Word Embeddings and Information Retrieval

Dwaipayan Roy

Thesis submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

to

Computer and Communication Science Division
Indian Statistical Institute, Kolkata

Supervisor: Dr. Mandar Mitra
Co-supervisor: Dr. Debasis Ganguly

June 2019

To Everyone who believed that I could do this...

Acknowledgements

It all started when I got the opportunity to work under Dr. Mandar Mitra during my Bachelor's project. I started working with him on a small project without realising I would be doing PhD on that topic. After qualifying for the PhD entrance examination at ISI Kolkata, I approached Dr. Mitra for being my supervisor. After initial hesitation, he finally agreed on supervising me but warning (with his usual modesty) about the lack of knowledge on recent works (which was not true), that he would only be able to fight together with me for my PhD work. For the past several years, he continued doing so, sitting next to me, breaking the conventional barrier of supervisor-student relationship in my mind. I learned unaccountable academic as well as non-academic concepts from him. Having Dr. Mitra as my PhD supervisor has been the best thing that happened to me in my entire PhD journey.

I will be eternally indebted to Dr. Debasis Ganguly for helping me understand about the IR research and for his polite response to my naive questions on general IR topics. He also helped me understanding Lucene which has been used for all the works reported in this thesis. I will remember his motivation when we were having a series of failures. Thanks to Prof. Gareth Jones for arranging internship supports to visit Dublin City University, Ireland where I got the opportunity to work closely with Debasis and Gareth on word embeddings which eventually turned out to be my thesis topic. I also express my gratitude to IR lab seniors Dipa di, Kripa da and Jia da.

PhD is a journey with regular rejections and occassional success, surviving which one needs to have extensive peer support as well. Thanks to Abhisek, Anabik, Ayan da, Chandan da, Jayati, Partho da, Soumen, and Suchana who were always there to cheer me up during the time of failures and helped me to survive this bumpy ride. Thanks to Sankar da for taking care of so many non-academic issues without which, it would have been difficult for me to focus on the work.

Finally, I would like to thank the persons for whom working for a PhD had become a reality. I have no words to express my gratitude to my brother Aniruddha and Sanjay and my wife Sukti who have always been the source of inspiration. Without their support, I never could have been able to complete this journey. A special thanks to Sukti for her patience and for the support with household duties when I was busy doing thesis works. Thanks to our parents for supporting me unconditionally and for understanding my work pressure without which, this thesis would not have been possible.

Abstract

Traditional information retrieval models work based on statistical co-occurrence and hardly consider any semantic relationships between terms during retrieval. In these models, the relevance of a document with respect to a query is measured by considering the exact term overlap between the query and the document. Natural language documents may use different words to report same concept, creating a *vocabulary mismatch* problem that hurts IR performance. The primary reasons for vocabulary mismatch are $1)$ variation of words due to the morphological events (both derivational and inflectional), and $2)$ use of different vocabulary for the same concept by the creator of documents and the user who is issuing the query. The morphological variational problem can be addressed by using a stemmer that changes all possible variations into a single stem.

To overcome the second drawback, in this thesis, we present a number of methods incorporating semantic relationships based on word embeddings to improve the retrieval experience. We have proposed a generalization of the traditional language model where the mutual independence of occurrence of a pair of words no longer holds. Utilizing the term vectors, the proposed method considers generation of semantically similar terms either from the document or from the collection. Consequently, a number of query expansion techniques, based on word embeddings, are proposed that consider embedding based similar terms as probable expansion terms. Additionally, a word embedding based relevance feedback method is proposed. Based on kernel density estimation, the proposed feedback model enables incorporation of semantic relations by exploiting term compositionality with embedded vectors. The results show the effectiveness of our proposed algorithms over state-of-the-art retrieval models.

If the performance of any given method for a particular query can be estimated in advance (or during retrieval), it may enable us to tailor the retrieval strategy to individual queries, so that the overall effectiveness of the system is improved. To be useful in practice, i.e., to avoid being reduced to a trial-and-error approach, this estimation must not make use of any ground truth / information provided by users about the relevance of different documents. This is the *Query Performance Prediction* (QPP) problem: to predict the effectiveness of an IR method for a given query and a collection of documents, without using relevance judgments. Based on word embeddings, further in this thesis, a novel query performance predictor is proposed. Experiments on benchmark datasets have shown the superiority, in terms of performance, of the proposed method over state-of-the-art performance predictors.

List of Publications

[1] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 795–798, New York, NY, USA, 2015. ACM.
Chapter 4 of this thesis is based on this paper.

[2] Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. Using word embeddings for automatic query expansion. In *Proc. of NeuIR Workshop, collocated with SIGIR*, 2016.
Chapter 5 of this thesis is based on this paper.

[3] Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth J.F. Jones. Word vector compositionality based relevance feedback using kernel density estimation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1281–1290, New York, NY, USA, 2016. ACM.
Chapter 6 of this thesis is based on this paper.

[4] Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth J.F. Jones. Estimating gaussian mixture models in the local neighbourhood of embedded word vectors for query performance prediction. *Information Processing and Management*, 2018. Under review.
Chapter 7 of this thesis is based on this paper.

[5] Dwaipayan Roy, Mandar Mitra, and Debasis Ganguly. To clean or not to clean: Document preprocessing and reproducibility. *ACM Journal of Data and Information Quality*, 2018. To appear.
Chapter 3 of this thesis is based on this paper.

[6] Dwaipayan Roy, Debasis Ganguly, Sumit Bhatia, Srikanta Bedathur, and Mandar Mitra. Using Word Embeddings for Information Retrieval: How Collection and Term Normalization Choices Affect Performance. In *Proceedings of the 27th ACM International on Conference on Information and Knowledge Management*, CIKM '18, to appear, New York, NY, USA, 2016. ACM.
Chapter 5 of this thesis is based on this paper.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

The amount of information produced in digital form (as text, images, audio, video etc.) has seen an exponential growth in the last few decades, creating a need for efficient handling of this information. Search engines have emerged and evolved to meet this need, by helping people to quickly find relevant items from the vast amounts of information available. Indeed, search engines have become an integral part of our day to day lives: people use search engines on various platforms, from mobile phones to personal computers (PCs), to look for information both online (e.g., to check the local weather forecast), and offline (e.g., to search for a .doc file stored in their phone / PC). Among the different formats in which information is stored digitally, text continues to be widely used. The primary task of a text-based search engine can be broadly described as follows: given a user's query, find textual documents that provide relevant information to the user.

A great deal of research in *Information Retrieval* (IR) focuses on designing and testing the algorithms that drive a search engine. As a discipline, IR may thus be loosely defined as the science behind searching. Given a query Q, most IR models compute a score for each document that is intended to be indicative of the document's usefulness. The documents are then presented to the user (often along with the scores) in decreasing order of presumed usefulness. For a document, the score (commonly referred to as the *similarity* score) corresponding to query Q is typically computed by considering terms that are common to both the document and the query. Search algorithms that compute document scores on the basis of terms present in both documents and the query are called term-overlap based searching algorithms.

Unfortunately, a query and a relevant document may use different terms to refer to the same concept. For example, a document that is relevant to a query on *automobile emissions* may not use the terms *automobile* and *emissions* at all; instead, it may use the term *car exhausts* when discussing the same subject. When this happens, a simple, term-overlap based search algorithm will likely fail to retrieve such relevant documents. This is termed the *vocabulary mismatch* problem. In many cases where term-overlap based search algorithms fail to provide satisfactory results, the poor performance may be attributed to vocabulary mismatch. This problem is often particularly acute for short queries.

Query Expansion (QE) is one standard approach that attempts to overcome the vocabu-

lary mismatch problem. In this technique, a set of additional terms that are related to the user's information need are added to the initial query. These terms (called *expansion terms*) are expected to increase the chance of a match between the query and a hitherto unretrieved, relevant document. They are also expected to improve the match between the query and retrieved, relevant documents.

For many common search activities, however, users are either unaware of good expansion terms (consider searching for a solution to a knotty computer-related problem, for example), or are simply unwilling to make the effort required to provide additional query terms. Therefore, a query expansion algorithm should try to solve the problem of vocabulary mismatch without any explicit additional input from the user. A sub-field of research on IR, commonly known as *Automatic Query Expansion* (AQE), tries to address this problem, and has been an active topic of research over the last few decades. Since fully automatic QE methods do not involve any human supervision, they are prone to error, in the sense that they sometimes add unrelated, irrelevant terms to a user's query, changing the focus of the query. This is known as *query drift*. In such cases, query expansion causes performance deterioration rather than improvement.

Given an oracle, it would be possible to know in advance which queries would be benefitted, and which ones would be hurt by AQE. A system would then be able to "maximise" performance, by selecting, for each query, the appropriate retrieval strategy for the particular query at hand (in this case, by deciding whether or not to expand the query automatically). More generally, if the performance of any given method for a particular query could be estimated in advance (or during retrieval), it might be possible to tailor the retrieval strategy to individual queries, so that the overall effectiveness of the system is improved. To be useful in practice, i.e., to avoid being reduced to a trial-and-error approach, this estimation must not make use of any ground truth / information provided by users about the relevance of different documents. This is the *Query Performance Prediction* (QPP) problem: to predict the effectiveness of an IR method for a given query and a collection of documents, without using relevance judgments.

In this thesis, we investigate novel approaches for both Automatic Query Expansion and Query Performance Prediction. The common idea underlying the proposed approaches is the use of *word embeddings*. Broadly, a word embedding maps each word in a text collection to a k-dimensional[1] vector over the real numbers in such a way that semantically related words are mapped to "similar" vectors. In the following sections, we discuss the motivating factors that led to our exploration of word embeddings (Section 1.1), and present a more precise statement of the research questions addressed in this thesis (Section 1.2).

1.1 Motivation

As outlined above, word embedding techniques take a raw, unannotated text corpus, and map each term of the corpus to a k-dimensional vector (k is typically very small compared to the number of distinct terms in the corpus). The terms of the vocabulary are said to be *embed-*

[1] Here, k is a parameter.

ded in the k-dimensional abstract space, and the vector corresponding to a word is called its *embedding* or *embedded vector*. A word embedding attempts to reflect the underlying semantic relationships between terms in the following sense: terms that are strongly semantically related are expected to be in close vicinity in the embedding space.

Word embedding techniques have been studied in IR at least as far back as 1990 [50]. Interest in the use of word embeddings has been rekindled thanks to relatively recent research in deep neural network based approaches, which have opened up a plethora of opportunities for researchers in different fields of Computer Science, from Computer Vision to Speech Recognition, to Natural Language Processing. In particular, work by Mikolov et al. [103], and the availability of efficient tools to compute embeddings from a corpus[2] have inspired a slew of work that seeks to apply word embeddings to IR problems [110]. The work presented in this thesis studies the application of word embeddings to Automatic Query Expansion (AQE) and Query Performance Prediction (QPP).

Automatic query expansion. As discussed above, the objective of AQE is to find words that are semantically related to a given user query (say Q). Consider word embeddings from an appropriate corpus. If w is a word whose embedding lies close to the embeddings of the query words, then w is likely to be semantically related to Q. It is, therefore, likely to be a useful expansion term. We explore this simple idea in greater detail in Chapters 4–6.

Query performance prediction. The quality of search results returned for a query depends on a very large number of factors. In our work, we focus on query ambiguity. If a query is ambiguous, then it may be difficult for a search engine to return satisfactory results for the query. A classic example of an ambiguous query is '*python*': the user may either be asking for information about a kind of reptile, or about Python, the programming language (or possibly about some other, less commonly known sense of the term python). While the information need in the mind of the user is clear, an IR system does not usually have any insight into this need. Thus, the top ranked documents retrieved by the system may contain documents about different possible senses of the query. It may well happen that only a few among these documents, i.e., the ones which are related to the intended sense of the query, are relevant to the user. This can lead to poor IR effectiveness.

We use word embeddings to quantify query ambiguity, and use this quantitative estimate of ambiguity in turn to predict query difficulty. To explain how embeddings may be used to measure ambiguity, we consider, for simplicity, a query containing the single term q. Let $W = \{w_1, w_2, \ldots, w_m\}$ be words whose embeddings lie close to the embedding for q. If q is an unambiguous term, then $w_1, w_2, \ldots, w_m$ are all likely to be semantically related to the single sense of q, and therefore to each other. On the other hand, if q is an ambiguous term like *python*, then some words in W are expected to be related to programming, while the others are likely to be related to the reptile. Thus, we expect the words within W to form two groups, one corresponding to each major sense of *python*. Extending this hypothesis to the embedding space, we expect that, if the vector embeddings of words in W are clustered, the number of clusters formed will be roughly indicative of the number of senses of q. In other

[2] Example: the word2vec software package.

words, it may be taken to be a measure of the ambiguity of q. This idea is studied in greater detail in Chapter 7.

Term compositionality in Information Retrieval

Word embeddings have another attractive feature that promises to be useful in IR applications. The addition of vectors in the embedding space has been shown to correspond nicely to a composition of the semantics of the corresponding words [103]. For example, if the embeddings of the terms *german* and *airline* are added together, the vector so obtained is close to the vector for the term *Lufthansa*.

IR systems that are based on term overlap commonly look for matches on individual words. However, a match on a multi-word construct (often referred to as a *phrase* in IR terminology) can often be more strongly indicative of usefulness. For example, a match on the phrase *blood pressure* would generally be regarded as a more specific match than a combination of individual matches on the words *blood* and *pressure*. To leverage such matches, IR systems sometime consider phrase matches, or term proximity during retrieval. The observation that simple vector addition of word embeddings can be used to capture the meaning of a set of terms taken together opens up new possibilities of matching concepts which are not adequately represented by single words alone. In our work on AQE and QPP, we make use of this compositionality property of word vectors.

1.2 Research questions and contributions

The overall research question addressed in this thesis is whether the performance of different IR methods can be improved by using semantic relations captured by word embeddings, and by utilizing the property of compositionality of embeddings. Particularly, we try to improve retrieval performances of baseline and query expansion methods as well as the accuracy of query performance predictors using embeddings. The specific research questions addressed in the respective sections of the thesis is outlined in the following section.

1.2.1 Chapter 3: Document preprocessing for IR

Before starting our study of applying word embeddings to IR, we faced a seemingly basic, operational question. When indexing the documents in any collection, an IR system has to decide which parts of a document to tokenise, what characters to regard as token delimiters, the nature of the tokens themselves (e.g., words, n-grams), which tokens to discard, if any (e.g., strings consisting of numerals only), etc. We refer to these tasks collectively as *document preprocessing* or *parsing*. Published articles generally do not provide specific details about preprocessing, suggesting that these details are probably not important, but our preliminary experiments suggested that document preprocessing choices can *significantly* affect retrieval effectiveness. This prompted us to consider more carefully the choices made regarding which parts of documents to index, and the impact of these choices on retrieval effectiveness. For document collections consisting primarily of relatively clean, textual content, the choice is reasonably clear: the entire document should probably be indexed. In contrast, collections

consisting mostly of Web documents generally contain a significant amount of markup and meta-content. These portions of the document — informally referred to as *noise* in this thesis — should probably be discarded during preprocessing. This hypothesis led to the following questions.

- Should noise be removed from documents (particularly Web documents) prior to indexing? How much difference does noise-removal make? Does the answer depend on the retrieval model applied?
- What is the effect of noise removal on query expansion (QE) techniques?

Our experiments show that removing noise during document parsing can indeed significantly affect retrieval effectiveness, *but this effect varies across retrieval models as well as test collections*. For example, metadata filtering was found to be generally beneficial when using BM25 [128], or language modeling [118, 82] with Dirichlet smoothing [175], but can significantly hurt performance if language modeling is used with Jelinek-Mercer smoothing. We also observed that, in general, the performance differences become more noticeable as test collections become more noisy. In a second set of experiments, we studied the effect of preprocessing on query expansion. In this case, once again, we found that it is generally better to remove markup before using documents for query expansion. Thus, preprocessing details actually appear to be important, their omission from published work notwithstanding.

1.2.2 Chapter 4: Generalized language model

With preprocessing concerns out of the way, we move on to our objective of exploring how word embeddings may be applied to IR. Our first approach attempts to incorporate semantic similarities between words (as reflected in the similarity between their embeddings) into the Language Modeling framework in a principled way.

According to the conventional language modeling approach, each document d may be regarded as an urn containing words. The smoothed[3] probability estimate of picking the query words from this urn is regarded as the document's similarity score. We construct a *generalized language model*, in which a query term t may be picked from the document d via two distinct event types.

- As in the conventional language model, t may occur as-is in d (or in the collection $\mathcal{C}$).
- Additionally, a different word t' may be picked first from d (or $\mathcal{C}$), and then transformed into t. The probability of transforming t' into t depends on the semantic similarity between the two words. As discussed above, this semantic similarity can be computed using vector embeddings. Thus, the probability of transforming *cigarette* to *smoking* should be relatively high, while the probability of transforming *cigarette* to *submarine* is expected to be low.

Our experiments show promising results on standard test collections: the proposed generalized language model outperforms the basic language modeling approach to IR.

[3] Smoothing is discussed in detail in Section 2.2.1.

1.2.3 Chapter 5: Query expansion techniques based on word embeddings

Chapter 5 presents a framework for query expansion that uses embeddings as outlined in Section 1.1: candidate expansion terms are obtained by applying a K-nearest neighbours approach in the embedding space. We explore two query expansion techniques within this framework. In the first variation, we search for the nearest neighbours of the query among the embeddings of all index terms in the collection. This method only makes use of the query terms, and does not depend on any retrieval method. Thus, this method can be categorized as a *pre-retrieval* query expansion technique. In the second variation that we propose, the search for nearest neighbours is restricted to the set of words contained in the top-ranked documents retrieved for the initial query by some retrieval method. For both the methods, the weights of the expansion terms are computed as a function of the vector similarity between the embeddings of the expansion terms and those of the query terms.

