park in San Antonio, Texas. Such is also the case for 'Detroit news', which is the name of an entity (i.e., newspaper), rather than a location name followed by the refiner 'news'.

Another type of false positive occurs when an entity is ambiguous, i.e., it belongs to more than one entity class. For example, in the query 'Hoffa search', the entity 'Hoffa' is annotated by DBpedia as a Film that is biographical and is based on the life of 'Jimmy Hoffa' (Politician). This results in having the refiner 'search' appear in the top Film noun refiners.

In the following section, a comparative analysis of the ranking based on the raw frequencies versus the ranking induced by HITS variants is presented. In Section 5.7, the list of refiners identified is discussed in greater detail to highlight the possible intentions behind refining entity-centric queries with the identified refineries.

Table 5.5: Top ten general nouns detected in entity-centric search queries

		Rank									
		1	2	3	4	5	6	7	8	9	10
Location	$H0$	sale	estate	home	map	hotel	property	newspaper	picture	history	car
	$H1$	map	city	history	newspaper	picture	state	sale	photo	news	government
	$H2$	map	hotel	sale	newspaper	estate	city	history	news	picture	flag
	$H3$	map	newspaper	hotel	flag	estate	city	news	picture	apartment	sale
	Freq	map	hotel	newspaper	sale	news	estate	attack	weather	job	home
Person	$H0$	picture	pic	lyric	photo	video	song	movie	bio	biography	music
	$H1$	lyric	picture	song	video	photo	music	movie	wife	pic	album
	$H2$	lyric	picture	pic	song	video	photo	music	movie	biography	bio
	$H3$	lyric	song	picture	biography	video	music	movie	wife	bio	book
	Freq	lyric	picture	pic	movie	video	song	photo	music	biography	bio
Edu Org	$H0$	baseball	football	team	basketball	softball	sport	swimming	schedule	hockey	site
	$H1$	baseball	football	team	sport	softball	basketball	schedule	soccer	swimming	stat
	$H2$	baseball	football	team	softball	basketball	sport	schedule	soccer	swimming	page
	$H3$	baseball	football	team	softball	sport	schedule	soccer	stat	basketball	online
	Freq	baseball	online	football	program	class	softball	sport	website	course	hospital
Org	$H0$	parts	accessory	com	store	lyric	code	engine	dealer	car	service
	$H1$	com	video	news	online	music	lyric	game	picture	code	store
	$H2$	lyric	parts	com	music	code	video	song	store	coupon	game
	$H3$	news	com	music	lyric	video	song	weather	online	page	game
	Freq	lyric	news	music	parts	com	code	store	video	coupon	game
Film	$H0$	movie	soundtrack	game	picture	book	trailer	story	music	review	song
	$H1$	search	dig	movie	trailer	soundtrack	book	picture	character	story	summary
	$H2$	movie	soundtrack	search	book	cast	dig	clip	game	trailer	review
	$H3$	dig	search	movie	trailer	headline	preview	quiz	remains	singer	scientist
	Freq	dig	movie	search	fish	dress	motel	soundtrack	game	book	lyric
Software	$H0$	game	cheat	code	download	online	walkthrough	guide	pc	movie	demo
	$H1$	game	cheat	code	guide	tip	walkthrough	pc	download	character	map
	$H2$	game	cheat	code	walkthrough	pc	hint	guide	video	download	tip
	$H3$	cheat	code	game	pc	walkthrough	hint	video	tip	guide	screenshot
	Freq	game	cheat	code	walkthrough	download	online	guide	video	pc	map

Table 5.6: Top ten adjectives detected in entity-centric search query

		Rank									
		1	2	3	4	5	6	7	8	9	10
Location	$H0$	real	rental	new	good	free	local	commercial	public	private	top
	$H1$	real	secret	rental	new	public	local	good	private	old	french
	$H2$	real	rental	new	secret	local	cheap	good	public	private	commercial
	$H3$	secret	real	rental	new	cheap	local	public	private	french	atomic
	Freq	secret	real	rental	cheap	public	good	new	free	local	current
Person	$H0$	nude	naked	topless	free	new	gay	pregnant	pic	sexy	hot
	$H1$	nude	naked	topless	free	new	pregnant	gay	hot	sexy	pic
	$H2$	nude	naked	topless	free	pregnant	pic	new	gay	sexy	hot
	$H3$	nude	naked	topless	free	pregnant	sexy	hot	pic	gay	fat
	Freq	nude	naked	free	new	pregnant	gay	good	official	pic	topless
Edu Org	$H0$	medical	financial	personal	good	athletics	academic	ecampus	high	free	veterinary
	$H1$	financial	medical	ecampus	high	athletics	personal	good	ok	average	dental
	$H2$	financial	ecampus	medical	high	personal	good	athletics	academic	ok	free
	$H3$	financial	ecampus	medical	high	ok	athletics	personal	good	tri-cities	blind
	Freq	financial	medical	ecampus	high	athletics	average	dental	ok	blind	regional
Org	$H0$	free	new	top	funny	own	cheap	more	corporate	audio	good
	$H1$	free	new	top	front	small	local	audio	private	antitrust	wireless
	$H2$	free	top	new	funny	front	antivirus	wireless	active	own	small
	$H3$	free	top	antivirus	active	wireless	new	front	antitrust	direct	local
	Freq	free	net	top	new	wholesale	real	wireless	funny	dental	local
Film	$H0$	free	new	last	big	short	amazing	official	funny	easy	original
	$H1$	free	new	official	funny	easy	last	amazing	big	short	original
	$H2$	free	new	official	funny	easy	big	last	short	amazing	original
	$H3$	ok	married	free	new	official	funny	easy	last	amazing	big
	Freq	free	ok	married	real	big	seventh	last	new	good	naked
Software	$H0$	free	grave	true	new	super	good	nude	final	official	white
	$H1$	free	new	good	final	unlockable	official	fake	old	true	super
	$H2$	free	true	official	new	fake	old	grave	good	final	unlockable
	$H3$	free	official	fake	old	true	new	good	final	unlockable	super
	Freq	free	super	new	top	downloadable	good	private	secret	key	official

Table 5.7: Top ten verbs detected in entity-centric search query

		Rank									
		1	2	3	4	5	6	7	8	9	10
Location	$H0$	use	live	camp	buy	find	sell	lodge	go	drive	die
	$H1$	use	camp	live	go	lodge	drive	shoot	walk	eat	surf
	$H2$	use	camp	live	lodge	buy	escort	drive	run	find	move
	$H3$	use	camp	lodge	live	drive	escort	move	walk	rent	eat
	Freq	use	camp	live	escort	lodge	show	find	make	hold	drive
Person	$H0$	die	go	born	marry	say	get	come	lose	listen	live
	$H1$	ask	marry	dot	shower	die	play	lose	come	be	believe
	$H2$	ask	born	marry	die	hit	play	come	lose	go	say
	$H3$	ask	marry	dot	born	hit	die	come	play	write	dunk
	Freq	ask	lift	born	die	live	listen	go	give	color	marry
Edu Org	$H0$	build	watch	locate	shoot	haze	offer	swim	graduate	continue	mount
	$H1$	mount	continue	fight	shoot	locate	offer	graduate	build	watch	haze
	$H2$	mount	fight	continue	shoot	build	locate	haze	offer	swim	graduate
	$H3$	mount	shoot	fight	continue	build	locate	haze	offer	swim	graduate
	Freq	mount	shoot	offer	locate	graduate	continue	find	swim	wrestle	build
Org	$H0$	use	buy	make	see	show	help	chat	live	play	add
	$H1$	use	clip	create	buy	miss	help	find	make	show	add
	$H2$	use	buy	see	sell	track	hook	clip	make	get	help
	$H3$	clip	miss	work	create	change	resume	use	scan	protect	buy
	Freq	use	clip	track	go	help	get	buy	chat	add	pay
Film	$H0$	color	play	call	draw	groom	print	sink	mean	roar	know
	$H1$	cast	sound	make	enter	finish	move	set	film	remake	listen
	$H2$	cast	sound	make	enter	move	remake	finish	set	film	listen
	$H3$	cast	sound	enter	make	finish	move	remake	set	film	listen
	Freq	cast	color	sound	cheat	make	die	enter	play	lose	find
Software	$H0$	cheat	help	hack	unlock	flash	make	buy	get	play	work
	$H1$	cheat	unlock	flash	make	buy	play	download	close	happen	walk
	$H2$	cheat	help	unlock	play	flash	buy	make	download	close	happen
	$H3$	cheat	unlock	play	flash	buy	make	download	close	happen	walk
	Freq	cheat	help	buy	make	play	hack	find	crack	want	get

5.6 Rank Comparison of Search Refiners

In order to quantify the differences in ranking based on frequency against the ranking produced using the different HITS weighting settings, the following three nonparametric correlation measures are employed:

O-Measure: given two lists L_1 and L_2 of length k, this measure quantifies the proportion of overlap between two lists, and it is given by the formula:

$$O = \frac{|\{L_1 \bigcap L_2\}|}{k}.$$

G-Measure: unlike O-Measures, this measure considers not only the overlapping elements of the two lists, but also the ranks of the elements in the lists. Spearman's footrule metric measures the distance between two permutations as the sum of the absolute values of the differences between the ranks of L_1 and L_2. Fagin et al. [38] presented a variation of Spearman's footrule metric to include the case where $O < 1$, i.e., there may be elements in one list that are not in the other. Fagin et al. [38] assigned $k+1$ as the rank of all the missing elements, and accordingly, the G-Measure is defined as follows:

$$G = 1 - \frac{F'}{F'_{max}}, \text{where}$$

$$F'_{max} = k(k+1), \text{and}$$

$$F' = 2(k-c)(k+1) + \sum_{i \in \{L_1 \bigcap L_2\}} |\tau_1(i) - \tau_2(i)| - \sum_{i \in \{L_1 \setminus L_2\}} \tau_1(i) - \sum_{i \in \{L_2 \setminus L_1\}} \tau_2(i),$$

where $\tau_j(i)$ denotes the rank of element i in the list $L_j (j = 1, 2)$, and $\{L_1 \setminus L_2\}$ is the relative complement of L_2 in L_1, that is, the set of all elements in L_1 that are not in L_2 ($\{L_2 \setminus L_1\}$ is the converse). Lastly, c is the number of elements that occur in L_1 and L_2, that is, $\mid \{L_1 \bigcap L_2\} \mid$.

M-Measure: introduced in [6], this assigns a higher similarity score to the lists when they share elements at the top ranks. This measure is defined by

$$M = 1 - \frac{M'}{M_{max}}, \text{where}$$

$$M_{max} = 2(\sum_{i=1}^{k}(\frac{1}{i} - \frac{1}{k+1})), \text{and}$$

$$M' = \sum_{i\in\{L_1 \bigcap L_2\}} |\frac{1}{\tau_1(i)} - \frac{1}{\tau_2(i)}| + \sum_{i\in\{L_1 \setminus L_2\}} |\frac{1}{\tau_1(i)} - \frac{1}{(k+1)}| + \sum_{i\in\{L_2 \setminus L_1\}} |\frac{1}{\tau_2(i)} - \frac{1}{(k+1)}|.$$

Table 5.8 lists the similarity scores using the O, G, and M measures between each of the HITS variants and the query frequency of the target refiners at rank $k = 50$.

One of the main factors that affect the similarity scores is the total number of refiners detected per entity class, from which the top 50 are compared. For example, for the class Educational Organisation, only 56 verb refiners and 74 adjective refiners are detected, and therefore the difference between the scores using the variation of HITS settings and query frequency is not large (i.e., 88% for verbs and 74% for adjectives over all the HITS variants).

When looking at the average similarity scores of the O measure, it can be seen that the overlap percentage lies between 49% and 67% for all search refiners. Moreover, at least 33% of the refiners among the various HITS rankings are different from those produced solely by query frequency. Thus, utilising HITS over click graphs reveals that there are some search refiners that do not appear in the top 50 most frequent refiners in the list of VFQs.

The O measure shows the size of the overlap regardless of the rank of the refiners in the top 50 lists. On the other hand, the G measure incorporates both the overlap size and the rank of the refiners in the lists. The similarity scores using these measures range between a minimum of 0.52 and a maximum of 0.64. Thus, although there are some refiners that appear in both rankings (i.e., HITS and query frequency), they are placed in different ranks.

Lastly, looking at the noun refiners, it can be seen that the M measure assigns higher similarity scores when the common refiners of both rankings appear at higher ranks in the top 50 refiners. In this measure, the maximum average similarity score increases to reach 71%, thus indicating that when using the weighted edges HITS ($H2$), the common refiners appear at higher ranks in both lists. However, as the total number of refiners decreases, as is the case with verbs and adjectives, the difference between the average score of the O measure using $H0$ and the similarity score of M when using $H1$ is relatively small.

In this section, using the three measures, O, G, and M, the ranking of refiners induced by each setting of the HITS algorithm is compared to that based on the refiners frequency in the query log. The results reached showed that as the different weighting schema employed by HITS varies the ranking of the refiners vary accordingly. In the following section, an in-depth analysis of the type of refiners associated with each class of named entities is presented.

Table 5.8: Comparison of the top 50 rankings of search refiners against query frequency

		Noun					Verb					Adjective				
		H0-F	H1-F	H2-F	H3-F	AVG	H0-F	H1-F	H2-F	H3-F	AVG	H0-F	H1-F	H2-F	H3-F	AVG
Location	O	0.62	0.48	0.66	0.52	0.57	0.58	0.38	0.60	0.38	0.49	0.66	0.44	0.64	0.48	0.56
	G	0.63	0.50	0.70	0.54	0.59	0.64	0.47	0.64	0.46	0.55	0.70	0.47	0.61	0.48	0.57
	M	0.42	0.54	0.80	0.67	0.61	0.69	0.67	0.78	0.67	0.70	0.56	0.59	0.56	0.73	0.61
Person	O	0.74	0.60	0.72	0.56	0.66	0.70	0.54	0.60	0.54	0.60	0.70	0.62	0.68	0.62	0.66
	G	0.75	0.63	0.76	0.60	0.69	0.73	0.50	0.55	0.50	0.57	0.76	0.71	0.75	0.67	0.72
	M	0.68	0.76	0.88	0.70	0.75	0.46	0.48	0.61	0.53	0.52	0.82	0.80	0.80	0.75	0.79
Edu Org	O	0.48	0.48	0.50	0.46	0.48	0.88	0.88	0.88	0.88	0.88	0.74	0.74	0.74	0.74	0.74
	G	0.47	0.49	0.47	0.49	0.48	0.78	0.82	0.79	0.80	0.80	0.69	0.79	0.75	0.79	0.76
	M	0.57	0.57	0.56	0.57	0.57	0.47	0.75	0.70	0.77	0.67	0.56	0.87	0.76	0.81	0.75
Org	O	0.54	0.46	0.62	0.44	0.52	0.56	0.48	0.46	0.34	0.46	0.62	0.30	0.56	0.30	0.45
	G	0.55	0.48	0.64	0.48	0.54	0.53	0.52	0.54	0.35	0.49	0.55	0.34	0.50	0.33	0.43
	M	0.37	0.40	0.71	0.48	0.49	0.53	0.63	0.59	0.28	0.51	0.61	0.54	0.60	0.51	0.56
Software	O	0.70	0.48	0.60	0.48	0.57	0.58	0.58	0.60	0.58	0.59	0.66	0.74	0.68	0.74	0.71
	G	0.77	0.57	0.70	0.57	0.65	0.52	0.51	0.52	0.51	0.52	0.52	0.53	0.52	0.53	0.52
	M	0.87	0.76	0.84	0.56	0.76	0.68	0.58	0.67	0.56	0.62	0.55	0.55	0.51	0.49	0.53
Film	O	0.60	0.44	0.54	0.46	0.51	0.60	0.68	0.62	0.68	0.65	0.64	0.66	0.66	0.66	0.66
	G	0.56	0.49	0.58	0.41	0.51	0.51	0.55	0.54	0.55	0.54	0.53	0.53	0.53	0.57	0.54
	M	0.37	0.49	0.48	0.55	0.47	0.32	0.57	0.58	0.58	0.51	0.53	0.50	0.52	0.51	0.51
Average	O	0.61	0.49	0.61	0.49		0.65	0.59	0.63	0.57		0.67	0.58	0.66	0.59	
	G	0.62	0.53	0.64	0.52		0.62	0.56	0.60	0.53		0.62	0.56	0.61	0.56	
	M	0.54	0.59	0.71	0.59		0.52	0.61	0.65	0.57		0.60	0.64	0.63	0.64	

5.7 Entity Refiners in Depth

In order to gain more insights, five randomly chosen queries for each of the top 20 search refiners per target entity class are checked manually. The manual analysis reveals that there are some common traits that these refiners possess across the different entity classes. In the following subsections, the general features observed on the usage of these search refiners are presented, along with examples of queries from the log file.

5.7.1 Noun Refiners

When common nouns are used to refine entity-centric queries, they often appear as one of the following intents:

1. To obtain listings of instances related to the entity. Some examples are a list of hotels in a location, movies of an actor, cast of a movie, products of a company, soundtracks of a film.

    - '*hotels* Irving Texas' (Location).
    - '*lyrics* by Shirley Caesar' (Person).
    - 'Walmart *stores*' (Org).
    - 'University of Georgia *football coaches*' (EduOrg).
    - 'Runescape *tips*' (Software).
    - 'Stir of Echoes *cast*' (Film).

2. To further specify the entity class when it may fall into more than one entity category. For example, Harry Potter the movie, the video game, or the book.

    - 'Wrightsville Pennsylvania *Government*' (Location).
    - 'Ann Nesby *singer*' (Person).
    - 'BMW *dealer*' (Org).
    - 'Bad Boys, *Movie*' (Film).

3. To specify the type of resource to be accessed related to the entity. For example, a *video* to watch a movie trailer; a *document* to read a song's lyrics, a person's biography, a video game walkthrough, or an *audio* to listen to a song.

- ‘China *map*’ (Location).
- ‘Shakira *videos*’ (Person).
- ‘Gucci Jewelry *catalog*’ (Org).
- ‘Bemidji High School *website*’ (Edu Org).
- ‘Masters of the Universe *trailer*’ (Film).
- ‘Tekken *video*’(Software).

5.7.2 Verb Refiners

The refiners whose POS is a verb may fall in the following categories:

1. To specify a virtual activity that can be performed on the web. For example, *buy* products online, *log in* or *sign up* into an account, *download* or *play* a video game online, *listen* to a song, or *watch* a video.
    - ‘*listen* Christina Aguilera’ (Person).
    - ‘AOL *chat* (Org).
    - ‘*watch* Matrix online’ (Film).
    - ‘*downloading* Halo 2 tunes’ (Software).
2. To specify a real-world activity related to an entity. For example, *lodging* or *camping* in a location, or *applying* to a university.
    - ‘*camping* in Iowa’ (Location).
    - ‘Stevie Wonder *playing* drums’ (Person).
    - ‘*install* motion sensor in Suzuki’ (Org).
    - ‘*graduated* from Mission College (Edu Org).
    - ‘Shrek *print* and *color*’ (Film).
3. To specify an action associated with a noun search refiner, such as finding jobs, stores, or restaurants in a Location.
    - ‘*find* job Singapore’ (Location).
    - ‘names of the books Edward Bloor *wrote*’ (Person).
    - ‘art classes *offered* at Gordon College’ (Edu Org).

- '*listen* to theme song from Jaws' (Film).
- 'Battlefield 2 map *unlock*' (Software).

4. To specify an event related to an entity. For example, *death* of a person, or an actor getting *married*.
    - '*shooting* in Virginia' (Location).
    - 'Juwan Howard *arrested*' (Person).
    - 'football player *killed* Albany State University' (Edu Org).

5.7.3 Adjective Refiners

Finally, when examining adjectives, it can be seen that they often appear in queries in the following manner:

1. To specify an attribute of the entity that the user included in the query. For example, *free* video game, movie, or song.
    - 'Britney Spears *pregnant*' (Person).
    - 'MTV in *Spanish*' (Org).
    - '*real* Silent Hill' (Film).
    - 'Sidebars *unlockable*' (Software).
2. To specify an attribute of another search refiner (i.e., noun, verb). For example, *official* trailer of a movie or a video game, a *personal* tutor in a university, a *new* release of a movie sequel, or a *local* grocery shop in a location.
    - 'Las Vegas *rental* homes' (Location).
    - '*new* Ciara lyrics' (Person).
    - 'AOL games *free*' (Org).
    - 'University of Louisville *medical* center' (Edu Org).
    - 'Godzilla *last* movie' (Film).
    - '*free* Kingdom Hearts II music' (Software).

Adjectives associated with Educational Organisation are often related to a department or module (e.g., 'medical', 'financial', or 'higher education'). On the other hand, for the case of entity classes that often involve a charging price, such as movies, video games, software, or products, the top ranked adjective refiner is 'free', regardless of the HITS variant used.