For our initial experiments with the QE techniques outlined above, we ran word2vec on the documents in the respective test collections to generate word embeddings. A number of variations on this theme are possible. In the second part of the chapter, we study the effect of these variations on the proposed QE method. Specifically, we consider the following issues.

- Choice of word vector training algorithm, e.g., word2vec vs. fastText[4] [21].
- Term normalization, i.e., should the words in the corpus be stemmed before training word vectors, or should the vectors be computed using an unstemmed corpus? In the latter case, an appropriate method is required to handle vectors corresponding to inflected variants of a word.
- Choice of document collection used to generate word vectors. If a generic collection of documents (such as Wikipedia[5]) is used to compute embeddings, how much difference does it make as compared to when embeddings are generated from the test collection in question?

In this connection, we also propose a quantitative measure for estimating the similarity of word vectors obtained under different settings,

1.2.4 Chapter 6: Pseudo relevance feedback based query expansion technique using word embeddings

A limitation of standard IR models is that the notion of term compositionality is restricted to the use of pre-defined phrases and term proximity. Standard text based IR models provide no easy way of representing semantic relations between terms that are not necessarily phrases, such as the equivalence relationship between 'osteoporosis' and the terms 'bone' and 'decay'. To alleviate this limitation, we introduce a pseudo relevance feedback (PRF) method which makes use of word embeddings. We leverage the fact that the vector addition of word embeddings corresponds to a semantic composition of the corresponding terms, e.g., addition of the vectors for 'bone' and 'decay' yields a vector that is likely to be close to the vector for the word 'osteoporosis'. Our proposed PRF model enables incorporation of semantic relations

[4] https://fasttext.cc/
[5] http://en.wikipedia.org

by exploiting term compositionality with embedded word vectors. We develop our model as a generalization of the relevance model (RLM) [91, 86]. Experiments on standard test collections show that the proposed generalization yields improvements over the standard relevance model.

1.2.5 Chapter 7: Word embedding based query performance predictor

As discussed in Section 1.1, the hypothesis underlying our proposed query performance predictor is that ambiguous queries are difficult. Some existing QPP approaches [75] take ambiguity into account with the help of static resources, such as WordNet [104]. We propose a novel word embedding based method for measuring the ambiguity of each query term by estimating how many 'senses' each word is associated with, within a given document collection. The key idea is to estimate a Gaussian Mixture Model (GMM) from the nearest word vectors of a query term. Each component of the mixture can be thought of as being associated with a particular sense of the query term. We then make use of the prior probabilities of the GMM components to measure how uniformly the words in this neighborhood are distributed.

The query predictor proposed in Chapter 7 is a *pre-retrieval* type predictor, i.e., it does not make use of the ranked list of documents retrieved for a query in order to estimate its difficulty. QPP approaches that use information from the top-ranked documents are categorized as *post-retrieval* predictors. Such predictors often outperform pre-retrieval predictors because of the additional information that they make use of. To achieve the best of both worlds, we propose a linear combination of a pre-retrieval predictor with a standard post-retrieval predictor function, called Normalized Query Commitment (NQC) [145], which uses the variance within the retrieval scores of the set of top-ranked documents to estimate query difficulty. Experiments on TREC data collections demonstrate that the proposed combination almost always outperforms other state-of-the-art predictors.

1.3 Thesis outline

The rest of the thesis is organized in the following way.

- Chapter 2 provides some basic background about IR and Word Embeddings that is related to the work presented in this thesis.

- Chapter 3 describes the effect of document pre-processing on IR effectiveness. Specifically, we consider the question of whether to remove or retain the meta-information from documents in a Web collection during indexing. Our experiments show that varying the pre-processing step for Web document collections results in significant changes in performance, both for baseline as well as query expansion based retrieval methods. The work presented in this chapter settles a basic operational question, thereby laying the foundation for the rest of the work reported in this thesis.

- Chapter 4 presents the generalized language modeling method that uses embedding similarity to incorporate semantic information into the standard language modeling

approach to IR. The proposed method is compared with the standard LM based retrieval model on several benchmark TREC collections.

- Chapter 5 describes some query expansion methods based on word embeddings. The proposed pre-retrieval query expansion method is shown to significantly outperform baseline methods on a number of test collections. Additionally, we study how the performance of the proposed QE method is affected by differences in choices made while training the word embeddings.

- Chapter 6 discusses a novel pseudo relevance feedback method that utilises word embeddings. Based on kernel density estimation, the proposed method uses the compositionality of terms to capture conceptual semantics. The proposed method is then used for query expansion, and is found to outperform RM3 [86], a very well-known query expansion approach.

- Chapter 7 proposes a novel query performance predictor that uses word embeddings. The proposed method is a pre-retrieval predictor, i.e., it does not depend on any retrieval model / function. To incorporate the post-retrieval information in the prediction model, a state-of-the-art predictor is linearly combined with the proposed predictor, resulting in a robust predictor. Experimental results on several benchmark TREC datasets are presented to permit a comparison between the proposed predictor and several state-of-the-art predictors.

- Chapter 8 concludes the thesis. It highlights our primary observations, and suggests some directions for further study.

Chapter 2

Background

Information Retrieval (IR) refers to the science of searching. From locating a phone number in a telephone directory, to fetching information about the nearest hospital in a locality from a map, are all forms of IR. Until some time in the second half of the previous century, IR was mostly an activity that involved a small handful of persons (like librarians and para-legal professionals). Since the advent of the World Wide Web, an enormous number of people from diverse backgrounds and professions have been regularly engaged in activities like Web search or email search that involve IR.

In this chapter, we start by covering the basics of IR. An architectural view of the process of information retrieval is presented with a brief discussion. Later in this chapter, important retrieval models are briefly reviewed before presenting the evaluation procedure, which will ease the path for understanding the work presented in the rest of this thesis.

2.1 Information Retrieval: a primer

Even in the current age of rapid technological advancement, textual data continues to be among the most prevalent forms in which information is stored. In this thesis, we focus on the problem of *textual information retrieval*, which may be defined as follows:

> Given an information need, and a collection of documents stored in unstructured textual form, *Information Retrieval* is the process of finding documents from that collection that satisfy the information need.

The whole information retrieval process broadly consists of two parts, i) indexing and ii) retrieval. While indexing, the collection of documents (commonly known as a *corpus*), is processed and stored for efficient and fast retrieval. Removal of non-informative terms (commonly known as *stopwords*) and stemming are performed during processing the collection prior to indexing. The indexing process primarily involves generating an inverted index from the corpus. The inverted index consists of two parts, i) an *inverted list* (or, *posting list*), and ii) a *dictionary*. In an inverted list, each term of the vocabulary is stored along with a list of documents containing that term. The list containing all the terms of the vocabulary is called a dictionary. For fast retrieval, the dictionary is kept in the primary memory with pointers

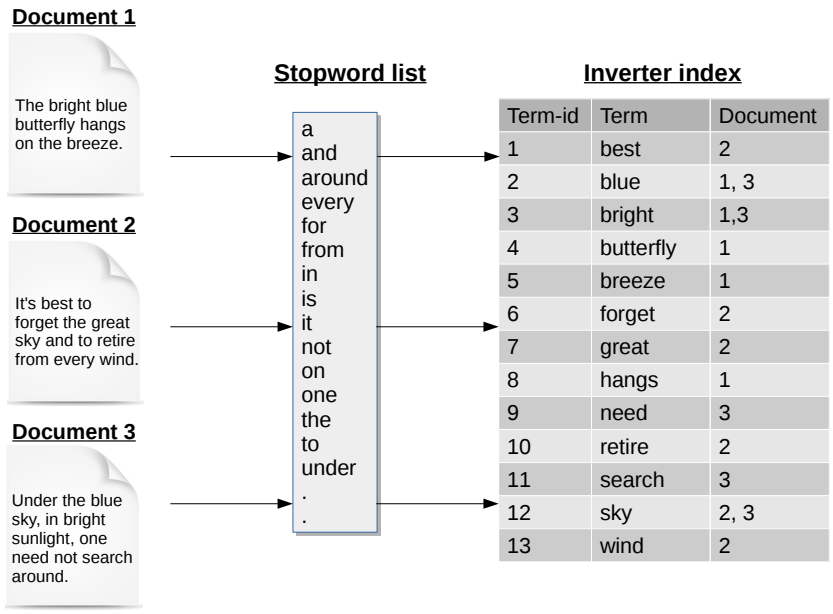


Figure 2.1: A conceptual model of indexing for IR.[1]

to each inverted lists. The inverted lists are stored in the secondary memory. For ranked retrieval, the inverted index contains a term weight for each document in which the term occurs. Additionally, the dictionary stores per-term collection statistics during indexing so that a single lookup in the dictionary can extract all necessary information to compute the retrieval score for a document during retrieval.

The process of indexing is broadly presented in Figure 2.1 where the collection consists of three documents.

2.2 Baseline retrieval models

The set of algorithms, that work behind a search engine to fetch potential relevant information, are called retrieval models / functions in research community. There are a number of retrieval functions that are known to be effective.

We now turn to the retrieval models that we considered in this thesis. We selected language model based retrieval methods, with two different smoothing techniques, namely Jelinek-Mercer smoothing, and Dirichlet smoothing [174, 175, 173]. These techniques are popular among researchers as baseline retrieval methods [171, 170, 178, 68, 111]. For some of our works, we also experimented with probabilistic and information theoretic methods, specifically BM25 [127, 128] and Divergence from Randomness [6] retrieval models. In the rest of this section, we briefly review these models and methods. In the next section, we will discuss about relevance feedback, and one of the popular state-of-the-art relevance feedback based query expansion technique (namely relevance based language model) that we used as baseline in some works reported in this thesis.

[1] Reproduced from http://www.writeopinions.com/inverted-index

2.2.1 Language modeling

The general idea of language modeling based retrieval can be described in the following way. Let Q be a query and d be a document. Let $\mathcal{D}$ represent the language model estimated from d. Then the score of document d with respect to query Q is given in decreasing order by the posterior probabilities $P(\mathcal{D}|Q)$. These posterior probabilities are estimated for each document d of the collection while indexing the collection, with the help of the prior probability $P(Q|d)$ according to the Bayes rule [117, 82, 175].

$$\begin{aligned} p(d|Q) &= \frac{p(Q|\mathcal{D}).p(\mathcal{D})}{\sum_{d' \in C} p(Q|\mathcal{D}').p(\mathcal{D}')} \propto p(Q|\mathcal{D}).p(\mathcal{D}) = p(\mathcal{D}).\prod_{q \in Q} p(q|\mathcal{D}) \\ &\propto \prod_{q \in Q} p(q|\mathcal{D}) \end{aligned} \tag{2.1}$$

The language model $\mathcal{D}$ associated with the document d is usually approximated by a unigram language model, i.e., $\mathcal{D} = \{p(w_i|d)\}_{i \in [1,|V|]}$ where V is the size of the vocabulary, and $p(w_i|d)$ represents the probability of randomly picking word w_i from document d.

Thus, the retrieval score of d for a given query Q can be written as

$$\begin{aligned} \text{Score}(Q,d) &= p(Q|\mathcal{D}) \\ &= \prod_{q \in Q} p(q|d) \end{aligned} \tag{2.2}$$

Jelinek-Mercer smoothing

To overcome the zero probability problem (when a query term is missing from d, its score becomes zero according to Equation 2.2), the language model $\mathcal{D}$ is smoothed by interpolating the Maximum Likelihood Estimate (MLE) of $p(w_i|d)$ with a background language model, estimated from the entire collection C, as in Equation 2.3.

$$\begin{aligned} p(Q|\mathcal{D}) &= \prod_{q \in Q} \left[\lambda p(q|d) + (1-\lambda) p(q|C)\right] \\ &= \prod_{q \in Q} \lambda \frac{tf(q,d)}{|d|} + (1-\lambda)\frac{cf(q)}{|C|} \end{aligned} \tag{2.3}$$

Here, $tf(q,d)$ and $cf(q)$ respectively indicates the count of term q in the document, and in the whole collection. $|d|$ and $|C|$ are the size of the document and the collection respectively. $\lambda = [0,1]$ is the interpolation parameter. This language modeling based retrieval model (Equation 2.3) is known as language model with Jelinek-Mercer smoothing or linear smoothing, and is referred to as JM or, LM-JM in the rest of this thesis.

Dirichlet smoothing

Another smoothing method that is also used by the researchers is the Dirichlet prior smoothing method that uses Bayesian estimation instead of MLE. The mathematical form of this model is similar to the language model with Jelinek-Mercer smoothing (see Equation 2.3),

except the interpolation parameter has a dynamic coefficient that changes according to document length (see Equation 2.4).

$$P(Q|\mathcal{D}) \quad = \quad \prod_{q \in Q} \frac{tf(q,d) + \mu p(q|C|)}{|d| + \mu} \tag{2.4}$$

Here, $tf(q,d)$ is the term frequency of q in document d, $|d|$ is the document length, and $p(q|C|)$ is the collection probability of q in the whole collection. μ is the interpolation parameter, and can be interpreted as the pseudo counts of words introduced through the prior. The practice is, μ is set in the range $[100, 5000]$. The language model based retrieval model presented in Equation 2.4 is known as language model with Dirichlet prior smoothing. In the rest of this thesis, Dir and LM-Dir are used interchangeably to refer to the same model that is presented in Equation 2.4.

2.2.2 BM25

BM25 is a traditional probabilistic retrieval model [127, 128] with better length normalization factors. Equation 2.5 mathematically presents the BM25 model to score a document d for a given query Q.

$$\text{Score}(Q,d) \quad = \quad \sum_{q \in Q} \log \frac{D - df(q) + 0.5}{df(q) + 0.5} \frac{tf(q,d)(k_1+1)}{tf(q,d) + k_1 \dot{(}1 - b + b\frac{|d|}{avgdl})} \tag{2.5}$$

In Equation 2.5, D is the number of documents in the collection, $df(q)$ is the number of documents in the collection containing the term q, $tf(q,d)$ corresponds to the count of term q in document d, and $avgdl$ is the average document length of the collection. b is a length normalization parameter, and k_1 is a positive tuning parameter that calibrates the document term frequency scaling [128].

2.2.3 Divergence from Randomness

The Divergence from Randomness (DFR) model has a number of associated parameters. For a given query $Q = \{q_1, \ldots q_k\}$, the within document term weight of a query term q ($q \in Q$) is defined as:

$$\text{w(q,d)} = \frac{cf(q)+1}{df(q)(tfn+1)} \big(tfn. \log_2 . \frac{D+1}{cf(q)+0.5} \big) \tag{2.6}$$

where,

- $cf(q)$ - collection frequency of q in the whole collection
- $df(q)$ - document frequency of q in the collection
- D is the number fo document in the collection

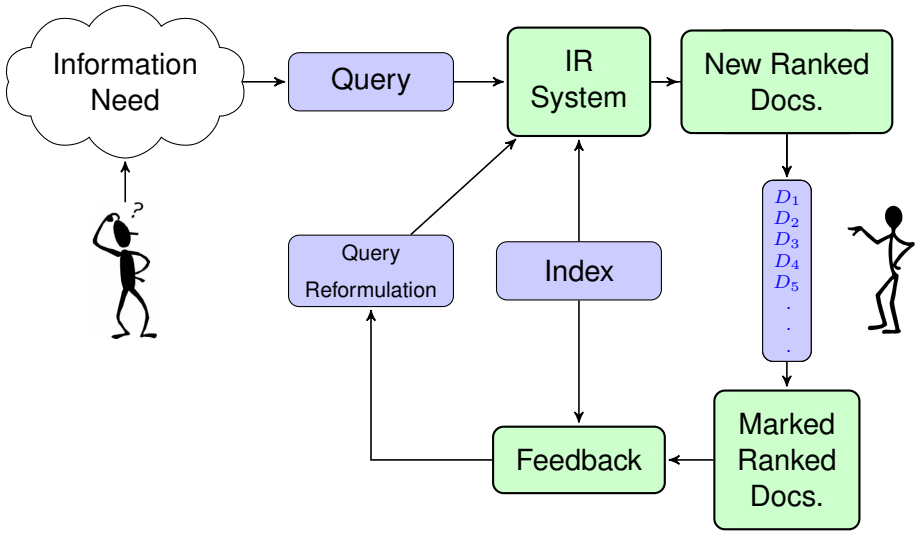


Figure 2.2: A conceptual view of explicit relevance feedback.

- tfn is the normalized term frequency, and defined as:

$$tfn = tf(q) \cdot \log_2(1 + c \cdot \frac{avgdl}{|d|})$$

with c being a parameter whereas, $|d|$ and $avgdl$ denotes the document length and average document length respectively.

The non-trivial details of the different parameters and normalization factor of DFR models are beyond the scope of this report. For a comprehensive discussion of DFR model , please refer to [80, 6, 4].

2.3 Relevance feedback and query expansion

One of the prevalent problem of each of the retrieval models discussed above, and in general a commonly known challenge for IR, is the problem of *vocabulary mismatch* [60]. Consider a user query Q having a relevant document D corresponding to it. Q and D may use different set of words (i.e. vocabulary) to refer to the same concept. In this practically likely event, retrieval models based on keyword overlapping may not be able to retrieve D in response to Q. A classic example of vocabulary mismatch occurs when two documents express the same topic, one using the phrase “atomic energy” whereas the other using “nuclear power”.

The general idea of *relevance feedback* is to involve the users in the retrieval process to improve final result. Particularly, after querying for an information need, the user gives a feedback on the relevance of the documents in an initial retrieved set of documents. Based on this feedback, the retrieval system modify the representation of the information need, and performs a second round of retrieval.

Based on the way the feedback is received, relevance feedback process can be categorized in three categories:

1. **Explicit relevance feedback:** Users specify search results as relevant and non-relevant.

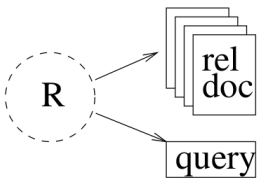


Figure 2.3: The query and its relevant documents are both random samples drawn from the underlying relevance model R (reproduced from [91]).

2. **Implicit relevance feedback:** Feedback is considered on the basis of user activity (e.g. previous searches).

3. **Pseudo relevance feedback:** Top n documents are considered as relevant.

The basic structure of explicit relevance feedback is graphically presented in Figure 2.2. However, in practice, explicit or implicit feedbacks are hard to get, and pseudo relevance feedback turns out to be very effective. In this section, we will discuss about expansion technique relying on pseudo-relevance feedback. Specifically, we will describe RLM, a popular acronym for relevance based language model [91, 86], that is popular among researchers as a query expansion technique based on pseudo relevance feedback. RLM is an association based method that remains an effective and widely used approach for query expansion. For a given query $Q = \{q_1, \ldots q_k\}$, the relevance model hypothesises that there is a latent probability distribution R that generates both the query Q, and its relevant documents (graphically presented in Figure 2.3). The relevance model R is then approximated on the basis of terms of Q, and set of known relevant documents. The terms of R with large probability weights can then be selected as the potential expansion terms. The better the estimation of R, the better performance of the query expansion technique. In the absence of training data, M top ranked pseudo-relevant documents are considered as the set of relevant documents. The terms of query Q serves as the exclusive evidence about the relevance model, that is $q_1, \ldots q_k$ are certainly generated from R. Thus, the probability density function of RLM, or the probability of sampling a term w from R, denoted by $P(w|R)$, is approximated by $P(w|Q)$, the conditional probability of observing w along with query terms $q_1, \ldots, q_k$.

$$P(w|R) \approx P(w|Q) = \frac{P(w,Q)}{P(Q)} \sim P(w,Q) = P(w, q_1, \ldots q_k) \tag{2.7}$$

Following Equation 2.7, estimating the probability density function $P(w|R)$ boils down to estimating the joint probability of observing w together with query terms $q_1, \ldots, q_k$. The joint probability $P(w,Q)$ is then estimated using the following two ways, i) *independent and identically distributed* (IID) sampling, and ii) *Conditional* sampling.

IID Sampling. In IID sampling, the term w is sampled conditionally, together with the query terms $q_1, \ldots, q_k$ from the same distribution, underlying a top ranked document. Therefore the probability estimation of $P(w|R)$ will follow Equation 2.8.

$$P(w|R) \approx P(w,Q) = \sum_{d \in D} P(w|d) \prod_{q \in Q} P(q|d) \tag{2.8}$$

Conditional Sampling. The term w is sampled conditionally, together with $q_1, \ldots, q_k$ from independent document models, each corresponding to a document retrieved at top rank. Following this process of sampling, $P(w|R)$ is estimated as shown in Equation 2.9.

$$P(w|R) = P(w) \prod_{q \in Q} P(q|w) = P(w) \prod_{q \in Q} \sum_{d \in D} P(d|w)P(q|d) \tag{2.9}$$

In Equation 2.8 and 2.9, D is the set of top M retrieved documents and, $Q = \{q_1, \ldots q_k\}$ is the query. The two variants of RLM in Equations 2.8 and 2.9 are referred to as **RM1** and **RM2**, respectively, in the literature [97].

The RLM, as proposed in [91], only considers the aspect of a term in the initial retrieved document list, without giving any special importance to the terms which are present in the query. Explicitly considering the role of query terms, in [86], Jaleel et al. linearly interpolated the relevance model with the query model (see Equation 2.10) and achieved significant improvement over the traditional relevance model.

$$P'(w|R) = (1 - \phi)P(w|R) + \phi P(w|Q) \tag{2.10}$$

In Equation 2.10, the query model $P(w|Q)$ is computed using maximum likelihood estimation (MLE), and $P(w|R)$ can be computed using either Equation 2.8 (RM1) or Equation 2.9 (RM2); $\phi \in [0, 1]$ is the linear interpolation parameter, that controls the contribution from both the language models. These are known as RM3 and RM4, respectively. Since RM3 has been shown to empirically outperform RM4 (and hence also RM1 and RM2, i.e., the respective models without the query mixture components) [97], we use the RM3 version of RLM (see Equation (2.11) as baselines in our experiments, and simply refer to it as RLM (or RM3 interchangeably) throughout the rest of the thesis.

$$P'(w|R) = (1 - \phi)\Big(\sum_{d \in D} P(w|d) \prod_{q \in Q} P(q|d)\Big) + \phi P(w|Q) \tag{2.11}$$

From the estimated probability distribution model R, top N terms with the highest probability distribution weights $P'(w|R)$ are chosen to be the expansion terms (with N being a parameter). The weight of the N expansion terms are then sum normalized to one before performing the retrieval with query expansion. Thus it is crucial for relevance model estimation to assign high weights to terms which are exclusive to the relevant documents, and to avoid assigning significant weights to common terms. Therefore, any form of filtering of the documents (in our case, metadata exclusion) may lead to deviation in performance.

For performing the initial retrieval, and retrieval with the expanded query, any retrieval model can be used. Following the trend in literature ([91, 86, 97]), as RLM is a language model based QE method, in the experiments reported in this thesis, we perform both, baseline and expanded retrievals using the language model with the two different smoothing methods, namely Jelinek-Mercer (JM) and Dirichlet (Dir) smoothing.

2.4 Word embeddings

In this section, we will start by discussing the fundamental background of word embedding techniques primarily focussing on *word2vec*, one of the most popular embedding technique.

After discussing about the structure of the embedding model, the application of embeddings are discussed later in this chapter, focussing on Information Retrieval.

2.4.1 Word embeddings

Although the idea of using vector representation for terms has existed for some time [40], the interest on using word embedding has expanded in various areas of natural language processing in recent years following the introduction of the word2vec algorithm by Mikolov et al. [103]. Word embedding technique, such as *word2vec* [103] (or *GloVe* [116]) represents words as vectors in an abstract dimensional space. Given a large amount of unlabeled training data, *word2vec* learns a low dimensional vector representation of words such that the similarity between the word vectors can reflect the semantic similarity between the constituent terms. The resultant representation is also shown to replicate many linguistic regularities like *conceptual combination* and *laws of analogy* by simple addition and subtraction of word vectors. The following are some of the useful properties of word embeddings:

1. Word embedding enables approximation of the *semantic similarity* between two words by the cosine similarity between their corresponding vectors.

2. The effect of *conceptual composition* can be captured by simple addition of the embedded vectors, e.g., the resultant vector after addition of vec('Russia') and vec('river'), will be close to vec('Volga').

3. The embedded vectors are found to obey the *laws of analogy*. For example, it gives a vector which is close to the representation of vec('Rome') as a result of the operation vec('Paris') − vec('France') + vec('Italy').

4. The embedded vectors can be used as features for various supervised text processing tasks such as document classification, named entity recognition, and sentiment analysis. The underlying semantic information in these vectors make them a powerful features for these tasks.

In our work, we mainly exploit the first key feature, i.e. that of the semantic distances between the terms in order to derive the similarity function to rank a document.

There are two word2vec models to represent word embeddings, *i)* skip-gram model and *ii)* continuous bag-of-words (CBOW) model. Both of these models use a neural network with a single hidden layer. The skip-gram model *predicts the context, given a word* whereas the CBOW model *predicts the word, given its context.* A graphical view of both the models are presented in Figure 2.4).

The key algorithm by which the neural network in word2vec is trained is that of *negative sampling*. For each word, a positive evidence set $\mathcal{D}$ is defined consisting of the words in its context. Similarly, the words that are outside the context of the word forms a negative evidence set $\mathcal{D}'$. Given a pair of words (w_t, w_c), the probability that the word w_c is seen in the context of words of w_t then represented as

$$arg\,max_\theta \prod_{(w_t,w_c)\in\mathcal{D}} P(D_{w_t,w_c} = 1|w_t, w_c) \prod_{(w_t,w_c)\in\mathcal{D}'} P(D_{w_t,w_c} = 0|w_t, w_c) \tag{2.12}$$

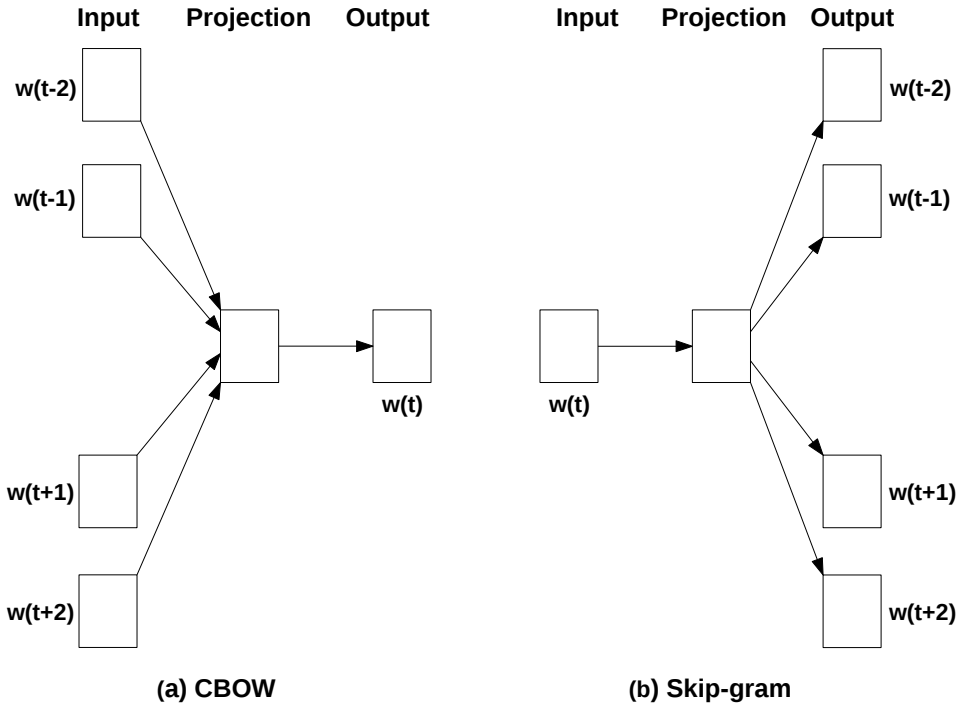


Figure 2.4: The graphical representation of the *skip-gram* and the *continuous bag-of-words* models of `word2vec`

.

The objective function for training the RNN aims to maximize the probability of sampling w_t from its context, i.e. $\mathcal{D}$, whereas minimizing the probability of sampling it from anywhere outside its context, i.e. $\mathcal{D}'$. In terms of vector representation of the words, the optimization function can be written as shown in Equation 2.13.

$$arg\ max_\theta \sum_{(w_t,w_c)\in\mathcal{D}} log\sigma(v_{w_c} \cdot v_{w_t}) - \sum_{(w_t,w_c)\in\mathcal{D}'} log\sigma(v_{w_c} \cdot v_{w_t}) \tag{2.13}$$

In Equation 2.13, $v(w)$ represents the embedded representation of the word w, and σ is the sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$). The objective function shown in Equation 2.13 ensures that the inner product between the vectors for the word w_t and its context w_c is maximized, i.e. these vectors become more similar, whereas the vector for the word w_t becomes more dissimilar with other vectors outside its context.

2.4.2 Word embeddings for IR

Word embeddings and neural network based techniques have gained a significant interest in the text processing community in recent times. Apart from the application in natural language processing (NLP) ([138, 53, 63, 87]), these techniques have proved their usefulness for many information retrieval tasks as well. The works in this directions are mainly focussed in overcoming the vocabulary mismatch problem in IR (see Section 2.3). One of the initial publications in this direction was published in 2009 where Salakhutdinov and Hinton [137] used a deep auto-encoder for semantic modeling to search related documents. In another earlier

work on application of embeddings for IR [39], Clinchant and Perronnin used the latent semantic indexing [50] (popularly known as LSI, is another embedding technique that captures semantics of words using singular value decomposition on a word count matrix) based embeddings to get Fisher Vectors (using [85]) which are finally compared using cosine similarly for document ranking.

For other tasks, e.g. in [157] Vulić and Moens combined embeddings with a context-predicting distributional semantic model (DSM) for croos-lingual information retrieval by representing query and document as sum of embeddings learned from a bilingual document collection. In another work on ranking [132], documents are modelled as mixture distribution of concepts using K-means clustering which is performed on the basis of the embeddings of the terms in the document. The documents are then scored on the basis of likelihood of a query to be generated from the document which is computed by the average inter-similarity of query terms to the centroids of clusters in the document. In a similar work [22], Boytsov et al. considered cosine similarity between the averaged word vectors of query with the document as a similarity function, in k-NN based retrieval. A neural translation model is proposed in [183] that integrates embeddings into the translation model based retrieval method proposed by [18].

The neural network based deep structured semantic model (DSSM) was proposed for the ad-hoc search task in [84] which was the pioneering model of a set of consequent works [105, 106, 166]. In embedding based query language models [169], Zamani and Croft proposed two query language models and an extended relevance model in which term weightings are computed using word embedding. In another work, Zamani and Croft [170] developped methods for query embedding based on the embedded vectors of individual vocabulary terms. They applied the proposed models for query expansion and query classification tasks. In [178], Zheng and Callan proposeed construction of term features directly from word vectors and model the target term weight as a regression problem.

Utilizing the semantic similarity of terms in the abstract space, there are different techniques proposed for query expansion. For example, a heuristic model for per-term query expansion using embedded vectors is proposed in [3]. For the given query, terms semantically similar to query words are considered as the expansion terms and their weights are set in proportion to their frequency in the expanded query. In [136], some k-NN based QE approaches were proposed in which, the expansion term weights were set according to their semantic similarity to the query terms (to be elaboratively discussed in Chapter 5). In a similar work, [126] recommended choosing expansion terms based on a global similarity threshold rather than by k-NN. An implicit query expansion method was proposed in [56] where they trained the embedding model on the topic-constrained, target corpora and claimed to achieved better performance than the global embedding based method. Application of word embeddings for personalised query expansion is explored in [7] as well where, a k-NN based approach in used for the task of social book search.

Among other tasks, IR researchers have applied word embeddings for different tasks, such as search result diversification [109, 162], query suggestion [150], query auto completion [26], proactive search [96], query auto-completion [26], query classification [170], question an-

Collection Name	Documents	# documents	# topics	# rel-docs
TREC123	Tipster disks 1, 2	741,856	150 (51-200)	37836
TREC678	Tipster disks 4, 5 exclude docs. from CR	528,155	150 (301-450)	13692
Robust	Tipster disks 4, 5 exclude docs. from CR	528,155	100 (601-700)	3720
TREC910	WT10G	1,692,096	100 (451-550)	5980
GOV2	GOV2	25,205,179	150 (701-850)	26917
ClueWeb09B	ClueWeb09 Disk1 - English	50,220,423	200 (1-200)	11037
ClueWeb09B-S70	ClueWeb09 Disk1 - English exclude docs. with spam score < 70	29,038,220	200 (1-200)	6847

Table 2.1: Overview of datasets used in experiments reported in this thesis.

swering [165], expert retrieval [155], product search [154], similar item finding [179], sponsored search [176, 12], content-based recommendation [100] etc. A comprehensive survey of applications of word embeddings in information retrieval can be found in [110].

2.5 Evaluation methodology

2.5.1 Datasets

The proposed retrieval methods are designed for efficient document retrievals. Hence, to empirically validate the effectiveness of the proposed methods, experimented are done on datasets provided by TREC[2]. Specifically, the ad-hoc data collections comprising of news documents and web documents are considered. For the benchmark TREC collections, the title fields are used for retrieval.

As part of the news collections, the early TREC ad-hoc datasets are used. Specifically, the Tipster disks 1, 2 (used in TREC 1, 2 and 3 ad-hoc track), and Tipster disks 4, 5 excluding Congressional Record (used in TREC 6, 7, 8 ad-hoc track, and TREC 2003, 2004 robust track). Other than the news corpora, web collections namely WT10G, GOV2 and ClueWeb09-B are used for experimentations. WT10G is a well crafted Web collection containing about 1.5 million Web documents [14]. It was used for the TREC 9 and 10 Web Track tasks [43, 42]. GOV2 is a collection of 25 million webpages from the `.gov` domain. It was used in the TREC Terabyte Track [36, 37, 24]. ClueWeb09 - category B, containing about 50 million Web documents in English, was used for the Web Track at TREC from 2009 to 2012 [32, 35, 33, 34]. As expected in a sizeable sample of pages crawled from the Web, this collection contains a non-trivial proportion of spam documents. A spam filter for this collection was presented in [41][3]. An

[2] https://trec.nist.gov
[3] https://plg.uwaterloo.ca/~gvcormac/clueweb09spam/

overview of these datasets is provided in Table 2.1. The official *trec_eval* tool[4] has been used for evaluating the performance.

2.5.2 Evaluation metrics

In this section, we discuss about the metrics that are used to evaluate the performance of the baseline, as well as the proposed methods.