5.8 Discussion

In this chapter, an exploratory analysis of the linguistic structure of entity-centric queries was presented. More specifically, this chapter presented an in-depth analysis of the search refiners, i.e., the remainder of the query after excluding the detected named entity string. Three main categories of search refiners: nouns, adjectives, and verbs were detected via Part-of-speech tagging search queries. This analysis led to the following findings:

- The click graphs created using the click-through data from the query log can be utilised to implement different variants of the HITS algorithm that allow the ranking of search refiners to be based on the resources that the users examined to satisfy their search need.

- The different variants of weighted HITS algorithm result in different rankings of the refiners that consider the following:

  (a) The click frequency of a URL and the frequency with which a refiner is used to convey a search need related to the target entity class (Weighted Vertices $H1$).

  (b) The frequency of a common search pattern related to a target entity class represented by the weight of edges in the click graph. The weight of the edge is set to the frequency with which a specific refiner is utilised, to convey a search need related to the target entity class and the corresponding clicked URL satisfying that need (Weighted Edges $H2$).

  (c) The frequency of both the refiner and the URL within a target entity class weighted by the count of having a refiner and URL commonly utilised within the target entity class (Weighted Edges $H3$).

- As a result of applying the HITS algorithm to the click graphs, a ranking of the most useful resources that the users visited to satisfy their information need can be obtained. For instance, Table 5.9 lists the top five URLs clicked for each of the search refiner nouns, verbs, and adjectives for the two classes Film and Software. Nevertheless, the ranking of the resources is outside the scope of this thesis.

- Although there are some false positives for the identified refiners, there is still a considerable proportion of refiners that can be utilised to reveal the search intent behind users' entity-centric search queries. These search refiners possess some common traits across different entity classes.

Table 5.9: Top five URLs associated with Film and Software entities

	Noun	Verb	Adjective
Film	imdb.com	scholastic.com	imdb.com
	us.imdb.com	harrypotter.warnerbros.com	us.imdb.com
	movies.yahoo.com	picnic-world.info	imdb.com
	imdb.com	bizarre-rituals.com	funwavs.com
	rottentomatoes.com	.blogspot.com	amazon.com
Software	gamespot.com	zone.msn.com	gamespot.com
	cheats.ign.com	download.com	cheats.ign.com
	ps2.ign.com	en.wikipedia.org	cheatscodesguides.com
	pccheatz.com	runescape.getnow.be	ps2.ign.com
	xbox.ign.com	gamespot.com	pccheatz.com

Chapter 6

Query Log-Driven Named Entity Taxonomy

6.1 Overview

In this chapter, a named entity taxonomy derived from a search query log is presented. First, in Section 6.2, a review of the taxonomies used for search query classification is presented. Section 6.3 presents the distribution of the named entity classes by manually annotating a random sample of queries selected from the query log. Manual annotation is compared to that achieved automatically using DBpedia as described in Section 6.4. In Section 6.5, the derived taxonomy of named entity classes is presented along with a set of possible search intents behind search queries containing the entities that belong to that class. Finally, Section 6.6 summarises the main findings of the analysis of the distribution of entity classes found in the query log.

6.2 Taxonomies of Search Queries

Previous work on query log analysis presented different approaches to query classification. In general, the definition of the taxonomy to which search queries are classified is based on one of the following: (i) the domain to which a search query belongs, e.g., Entertainment, Fashion, News, or Politics; (ii) the user's search goal behind the submitted search query; and (iii) the class of the named entity that a search query may contain.

Early work on named entity recognition and classification used coarse-grained named entity classes, such as Person, Location, Organisation, which have been referred to as "enamex" since the Message Understanding Conference (MUC-6). For domain-specific applications, new named entity classes were studied, such as names of Diseases, Projects, Proteins, and Addresses. Sekine and Nobata [92] designed a hierarchical named entity taxonomy consisting of 200 types of named entities that target open-domain applications, which consist of very specific named entity types, such as Tunnel, Bridge, Spaceship, and Color.

When named entity recognition was first applied to search queries, the classes chosen were either fine-grained classes, such as Athlete, Actor, as in [49], Cities, Countries and Drugs as in [80], Car Models as in [33], and Movies, Video Games, and Books as in [44], or more general classes, such as Location and Person as in [80]. In this thesis, for evaluation purposes, the choice of the target entity classes is based on the most frequent classes of the detected named entities that exactly match a DBpedia result and was classified by the DBpedia ontology (see Chapter 4, Section 4.4).

In this chapter, a study of the types of named entities that users would generally employ in search queries is undertaken, and an estimation of each class frequency is presented. Thus, in the following section, the classes of named entities that are correctly extracted by the QNER method presented in Chapter 3 are examined manually and compared with the DBpedia ontology classes for the exactly matched instances.

6.3 Manual Annotation of Candidate Named Entities

Using the evaluation sample used to evaluate the QNER approach presented in Chapter 3, each correctly extracted candidate named entity is annotated manually with a named entity class. In this set, all extracted named entities were included, regardless of how accurate the boundaries. Based on the context of the named entity in the query (if it exists) and the snippets of the top eight results related to the query, each entity is annotated with a named entity class. Table 6.1 presents the classes of the named entities and their percentages detected in the 1,000 query sample.

As demonstrated in the table, the most frequent named entity type observed is ‘Location’, which includes the names of countries, cities, towns, districts, and provinces. In the sample, Location named entities are usually associated with entities of type Facility, such as ‘Riviera Maya Hotel’. All named entities, such as hotels, resorts, hospitals, universities, and restaurants are annotated manually as ‘Facility’. In addition, named entities of the type ‘Person’ include all athletes, actors, historical figures, TV presenters, and people’ names found in Linkedin or Facebook. Moreover, the entities that are related to ‘TV’ include channels, series, and programs, whereas Music includes album, song, and band names. The list of entity types in Table 6.1 represents 96.89% of the candidate named entities extracted from the 1,000 query sample. Approximately 3% are labelled as ‘Other’, because of their very small overall percentage

Table 6.1: Manual analysis of candidate named entities classes

NE Class	Example	%
Location	Alabama, Australia, Georgia	26.8293
Facility	Casa Grande Restaurant, Hard Rock Casino & Resort	18.1818
Company	Hummel Bros, Lexus, Nelson Industries	15.5211
Person	Ben Affleck, Christina Aguilera, Donna Lopez	10.7539
Product	iPod, Motorola 6208	7.0953
Website	Careerbuilder, Gamespot, MSN	3.6585
Software	Windows XP, PowerPoint, Corel Photo Album 6	3.2151
TV show/channel	abc, American Idol, Sienfield	2.439
Musical Entity	Arctic Monkeys, Blue, Pink Floyd	2.3282
Medical Entity	Crohns, Dawson's, Juvenile Diabetes	1.7738
Video Game	Halo 2, Final Fantasy, God of War	1.2195
Newspaper/Magazine	Graham Leader, the Detroit	1.1086
Sport	Major League Baseball All-Star game, Nascar 1983	1.1086
Movie	Curse of the Omen, Little Mermaid, Match Point	0.8869
Book	Marvel Mangaverse	0.7761
Other	Animal, Scientific Concept, Language	3.1042

compared with the more frequent named entity types. The 'Other' category includes instances such as:

(i) Animal breeds, such as 'Bluetick Coonhounds'.

(ii) Languages and Nationalities that are proper nouns, such as 'Latin', 'English', 'Chinese'.

(iii) Religions, which are also proper nouns detected as candidate named entities.

In order to compare the manual annotation to the annotation derived from the DBpedia ontology, the class of each DBpedia result whose string matches a candidate named entity string is analysed in the following section.

6.4 DBpedia Annotation of Extracted Candidate Named Entities

The structure of the DBpedia ontology is hierarchical, for example, the extracted named entity 'Vivien Cardone' is classified as an Actor that is an Artist, that is a Person; and the entity 'London' belongs to the class Settlement, which in turn belongs to Place, that is a Populated Place.

Table 6.2 presents the percentages of named entity instances classified by the DBpedia ontology for the 24% instances whose manual evaluation is an 'Accurate NE' with DBpedia results whose label exactly matches the extracted named entity (see Chapter 3 Table 3.5).

By examining the classified instances, it can be seen that only 58% of all instances with an exactly matching DBpedia result were classified into the DBpedia ontology. Instances such as 'Minnesota Department of Agriculture', 'Fashion Expo', 'Qwest Wireless', and 'Garden City

Table 6.2: Exactly matched DBpedia results classification

Class Name	%	Class Name	%	Class Name	%
Administrative region	13.475	Athlete	1.413	Beverage	0.353
City	7.801	Broadcast	1.060	Body of Water	0.353
Company	6.738	Educational Institution	1.060	Disease	0.353
Artist	4.255	Musical Work	1.060	Film	0.353
Band	3.180	Website	1.060	Language	0.353
Actor	2.827	Colour	0.707	Mountain Range	0.353
Television Show	2.473	Comics Character	0.707	Organisation	0.353
Country	2.120	Office Holder	0.707	Reptile	0.353
Place	2.120	Software	0.707		
Person	1.767	Album	0.353		
		Total			58.657

Hotel' have a DBpedia result whose label exactly matches the extracted candidate named entity, yet they are not classified.

Furthermore, other candidate named entities can belong to more than one class in the DBpedia ontology. This is because the named entities are analysed or evaluated without considering their context, which may resolve such ambiguity. Table 6.3 shows the DBpedia classes for the instances of named entities of the type Location, Person, Facility, and Company. Under the manual annotation for Location, in addition to Administrative Region, City, and Country, there were entities such as 'Las Vegas' that refer to a Location or a TV series. In addition, for a query such as 'Cheap Tickets from OKC to ATL', the candidate named entity 'ATL' is classified by DBpedia as a Film or Band Name, whereas OKC refers to Oklahoma and ATL refers to Atlanta. Similarly, in the query 'Sticky Fingers sauce', the entity 'Sticky Fingers' refers to a diner in USA as well as an album released by the Rolling Stones band.

For every identified entity class by manual annotation, the corresponding DBpedia ontology classification is checked. Table 6.4 lists the percentage of instances that are correctly classified, misclassified, and not classified by the DBpedia ontology. Out of the 31.56% instances labelled manually as a named entity of type Location, 24.8% are classified correctly, whereas 1.4% is misclassified.

There are three main issues when evaluating a named entity recognition system using a source such as DBpedia as a gazetteer of named entities:

(a) The gazetteer may not contain the named entity, which was also observed by Jain and Pennacchiotti in [49], where 61% of their named entity extractions were correct, although Wikipedia had missed them; and

(b) The candidate named entity may belong to more than one entity class, and this justifies using only those instances that belong to a single named entity class in the ontology

Table 6.3: Mapping the manual classification of named entities to the DBpedia ontology (%)

Manual Classification	**Location**		**Person**	
	Correct Classification		**Correct Classification**	
	Administrative region	41.573	Artist	28.5714
	City	24.719	Actor	19.048
	Country	6.742	Person	11.905
DBpedia	Place	4.494	Athlete	9.524
Ontology	Body of Water	1.124	Office Holder	4.762
Classification	**Misclassification**		**Misclassification**	
	Television Show	2.247	Administrative Region	2.381
	Film	1.124	Band	2.381
	Organisation	1.124		
	Not Classified	16.854	**Not Classified**	21.428

Manual Classification	**Facility**		**Company**	
	Correct Classification		**Correct Classification**	
	Educational Institution	7.692	Company	50
	Place	5.1282	Beverage	2.778
	Misclassification		**Misclassification**	
DBpedia	Album	2.564		
Ontology	Company	2.564		
Classification	Mountain Range	2.564		
	Musical Work	2.564		
	Television Show	2.564		
	Not Classified	74.359	**Not Classified**	47.2222

when evaluating the Vector Expansion approach to QNEC presented in Chapter 4.

(c) Mapping the name of classes to other named entity ontologies is not an easy task.

In this section, the manual classification of named entities in search queries is compared to that of the DBpedia ontology. In the following section, using the manually annotated sample of named entities detected in search queries, a taxonomy of named entities is derived. For each named entity class in the taxonomy, a list of possible search intents, derived from the work of [21, 51, 91], is presented.

6.5 Query Log-Driven Named Entity Taxonomy

The objective is to define a comprehensive named entity taxonomy along with the possible search intents behind a search query that contains an entity of the identified class. To this end, in this thesis, every named entity class in the taxonomy can be described in light of the following:

(a) The domain or topic in which the named entity operates, such as Entertainment, Business, Art, or Fashion.

(b) Attributes derived from search queries that include the search refiners the users often include in their search query observed through the analysis presented in Chapter 5.

Table 6.4: Results of the DBpedia and manual classifications (%)

Manual Classification		DBpedia Ontology Classification		
Class Name	Percentage	Correctly Classified	Misclassified	Not Classified
Location	31.56	24.823	1.418	5.319
Person	14.894	10.993	0.709	3.191
Facility	13.83	2.482	1.064	10.284
Company	12.766	6.738	0.000	6.028
Music	4.61	3.191	0.355	1.064
TV Show/Channel	4.255	2.128	0.000	2.128
Product	3.191	0.000	0.709	2.482
Software	3.191	0.709	0.355	2.128
Website	2.837	1.064	0.355	1.418
Medical	2.482	0.355	0.355	1.773
Video Game	1.064	0.000	0.355	0.709
Book	0.709	0.000	0.000	0.709
Newspaper/Magazine	0.355	0.000	0.000	0.355
Sport	0.355	0.000	0.000	0.355
Other	3.901	0.355	0.000	3.546
Total	100	52.838	5.675	41.489

(c) Search goals behind search queries.

For example, consider the class of named entities Person. In the domain of Entertainment, possible entities are singers, actors, TV presenters, or directors, whereas in the domain of Politics, the entities would be politicians, queens, presidents, or ministers. Because the domains differ, the attributes that describe the entity would be different. For example, a majority of person entities would have generic attributes with which most named entities, regardless of their types, are associated, such as 'biography', 'born', or 'age'. On the other hand, the attributes 'lyrics', 'songs', or 'albums' are most likely associated with singers, whereas actors would be associated with 'movies', 'new movie', 'movie trailer', or 'Oscar nomination'. Finally, because the attributes differ, the search intent interpretations also differ. When an actor name is followed by 'movie trailer', it is highly likely that the intention behind the search query is to watch a video stream of the trailer, whereas the attribute 'biography' can be used to induce an informational search intent or the intent to navigate to a specific web page to read a person's biography.

The creation of the taxonomy is inspired from the following existing taxonomies: (i) Sekine and Nobata's named entity taxonomy [92], and the hierarchical taxonomy of the DBpedia ontology, and (ii) search intent taxonomies presented by Broder [21], Rose and Levinson [91], and Jansen et al. [51].

Figure 6.1 shows the query log-derived named entity taxonomy. In the following paragraphs, a description and a list of possible search goals is presented for each named entity class.

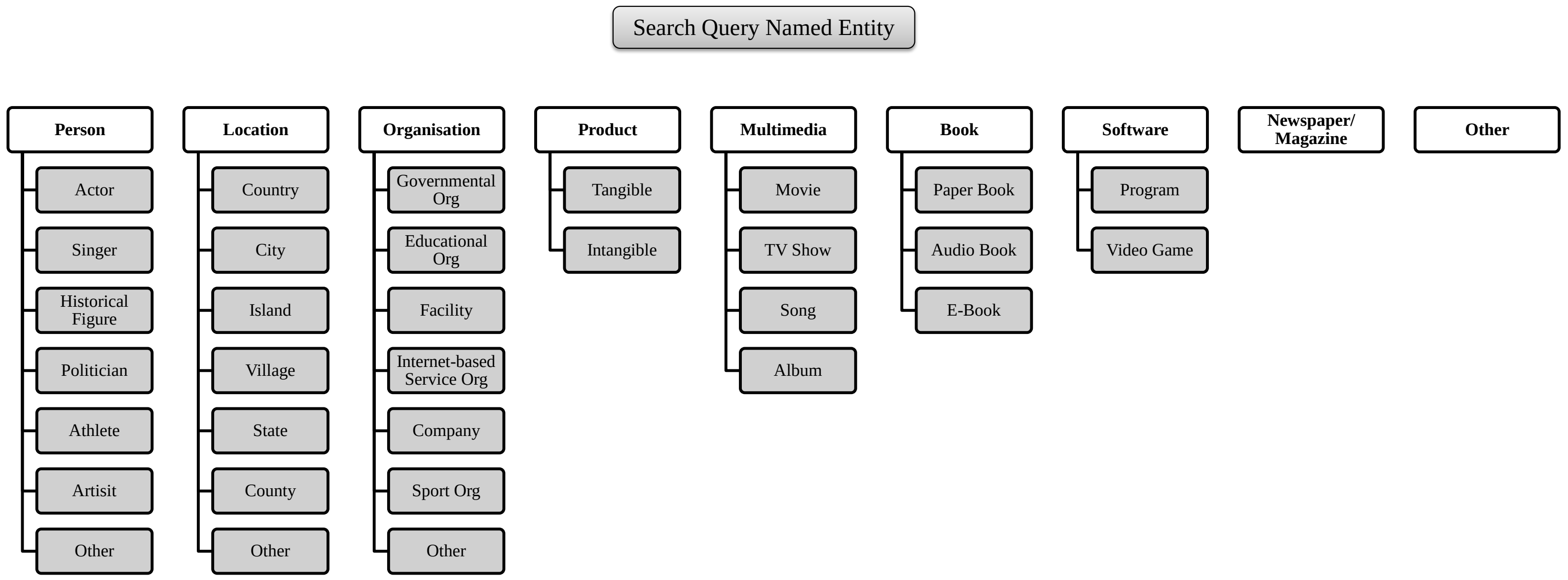


Figure 6.1: Named entity taxonomy derived from search query log

Person

Description: as presented earlier, given the domain in which the named entity operates the search refiners or attributes differ. Therefore, there are generic search refiners that apply to most named entities of the class Person, such as 'biography', 'bio', 'age', 'height', and 'spouse'. On the other hand, there are domain-specific search refiners, such as 'album', 'songs', and 'lyrics', for singers, and 'movies' and 'tv series' for actors. Nevertheless, a person can operate in more than one domain, for example, Robin Williams is an actor and a comedian, so there are attributes such as 'stand up show' and 'movies'. Furthermore, there are exceptions, i.e., not all person named entities mentioned in the search queries share the generic attributes, e.g., 'biography', especially for non-celebrities that include people on Facebook or Linkedin. In addition, mentions of fictional characters that appear in novels or movies whose search refiners are not necessarily similar to the majority of the person named entities in search queries.

Search Intents: a search query with a person's name may have the following search intents:

- Navigational: navigate to an official web page or website.
- Informational: to gain information related to a person, which can be:
  - Direct answer.
  - List related work.
  - Read related news.
- Access a web media resource, such as images or videos.
- Download a web media resource, such as images or videos.

Location

Description: Similarly to person entities, there are some generic search refiners among location references regardless of their type. For example, 'map' and 'population' are shared by countries, cities, and states. On the other hand, 'flag', 'president', and 'capital' are more likely to be specific attributes of countries.

Search Intents: a search query with a location's name can have the following search intents:

- Navigational: navigate to an official web page or website.
- Informational: gain information related to a location, which can be:

- Direct answer, such as 'flag', 'capital', or 'popluation' of a country name.
- List related facilities within a location, such as 'hotels', 'parks', or 'restaurants'.
- Read related news.

- Access a web resource, such as images, videos, or map.

Organisation

Description: search refiners that surround the name of an organisation are different given the type of organisation. For example, Educational Organisation are often associated with words such as 'registration', and 'courses', whereas for commercial companies, the refiners include, e.g., 'retail stores', 'web page', or a general class of the products that a company retails, such as 'Lexus used parts' or 'Chanel bags'. On the other hand, for Internet-based service companies, such as Google, Yahoo, or Bing, words such as 'mail', 'maps', 'news' or other web-mediated services provided by the companies are often associated with the named entities. Moreover, facilities include the names of banks, hotels, parks, or restaurants, and would be associated with search refiners such as 'location', 'near me', or 'direction'. Often, sport organisations, such as football or basketball teams, are often associated with refiners, such as 'game', 'scores', or 'players'. Generic search refiners would be 'profile' or 'contact information'.

Search Intents: a search query with an organisation's name can have the following search intents:

- Navigational: navigate to an official web page or website, and in the case of Internet-mediated services, navigate to the official web page that provides that service.
- Informational: gain information related to an organisation, which can be:
  - Direct answer, e.g., contact information, CEO, or director.
  - List of related products, services, or retail stores related to the organisation.
  - Read related news.
- Access a web resource, such as images or videos.
- Find location on map.

Multimedia

Description: the reason behind grouping movies, songs, TV shows, or TV programs into a single category referred to as multimedia is that they all have two generic search refiners or intents, ‘download’ and ‘online stream’, which are the most general web-mediated actions that can be performed when searching for a multimedia name. Nevertheless, each subcategory would have specific search intents. In general, movies, TV shows, and TV programs have search refiners, such as cast and director; however, songs refiners include, ‘singer’, ‘lyrics’, or ‘writer’.