Let us consider a set of queries $\mathbb{Q}$ corresponding to a set of information needs. For every query $Q \in \mathbb{Q}$, let Rel_Q be the total number of relevant documents of Q present in the collection. Now, a retrieval function $\mathcal{F}$ retrieves a set of n documents as a ranked list $L = \{d_1, d_2, \ldots, d_n\}$ ordered by their retrieval scores, i.e. $score(d_i, Q) \geq score(d_j, Q)$ for all $i < j$.

Mean Average Precision

Mean Average Precision (MAP) yields a single-figure measure considering retrieval quality across recall levels. Among the other metrics, MAP has been shown to posses satisfactory discriminative property with notable stability[99]. Given a single information need, *average precision* indicates the average of the precision values after each relevant document is retrieved in the top K retrieved documents. For a set of information needs, the mean precision values are again averaged to produce mean average precision (MAP) for the set of information needs.

The rank of each document is then defined to be the position at which they are retrieved, i.e. $Rank(d_i) = i,\ i = 1, \ldots, n$. If $RelRet_Q = \{d_1, d_2, \ldots, d_m\}$ denotes the m relevant documents ($n \geq m$) retrieved among the documents of Ret_Q ($Ret_Q \subseteq RelRet_Q$), then the average precision (AP) for query Q is given by:

$$AP(Q) = \frac{1}{|Rel_Q|} \sum_{d_m \in RelRet_Q} \frac{m}{Rank(d_m)} \tag{2.14}$$

The mean average precision of $\mathbb{Q}$ is then defined as:

$$MAP(\mathbb{Q}) = \frac{1}{|\mathbb{Q}|} \sum_{Q \in \mathbb{Q}} AP(Q) \tag{2.15}$$

In mean average precision, the metric is calculated on the top n retrieved documents, where the value of n is generally set to 1000.

Precision at rank k ($P@k$)

The precision at depth k is calculated by:

$$P@k = \frac{Relevant(L_k)}{k} \tag{2.16}$$

In Equation 2.16, $Relevant(L_k)$ denotes the number of relevant documents in the rank list L upto the rank k. Similar to MAP, for a given topic set $\mathbb{Q}$, $P@k$ is computed by averaging $P@k$ for all the queries. In this thesis, we set $k = 5$ to evaluate the rank at depth 5.

[4] https://trec.nist.gov/trec_eval/

Recall at rank k

The Recall at depth k for query Q is given by:

$$Recall@k = \frac{Relevant(L_k)}{Rel_Q} \tag{2.17}$$

For the entire topic set $\mathbb{Q}$, the recall at rank k is calculated by averaging the values over all the queries. In the results presented in this thesis, we set the value of k to 1000.

Significance Testing

To infer whether one method is statistically significantly better than another, we use paired t-tests at 95% confidence level ($p < 0.05$), on the basis of the per query average precision values.

Chapter 3

Document preprocessing *

3.1 Introduction

Reproducibility of empirical results is a fundamental requirement of disciplines such as Information Retrieval (IR) in which experimentation plays a major role. Experimentation in systems-oriented IR research typically involves two major aspects: an IR system, and one or more test collections. Reproducibility is partly ensured by the widespread use of standardized test collections, experimental protocols and evaluation measures, such as those provided / defined by TREC[2], NTCIR[3], CLEF[4], INEX[5], and FIRE[6]. However, apart from test collections, experimentation also generally involves a retrieval system. Because most retrieval engines are relatively complex, multi-component, highly configurable systems, precisely reproducing a set of experimental results can be challenging in the absence of a detailed description of the retrieval engine used, and its settings. Such a description should preferably cover at least the following components of the IR system.

1. Document tokenization / parsing method used:[7] which parts of a document are tokenized, what characters are regarded as token delimiters, the nature of the tokens themselves (e.g., words, n-grams), which tokens are discarded, if any (e.g., strings consisting of numerals only), etc.

2. Stopword list used, if any.

3. Stemming algorithm used, if any.

4. Additional indexing units (e.g., phrases and named entities) identified, if any, and details of the identification method used.

5. Details of retrieval model used (e.g., language modeling, smoothing method used, values of parameters, etc.).

*Some material from [135] has been reused in this chapter.
[2] http://trec.nist.gov/
[3] research.nii.ac.jp/ntcir/
[4] www.clef-initiative.eu
[5] http://inex.mmci.uni-saarland.de/about.html
[6] fire.irsi.res.in/
[7] For simplicity, we refer to this step as *parsing* henceforth.

6. Information about other techniques (e.g., reranking, query expansion) that are used on top of a basic, keyword-matching approach to ranked retrieval.

In practice, however, many of the above details are often missing from the system descriptions provided in scholarly articles [93, 54]. In most cases, authors mention only the retrieval model and the engine that was used (particularly when it is an open-source engine such as Indri, Lucene or Terrier) along with some additional information, such as the stopword list used, and the stemming method employed. This has become a standard practice, since the proposed method and/or research findings are the primary focus in published papers, and the publication system does not, as yet, provide a convenient method for recording the mundane details required to reproduce experimental results.[8] These missing details lead to potential reproducibility issues, since the various components of an IR system may, in general, have a significant effect on the overall performance of the system [93, 54, 59].

In this study, we focus in particular on the parsing step (Item 1 above). More specifically, we are interested in the choices made regarding which parts of documents to index, and the impact of these choices on retrieval effectiveness.

A sampling of recent publications ([15, 8, 48, 51], for example) in leading IR venues shows that papers rarely include complete information about document indexing. Other papers [171, 170, 49] mention whether stopwords were removed and whether words were stemmed, but omit details regarding which parts of documents were indexed.

For document collections consisting primarily of relatively clean, textual content (e.g., news-wire collections such as WSJ, AP and SJMN included in Tipster Disks 1–3), the choice is reasonably clear: the entire document should probably be indexed. In contrast, documents in web corpora (e.g. WT10G and ClueWeb) generally contain, in addition to actual information, a significant amount of markup and meta-content (HTML tags along with their options, style specifications, Javascript fragments, etc.), that are used by a browser to visually render the document, but which are unrelated to the information contained in the document. These portions of the document — informally referred to as *noise* in the thesis — should probably be discarded during parsing.

Our goal in this study is to confirm the above hypothesis, and thereby to arrive at definitive guidelines with regard to document parsing. We seek to do this by quantifying the impact of noise removal and content selection on system performance, both for baseline retrieval methods, as well as methods involving query expansion. To summarise, the questions that we intend to address are the following.

Question 1. Should documents (particularly documents in Web collections) be cleaned prior to indexing? How much difference does noise-removal make? Does the answer depend on the retrieval model applied?

Question 2. What is the effect of noise removal on query expansion (QE) techniques?

Our experiments show that removing noise during document parsing can indeed significantly affect retrieval effectiveness, *but this effect varies across retrieval models as well as test collec-*

[8] Early TREC proceedings included System Summaries or System Descriptions in an attempt to record some of the above information; however, this practice also seems to have petered out.

tions. In particular, for reasons discussed in Section 3.6, noise removal *adversely* affects the effectiveness of at least one retrieval model for some collections.

Thus, there does not appear to be a single, obvious answer to the question of whether noise removal is desirable; researchers might reasonably make different choices in different situations with regard to noise removal. For example, even though our findings suggest noise removal is, on the whole, to be preferred, [149] provides code that indexes *all* content inside the container tag (`<DOC> ... </DOC>`) for each document.

Given the above variations in praxis, and in the absence of definite information about choices made with regard to noise removal, one might have to resort to guesswork in order to reproduce a given set of experimental results. Secondly, an improper choice of parsing method can lead to poor results / weak baselines, which may in turn lead to incorrect conclusions about the benefits of using various retrieval techniques [9] (like query expansion, for example).

The rest of this chapter is organized as follows. In the next section, we discuss previous work in order to provide a context for this study. In Section 3.3, we describe the method that we used to detect noise in documents. Section 3.5 describes our experimental setup and the test collections used. Section 3.6 compares and discusses the performance obtained for different retrieval models with and without noise removal. Section 3.7 concludes the chapter with directions for future work.

3.2 Related work

In this section, we look at earlier work that deals with parsing of documents in general, and of web pages in particular, from an indexing perspective.

3.2.1 Parsing of documents and Web pages

The test collections used for the ad hoc task during the early years of TREC consisted of documents that were marked up using SGML. Some participating teams (e.g., Cornell University) constructed a list of the various tags present in documents, and prepared a hand-crafted set of rules regarding which tags signify indexable content, and which tags enclose content that should be discarded. Simple safeguards were put in place to handle syntactic errors (e.g., missing closing tags).

To the best of our knowledge, no such list is available for the Web collections (e.g. WT10G and ClueWeb). Therefore, researchers need to decide for themselves which portions of Web documents are to be indexed, and which portions are to be discarded. The community must have faced this issue during the early years of the TREC Web Track [76, 43, 42, 32, 35], but the research focus was primarily on link analysis and differential weighting for different parts of Web pages, rather than on parsing, per se [89, 76, 43, 42].

Of course, parsing of Web pages and extraction of useful content, particularly in the presence of navigational and advertisement blocks, and syntactic errors, is a challenging task that has been extensively studied [121, 25, 69, 159, 152].

```
1  <HTML>
2  <HEAD>
3   <META CONTENT="asWedit 2.0" NAME="Generator">
4   <META CONTENT="lorna" NAME="AUTHOR">
5   <LINK HREF="mailto:lcampbell@cims.co.uk" REL="made">
6   <META CONTENT="Thur 8 Aug 1996" HTTP-EQUIV="created">
7   <META CONTENT="Scottish Enterprise, Scotland, Scottish Business, Locate in Scotland,
        Scottish Trade, Scottish Enterprise News, locate, location, investment, Bargraphs,
        Cameras, CMOS Imaging, Consultancy, Contract Research, Diode, Display,
        Electronics, Fibre Optics, Image Sensing, Imaging, Indicators, Infra-Red,
        Infra-Red Gas Sensors, Laser, LCD, Led, Light sources, Optical, Optics, Opto
        Couplers, Opto Switches, Optoelectronics, Photonics, SLM, Smart Vision,
        Ultraviolet, Visible." NAME="keywords">
8
9   <TITLE>Scotland's key sectors -- optoelectronics</TITLE>
10
11 </HEAD>
12
13              <FRAMESET COLS="120,*">
14              <FRAMESET ROWS="230,*">
15              <FRAME SRC="../upperstrap.html" NORESIZE SCROLLING="auto" NAME="upperstrap">
16              <FRAME SRC="./sectors.lowerstrap.html" NORESIZE SCROLLING="auto"
                    NAME="lowerstrap">
17              </FRAMESET>
18              <FRAME SRC="./optoelectronics.body.html" NORESIZE SCROLLING="auto" NAME="body">
19
20              </FRAMESET>
21 </HTML>
```

Figure 3.1: Document WTX103-B39-174 from the WT10G collection. The visible content is marked by a rectangle. Note that this is a judged non-relevant document.

Web documents are also vulnerable to the problem of *spamdexing*,[9] or *keyword stuffing*: keywords added to a document for search engine optimization (SEO) may, in fact, be unrelated to its actual content. A great deal of work has been done on IR in the presence of adversarial information providers, and on detecting spam pages [30, 70, 31, 108, 41, 151]. Some work has also been done on studying the effect of spam (or "content manipulation") on classical retrieval models [122], and using content-based techniques to address this problem [98, 125]. Detecting and eliminating spam is beyond the scope of our investigation, however; we are simply interested in studying how baselines may be affected if noise in Web pages (non-content-related portions that are quite possibly legitimate) is removed prior to document indexing.

In the next section, we present the details of how we removed noise, following which, empirical results are presented showing that there is a relationship between noise removal and the performance of a retrieval engine.

3.3 Indexing Web documents

While processing documents from Web collections, we have broadly two options: either to index whole documents, or to discard portions that correspond to markup and meta-content prior to indexing. Recall from Section 1 that our hypothesis is that the second option should be the preferred alternative.

[9] https://en.wikipedia.org/wiki/Spamdexing

```
1  <HTML>
2  <HEAD>
3  <TITLE>Highlands and Islands of Scotland − Virtual Tour − Shetland</TITLE>
4  </HEAD>
5  <BODY BGCOLOR="#FFFFFF">
6  <IMG align=right width="120" height="180" VSPACE=5 HSPACE=5 alt="photo" SRC="shet6.jpg">
7  <H2>LAND OF THE VIKINGS</H2>
8  <P>
9  In the ninth century, the whole western world was rocked by the movement of Norsemen away from their own countries, their
       longships leaving the fjords for new lands across the sea. Their adventuring and colonisation in time took them to the Holy
       Land sailing south, Greenland and America heading west.
10 <P>
11 Much of northern England and Scotland − Caithness, the Western Isles, Orkney and Shetland − succumbed to these forays, and the
       Picts gave way to this powerful force that came and stayed. Not just warriors, but farmers and their families, and a new culture.
12 <P>
13 From 872 AD, a powerful Viking earldom had been established in neighbouring Orkney, and although actual Scandinavian rule in
       Shetland was to last until the mid fifteenth century, in reality that influence is still incredibly prevalent. Wherever the Vikings
       went they took their law and their language, and of Shetland's 50,000 place names most are Norn. Their local parliament was
       held at Lawting Holm, an islet in Tingwall Loch.
14 <P>
15 One spectacular and enduring reminder of Shetland's heritage is to be found at Jarlshof− regarded as one of the most interesting
       and complex archaeological sites in all Britain. A settlement buried in time, until a storm exposed the masonry of an entire
       village. Wheelhouses and brochs, hearths and troughs reflecting the way of life of a bygone era.
16 <P>
17 Equally enduring are the folklore and the sagas associated with many other legendary sites − the Broch of Mousa, the Loch of
       Girlsta, Haroldswick on Unst, the 'Bears Bait' on Fetlar, and the beach at Gulberwick, for instance, where two of Earl
       Rognvald's longships were wrecked en route for the Crusades in 1148. A million other secrets must still be locked away in
       mounds and under fields, beneath sandy beaches and on the seabed.
18 <P>
19 <IMG align=left width="280" height="170" VSPACE=5 HSPACE=5 alt="photo of longship burning"
       SRC="shet7.jpg">
20 <I>(photo: Burning the Up Helly Aa longship)</I>
21 <P>
22 As a reminder. if one were ever needed, of Shetland's Scandinavian past, every year the midwinter festival, Up Helly.Aa, features a
       procession of a thousand torch−carrying revellers' s, a squad of Vikings in horned helmets and full regalia, and a longship,
       dragged through the streets of Lerwick, before its ceremonial burning. There's more than a hint of myth and history in this
       extraordinary celebration.
23 <P>
24 <TABLE WIDTH=100% BORDER CELLPADDING=5 CELLSPACING=1>
25 <TD WIDTH=25%>
26 <TD WIDTH=25%>
27 <TD WIDTH=25% ALIGN=CENTER><A HREF="index.html">Shetland</A>
28 <TD WIDTH=25% ALIGN=CENTER><A HREF="../../index.html"><I>Gael</I>−net Home Page</A>
29 </TABLE>
30 <P>
31 <TABLE WIDTH=100%>
32 <TD>Comments and e−mail to <A HREF="mailto:info@gael-net.co.uk">info@gael−net.co.uk</a>
33 <TD ALIGN=RIGHT>Page updated 05/03/96<!-- by KM -->
34 </TABLE>
35 </BODY>
36 </HTML>
```

Figure 3.2: Document WTX075-B17-106 from the WT10G collection. This is a judged relevant document for the query *503 - Vikings in Scotland?*

As an example supporting this hypothesis, consider Figure 3.1 which shows document `WTX103-B39-174` from the WT10G web collection [76]. This is a judged non-relevant document[10] for topic 503 (*Vikings in Scotland?*) from the TREC 10 Web Track [77]. The actual (visible) content of the corresponding Web page is marked by a rectangle and constitutes only a small fraction of the overall document. The "keywords" section does contain terms related to the visible content (e.g., *Opto Couplers, Opto Switches, Photonics*), but because this section is often a target for keyword stuffing / spamdexing [151], it is questionable whether it should

[10]This means that the document was retrieved at a relatively high rank by at least one system participating in the TREC 10 Web Track, justifying its inclusion in the pool.

be indexed in the same way as the visible content of the document. The remaining words in the document (mostly HTML tags) are clearly unrelated to the document content, and are therefore unsuitable as indexing terms. Thus, it makes sense for the document preprocessor to remove this noise prior to indexing the document.

Note that, in some years, assessors viewed pages as they would be rendered in a browser (i.e., including frames, images, and other embedded content), rather than the plain source HTML view. This is an obvious challenge to indexing, but can explain the noted divergence between the textual content and the assessor's judgement. However, this is a relatively minor issue over the whole collection as this affects a minority of pages.

To quantify the impact of noise cleaning on system performance, we created two different indexes for each collection. For the *full* index, the original documents (including all markup and meta-content) were indexed; for the *clean* index, documents were run through a preprocessor that removes noise prior to indexing. In the rest of this section, we present the details of how these two document indexes were created.

3.3.1 Full index

For the *full* index (also called the *noisy* index in this thesis), whole documents were indexed as-is using Lucene 5.3.1. Other than a pre-defined set of stopwords, no words were discarded. We followed the basic guidelines provided in [149][11], with only one minor change. Unlike [149], we removed the additional document markup fields added by the collection creators (e.g., `<DOC>`, `<DOCHDR>`, `<DOCNUM>` for the WT10G and GOV2 collections, and the WARC information for the ClueWeb09 collection). These tags were not part of the original web pages. [12] Porter's stemmer is used to stem the words before indexing.