Search Intents: a search query with a multimedia name can have the following search intents:

- Navigate: navigate to an official web page or website.
- Informational: gain information related to a multimedia entity, which may be:
  - Direct answer, such as a song writer, or a movie director.
  - List the cast members of a movie/ TV show/ TV program, or list the songs of an album.
  - Read related news, for example, the release date of a movie.
- Access a web resource, such as images (more specifically, wallpapers of movies or albums).
- Watch online stream.
- Download track or video.
- Buy a multimedia track or video.

<u>Product</u>

Description: looking at named entities of the type Product, it can be seen that they fall into two general categories, tangible and intangible products. In the context of the web, intangible products refer to web-mediated services that require actions, such as subscriptions, registration, or logging, for example, ‘gmail’ or ‘hotmail’ services, which may be either free or has a price quote. On the other hand, tangible products are often associated with search intentions such as ‘retail stores’, ‘delivery tracking’, or ‘placing orders’. This is the major distinction between tangible and intangible products within the context of the Web. Generic search refiners that apply to all products regardless of their type would be reviews, service provider, or producing company.

Search Intents: A search query with a product’s name can have the following search intents:

- Navigational: navigate to official web page or website.

- Informational: gain information related to a product entity, which can be:
    - Direct answer.
    - List of retail stores, reviews, price quotes, or versions of the product.
    - Read related news.
    - Compare based on reviews or price quotes.
- Access a web resource, such as images or videos.
- Subscribe/Log to web mediated service.
- Purchase products online.

Book

Description: book entities have common search intents with product entities, because they are the products of publishing companies; however, they have distinctive attributes that set them apart from other tangible products, such as 'author', or 'publisher'. Furthermore, they can be both tangible 'paper book' and intangible 'audio book' or 'e-books'.

Search Intents: a search query with a book name can have the following search intents:

- Navigational: navigate to an official web page or website.
- Informational: gain information related to a book entity, which can be:
    - Direct answer, such as author name.
    - List of book stores, reviews, price quotes, or versions of the book.
    - Read related news.
    - Compare based on reviews or price quotes.
- Access a web resource, such as PDF files or images.
- Download audio or ebooks.
- Buy books online.

Software

Description: similar to books, software can be both tangible (CD pack) and intangible (downloadable content). Two major categories of software are defined in the taxonomy: programs and video games. Both of the categories can have the generic search intents such as 'download', whereas video games can have more specific intents, such as 'play online'.

Search Intents: a search query with a software name can have the following search intents:

- Navigate: navigate to an official web page or website.
- Informational: gain information related to a video game entity, which can be:
    - Direct answer.
    - List of retail stores, reviews, price quotes, or versions of the software and list of web pages that provide online playing options.
    - Read related news or article.
    - Compare based on reviews or price quotes.
- Access a web resource, such as images or videos.
- Buy software.
- Play video game online.

Newspaper or Magazine

Description: refers to the name of a newspaper or magazine that can be either available online, such as in electronic format, or in paper format.

Search Intents: a search query with a magazine name or newspaper name can have the following search intents:

- Navigational: navigate to an official web page or website.
- Informational: gain information related to the entity, which can be:
    - Direct answer, such as owner or publisher.
    - List of articles.
    - Read related news or article.

6.6 Discussion

In this chapter, an analysis of the distribution of entity classes of the identified named entities observed over a sample from the search query log was presented. Accordingly, a taxonomy of named entity classes was presented, along with the set of possible intentions that can be identified for each named entity class.

Although the derived taxonomy is based on a random sample selected from the query log, the following two limitations were observed:

1. Search intents are inherently subjective, and therefore the search goals defined in the search log-derived taxonomy do not comprise a definitive list. Furthermore, to validate the coverage of the taxonomy of entities with respect to users' search goals, a survey of users' search intents behind search queries that bear named entities needs to be conducted.

2. Search query logs and web content are evolving constantly. Therefore, relying on a query log released in 2006 is not sufficient for a comprehensive analysis of the distribution of entity classes identified, or for deriving a complete taxonomy of named entities.

Chapter 7

Conclusions and Future Work

This thesis presents an approach for named entity recognition and classification in search queries. The approach is tailored specifically to address the problems that are imposed by the nature of search queries. In this chapter, the thesis is summarised in Section 7.1, followed by the main contributions to the problem of named entity recognition in search queries in Section 7.2. After that, in Section 7.3, the limitations and constraints encountered in this research are described. Finally, the chapter is concluded by presenting future opportunities with this area of research in Section 7.4.

7.1 Summary of the Thesis

In Chapter 1, the problem of NER in search queries and the motivation behind it is presented. Due to the leverage that named entities may lend to search engines in order to realise the vision of the semantic web introduced by Tim-Berners Lee, search engines have targeted different techniques to incorporate and exploit named entities. One of these techniques is mining named entities to create a comprehensive resource of human knowledge in the form of named entities interconnected via relationships and described by attributes. Looking specifically at search query logs, the advantage of mining named entities from search queries is that query logs present a rich source of named entities. Furthermore, they depict named entities based on users' search trends.

Existing solutions to the problem of NER in search queries are very promising, yet this problem is also a very challenging one. Search queries are very short, i.e., on average consist of two to three keywords. In addition, users do not follow the orthographic rules of spelling when submitting search queries that may reveal whether the query contains a named entity. Moreover, search queries lack grammatical structure, and search keywords are often submitted without the use of function words. Finally, due to their brevity, search queries lack contextual clues that can be exploited to identify or classify a named entity.

The detection of named entities in search queries provides a means for search engines to gain a better understanding of search queries. Search engines have always strived to understand search queries in order to reveal and meet users' search intents. Therefore, before the introduction of named entities within the context of search queries, search engines used various techniques for understanding queries. Chapter 2 presents a survey of five tasks that aim for query comprehension, which include search intent discovery, query topical classification, query segmentation, query structural annotation, and query named entity recognition. The work presented in this thesis builds on these observations and findings of previous research within the context of search queries.

Regardless of the target task, query enrichment via search results lends itself to the dynamic nature of search queries and the Web, and thereby leads to facilitating the different tasks of search query understanding. Previous work in query segmentation and structural annotation revealed that both can be exploited to combine their strengths for recognising candidate named entities in search queries. Finally, when reviewing previous techniques to NER in search queries, they mainly relied on the query string to identify and classify named entities. Although the accuracy scored by their techniques is high, there remains the problem when the query consists of only the named entity string and there are no any contextual keywords or orthographic features that can be utilised for NER. Based on these observations, this thesis has presented new approaches to identify and classify named entities for search queries.

In Chapter 3, the notion of a candidate named entity is presented, which is defined as any sequence of search keywords that may refer to a named entity regardless of its class. Then, the three-stage model for detecting candidate named entities is described. The model consists of the following three stages: (i) grammatically annotate search queries to assign a part-of-speech and an orthographical feature to each keyword, (ii) segment search queries into the most probable phrases, and (iii) detect named entities using a small set of hand-crafted rules, which is defined to exploit grammatical features in order to identify sequences of keywords that may refer to candidate named entities whose boundaries are set by the query segmentation. The performance of the model is measured at both the query-level and named entity-level using manual and automatic evaluation with the aid of DBpedia [18].

Once candidate named entities are detected, the next step is to classify them into their repective classes as described in Chapter 4. Due to the lack of context in search queries, the snippets of the search results are used again to extract the contextual clues. However, since snippets are domain-independent and are not necessarily structured, a *Bag-of-Context-Words (BoCW)* representation of the contextual clues is used. For each candidate named entity, a BoCW is extracted and used to classify named entities. For the classification, a vector expansion algorithm is introduced, where the BoCW vectors of automatically discovered seeds, referred to as a Class Vector, are iteratively expanded by adding context words of those entities whose BoCW vector's similarity to the Class Vector is above a predetermined threshold. Using a subset of DBpedia-validated candidate named entities, the classification was evaluated using precision and recall over six target entity classes, namely, Person, Location, Educational Organisation, Organisation, Film, and Software. Moreover, the usefulness of the Class Vector created by the end of the expansion is quantified using precision@K. Using the full set of extracted candidate named entities, the similarity of its BoCW and the Class Vector of each target entity class is measured. Then, for each target entity class, the candidate named entities are ordered by their decreasing similarity, and the top 250 named entities are checked manually to measure precision@K. Finally, for performance comparison, the NER technique presented herein is compared to a baseline mining technique introduced in [80].

Given that named entities in search queries are detected and classified, in Chapter 5 an exploratory analysis of the linguistic components associated with a named entity string in search queries is presented. The analysis is based on the following heuristic: users further specify the search results related to their entity-centric search query via three types of search keywords, namely, common nouns, adjectives, and verbs. In order to identify this set of search keywords, called *search refiners*, the grammatical annotation of queries presented in Chapter 3 is exploited. First, observations were drawn from statistical analysis to answers the following questions: how many keywords do users employ to refine their entity-centric queries; what are most common keywords that are used as refiners; and what are the grammatical categories of these refiners? Then, for every type of search refiner and every target entity class, different ranking mechanisms were tested. For a target entity class, one may consider ranking refiners based on how frequently they are employed across users who submitted queries with entities belonging to the same class. However, most frequently employed refiners do not necessarily represent the search intents of the users. Therefore, via search patterns, i.e., the search refiner used and the corresponding clicked URL recorded in the click-through data in the search log, the search refiners are ranked using variants of Kleinberg's HITS algorithm [59].

Finally, in Chapter 6, a taxonomy of named entity classes resulting from a sample of manually annotated search queries that bear named entities is presented. In the taxonomy, an attempt to identify the search intent behind users' search queries is described, given that a named entity of a target entity class is identified.

7.2 Summary of Contributions and Findings

In order to recognise and classify named entities in search queries, the thesis makes the following contributions:

- A novel candidate named entity recognition approach that combines the strengths of query grammatical annotation and query segmentation.

- A novel named entity classification approach that presents the contextual clues found in the snippets as BoCW vectors in a common vector space. Then, with the aid of automatically selected seeds, the BoCWs of these seeds, referred to as Class Vectors, are iteratively expanded as new candidate named entities are classified. A candidate named entity is classified to a target entity class, if the similarity of its BoCW is equal to or greater than a predefined threshold.

- An exploratory analysis of three types of search refiners that users often employ in their entity-centric queries to refine the search results related to a named entity.

- A taxonomy of named entities derived from the query log, which reflects the possible search intents that each class of entities may reveal about an entity-centric query.

The research conducted in this thesis leads to the following findings:

Query enrichment: The top-n snippets related to a query often provide enough context to identify and classify named entities that appear in that query. Moreover, query enrichment solves the problem when a query lacks contextual clues for NER, which previous approaches, such as [49, 44, 80], left unresolved.

Categories of entity-centric search queries: Given a target candidate named entity to be labeled with a target entity class, a search query can fall into one of the following categories: Focused Query, Very Focused Query, and UnFocused Query. In the case of Focused Queries, using a flexible representation of the contextual clues in the form of a BoCW can effectively classify named entities in search queries using the vector expansion algorithm.

Vector Expansion Modes: Single Class Vector (SCV) and Multi sub-Class Vector (MCV) are two modes of expansion through which named entities were classified. The two modes affect the precision and recall scores that can be achieved by the vector expansion approach. The SCV ensures higher precision during the expansion, yet it biases the expansion towards a specific semantic space within the target entity class and thereby scored lower recall. On the other hand, MCV scored a higher recall, since each fertile seed is expanded independently of the other seeds and thereby each seed's expansion covers a larger portion of the semantic space of the target entity class; however, the improved recall is at the cost of lower precision. This is due to the fact that the chances of semantic drift are higher when using MCV expansion mode.

Search refiners and HITS algorithm: Tagging each search keyword with the corresponding part-of-speech revealed how users refine their entity-centric queries with three types of keywords, nouns, adjectives, and verbs. Then, using the click graphs created from the click-through data from the query log, different variants of the HITS algorithm were implemented. Accordingly, the various search refiners were ranked based on the resources that the users examined to satisfy their search needs.

7.3 Limitations and Constraints

Search Log: Due to the potential breach of users' privacy, acquiring a large-scale web search log from a commercial search engine is not an easy task. In 2006, the AOL search engine disclosed twenty million search queries issued by 650,000 users over a three-month period for research purposes [2, 7]. While AOL anonymised users' identities with a unique ID number to identify each user, search traits of users can be identified by the queries they submitted, since individual users' searches can be identified over that period. For example, user X is overweight as X searched for a diet to lose weight, searched for restaurants nearby, searched for tickets to travel, and personal preferences that were meant to be kept private were revealed. Accordingly, search engines face the risk that releasing a search log may lead to accidental or malicious disclosure of users' information [26].

While using a relatively old search log is a limitation in this work, the proposed approach for named entity recognition and classification in search queries and the following exploratory analysis of search refiners is nevertheless applicable to search logs regardless of their release time. However, the drawn statistics may vary due to the dynamic nature of search queries and constant shifts in search topics.

Candidate named entity detection in search queries: Grammatical annotation and segmentation of search queries are both major techniques that were used here for detecting candidate named entities in search queries. Herein, the performance of both techniques employed was not measured individually; however, the performance was estimated given the accuracy of the candidate named entities detection.

BoCW and Vector Expansion: The BoCW representation of the contextual clues and the use of vector expansion to classify named entities was highly successful for named entities extracted from Focused Queries. However, the performance of the expansion decreases when applied to named entities extracted from Very Focused Queries and significantly decreases for those appearing in UnFocused Queries.

Deterministic Classification: Since search queries and named entities are highly ambiguous, assigning a probability to each named entity class lends itself naturally to the ambiguity of the named entities. In this thesis, a deterministic approach for classification was presented, where for every target entity class, a named entity either belongs or does not belong to that class.

7.4 Directions for Future Work

Weight of Contextual Clues: In the BoCW of each candidate named entity, each context word is assigned a weight, which is set to the frequency of that context word in the snippets set. In order to improve the performance, one may consider the following definition of the weight:

- Distance from a named entity mention in a sentence: The weight of each keyword can be discounted by how far it is from the target named entity mention in a snippet. The rationale behind this is that the further a context word is from a named entity, within a sentence, the less relevant it is to the target named entity.

- Rank of the related result: The weight can be discounted by the rank of the snippet from which the context word is extracted. The rationale behind this is that the lower the rank of the search result from which a context word is extracted, the less related the search result is to the user's search intent, and thereby less descriptive the context word of the named entity.

- Mention of other named entities: The weight can be discounted by the mention of other named entities in the same sentence where the target entity appeared.

- Inverse Class Frequency (ICF): After the selection of the fertile seeds, the weight of context words can be updated by discounting the weight of context words that appear in more than one named entity Class Vector. This is achieved by utilising the notion of Inverse Class Frequency (ICF). If C is the number of target named entity classes, and C_w the number of class vectors having w as a context word, then the inverse class frequency of a context word W is as follows.

$$ICF_w = \log \frac{C}{C_w}$$

Tailoring a user's search session: Revealing search intent by revealing the linguistic composites constituting the search query is the focus of current research for web search queries. Therefore, new techniques are needed to leverage the identified components in order to customise a user's search experience accordingly. The exploratory analysis of these components, which is presented in Chapter 5, revealed that these components can be utilised to reveal a search intent. This can further be extended to provide a mechanism for search results diversity [32].

Appendix A

Resources

A.1 Determining Sample Size

When determining a sample size, one should consider the tradeoff between accuracy and cost. A large sample yields an accurate representation of the target dataset; however, the excessive analysis required for a large sample is costly. Moreover, before determining the sample size, the following factors should be considered [24]:

- Margin of error: what is the error rate that a researcher is prepared to tolerate in the results drawn from the sample?

- Confidence level: how confident must a researcher be that the results drawn from the sample are true?

- Standard deviation: how much variance is the researcher expecting in the results drawn?

Accordingly, the sample size can be calculated using Cochran's equation [24]:

$$\frac{Z^2 p(1-p)}{E^2}.$$

The parameter $p(1-p)$ estimates the sample standard deviation, where p is the best estimate of the chance that a search query will be selected from a set of search queries. Because queries were selected randomly, each query had a 50/50 chance to be chosen, with a confidence level of 95% (i.e. $Z = 1.96$) and a maximum tolerable error rate of 0.031 (E). Therefore, the size of the validation and evaluation samples used in Chapter 3 was set to 1,000 search queries.

A.2 Segments Probability

To identify the probability of each possible segment in a search query as described in Chapter 3, each query that consisted of n keywords can have a maximum of $\frac{n(n+1)}{2}$ distinct segments. For example, the following queries have the corresponding distinct segments:

- 'london': `[london]`.

- 'london university': `[london]`, `[university]`, and `[london university]`.

- 'university of london': `[university]`, `[of]`, `[london]`, `[university of]`, `[of london]`, and `[university of london]`.

The probability of each segment, given a web n-gram model, is measured using the Bing web n-gram model. The Bing web n-gram model contains a maximum of 5-grams, and the probabilities of the segments whose order is greater than five were estimated using the chain rule along with the Markov assumption. For example, consider the sequence of keywords, 'a b c d', and assuming that a language model contains a maximum of 3-grams. According to the chain rule,

$$P(abcd) = P(a)P(b|a)P(c|ab)P(d|abc).$$

The probability $P(d|abc)$ can be estimated using a third-order Markov chain, which is $P(d|abc) \approx P(d|bc)$. The conditional probability of the keyword d given the preceding search keywords bc is estimated by:

$$P(d|bc) \approx \frac{P(bcd)}{P(bc)}.$$

A.3 DBpedia

DBpedia is a project whose aim is to extract structured information from Wikipedia [18]. The current DBpedia version (DBpedia 2014) describes 4.58 million "things" with 583 million "facts". The version used in this thesis (DBpedia 2012) describes 3.64 million 'things' from which 1.83 million instances are classified into a hierarchical ontology including entities such as Persons, Places, and Organisations.

DBpedia provides a lookup service that can be used to locate DBpedia URIs, where each URI identifies a 'thing' in the DBpedia dataset; the URI refers to a Wikipedia article from which data is extracted. Each extracted named entity is validated using this look up service, where a set of results related to the extracted entities is retrieved. Each related result consists of a label, which is the string representing the instance found, a description, a class in the DBpedia ontology, a Wikipedia category, and the number of inlinks within Wikipedia pointing to this result.

Each unique candidate named entity extracted by QNER is submitted to DBpedia for validation. The label of the DBpedia result is either exact match, partial match, or does not match the entity string. The results retrieved from DBpedia for each extracted named entity are used for evaluation as presented in Chapters 3 and 4.

A.4 GATE Toolkit

GATE stands for General Architecture for Text Engineering and it is “an infrastructure for developing and deploying software components that process human language” [28]. It is an open source software that includes comprehensive tools for natural language text processing.

In this thesis, GATE was incorporated to grammatically annotate the top-n snippets related to a query by assigning to each word a Part-of-Speech and an orthographic feature. In order to achieve that, a processing pipeline was constructed using the following GATE resources:

RegEx Sentence Splitter, which identifies the boundaries of sentences within each snippet. This is particularity useful when measuring the probability of a segment using the snippets, where every segment should not cross the boundaries of the sentences.

Tokeniser, which splits each phrase into tokens and each token can be:

- a sequence of letters including a hyphen, and each can have one of the following orthographic attributes: uppercase Initial, all lowercase letters, all capitalised, or a mixture;
- a Number as a sequence of consecutive digits;
- a symbol;
- a punctuation; or
- space token (i.e. white space).

Part-of-Speech Tagger, which is ‘a modified version of the Brill tagger’, and assigns a Part-of-Speech to each word. It utilises a default lexicon and a set of rules that is based on training the tagger using the Wall Street Journal corpus.

Using this processing pipeline, each snippet is grammatically annotated and each search keyword in the query is assigned a POS and an ORTH feature whose scores are the highest as described in Chapter 3 Subsection 3.2.1.

Moreover, once queries are grammatically annotated and segmented, candidate named entities were detected using the GATE JAPE transducer. The transducer loads the JAPE grammar and builds a Finite State Machine (FSM) over it. The JAPE grammar consists of eight rules, which are presented in Appendix B. Using the these rules, the transducer creates the annotations over search queries using the POS and ORTH features assigned by the grammatical annotation stage of QNER to identify candidate named entities.