3.3.2 Clean index

To generate the *clean* index, we ran documents through a preprocessor that removed the following specific types of "noise" prior to indexing.

1. **HTML tags along with their attributes:** HTML tags are generally not included in stopword lists. Thus, if they are not removed by the preprocessor, the tags in a document will be indexed as a part of the document's content. Since HTML tags are unlikely to occur in a user query, they might not participate in a query-document match, but they do indirectly influence document ranking by increasing a document's length.

   Further, if a method such as the Relevance Model [91]) is used for query expansion, these tags may constitute a significant part of the expanded query (as we will see in Section 3.5) due to the high probability of their co-occurring with query terms in the pseudo-relevant documents.

[11] This has also been a suggested choice in a number of workshops [13, 160].

[12] Our preliminary experiments (results not reported in this thesis) show that such a seemingly unimportant decision (about whether the preamble is indexed or eliminated during preprocessing) can also have a significant impact on performance, particularly for LM-JM based initial retrieval, as well as for LM-JM based RM3. For the full index used in our experiments, we chose to remove the preamble tags, as they are not part of the original document.

2. **URLs and email addresses:** Even though URLs and email addresses typically occur as attributes of tags (e.g., they occur in the HREF attribute of <A> tags, or the SRC attribute of <FRAME> tags), and are therefore included in the first type of noise mentioned above, we consider them separately because the URLs in a document may yield terms that match query terms. However, these terms may or may not be related to the actual content of the document. For example, a URL in the document in Figure 3.1 contains the term *optoelectronics*, which is related to the visible content of the page. In the document shown in Figure 3.2, however, an unrelated term *index* occurs in URLs. Thus, matches between query terms and terms extracted from URLs may have either a positive or a negative impact on retrieval effectiveness. In any case, the text of the URLs is usually not explicitly displayed by Web browsers, so the URL strings do not necessarily convey any relevant information to the end user.

In order to identify and remove the above kinds of noise from documents, we used *jsoup*[13], an HTML parser for Java. According to the jsoup home page, it is "designed to deal with all varieties of HTML found in the wild; from pristine and validating, to invalid tag-soup; jsoup will create a sensible parse tree." Since several types of syntactic errors — non-standard tags, unbalanced tags, incorrect encoding of HTML files — occur frequently in Web pages (Section 3.2.1), this feature is important. Further, jsoup provides a number of options to extract parts of HTML documents like URLs, email addresses, visible text, etc. As discussed above, all contents of an HTML page other than the visible text may be considered as noise. We therefore used jsoup to explicitly remove all such noise, and to extract only the visible textual content from web pages in order to construct the clean index.

Table 3.1 presents some statistics related to the full and clean indexes for standard Web collections. The 'Markup-to-Length Ratio' column is of particular interest. This column shows how much of the original collection is eliminated via noise removal. To generate the figures for this column, we looked at the total number of tokens (after stopword removal and stemming) in the clean and the full index across all the documents in a given collection. Table 3.1 clearly shows that tags, their attributes, scripts and similar noise account for a large proportion of all these Web collections. For ClueWeb09B, a massive 84.28% of all the terms in the collection are eliminated via the cleaning process. As expected, the number of distinct terms is also substantially smaller in the clean index than in the full index for all collections. While the impact of this cleaning on retrieval effectiveness will be studied in detail in Section 3.6, these numbers show that noise removal is a good idea simply from the perspective of computational resources required to process a collection.

3.4 Approaches investigated

To experimentally validate our claim, that there is a substantial possibility of performance variation due to inclusion and exclusion of metadata in index, we experimented with various retrieval models. Specifically, we selected language model based retrieval methods (with

[13] https://jsoup.org/

TREC collection	No. of docs	Markup-to-Length Ratio	No. of Distinct Terms	
			Full	Clean
WT10G	1,692,096	43.67%	10,055,958	7,282,492
GOV2	25,205,179	68.09%	85,590,568	59,513,645
ClueWeb09B-Spam 70	29,038,220	81.33%	151,574,283	43,487,205
ClueWeb09B	50,220,423	84.28%	418,246,153	118,442,851

Table 3.1: Statistics relating to the full and clean indexes for various Web collections.

two different smoothing technique, namely Jelinek-Mercer smoothing, and Dirichlet smoothing [174, 175, 173]), probabilistic methods BM25 [127, 128], and information theoretic methods named Divergence from Randomness [6] retrieval models. Finally, we also studied the impact of noise removal on query expansion based on the relevance based language model [91, 86]. All the methods are previously discussed in Chapter 2.2 and 2.3.

For performing the initial retrieval, and retrieval with the expanded query, any retrieval model can be used. Following the trend in literature ([91, 86, 97]), as RLM is a language model based QE method, we perform both, baseline and expanded retrievals using the language model with the two different smoothing methods, JM and Dir.

In the following section, we describe the basic experimental setup that we followed for these experiments. Also, the dataset for the evaluation as well as, the evaluation metrics used are outlined. Later, in Section 3.6, the empirical evaluation of the different methods used for indexing is presented.

3.5 Experimental setup

We used Lucene 5.3.1 for our experiments. The document preprocessor described in Section 3.3 was incorporated into the Lucene indexing pipeline. We also implemented RM3 [86], the relevance model based query expansion method, within Lucene. For both the clean and the full index, common terms were eliminated using the SMART stopword list [140]; the remaining words were stemmed using Porter's stemmer. For retrieval, Lucene's native implementations of the baseline methods (Section 3.4) were used. Only the title field of queries was considered. Retrieval effectiveness was measured and compared using mean average precision (MAP) and precision upto rank 5 (P@5), but other evaluation metrics (e.g., *NDCG* and *recall*) exhibit similar variations. All the significance tests are performed using paired t-test at 95% confidence level ($p < 0.05$).

3.5.1 Dataset overview

Since noise is mostly a concern for Web corpora, we used TREC Web datasets — specifically WT10G, GOV2 and ClueWeb09B — for our experiments. WT10G is a well crafted Web collection containing about 1.5 million Web documents [14]. It was used for the TREC 9 and 10 Web Track tasks [43, 42]. GOV2 is a collection of 25 million webpages from the `.gov` domain. It was used in the TREC Terabyte Track [36, 37, 24]. ClueWeb09 - category B, containing about 50 million Web documents in English, was used for the Web Track at TREC from 2009

TREC collection	Topic Set (Title only)	RunID	Number of topics	Average topic length (in words)
WT10G	TREC 9-10 Web Track Topics 451-550	WT10G	100	4.18
GOV2	TREC 2004-2006 Terabyte Track Topics 701-850	GOV2	150	3.08
ClueWeb09B - Spam70	TREC 2009-2012 Web Track Topics 1-200	CW09B-S70	200	2.48
ClueWeb09B	TREC 2009-2012 Web Track Topics 1-200	CW09B	200	2.48

Table 3.2: Overview of web collections used in our experiments.

to 2012 [32, 35, 33, 34]. As expected in a sizeable sample of pages crawled from the Web, this collection contains a non-trivial proportion of spam documents. A spam filter for this collection was presented in [41][14]. For our experiments with baseline retrieval methods, we used both the original collection, and a spam-filtered version. The filtered version was created by eliminating all documents with a spam score of less than 70% (as suggested in https://plg.uwaterloo.ca/~gvcormac/clueweb09spam/ and used in [163, 178, 111, 52]). However, we noticed that a number of judged relevant documents were eliminated by this process. Further, the results on both the filtered collection and the full collection follow a similar pattern by and large. Therefore, for the experiments on Relevance Feedback (presented in Sections 3.6.2), we report only the results for the entire ClueWeb09 - Category B collection without any spam-filtering. An overview of these datasets is provided in Table 3.2.

3.5.2 Parameter settings

The retrieval models used in this study have a number of associated parameters. As discussed in Sections 2.2.1 and 2.2.2, the language modeling based methods (LM-JM and LM-Dir) have one parameter each (λ and μ respectively), while BM25 has two parameters (k_1 and b). The DFR model is made up of three components, namely, *basic-model*, *information-gain normalization*, and *length normalization* ([6, 4] contains a detailed discussion about these components). All parameters of these baseline models were tuned separately for each topic set listed in Table 3.2 using 5-fold cross validation. Further, all the parameters were separately tuned for the clean and full indexes. The linear smoothing parameter λ of LM-JM was varied from 0.1 to 0.9 in steps of 0.1; the Dirichlet parameter μ was varied over the range $\{100, 200, 500, 1000, 2000, 2500, 3000, 5000\}$. The BM25 parameters ($k_1$ and b) were varied over $[0.1, 2.0]$ and $[0.0, 1.0]$ respectively, in steps of 0.1. For DFR, all possible combinations of the three different components were explored. The range of parameters tried for each baseline retrieval model is presented in Table 3.3.

For experiments involving query expansion (Section 2.3), the RM3 parameters were also selected using 5-fold cross validation. These parameters were tuned separately for the two document smoothing methods (LM-JM and LM-Dir), since these two methods are known to

[14] https://plg.uwaterloo.ca/~gvcormac/clueweb09spam/

Retrieval Model	Range of Parameter Variation
LM-JM	$\lambda = \{0.1 - 0.9\}$
LM-Dir	$\mu = \{100, 200, 500, 1000, 1500, 2000, 5000\}$
BM25	$k_1 = \{0.5 - 1.5\}$ $b = \{0.1 - 0.9\}$
DFR	BasicModel = *{BE, D, G, IF, In. Ine, P}* Info. gain normalization = *{L, B, NoNorm}* Length normalization = *{Uni., InvLen., Dir., Zip., NoNorm}*

Table 3.3: Different parameter/component settings of various retrieval methods tried during cross validation.

Model	Smoothing Parameter	RM3 Parameters		
	Range	M - # fdbk docs.	N - # expans. terms	ϕ - query wt.
LM-JM	$\lambda = \{0.7 - 0.9\}$	10	50, 60, 70, 80, 90	$\{0.3 - 0.7\}$
LM-Dir	$\mu = \{1000, 1500, 2000\}$	10	30, 40, 50, 60, 70	$\{0.3 - 0.7\}$

Table 3.4: Parameter settings tried during cross-validation for LM-JM and LM-Dir based RM3 feedback model.

behave differently for queries of different lengths [174]. We varied the number of expansion terms in the range from 50 to 90, and from 30 to 70 separately for LM-JM and LM-Dir respectively. We fixed the number of feedback documents at 10. The query mixing parameter (α of Equation 2.11) was varied in the range $\{0.3 - 0.7\}$ in steps of 0.1 for all the QE experiments.

The baseline retrieval results show that the performance of LM-JM drastically deteriorates for λ values less than 0.7. Similarly, increasing μ beyond 2000 resulted in gradual degradation of MAP for LM-Dir. Therefore, to keep the overall number of parameter combinations to be tried for RM3 to a tractable number, we limited the range for λ and μ to $\{0.7, 0.8, 0.9\}$ and $\{1000, 1500, 2000\}$ respectively. For a particular retrieval model (LM-JM or LM-Dir), the smoothing parameter selected for baseline retrieval was also used when performing retrieval with the expanded query. Table 3.4 shows the range of parameter values that were tried during cross validation for RM3.

3.6 Results and discussion

We first present the results obtained using the full and clean indexes for various baseline retrieval approaches. Next, in Section 3.6.2, we look at the effect of noise cleaning on RM3-based query expansion.

3.6.1 Baseline retrieval

Table 3.5 presents the performance of four different retrieval strategies, in terms of mean average precision (MAP) when the full and clean indexes are used. Our observations are summarized below.

Collection	Content Used	LM-JM	LM-Dir	BM25	DFR
WT10G	Full	0.1730	0.2115	0.2085	0.1888
	Clean	0.1500	0.2179	0.2098	0.2036
	% change	**-13.29**	+3.03	**+0.62**	**+8.30**
GOV2	Full	0.2298	0.2705	0.2797	0.2755
	Clean	0.2181	0.2953	0.2996	0.2897
	% change	-5.09	**+9.17**	**+7.11**	**+5.15**
CW09B-S70	Full	0.0907	0.1184	0.1269	0.1377
	Clean	0.0904	0.1558	0.1574	0.1646
	% change	-0.33	**+31.59**	**+24.03**	**+19.54**
CW09B	Full	0.1055	0.1266	0.1476	0.1560
	Clean	0.1122	0.1738	0.1941	0.1962
	% change	**+6.35**	**+37.28**	**+31.50**	**+25.77**

Table 3.5: Comparison of Mean Average Precision (MAP) for different retrieval models and indexing schemes on different topic sets. % changes in bold denote changes that are statistically significant (based on a paired t-test at the 5% level).

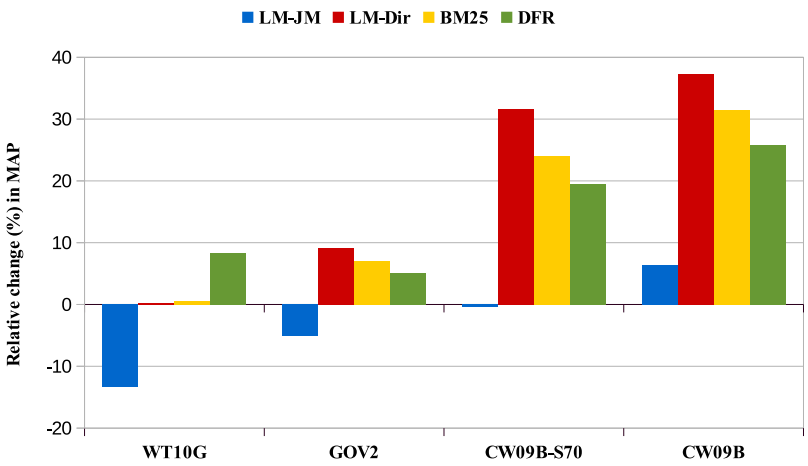


Figure 3.3: Performance variation for different retrieval models.

- On all collections except ClueWeb09B, noise removal has a (sometimes strong) detrimental effect on LM-JM. This was, prima facie, an unexpected result, but one that is explained below. Noise removal has no effect if spam is filtered out from the ClueWeb09B collection. Cleaning improves the performance of LM-JM only for the full ClueWeb09B collection, but the margin of improvement is much lower than what is observed for the other retrieval models.

- For LM-Dir and BM25, noise removal generally does not seem to have much impact when the test collection is relatively small and noise-free. Specifically, for the WT10G topic set, cleaning has minimal effect on the performance of LM-Dir and BM25. As the collections grow in size and become progressively noisier, cleaning has a more pronounced positive effect. For ClueWeb09B, cleaning improves MAP by as much as 30% or more.

- The effect of cleaning on DFR is significant for all the topic sets. As expected, the maxi-

Collection	Content Used	LM-JM	LM-Dir	BM25	DFR
WT10G	Full	0.3125	0.3502	0.3605	0.3551
	Clean	0.2467	0.3527	0.3622	0.3789
	% change	**-0.211**	+0.007	+0.005	+0.067
GOV2	Full	0.4634	0.5641	0.6069	0.6014
	Clean	0.3559	0.5766	0.5834	0.5945
	% change	**-0.232**	+0.022	-0.039	-0.011
CW09B-S70	Full	0.2021	0.2708	0.2882	0.3251
	Clean	0.2072	0.3374	0.3415	0.3815
	% change	**+0.025**	**+0.246**	**+0.185**	**+0.173**
CW09B	Full	0.1682	0.1846	0.2256	0.2246
	Clean	0.2000	0.2626	0.2667	0.2738
	% change	**+0.189**	**+0.423**	**+0.182**	**+0.219**

Table 3.6: Comparison Precision upto rank 5 (P@5) for different retrieval models and indexing schemes on different topic sets.

mum difference is observed for the complete, unfiltered ClueWeb09B collection.

- Combining the information in Table 3.5 with the third column of Table 3.1, we note that all four retrieval models exhibit a common trend: cleaning plays an increasingly useful role as collections become noisier. This trend is shown graphically in Figure 3.3.

- Spam filtering removes the most noisy documents, thereby taking care of some of the effects of noise cleaning. Cleaning thus produces somewhat smaller improvements for LM-Dir, BM25 and DFR on CW09B-S70 compared to the unfiltered ClueWeb09B collection. Nevertheless, the improvements are substantial in absolute terms (around 20% or more) and statistically significant.

- In case of language model based retrieval methods, Dirichlet smoothing consistently outperforms Jelinek-Mercer smoothing. This observation is consistent with the findings of [174, 175].

- When performance across all collections is taken into account, BM25 seems to be the most consistently effective retrieval model.

To evaluate the performance variation in terms early precision, we have reported P@5 in table 3.6. From the table, a similar observations are noticed. That is, noise removal is seen to adversely affect the performance of LM-JM, except for ClueWeb collections. For LM-Dir, the performance is always seen to be improving with the removal of noises for all collections. In fact, the relative improvements are also noticed to be proportional to the difference in size. For both the ClueWeb09B variants that we used for evaluation, are seen to be fortuitous to cleaning of the collection. Except for GOV2, the performance of BM25 and DFR are also seen to surpass when the so-caled noises are removed before retrieval.

We now turn to the apparently counter-intuitive behaviour of LM-JM. For the WT10G collection (with topic sets TREC 9,10), using the noisy, full index consistently produces *significantly* better results if JM smoothing is used (for the GOV2 collection, the noisy index also

yields better results, but the improvements are not statistically significant). This is best explained by considering the difference between JM and Dirichlet. Dirichlet smoothing (Equation 2.4) is document-length dependent: maximum likelihood estimates (MLEs) are less reliable for short documents, and are more heavily smoothed; conversely, MLEs from longer documents are more reliable, and are smoothed to a smaller extent. In contrast, JM smoothing is document length independent.