Appendix B

JAPE Rules for Candidate Named Entity Detection in Search Queries

```
Phase: candidateNamedEntityDetection
Input: Token SpaceToken
Options: control = appelt

/*
 * Rule[01]: General Entities: A sequence of proper nouns
 *
 * Example:
 *  "Nissan Maxima"
 *  "New York"
 */

Rule: candidateNamedProperNoun
Priority: 90
(       (({Token.string == "the"} {SpaceToken})?)
    (
                (({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                (({Token.string == "'s"} ({SpaceToken})?)?))
                ({SpaceToken})?
        )+
):candidateNE
-->
{
  gate.AnnotationSet candidateNE =
      (gate.AnnotationSet)bindings.get("candidateNE");
  gate.FeatureMap features =
       Factory.newFeatureMap();
  features.put("id", "01");
  features.put("rule","GeneralEntities");
  outputAS.add(candidateNE.firstNode(), candidateNE.lastNode(),
      "Candidate NE", features);
}
```

```
/*
 * Rule[02]: Entities with Connectors: a connector followed and
 *       preceded by proper nouns connectors: of,&,de,for
 *
 * Example:
 *  "rose & womble"
 *  "battle for manila"
 *  "anthony de mello"
 *  "soldier of fortune", "curse of the omen"
 */

Rule: candidateNamedProperNoun
Priority: 100
(
        (
                (({Token.string == "the"} {SpaceToken})?)
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                ({SpaceToken})
        )+
        (
                {Token.string == "of"}
                |
                {Token.string == "&"}
                |
                {Token.string == "de"}
        |
                {Token.string == "for"}
        )
        ({SpaceToken})
        (
                (({Token.string == "the"} {SpaceToken})?)
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                (({Token.string == "'s"} ({SpaceToken})?)?)
                ({SpaceToken})?
         )+
):candidateNE
-->
{
  gate.AnnotationSet candidateNE =
      (gate.AnnotationSet)bindings.get("candidateNE");
  gate.FeatureMap features =
       Factory.newFeatureMap();
  features.put("id", "02");
  features.put("rule","EntitiesWithConnectors");
  outputAS.add(candidateNE.firstNode(), candidateNE.lastNode(),
  "Candidate NE", features);
}
```

```
/*
 * Rule[03]: Named Entities with Numbers
 *
 * Example:
 *   "channel 5"
 *   "john deere 214"
 *   "microsoft office 2006"
 *   "starship troopers 2"
 *   "halo 2"
 *   "internet explorer 6.0"
 *   "msn messenger 8.0"
 *   "office 4.2.1"
 */

Rule: candidateNamedEntityCardinalNumber
Priority: 100

(
        (
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                ({SpaceToken})?
        )+
        (
                {Token.POS == CD, Token.kind == number}
        )
        (
                ({Token.string=="."}{Token.POS == CD, Token.kind == number})?
        )+
)
:candidateNE
-->

{
  gate.AnnotationSet candidateNE =
      (gate.AnnotationSet)bindings.get("candidateNE");
  gate.FeatureMap features =
       Factory.newFeatureMap();
  features.put("id","03");
  features.put("rule","NamedEntityWithNumbersI");
  outputAS.add(candidateNE.firstNode(), candidateNE.lastNode(),
        "Candidate NE", features);
}
```

```
/*
 * Rule[04]: Named Entities with Numbers
 *
 * Example:
 *   "1997 chevrolet monte carlo"
 */

Rule: candidateNamedEntityCardinalNumber
Priority: 100

(
        (
                ({Token.POS == CD, Token.kind == number})
                ({SpaceToken})?
        )
        (
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                ({SpaceToken})?
        )+
)
:candidateNE
-->

{
  gate.AnnotationSet candidateNE =
      (gate.AnnotationSet)bindings.get("candidateNE");
  gate.FeatureMap features =
       Factory.newFeatureMap();
  features.put("id","04");
  features.put("rule","NamedEntityWithNumbersII");
  outputAS.add(candidateNE.firstNode(), candidateNE.lastNode(),
      "Candidate NE", features);
}
```

```
/*
 * Rule[05]: Named Entities with Numbers
 *
 * Example:
 *  "first american bank"
 *  "first avenue"
 *  "swift first aid"
 */

Rule: candidateNamedEntityCardinalNumber
Priority: 100

(
        ((
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                ({SpaceToken})?
        )?)+
        (
          ({Token.POS == LS, Token.ORTH == upperInitial, Token.kind == word})
                |
          ({Token.POS == CD, Token.ORTH == upperInitial, Token.kind == word})
         )
        ({SpaceToken})
        (
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                ({SpaceToken})?
        )+
)
:candidateNE

-->

{
  gate.AnnotationSet candidateNE =
      (gate.AnnotationSet)bindings.get("candidateNE");
  gate.FeatureMap features =
       Factory.newFeatureMap();
  features.put("id","05");
  features.put("rule","NamedEntityWithNumbersIII");
  outputAS.add(candidateNE.firstNode(), candidateNE.lastNode(),
      "Candidate NE", features);
}
```

```
/*
 * Rule[06]: Detect candidate named entity that has a mixed caps
 *           contained in a segment
 *
 * Example:
 *  "eBay"
 *  "iPhone"
 */

 Rule: CandidateNEMixedCaps
Priority: 100

(
        (({Token.POS == NN , Token.ORTH == "mixedCaps", Token.kind == "word"})
         ({SpaceToken})?)+
)
:candidateNE

-->
{
  gate.AnnotationSet candidateNE =
      (gate.AnnotationSet)bindings.get("candidateNE");
  gate.FeatureMap features =
       Factory.newFeatureMap();
  features.put("id", "06");
  features.put("rule","NamedEntityBasedOnOrthI");
  outputAS.add(candidateNE.firstNode(), candidateNE.lastNode(),
     "Candidate NE", features);
}
```

```
/*
 * Rule[07]: Detect candidate NEs with adjectives being preceded and
 *                       followed by proper nouns contained in a segment
 *
 * Example:
 *  "America's Next Top Model"
 *  "The Big Bang Theory"
 */

Rule: cadidateNamedEntityAdjectiveRule
Priority: 100


(
        ({Token.string == "the"} {SpaceToken})?
        ((
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                (({Token.string == "'s"} ({SpaceToken})?)?)
                ({SpaceToken})?
        )?)+
        (
                {Token.POS == JJ, Token.ORTH == upperInitial}
        )+
        ({SpaceToken})
        (
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                (({Token.string == "'s"} ({SpaceToken})?)?)
                ({SpaceToken})?
        )+
):candidateNE

-->
{
 gate.AnnotationSet candidateNE =
      (gate.AnnotationSet)bindings.get("candidateNE");
  gate.FeatureMap features =
       Factory.newFeatureMap();
  features.put("id","07");
  features.put("rule", "NamedEntityBasedOnOrthII");
  outputAS.add(candiateNE.firstNode(), candiateNE.lastNode(),
     "Candidate NE", features);
}
```

```
/*
 * Rule[08]: Entity based on orthography
 *
 * Example:
 *  "alfred l. kroeber"
 *  "andrew j. williams"
 *  "st. louis"
 */

Rule: cadidateNamedEntityTitlesDotRule
Priority: 100


(
        ((
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                ({SpaceToken})?
         )+
        (
                {Token.string == "."}
                ({SpaceToken})?
        ))+
        (
                ({Token.POS == NNP}
                |
                {Token.POS == NNPS})
                ({SpaceToken})?
        )+

):candidateNE

-->
{
 gate.AnnotationSet candidateNE =
      (gate.AnnotationSet)bindings.get("candidateNE");
  gate.FeatureMap features =
       Factory.newFeatureMap();
  features.put("id","08");
  features.put("rule", "EntityBasedOnOrthIII");
  outputAS.add(candiateNE.firstNode(), candiateNE.lastNode(),
     "Candidate NE", features);
}
```

Appendix C

Publications

[1] Areej Alasiry, Mark Levene, and Alexandra Poulovassilis. "Detecting candidate named entities in search queries". *In Proceedings of the* 35th *International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1049-1050, New York, NY, USA, 2012. ACM.

[2] Areej Alasiry, Mark Levene, and Alexandra Poulovassilis. "Extraction and evaluation of candidate named entities in search engine queries". *In Proceedings of the* 13th *International Conference on Web Information Systems Engineering*, WISE'12, pages 483-496, Berlin, Heidelberg, 2012. Springer-Verlag.

[3] Areej Alasiry, Mark Levene, and Alexandra Poulovassilis. "Mining named entities from search engine query logs". *In Proceedings of the* 18th *International Database Engineering & Applications Symposium*, IDEAS '14, pages 46-56, New York, NY, USA, 2014. ACM.

Bibliography

[1] Lada A Adamic. Zipf, power-laws, and pareto- a ranking tutorial, 2008.

[2] Michael Arrington. Aol proudly releases massive amounts of private data. *TechCrunch*, 6 Aug 2006. Available: `http://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/`.

[3] Ricardo Baeza-Yates, Aristides Gionis, Flavio Junqueira, Vanessa Murdock, Vassilis Plachouras, and Fabrizio Silvestri. The impact of caching on search engines. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 183–190, New York, NY, USA, 2007. ACM.

[4] Jing Bai, Yi Chang, Hang Cui, Zhaohui Zheng, Gordon Sun, and Xin Li. Investigation of partial query proximity in web search. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 1183–1184, New York, NY, USA, 2008. ACM.

[5] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[6] Judit Bar-Ilan. Comparing rankings of search results on the web. *Inf. Process. Manage.*, 41(6):1511–1519, December 2005.

[7] Michael Barbaro. A face is exposed for aol searcher no. 4417749. *The New York Times*, 9 Aug 2006. Available: `http://www.nytimes.com/2006/08/09/technology/09aol.html?pagewanted=all&_r=0`.

[8] Cory Barr, Rosie Jones, and Moira Regelson. The linguistic structure of english web-search queries. In *Proceedings of the Conference on Empirical Methods in Natural Lan-*

guage Processing, EMNLP '08, pages 1021–1030, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[9] Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, David Grossman, and Ophir Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 321–328, New York, NY, USA, 2004. ACM.

[10] Steven M. Beitzel, Eric C. Jensen, David D. Lewis, Abdur Chowdhury, and Ophir Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM Trans. Inf. Syst.*, 25(2), April 2007.

[11] Michael Bendersky and W. Bruce Croft. Analysis of long queries in a large scale search log. In *Proceedings of the 2009 Workshop on Web Search Click Data*, WSCD '09, pages 8–14, New York, NY, USA, 2009. ACM.

[12] Michael Bendersky, W. Bruce Croft, and David A. Smith. Two-stage query segmentation for information retrieval. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 810–811, New York, NY, USA, 2009. ACM.

[13] Michael Bendersky, W. Bruce Croft, and David A. Smith. Structural annotation of search queries using pseudo-relevance feedback. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1537–1540, New York, NY, USA, 2010. ACM.

[14] Michael Bendersky, W. Bruce Croft, and David A. Smith. Joint annotation of search queries. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 102–111, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[15] Shane Bergsma and Qin Iris Wang. Learning noun phrase query segmentation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 819–826, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[16] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.* Harper San Francisco, 1st edition, 1999.

[17] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: A high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, pages 194–201, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[18] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7(3):154–165, September 2009.

[19] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[20] David J. Brenes, Daniel Gayo-Avello, and Rodrigo Garcia. On the fly query segmentation using snippets. In *Proceedings of the Spanish Conference on Information Retrieval*, CERI '10, pages 259–266, 2010.

[21] Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, September 2002.

[22] Andrei Z. Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 231–238, New York, NY, USA, 2007. ACM.

[23] Jackie Chi Kit Cheung and Xiao Li. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 383–392, New York, NY, USA, 2012. ACM.

[24] William G. Cochran. *Sampling Techniques, 3rd Edition.* John Wiley, 1977.

[25] Michael Collins and Yoram 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.

[26] Alissa Cooper. A survey of query log privacy-enhancing techniques from a policy perspective. *ACM Trans. Web*, 2(4):19:1–19:27, October 2008.

[27] Jim Cowie. CRL/NMSU: Description of the CRL/NMSU systems used for MUC-6. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 157–166, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[28] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. *Text Processing with GATE (Version 6).* 2011.

[29] Honghua (Kathy) Dai, Lingzhi Zhao, Zaiqing Nie, Ji-Rong Wen, Lee Wang, and Ying Li. Detecting online commercial intention (oci). In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 829–837, New York, NY, USA, 2006. ACM.

[30] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[31] Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. Analysis of named entity recognition and linking for tweets. *Inf. Process. Manage.*, 51(2):32–49, 2015.

[32] Marina Drosou and Evaggelia Pitoura. Search result diversification. *SIGMOD Rec.*, 39(1):41–47, September 2010.

[33] Junwu Du, Zhimin Zhang, Jun Yan, Yan Cui, and Zheng Chen. Using search session context for named entity recognition in query. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 765–766, New York, NY, USA, 2010. ACM.

[34] Keith R. Eberhardt and Michael A. Fligner. A comparison of two tests for equality of two proportions. *The American Statistician*, 31, 1977.

[35] Andreas Eiselt and Alejandro Figueroa. A two-step named entity recognizer for open-domain search queries. In *Proceedings of the Sixth International Joint Conference on*

Natural Language Processing, pages 829–833. Asian Federation of Natural Language Processing, 2013.

[36] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, December 2008.

[37] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134, June 2005.

[38] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 28–36, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[39] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Tech Report HPL-2003-4, HP Laboratories*, 2004.

[40] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[41] Kuzman Ganchev, Keith Hall, Ryan McDonald, and Slav Petrov. Using search-logs to improve query tagging. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 238–242, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[42] Jianfeng Gao, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. Dependence language model for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 170–177, New York, NY, USA, 2004. ACM.

[43] Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[44] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 267–274, New York, NY, USA, 2009. ACM.

[45] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Braeutigam. The power of naive query segmentation. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 797–798, New York, NY, USA, 2010. ACM.

[46] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. Query segmentation revisited. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 97–106, New York, NY, USA, 2011. ACM.

[47] Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. Understanding user's query intent with wikipedia. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 471–480, New York, NY, USA, 2009. ACM.

[48] Alpa Jain and Marco Pennacchiotti. Open entity extraction from web search query logs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 510–518, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[49] Alpa Jain and Marco Pennacchiotti. Domain-independent entity extraction from web search query logs. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 63–64, New York, NY, USA, 2011. ACM.

[50] Bernard J. Jansen. Search log analysis: What it is, what's been done, how to do it. *Library & Information Science Research*, 28(3):407 – 432, 2006.

[51] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. Determining the informational, navigational, and transactional intent of web queries. *Inf. Process. Manage.*, 44(3):1251–1266, May 2008.

[52] Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32(1):5–17, April 1998.

[53] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *In Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, The Netherlands: North-Holland, May 1980.

[54] Daxin Jiang, Jian Pei, and Hang Li. Mining search and browse logs for web search: A survey. *ACM Trans. Intell. Syst. Technol.*, 4(4):57:1–57:37, October 2013.

[55] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA, 2006. ACM.

[56] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

[57] In-Ho Kang and GilChang Kim. Query type classification for web document retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 64–71, New York, NY, USA, 2003. ACM.

[58] T. Kanungo and D. Metzler. System and method for automatically ranking lines of text, August 23 2011. US Patent 8,005,845.

[59] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[60] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[61] Lillian Lee. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 25–32, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

[62] Uichin Lee, Zhenyu Liu, and Junghoo Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 391–400, New York, NY, USA, 2005. ACM.

[63] Mark Levene. *An Introduction to Search Engines and Web Navigation (2Nd Edition)*. Wiley-Blackwell, 2010.

[64] K. Li, Y. Li, Y. Zhou, Z. LV, and Y. CAO. Knowledge-based entity detection and disambiguation, July 4 2013. WO Patent App. PCT/US2012/069,604.

[65] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st Annual International ACM SIGIR Conference on*

Research and Development in Information Retrieval, SIGIR '08, pages 339–346, New York, NY, USA, 2008. ACM.

[66] Yanen Li, Bo-Jun Paul Hsu, ChengXiang Zhai, and Kuansan Wang. Unsupervised query segmentation using clickthrough for information retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 285–294, New York, NY, USA, 2011. ACM.

[67] Yanen Li, Bo-June Paul Hsu, and ChengXiang Zhai. Unsupervised identification of synonymous query intent templates for attribute intents. In *Proceedings of the 22nd ACM international conference on Conference on information &#38; knowledge management*, CIKM '13, pages 2029–2038, New York, NY, USA, 2013. ACM.

[68] Ying Li, Zijian Zheng, and Honghua (Kathy) Dai. Kdd cup-2005 report: Facing a great challenge. *SIGKDD Explor. Newsl.*, 7(2):91–99, December 2005.

[69] Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 768–774, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[70] Thomas Lin, Patrick Pantel, Michael Gamon, Anitha Kannan, and Ariel Fuxman. Active objects: Actions for entity-centric search. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 589–598, New York, NY, USA, 2012. ACM.

[71] W.Y. Ma, X. Xiao, and X. Xie. Classifying functions of web blocks based on linguistic features, February 22 2011. US Patent 7,895,148.

[72] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[73] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, USA, 1999.

[74] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[75] Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 472–479, New York, NY, USA, 2005. ACM.

[76] G. Mishne, R. Stata, and F. Peng. Search query disambiguation, August 12 2010. US Patent App. 12/367,114.

[77] Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. Unsupervised query segmentation using only query logs. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 91–92, New York, NY, USA, 2011. ACM.

[78] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007. Publisher: John Benjamins Publishing Company.

[79] Paul Ogilvie and Jamie Callan. Combining document representations for known-item search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 143–150, New York, NY, USA, 2003. ACM.

[80] Marius Paşca. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 683–690, New York, NY, USA, 2007. ACM.

[81] Marius Paşca and Benjamin Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of ACL-08: HLT*, pages 19–27, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[82] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 613–619, New York, NY, USA, 2002. ACM.

[83] Patrick Pantel, Thomas Lin, and Michael Gamon. Mining entity types from query logs via user intent modeling. In *Proceedings of the 50th Annual Meeting of the Associa-*

tion for Computational Linguistics: Long Papers - Volume 1, ACL '12, pages 563–571, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[84] Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1400–1405. AAAI Press, 2006.

[85] Marius Pasca and Benjamin Van Durme. What you seek is what you get: Extraction of class attributes from query logs. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2832–2837, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[86] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 771–780, New York, NY, USA, 2010. ACM.

[87] P. Resnik. Selectional preference and sense disambiguation. pages 52–57, 1997.

[88] Ellen Riloff and Jay Shoen. Automatically acquiring conceptual patterns without an annotated corpus. In *In Proceedings of the Third Workshop on Very Large Corpora*, pages 148–161, 1995.

[89] Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. Query segmentation for web search. In *Proceedings of the 12th International Conference Companion on World Wide Web (Posters)*, WWW '03, 2003.

[90] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1524–1534, 2011.

[91] Daniel E. Rose and Danny Levinson. Understanding user goals in web search. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 13–19, New York, NY, USA, 2004. ACM.

[92] Satoshi Sekine and Chikashi Nobata. Definition, dictionaries and tagger for extended named entity hierarchy. In *LREC*. European Language Resources Association, 2004.

[93] Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. Query enrichment for web-query classification. *ACM Trans. Inf. Syst.*, 24(3):320–352, July 2006.

[94] Ben Shneiderman, Don Byrd, and W. B Croft. Clarifying search: A user-interface framework for text searches. Technical report, 1997.

[95] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, September 1999.

[96] Amanda Spink, Bernard J. Jansen, Dietmar Wolfram, and Tefko Saracevic. From e-sex to e-commerce: Web search changes. *Computer*, 35(3):107–109, March 2002.

[97] Amanda Spink, Dietmar Wolfram, Major B. J. Jansen, and Tefko Saracevic. Searching the web: The public and their queries. *J. Am. Soc. Inf. Sci. Technol.*, 52(3):226–234, February 2001.

[98] Bin Tan and Fuchun Peng. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 347–356, New York, NY, USA, 2008. ACM.

[99] A.A. Verstak and A. Acharya. Generation of document snippets based on queries and search results, March 27 2012. US Patent 8,145,617.

[100] David Vogel, Steffen Bickel, Peter Haider, Rolf Schimpfky, Peter Siemen, Steve Bridges, and Tobias Scheffer. Classifying search engine queries using the web as background knowledge. *SIGKDD Explor. Newsl.*, 7(2):117–122, December 2005.

[101] Vishnu Vyas, Patrick Pantel, and Eric Crestan. Helping editors choose better seed sets for entity set expansion. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 225–234, New York, NY, USA, 2009. ACM.

[102] Kuansan Wang, Christopher Thrasher, Evelyne Viegas, Xiaolong Li, and Bo-june (Paul) Hsu. An overview of microsoft web n-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, HLT-DEMO '10, pages 45–48, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[103] Gu Xu, Shuang-Hong Yang, and Hang Li. Named entity mining from click-through data using weakly supervised latent dirichlet allocation. In *Proceedings of the 15th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pages 1365–1374, New York, NY, USA, 2009. ACM.

[104] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[105] Xiaoxin Yin and Sarthak Shah. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1001–1010, New York, NY, USA, 2010. ACM.

[106] H. Zhou, K. Bharat, M. Schmitt, M. Curtiss, and M. Mayer. Query rewriting with entity detection, October 6 2005. US Patent App. 10/813,572.

[107] Ingrid Zukerman and Bhavani Raskutti. Lexical query paraphrasing for document retrieval. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.