Consider a single-term query $Q = \{q\}$ and two documents d_1 and d_2. Suppose that d_1 is very short (too short to be relevant), but contains a chance occurrence of the query term q; also suppose that d_2 is a longer, useful document that contains more occurrences of q, but also many more additional terms. The Maximum Likelihood Estimate (MLE) of $p(q|d)$ would typically be much higher for d_1 than for d_2. When JM smoothing is used (Equation 2.3), d_1 would get a much higher score compared to d_2. Thus, JM has a bias in retrieval towards short documents. It is well-known (from studies by [147, 146], for example) that retrieval functions (such as JM) that are biased in favour of short documents perform relatively poorly. This explanation is also consistent with earlier analyses explaining the advantage of Dirichlet over JM [148, 95].

This explanation can be extended to account for why JM performs better over a full index. Consider TREC topic 503 (*Vikings in Scotland?*), and two documents d = WTX103-B39-174 (shown in Figure 3.1) and d' = WTX075-B17-106 (shown in Figure 3.2). d is short, uninformative and non-relevant, while d' contains less metadata, and is relevant. When markup / meta-content is removed, short documents like d become very short (with possibly as few as 3-5 words). If such an extremely short document contains even a single occurrence of one query term q, $P_{MLE}(q|d)$, the maximum likelihood estimate of the probability of q being generated by the document model d, becomes unnaturally high. For our example, $P_{MLE}(scotland|d) = 0.25$. In contrast, $P_{MLE}(scotland|d')$ has a far more modest value of 0.026, even though d' is actually useful, and has as many as 34 occurrences of *scotland*. From Equation 2.3, it is clear that d would be ranked far above d' if JM smoothing were used.

When full documents are used, all documents appear longer because each document contains its fair share of noise. Thus, the values of $P_{MLE}(q|d)$ are more moderate, and documents with little useful content no longer have an unreasonable advantage. Therefore, retrieval effectiveness improves.

Given this anomalous behaviour of LM-JM, and given that researchers continue to use LM-JM, the recommendations of [174, 175] notwithstanding (see [58, 64, 124, 65, 114, 11] for example), we believe that it is important from the reproducibility perspective to report the parsing method used for a particular study.

3.6.2 Pseudo feedback based query expansion

In this section, we examine the differences in improvement obtained when either the full or the clean index is used during pseudo relevance feedback (PRF) based QE. Specifically, we consider RM3 based QE (discussed in Section 2.3) for our study. Since we experimented with 2 separate indexes for the baseline retrieval phase, we actually get a total of four possible combinations. We refer to these combinations as full-full, full-clean, clean-full, and clean-clean.

The first part of a combination's name signifies which index was used for initial retrieval, while the second specifies which index was used during QE.

It is already a known fact that the relevance model is prone to choosing common terms [46, 78, 38]. Dropping stopwords from the collection during indexing partially addresses the problem. However, terms which are common in the top-ranked documents, but not present in the stopword list (e.g., HTML tags in web documents) are quite often selected as expansion terms by the relevance model due to their high association with query terms in these documents. For this reason, noise cleaning is likely to be important for QE: it is expected to eliminate at least some inappropriate terms from being candidate expansion terms.[15]

The results in terms of MAP and P@5 for RM3 (Table 3.7 and 3.8) are in line with our expectations. For a given baseline index (either clean or full), using the clean index for feedback works better than using the full index in terms of both MAP and P@5. The improvement is sometimes minuscule (e.g., full-clean vs. full-full for LM-Dir on GOV2), but often significant. As discussed above, this is easily explained: tags and other noisy terms are included in the query if the full index is used during QE. In contrast, the terms selected for QE from the clean index are generally much more related to the actual intent of the query. The improvement in P@5 indicates the change in early precision when the clean indexes are used for feedback. Table 3.9 illustrates this phenomenon using one example query from each of the four test collections used in our experiments. For each query, the top 15 expansion terms (having the highest $P(w|r)$ weights according to Equation 2.11) are shown. It is clear from the table that many / most of the terms selected from the full index are tags.

For WT10G topic set, the best result for LM-JM is obtained when the full index is used for the initial retrieval (as this provides a much better starting point than the clean index), and the clean index is used for feedback. Further, these results are statistically significantly better than the RM3 results for the other combinations (full-full, clean-full and clean-clean). In case of GOV2, although the initial retrieval was inferior when the clean index was used (a 5% difference in MAP that is not statistically significant), the clean-clean combination produced the best performance among all combinations. For ClueWeb09B, initial retrieval from the clean index produces significantly better results; not surprisingly, the use of the clean index for both the initial retrieval as well as feedback yields the best results.

When the baseline retrieval is performed using LM-Dir, the clean index produces better results for all topic sets (See Figure 3.3). Once again, it is not surprising that post-QE performance is generally best for the clean-clean combination. The only exception to this pattern is that, the full-clean combination achieved the best performance for WT10G, but it is only marginally better than the clean-clean combination.

As a general observation from Table 3.7, we conclude that the removal of noise during indexing can produce significantly different results for QE-based retrieval methods, irrespective of the index chosen for initial retrieval.

[15] One might argue that these 'useless' expansion terms do not hurt effectiveness because of the IDF factor that comes into play during the final retrieval using the expanded query. However, if the number of expansion terms is kept fixed, then these terms would elbow out more useful terms from the expanded query. In order to reap the potential benefits of QE in the presence of such terms, one would need to increase the total number of expansion terms. This would adversely affect efficiency, if not effectiveness.

Collection	Baseline index	Feedback index	JM			Dir		
			Baseline	RM3	% change	Baseline	RM3	% change
WT10G	Full	Full	0.1730	0.1671	-3.41	0.2115	0.2208	+4.40
		Clean		**0.2017**^{1,3,4}	+16.59		**0.2367**^{1}	+11.91
	Clean	Full	0.1500	0.1575	+5.00	0.2179	0.2334	+7.11
		Clean		0.1796	+19.73		0.2364	+8.49
GOV2	Full	Full	0.2298	0.2459	+7.00	0.2705	0.2920	+7.95
		Clean		0.2563	+11.53		0.2943	+8.80
	Clean	Full	0.2181	0.2266	+3.90	0.2953	0.3156	+6.87
		Clean		**0.2638**^{1,2,3}	+20.95		**0.3269**^{1,2,3}	+10.70
CW09B	Full	Full	0.1055	0.1236	+17.16	0.1266	0.1277	+0.87
		Clean		0.1360	+28.91		0.1422	+12.32
	Clean	Full	0.1122	0.1249	+11.32	0.1738	0.1759	+1.20
		Clean		**0.1540**^{1,2,3}	+37.25		**0.1882**^{1,2,3}	+8.29

Table 3.7: Comparison of Mean Average Precision (MAP) for baseline and RM3 for different smoothing methods and indexing schemes on different topic sets. The RM3 results in bold denote the maximum improvement over baseline as compared to the other configurations for the same topic set and retrieval model. Note that for all collections, the maximum improvements are achieved when the clean index is used for pseudo relevance feedback. The superscripts ^{1,2,3,4} denote a statistically significant difference between the corresponding value and the MAP for the full-full, full-clean, clean-full and clean-clean combinations respectively.

Collection	Baseline index	Feedback index	JM			Dir		
			Baseline	RM3	% change	Baseline	RM3	% change
WT10G	Full	Full	0.3125	0.3091	-0.011	0.3502	0.3489	-0.0037
		Clean		**0.3427**^{1,3,4}	0.097		**0.3851**^{1}	0.0997
	Clean	Full	0.2467	0.2702	0.095	0.3527	0.3698	0.0485
		Clean		0.266	0.078		0.3729	0.0573
GOV2	Full	Full	0.4634	0.4938	0.066	0.5641	0.5903	0.0464
		Clean		**0.5255**^{1,2,3}	0.134		0.5876	0.0417
	Clean	Full	0.3559	0.3628	0.019	0.5766	0.5821	0.0095
		Clean		0.4028	0.132		**0.6110**^{1,2,3}	0.0597
CW09B	Full	Full	0.1682	0.2154	0.281	0.1846	0.1733	-0.0612
		Clean		0.2144	0.275		0.1641	-0.1111
	Clean	Full	0.2000	0.1928	-0.04	0.2626	0.2287	-0.1291
		Clean		**0.2462**^{1,2,3}	0.231		**0.2646**^{1,2,3}	0.0076

Table 3.8: Comparison of Precision at rank 5 (P@5) for baseline and RM3 for different smoothing methods and indexing schemes on different topic sets. The RM3 results in bold denote the maximum improvement over baseline as compared to the other configurations for the same topic set and retrieval model. Note that for all collections, the maximum improvements are achieved when the clean index is used for pseudo relevance feedback. The superscripts ^{1,2,3,4} denote a statistically significant difference between the corresponding value and the P@5 for the full-full, full-clean, clean-full and clean-clean combinations respectively.

499 - “pool cue”			
full index		clean index	
w	$P(w\|R)$	w	$P(w\|R)$
pool	0.3055	pool	0.3215
cue	0.3032	cue	0.3127
td	0.0369	tabl	0.0111
br	0.0297	1996	0.0108
href	0.0295	ball	0.0100
font	0.0165	page	0.0091
tr	0.0126	time	0.0078
center	0.0126	inform	0.0077
http	0.0105	web	0.0071
img	0.0102	home	0.0067
li	0.0101	plai	0.0066
src	0.0100	game	0.0063
align	0.0097	make	0.0062
html	0.0092	line	0.0061
width	0.0092	year	0.0061

529 - “history on cambodia?”			
full index		clean index	
w	$P(w\|R)$	w	$P(w\|R)$
cambodia	0.2151	cambodia	0.2455
histori	0.2113	histori	0.2301
td	0.0480	1	0.0310
href	0.0386	yahoo	0.0187
br	0.0374	countri	0.0174
font	0.0217	inform	0.0172
html	0.0211	lao	0.0168
1	0.0185	societi	0.0150
width	0.0167	cultur	0.0148
titl	0.0167	2	0.0129
tr	0.0164	search	0.0118
h3	0.0155	region	0.0102
center	0.0150	khmer	0.0098
http	0.0148	1996	0.0097
li	0.0138	centuri	0.0096

146 - “sherwood regional library”			
full index		clean index	
w	$P(w\|R)$	w	$P(w\|R)$
librari	0.1391	librari	0.1976
region	0.1333	region	0.1532
sherwood	0.1333	sherwood	0.1500
href	0.0435	link	0.0183
td	0.0406	1	0.0173
class	0.0368	home	0.0156
div	0.0312	alexandria	0.0151
li	0.0300	2008	0.0144
http	0.0278	counti	0.0138
tr	0.0216	inform	0.0133
span	0.0206	number	0.0131
titl	0.0196	locat	0.0129
0	0.0181	2	0.0127
width	0.0164	contact	0.0125
id	0.0143	virginia	0.0124

847 - “portug world war ii”			
full index		clean index	
w	$P(w\|R)$	w	$P(w\|R)$
ii	0.0999	war	0.1065
portug	0.0999	ii	0.0999
war	0.0999	portug	0.0999
world	0.0999	world	0.0999
td	0.0860	1	0.0368
tr	0.0316	state	0.0236
font	0.0314	data	0.0225
href	0.0306	2	0.0218
0	0.0258	inform	0.0161
width	0.0216	3	0.0161
align	0.0198	servic	0.0150
img	0.0177	4	0.0141
1	0.0166	5	0.0128
br	0.0165	2003	0.0126
class	0.0163	nation	0.0121

Table 3.9: Top 15 terms of some expanded queries, with associated weights. LM-JM was used for the initial retrieval, before queries were expanded using RM3. Notice that, most of the expansion terms, when selected from the full index, are unimportant terms from meta-content.

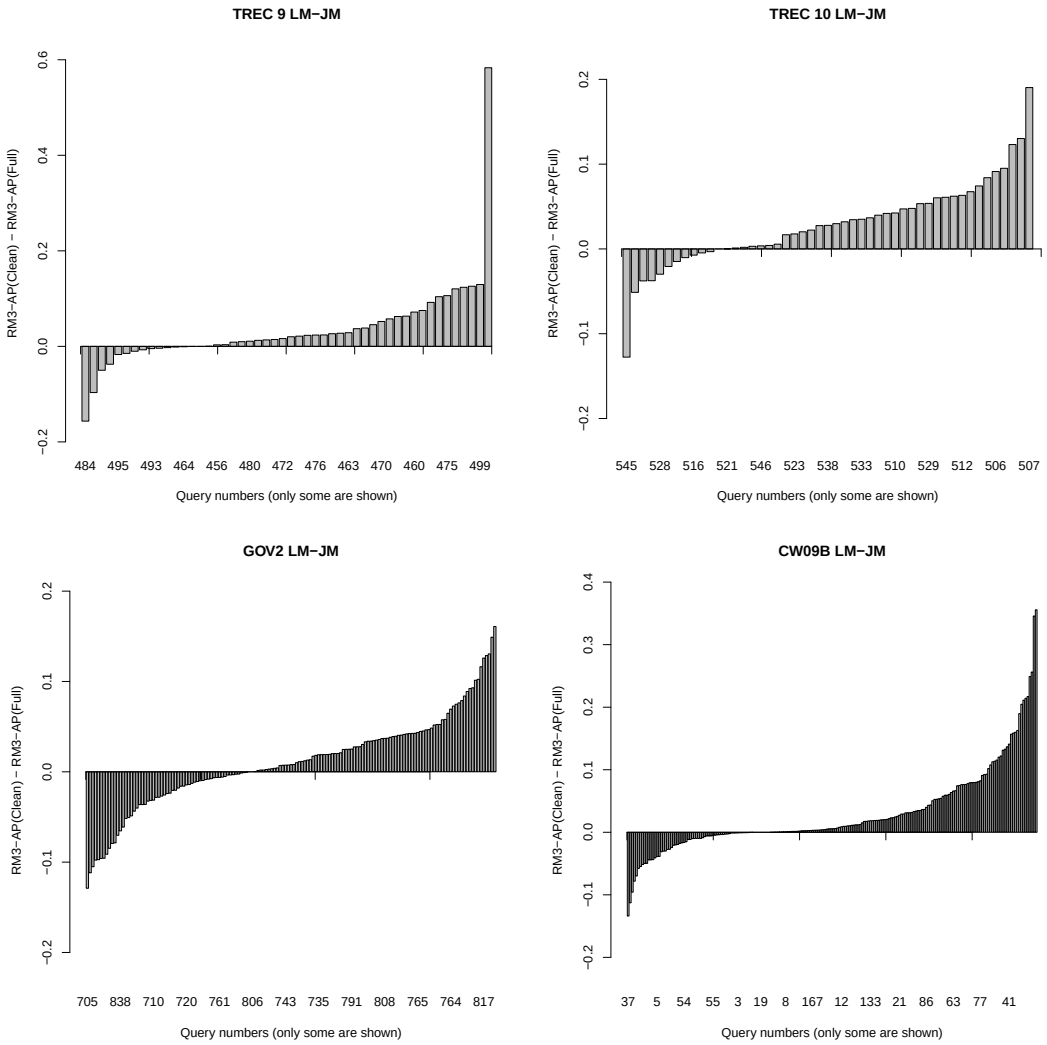


Figure 3.4: Difference in AP for expanded queries from different topic sets. In all cases, the initial retrieval was performed using LM-JM on the full index. Queries were expanded using the full and the clean indexes, and the difference in performance is plotted along the X-axis.

Of course, as with many other IR techniques (such as stemming or stopword removal), noise removal is not beneficial for all queries. A per-query analysis of the performance variation of RM3 across document preprocessing strategies (specifically, full-full and full-clean) is shown graphically in Figure 3.4. In the figure, each vertical bar corresponds to one query, and represents the difference in AP observed when the query is expanded using the clean and the full index. Bars above the X-axis correspond to queries for which expansion from the clean index works better. From Figure 3.4, we note that for some queries, exclusion of metadata leads to poorer performance. We hope to take a closer look at these queries in future work in order to explain why this might be happening.

3.7 Summary

The main questions that our study was intended to address were introduced in Section 3.1. They are:

Question 1. Should documents be cleaned prior to indexing? How much difference does noise-removal make?

Question 2. What is the effect of noise removal on query expansion (QE) techniques?

We specifically focus on the TREC Web collections used for text retrieval research, since these contain a non-trivial amount of noise in the form of meta-tags, hyperlinks etc. How this noise is handled during indexing turns out to be an important issue for the outcome of both baseline retrieval as well as feedback based models. Based on our experimental results, we conclude with the following observations in response to the above questions.

1. For most retrieval models, documents should indeed be cleaned prior to indexing. Cleaning generally results in improvements in MAP. Depending on the retrieval model and the amount of noise present in a corpus, these improvements may be very substantial (we observed an improvement of about 37% in case of LM-Dir on the ClueWeb09B collection).

2. Using a clean index during pseudo relevance feedback based query expansion is also seen to be beneficial overall.

Language modeling with Jelinek-Mercer smoothing exhibits anomalous behaviour with regard to the above observation. Other than on extremely noisy collections (ClueWeb09B), cleaning adversely affects MAP, sometimes significantly so.

This anomaly would normally not be significant, since LM-Dir should generally be preferred over LM-JM [174, 175]. However, since researchers continue to use LM-JM, we have included results obtained using LM-JM in the rest of this thesis in order to faciliate comparison with related work. Following the findings of this chapter, for the results reported in this thesis involving web collections, we use i) full index when LM-JM is used for retrieval (except for ClueWeb09B), ii) clean index when LM-Dir is used for retrieval, and iii) the clean index for all collections while performing feedback.

Chapter 4

Generalized language model *

4.1 Introduction

Vocabulary mismatch, the inherent characteristic of using different but semantically similar terms across documents about the same topic or between a query and its relevant documents (as highlighted in Section 2.1), is a difficult problem to solve in the context of information retrieval. The query expansion techniques solely rely on the co-occurrence information of terms with the query terms to overcome this problem. However, there is no way these techniques be able to utilize the semantic similarity between terms. As a first step towards utilizing the semantic relationship of terms as captured by word embeddings, in this chapter, we will discuss about a novel retrieval model that incorporates the semantically similar terms in the standard LM framework.

The working principle of most standard retrieval models in IR involves an underlying assumption of term independence, e.g. the vector space model (VSM) [139] assumes that the documents are embedded in a mutually orthogonal term space, while probabilistic models, such as the BM25 [128] or the language model (LM) [173] assume that the terms are sampled independently from documents. Standard approaches in IR take into account term association in two ways, one which involves a ***global*** analysis over the whole collection of documents (i.e. independent of the queries), while the other takes into account ***local*** co-occurrence information of terms in the top ranked documents retrieved in response to a query. The local approach corresponds to the relevance feedback step for IR which we do not investigate in this chapter. Later in Chapter 5 and Chapter 6, we will discuss some approaches that utilizes the local information as well.

Existing global analysis methods such as the latent semantic indexing (LSA) [50] or latent Dirichlet allocation (LDA) [20] only take into account the co-occurrences between terms at the level of documents instead of considering the context of a term. Since the word embedding techniques that we introduced in the beginning of this thesis, leverage the information around the local context of each word to derive the embeddings (two words have close vector representations if and only if they are used in similar contexts), we believe that such an approach can potentially improve the global analysis technique of IR leading to better retrieval

*Some material from [62] has been reused in this chapter.

effectiveness.

The research problem, which we investigate in this chapter, is whether the word embedding techniques can improve retrieval effectiveness. Instead of attempting ad-hoc term weighting approaches, we propose a generalized version of the LM, in which we explicitly model pairwise term transformation probabilities across a noisy channel, these probability values being estimated from word embeddings.

The rest of this chapter is organized as follows. Section 4.2 discusses some works that are related to the work presented in this chapter. In Section 4.3, keeping a connection with the standard linear-smoothed language modeling retrieval method, we propose the generalization of the language model. We will refer to this proposed method as **GLM** in the rest of this chapter. The empirical evaluation result of GLM on standard TREC benchmark datasets, is then presented in Section 4.4. Finally, Section 4.5 concludes the chapter.

4.2 Related work

Latent semantic analysis (LSA) [50] is a global analysis technique in which documents are represented in a term space of reduced dimensionality so as to take into account inter-term dependencies. Generalized VSM (GVSM) is another approach to model inter-term dependencies in VSM retrieval model, in which the dot product similarity between two document vectors considers contributions from the angles between two terms (it is assumed that the terms are not orthogonal) [161]. More recent techniques such as the latent Dirichlet allocation (LDA) represent term dependencies by assuming that each term is generated from a set of latent variables called the *topics* [20]. A major problem of these approaches is that they only consider word co-occurrences at the level of documents to model term associations, which may not always be reliable. In contrast, the word embeddings take into account the local (window-based) context around the terms [103], and thus may lead to better modeling of the term dependencies. Berger et al. [18] used statistical translation model to incorporate synonomy into the document language model achieveing effects similar to query expansion. Cao et al. [27] tried to extend existing LM by incorporating term dependencies by using co-occurrence and WordNet relations.

Moreover, most of these global analysis approaches, e.g. LDA, have been applied in IR in an ad-hoc way for re-assigning term weights without explicitly representing the term dependencies as an inherent part of an IR model. For example, an LDA document model (term sampling probabilities marginalized over a set of latent topic variables) is linearly added as a smoothing parameter to the standard LM probability [158], as a result of which the term dependencies are not clearly visible from the model definition. Contrastingly, in this work, we intend to directly model the term dependencies as a part of an IR model. In this respect, our work is somewhat similar to the work in cross language IR, where a translation model was proposed to model the transformations of terms from the source language (query) to the target language (documents in collection) [18].

Recently, a concept language model has been proposed in [182] which combines the probability of the document generating the concept expressed by the query, and the traditional language model probability of the document generating the query terms. In the paper, Zou

et al. use a word embedding space to express concepts. The weighted cosine distance is used to estimate the similarity between two vectors. A modification of paragraph vector model is used in [2] for improving IR performance. In another similar work [169], Zamani and Croft propose query language models based on embedding similarity, where they transformed the cosine similarity scores by the sigmoid function and used it as a final similarity score.

4.3 A generalized language model

In this section, we propose the generalized language model (GLM) that models term dependencies using the embedding of terms.

4.3.1 Term transformation events in language modeling

Following the discussion in Section 2.2.1, in language model based retrieval methods, for a given query Q, documents are scored based on the prior probability $P(Q|d)$.

$$\begin{aligned} \text{Score}(Q,d) &= p(Q|\mathcal{D}) \\ &= \prod_{t \in Q} p(t|d) \\ &= \prod_{t \in Q} \lambda \hat{p}(t|d) + (1-\lambda)\hat{p}(t|C) \\ &= \prod_{q \in Q} \lambda \frac{tf(t,d)}{|d|} + (1-\lambda)\frac{cf(t)}{|C|} \end{aligned} \tag{4.1}$$

In Equation 4.1, the set C represents a universe of documents (commonly known as the *collection*), $\hat{p}(t|d)$ and $\hat{p}(t|C)$ denote the maximum likelihood estimated probabilities of generating a query term t from the document d and the collection respectively, using frequency statistics. The probabilities of these two (mutually exclusive) events are denoted by λ and $1-\lambda$ respectively. The notations $tf(t,d)$, $|d|$, $cf(t)$ and $|C|$ denote the term frequency of term t in document d, the length of d, collection frequency of the term t and the collection size respectively.

As per Equation 4.1, terms in a query are generated by sampling them independently from either the document or the collection. We propose the following generalization to the model. Instead of assuming that terms are mutually independent during the sampling process, we propose a generative process in which a noisy channel may transform (mutate) a term t' into a term t. More concretely, if a term t is observed in the query corresponding to a document d, according to our model it may have occurred in three possible ways, shown as follows.

1. **Direct term sampling**: Standard LM term sampling, i.e. sampling a term t (without transformation) either from the document d or from the collection.

2. **Transformation via Document Sampling**: Sampling a term $t'(t' \neq t)$ from d which is then transformed to t by a noisy channel.

3. **Transformation via Collection Sampling**: Sampling the term t' from the collection which is then transformed to t by the noisy channel.

1. Direct term sampling

This is the standard language modeling term sampling; a term t in the query can be generated from either the document or from the collection, following Section 4.3.1

2. Transformation via document sampling

Let $P(t,t'|d)$ denote the probability of generating a term t' from a document d and then transforming this term to t in the query.

$$P(t,t'|d) = P(t|t',d)P(t'|d) \tag{4.2}$$

In Equation 4.2, $P(t'|d)$ can be estimated by maximum likelihood with the help of the standard term sampling method as shown in Equation 4.1. For the other part, i.e. transforming t' to t, we make use of the cosine similarities between the two embedded vectors corresponding to t and t' respectively. More precisely, this probability of selecting a term t, given the sampled term t', is proportional to the similarity of t with t'. Note that this similarity is independent of the document d. This is shown in Equation 4.3, where $sim(t,t')$ is the cosine similarity between the vector representations of t and t' and $\Sigma(d_t)$ is the sum of the similarity values between all terms with the term t occurring in document d, which being the normalization constant, can be pre-computed for each document d.

$$P(t|t',d) = \frac{sim(t,t')}{\sum_{t'' \in d} sim(t,t'')} = \frac{sim(t,t')}{\Sigma(d_t)} \tag{4.3}$$

Consequently, we can write Equation 4.2 as

$$P(t,t'|d) = \frac{sim(t',t)}{\Sigma(d_t)} \frac{tf(t',d)}{|d|} \tag{4.4}$$

Equation (4.4) favours those terms t's that are not only tend to co-occur with the query term t within d, but are also semantically related to it. Thus, words that are used in similar contexts with respect to the query term t over the collection, as given by the vector embedding, are more likely to contribute to the term score of t. In other words, Equation 4.4 takes into account how well an observed query term t *contextually fits* into a document d. A term contextually fits well within a document if it co-occurs with other semantically similar terms. Terms, score high by Equation 4.4, potentially indicate a more relevant match for the query as compared to other terms with low values for this score.

3. Transformation via collection sampling

Let the complementary event of transforming a term t', sampled from the collection instead of a particular document, to the observed query term t be denoted by $P(t,t'|C)$. This can be estimated as follows.

$$P(t,t'|C) = P(t|t',C).P(t'|C) = P(t|t',C).\frac{cf(t')}{cs} \tag{4.5}$$

Now $P(t|t',C)$ can be estimated in a way similar to computing $P(t|t',d)$, as shown in Equation 4.3. However, instead of considering all (t,t') pairs in the vocabulary for computation, it is

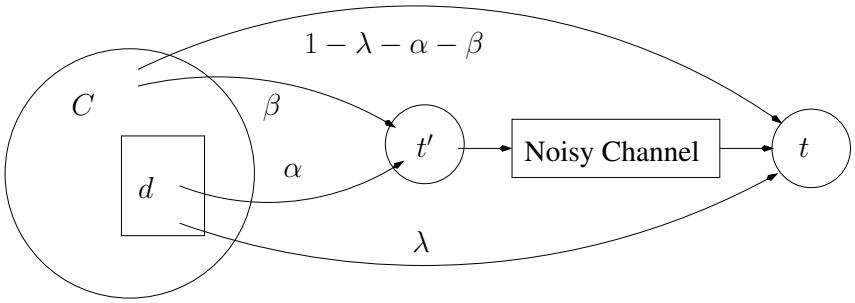


Figure 4.1: Schematics of generation of a query term t in the proposed Generalized Language Model (GLM). GLM degenerates to LM-JM (Equation 2.3) when $\alpha = \beta = 0$.

reasonable to restrict the computation to a small neighbourhood of terms around the query term t, say N_t because taking too large a neighbourhood may lead to noisy term associations. This is shown in Equation 4.6.

$$P(t|t', C) = \frac{sim(t, t')}{\sum_{t'' \in N_t} sim(t, t'')} = \frac{sim(t, t')}{\Sigma(N_t)} \tag{4.6}$$

Therefore, Equation (4.5) can be written as

$$P(t, t'|C) = \frac{sim(t, t')}{\Sigma(N_t)} . \frac{cf(t')}{cs} \tag{4.7}$$

While $P(t, t'|d)$ measures the contextual fitness of a term t in a document d with respect to its neighbouring (in the vector space of embedded terms) terms t' in d, $P(t, t'|C)$, on the other hand, aims to alleviate the vocabulary mismatch between documents and queries in the sense that for each term t in d it expands the document with other related terms t's. From an implementation perspective, $P(t, t'|d)$ reweights existing document terms based on their contextual fit, whereas $P(t, t'|C)$ expands the document with additional terms with appropriate weights.

4.3.2 Combining the events

Finally, to place all the events together in the LM generation model, let us assume that the probability of observing a query term t without the transformation process (as in standard LM) be λ. Let us denote the probability of sampling the query term t via a transformation through a term t' sampled from the document d with α, and let the complementary probability of sampling t' from the collection be then β, as shown schematically in Figure 4.1. The LM term generation probability in this case can thus be written as shown in Equation 4.8. This is a generalized version of the standard LM, which we now henceforth refer to as *generalized language model* (GLM), that takes into account term relatedness with the help of the noisy channel transformation model, which in turn uses the word embedding to derive the likelihood of term transformations. Note that the GLM degenerates to standard LM by setting α and β to zero, i.e. not using the transformation model in the term generation process. In Equation (4.8), $P(t')$, the prior probability of selection of t' is considered as uniform and

hence can be ignored for ranking purposes.

$$\begin{aligned} weight(t,d) &= \lambda P(t|d) + \alpha \sum_{t' \in d} P(t,t'|d)P(t') + \beta \sum_{t' \in N_t} P(t,t'|C)P(t') + (1-\lambda-\alpha-\beta)P(t|C) \\ &= \lambda \frac{tf(t,d)}{|d|} + \alpha \sum_{t' \in d} \frac{sim(t,t')}{\Sigma(d_t)} \frac{tf(t',d)}{|d|} + \beta \sum_{t' \in N_t} \frac{sim(t,t')}{\Sigma(N_t)} \frac{cf(t')}{cs} + (1-\lambda-\alpha-\beta)\frac{cf(t)}{cs} \end{aligned} \tag{4.8}$$

To incorporate the IDF effectiveness into the model, following the exposition presented in [175], Equation (4.8) can be divided by the collection probability component. This collection normalization makes Equation (4.8) into Equation (4.9).

$$\begin{aligned} weight(t,d) = 1 + \frac{\lambda}{1-\lambda-\alpha-\beta} \frac{P(t|d)}{P(t|C)} + \frac{\alpha}{1-\lambda-\alpha-\beta} \frac{\sum_{t' \in d} P(t,t'|d)P(t')}{P(t|C)} + \\ \frac{\beta}{1-\lambda-\alpha-\beta} \frac{\sum_{t' \in N_t} P(t,t'|C)P(t')}{P(t|C)} \end{aligned} \tag{4.9}$$

For a given query $Q = \{q_1, \ldots, q_k\}$, the score of any document d will follow Equation (4.10).

$$Score(Q,d) = \sum_{q \in Q} log\big(weight(q,d)\big) \tag{4.10}$$

4.3.3 Relation between GLM and Query Expansion

While scoring a document d from a collection C, the weight of a query term t following GLM (Equation 4.10) depends on the statistical count of term t in d, C (respectively $P(t|d)$ and $P(t|C)$) as well as terms t' semantically similar to t ($t' \in N_t$). That is, the scoring function also considers the occurrence of semantically similar terms in the documents through document transformation, as well as occurrence in collection through collection transformation which is actually a weak form of document expansion. In case of query expansion techniques, the initial query is expanded by adding N related terms and hence, N additional posting lists (for each term of the expanded query) are needed to be traversed during retrieval, resulting in higher latency. In contrast, the proposed GLM technique expands the document with semantically similar terms while indexing, and hence does not require passing through additional posting lists other than for the query terms. This gain in efficiency happens at the cost of retrieval effectiveness; the loss in effectiveness is quantified by comparing GLM with QE methods in Chapter 5.

4.3.4 Implementation outline

An efficient approach to get the neighbours of a given term is to store a pre-computed list of nearest neighbours in memory for every word in the vocabulary. After this step, for each document d in the collection, we iterate over term pairs (t,t') and assign a new term-weight to the term t representing the document sampling transformation according to Equation 4.4. Then we iterate again over every term t in d and use N_t, pre-computed nearest neighbours of t to compute a score for the collection sampling transformation, as shown in Equation 4.7. To account for the fact that these transformation probabilities are symmetrical, we add the

term t' to d. Note that it is not required to add the term t' in case of the document sampling transformation event because t' is already present in d. Similar to the LM probabilities ($P(t|d)$ and $P(t|C)$), all this additional transformation probabilities, specifically $\sum_{t' \in d} P(t, t'|d)$ and $\sum_{t' \in N_t} P(t, t'|C)$ of Equation (4.8) can stored in the form of a posting lists for efficient retrieval. The workflow of the procedure of making the index for retrieval in presented in Algorithm 1.

While scoring a document d for a (say) single-term query q, three separate posting list lookups, specifically i) LM-posting, ii) document transformation posting, and iii) collection transformation posting are performed. The weights of q in each of the posting lists are then merged and added after multiplying by the appropriate smoothing values (respectively, $\frac{\lambda}{1-\lambda-\alpha-\beta}$, $\frac{\alpha}{1-\lambda-\alpha-\beta}$ and $\frac{\beta}{1-\lambda-\alpha-\beta}$). The score for the document is then finalized by taking a logarithm of the summed weights.

4.4 Evaluation

In this section, we conduct experiments to evaluate the effectiveness of the proposed method. We first describe the experimental settings and then the results of our experiments.

4.4.1 Experimental setup

The experiments were conducted on the standard TREC news and web collections. More specifically, we use TREC disks 1, 2 with query sets TREC 1, 2, 3 and disks 4, 5 (excluding CR) with query sets from TREC 6, 7, 8 and the Robust tracks. For experimentation on web data, we used WT10G collection with corresponding topic sets (TREC 9 and 10). Information about the document and the query sets is outlined in Table 2.1. We have observed in Chapter 3 that use of full index yields better performance on web corpora when Jelinek-Mercer smoothed language model is used for retrieval. As the baseline as well as the proposed GLM is based on LM-JM, for the reported experiments on web corpora, the full index is used for retrieval.

We implemented GLM using the Lucene[2] IR framework. As our baseline retrieval models, we used standard LM with Jelinek Mercer smoothing [82, 175], which is distributed as a part of Lucene.

4.4.2 Parameter settings

The parameter λ of the LM baseline was empirically set to 0.7 and 0.6 for TREC news collections and TREC web collection respectively (after varying it within a range of $[0.1, 0.9]$). The word embedding for the experiments reported in this section were obtained on the respective TREC document collections with the parameter settings as prescribed in [103], i.e., we embedded the word vector in a 200 dimensional space, and used continuous bag-of-words with negative sampling. The neighbourhood N_t of the GLM (Equation 4.8) was set to 3, i.e., for each given term in a document, we consider adding at most 3 related terms from the collection.

[2] http://lucene.apache.org/core/

Algorithm 1: Generalized Language Model: Expanded Indexing

Input : Document Collection C with D documents, and V unique terms
Output : Expanded index having three posting lists ($posting_{lm}, posting_{doc}, posting_{col}$) with corresponding term weights

$cs \leftarrow$ collection size
for $t \in V$ **do**
  $vec(t) \leftarrow$ vector embedding of t;
  $NN(t) \leftarrow$ compute K nearest neighbors of $vec(t)$;
  $ncf[t] \leftarrow cf(t)/cs$; `/* normalized collection frequency */`
for $d \in D$ **do**
  doclen $\leftarrow$ no. of tokens in d;
  `/* Term-weight calculation:transformation via doc sampling */`
  for *each unique term* $t \in d$ **do**
    $simSum[t] \leftarrow 0$;
    $ntf[t] \leftarrow tf(t,d)/doclen$;
    $\mathbf{t} \leftarrow vec(t)$;
    for *each unique term* $t' (\neq t) \in d$ **do**
      $\mathbf{t'} \leftarrow vec(t')$;
      $simSum[t] \leftarrow simSum[t] +$ cos-sim$(\mathbf{t}, \mathbf{t'})$;
  for *each unique term* $t \in d$ **do**
    $trans_{doc}[t] \leftarrow 0$;
    $\mathbf{t} \leftarrow vec(t)$;
    for *each unique term* $t' (\neq t) \in d$ **do**
      $\mathbf{t'} \leftarrow vec(t')$;
      $sim \leftarrow$ cosine-sim$(\mathbf{t}, \mathbf{t'})$;
      $trans_{doc}[t] \leftarrow trans_{doc}[t] + ntf[t] * sim/simSum[t]$;
    $posting_{lm}.add(ntf[t]/ncf[t])$;
    $posting_{doc}.add(trans_{doc}[t]/ncf[t])$;
    `/* Doc expansion for transformation via collection sampling */`
    $nn\_sim\_norm[] \leftarrow 0$;
    foreach $n \in NN(t)$ **do**
      $\mathbf{n} \leftarrow vec(n)$;
      $trans_{col}[n] \leftarrow ncf[t]$;
      $nn\_sim\_norm[n] \leftarrow nn\_sim\_norm[n] +$ cosine-sim$(\mathbf{t}, \mathbf{n})$;
    foreach $n \in NN(t)$ **do**
      $\mathbf{n} \leftarrow vec(n)$;
      $sim \leftarrow$ cosine-sim$(\mathbf{t}, \mathbf{n})$;
      $trans_{col}[n] \leftarrow trans_{col}[n] * sim/nn\_sim\_norm[n]$;
      $posting_{col}.add(trans_{col}[n]/ncf[n])$;

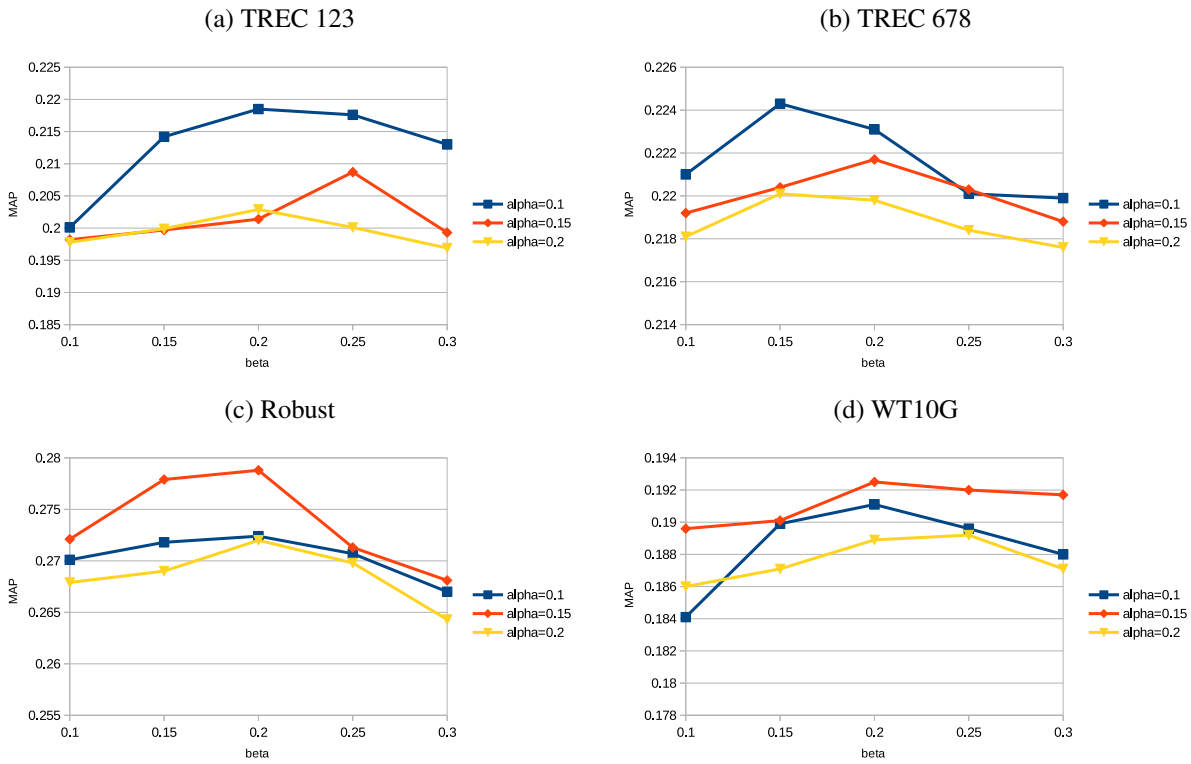


Figure 4.2: Effect of varying the GLM parameters α and β on the MAP values for the TREC query sets.

Query	Method	Metrics		
		MAP	P@5	Recall
TREC 123	LM	0.1967	0.4213	0.4477
	GLM	**0.2104**†	**0.4562**†	**0.4893**†
TREC 678	LM	0.2189	0.4160	0.5300
	GLM	**0.2214**†	**0.4187**	**0.5327**
Robust	LM	0.2658	0.4364	0.7881
	GLM	**0.2777**†	**0.4586**†	**0.7990**
WT10G	LM	0.1747	0.3152	0.6377
	GLM	**0.1906**†	**0.3782**†	**0.6689**†

Table 4.1: Comparative performance of GLM on the basis of mean average precision (MAP) precision at 5 (P@5) and recall at 1000. A † indicates the significance of the metric value with respect to the baseline LM based retrieval model.

4.4.3 Results

In Table 4.1, we show the optimal results obtained with GLM after performing 5-fold cross validation on the GLM parameters α and β. Both α and β are varied in steps of 0.05. From Table 4.1, we observe that GLM consistently and significantly outperforms the LM based retrieval method for all the query sets with respect to mean average precision (MAP), precision at rank 5 and recall at rank 1000.

While performing the initial retrieval using LM-JM, we noticed that the optimal performance is attaining when the smoothing parameter λ is set 0.7 and 0.6 respectively for news

and web queries. Hence, we fixed λ to values corresponding to the respective topic sets while setting α and β accordingly to ensure that $\alpha+\beta+\lambda < 1$ for all the query sets used in our experiments. The results of varying the parameters in the specified range are shown in Figure 4.2. It can be seen that the optimal values of α and β depend on the query set, e.g. for the TREC 123 query set (Figure 4.2a), the optimal results are obtained for $(\alpha, \beta) = (0.1, 0.2)$, whereas setting $(\alpha, \beta) = (0.1, 0.15)$ produces best performance for TREC 678 (Figure 4.2b). For Robust and WT10G topic sets, the optimal parameters setting is $(\alpha, \beta) = (0.15, 0.2)$ (Figure 4.2c and 4.2d respectively). It can be observed that a reasonable choice for these parameters is in the range $[0.1, 0.2]$, which means imparting more or less uniform weights to both of the transformation events.

The proposed method considers the occurrence of terms, which are semantically similar to the query terms, for ranking. While the model works on the basis of term transformation (via document and via collection, considered separately), they are less likely to be as effective as query expansion techniques. In the next chapter, we would compare the discuss on the relative performance of GLM and expansion methods.

4.5 Summary

In this chapter, we proposed a generalized version of the language model for IR. Our model considers two possible cases of generating a term from either a document or the collection and then changing it to another term after passing it through a noisy channel. The term transformation probabilities of the noisy channel, in turn, are computed by making use of the distances between the word vectors embedded in an abstract space. We argue that this model has two fold advantage, *firstly* it is able to estimate how well a term fits in the context of a document (by transformation via document sampling), and *secondly* it is able to decrease the vocabulary gap by adding other useful terms to a document (transformation via collection sampling). Empirical evaluation shows that our method significantly outperforms the standard language modeling based baseline.

Chapter 5

Query expansion using word embedding *

5.1 Introduction

In recent times, the IR and Neural Network (NN) communities have started exploring the application of deep neural network based techniques to various IR problems. A few studies have focused in particular on the use of *word embeddings* generated using deep NNs. As discussed in Section 2.4, a word embedding is a mapping that associates each word or phrase occurring in a document collection with a vector in $\mathbb{R}^d$, where d is the dimension of the abstract space and significantly lower than the size of the vocabulary of the document collection. If a and b are two words, and $\mathbf{a}$ and $\mathbf{b}$ are their embedded vectors in the d dimensional abstract space, then the similarity or distance between $\mathbf{a}$ and $\mathbf{b}$ is expected to be a quantitative indication of the semantic relatedness between a and b. Various techniques for creating word embedding — including Latent Semantic Analysis (LSA) [50] and probabilistic LSA [83] — have been in use for many years. However, interest in the use of word embedding has been recently rekindled thanks to the work by Mikolov et al. [103], and the robustness of the proposed *word2vec* tool. It has been reported [103] that the semantic relatedness of words is generally accurately captured by the vector similarity between the corresponding embedding produced by this method (discussed in Section 2.4).

Query Expansion (QE) is a popular and effective technique to overpower the hindrance problem of vocabulary mismatch for IR. Since a number of QE approaches try to find words that are semantically related to a given user query, it should also be possible to leverage word embeddings for query expansion. Let Q be a given user query consisting of the words $q_1, q_2, \ldots, q_m$. Let $w_1^{(i)}, w_2^{(i)}, \ldots, w_k^{(i)}$ be the k nearest neighbours (kNN) of q_i in the embedding space. Then, these $w_j^{(i)}$s constitute a set of obvious candidates from which terms may be selected and used to expand Q. Of course, instead of considering terms that are proximate neighbours of individual query words, it is generally preferable to consider terms that are close to the query as a whole. This idea has been used in a number of traditional, effective QE techniques, e.g., LCA [164] and RM3 [86]. However in these techniques, expansion terms are selected on the

*Some material from [136, 131] has been reused in this chapter.

basis of their association with all query terms, and any semantic relationships between terms are ignored.

Embedding vocabulary terms as dense, real valued vectors in a low dimensional space has been shown to successfully encapsulate semantic concepts associated with them [103]. Semantic relationships captured by such *word embeddings* have been used to improve the performance of various information retrieval tasks such as document ranking [169, 1], query reformulation [67, 178], relevance feedback [56], and end-to-end deep neural ranking models [68], session modeling [101], etc.

In this work, we explore the possibility of applying embedding for IR. Based on the semantic similarity as generated by embedding models, we describe two models in the next section. Experiments on a number of TREC collections show that these QE methods generally yield significant improvements in retrieval effectiveness when compared to using the original, unexpanded queries. However, they are generally significantly inferior to PRF based QE methods. We discuss these results in greater detail in Section 5.4.5.

In an embedded space of words, vectors for semantically related terms lie in close proximity to each other. Thus, the distance between embedded vectors of two terms can be taken as a measure of semantic relatedness between the terms. However, the choices made during the training phase can result in different representative vectors resulting in the *relative* semantic relatedness between terms not getting preserved across different embedding spaces. Different studies on the application of word embeddings to IR problems have made different choices for learning the word embeddings (i.e. which collection to use, what preprocessing to apply, etc.). This highlights the lack of established standards in this regard. On one hand, we have results that show that query-specific embedding trained on a subset of relevant document are superior over globally trained embedding [56], on the other hand, retrieval evaluations are conducted using embedding learned over an external collection such as Wikipedia [169]. The pre-TREC [71] IR community suffered from the lack of a similar set of 'best practices' (e.g. stopword removal is effective, too aggressive stemming often degrades retrieval quality). The best practices gradually became established after a few years of the TREC evaluation forum.

While efforts have been made to study the sensitivity of word embeddings to different model parameters (e.g. embedding dimensions, context window size, skip-gram v/s cbow, etc.) and their impact on ad hoc retrieval performance [183], little attention has been paid to investigate the effects of the nature of the *training collection* and *term normalization* used to learn the embedded vectors. As a byproduct of this work, we focus on this aspect and study *how the choices made with respect to the underlying corpus and its pre-processing impact the performance of downstream IR tasks*. Since the embedding models rely on context around the terms in training corpus, these choices are crucial as they directly affect the *contextual information* around each term and thus, can result in significantly different embedding spaces (Section 5.3). While a large, generic external corpus may be able to provide diverse contextual information to learn good embedding representations, the target corpus (i.e., the collection on which retrieval is to be executed) will be topically more relevant. Likewise, normalizing terms (via stemming) can help aggregate the contexts of different morphological forms to-

gether and thus, may help learn better representations of these terms.

In order to study how the above choices effect the learned embeddings, we first describe two measures to compare embeddings learned under different settings (Section 5.3). Next, we use the proposed word embedding-based query expansion method (to be discussed in Section 5.2) to analyze the impact of embedding variations on the retrieval performance on three different test collections. To study the impact of collection choice, we compare retrieval performance obtained using the embeddings trained on the target collection as well as Wikipedia (a generic, external collection). To understand the effect of term normalization, we report results using embeddings learned from stemmed and unstemmed collections. In addition, we also describe an intermediate *composition* method that can be used to simulate the effect of stemming in cases where pre-trained word vectors on unstemmed corpus are available and re-training may not be feasible. We report all results with two different word embedding models, (i) **word2vec** [103], one of the most commonly used word embedding model; and (ii) **fastText** [21], a word embedding model that utilizes sub-word information for learning term representations. We decided to study fastText as it performs implicit term normalization by composing skipgram vectors of character n-grams [21].

The remainder of this chapter is organized as follows. In Section 5.2, we propose some query expansion methods based on word embeddings. Section 5.3 discusses ways to compare different embedded representations based on the QE method. We present and discuss the performance of the proposed QE methods on benchmark datasets in Section 5.4. Using the proposed method, we compare the retrieval performance achieved using different word vector representations learned under different settings and analyze the reasons for these differences in Section 5.5. Section 5.6 concludes this chapter with some suggested directions for future work.

5.2 Word embedding based query expansion

One of the goals in this work is to study how word embeddings may be applied to QE for ad hoc retrieval. Specifically, we are looking for answers to the following research questions.

(RQ_1) Does QE using the nearest neighbours of query terms improve retrieval effectiveness?

(RQ_2) How does embedding based QE perform compared to an established QE technique based on relevance feedback like RM3 [86]?

We will start this section by exploring two different embedding based QE methods. We describe two QE methods that use the embeddings of individual query terms. The first method is a kNN based QE method that makes use of the basic idea outlined in Section 5.1. Unlike pseudo relevance feedback (PRF) based QE methods, this method does not require an initial round of retrieval. The second approach we tried is a straightforward variation of the first approach that uses word embeddings in conjunction with a set of pseudo relevant documents. Finally, we describe how we obtain the embeddings of an extended query term set by using compositionality of terms. The nearest neighbours of this extended query term set are then used for QE. For both methods, we used *word2vec* [103] for computing word embedding.

5.2.1 Pre-retrieval kNN based approach (QE_{pre})

Let the given query Q be $\{q_1, \ldots, q_m\}$. In this approach, we define the set C of candidate expansion terms as

$$C = \bigcup_{q \in Q} NN(\mathbf{q}) \tag{5.1}$$

where $\mathbf{q}$ represents the embedding of q, and $NN(\mathbf{q})$ is the set of K terms that are closest to q in the embedding space. For each candidate expansion term w in C, we compute the mean inner product similarity between w and all the terms in Q following Equation (5.2). The availability of vector for each term in the vocabulary makes it possible to measure the similarity between the two terms.

$$Sim(w, C) = \frac{1}{|C|} \sum_{q_i \in C} \mathbf{w}.\mathbf{q_i} \tag{5.2}$$

On the basis of this mean score of Equation (5.2), all terms in C are sorted and the top K terms, having the highest similarities, are selected as the actual expansion terms. The expansion term weights of the K selected terms are computed by normalizing the expansion term score (mean similarity with respect to all the terms in Q) by the total score obtained by summing over all top K expansion terms, as presented in Equation (5.3).

$$FW(w) = \frac{Sim(w, C)}{\sum_{w \in Q_K} Sim(w, Q)} \tag{5.3}$$

5.2.2 Post-retrieval kNN based approach (QE_{post})

In the pre-retrieval approach, no feedback information is utilized while selecting the expansion terms. Also, the candidate expansion terms are selected from the whole vocabulary. In our next approach, we use a set of pseudo-relevant documents (PRD) — documents that are retrieved at top ranks in response to the initial query — to restrict the search domain for the candidate expansion terms. Instead of searching for nearest neighbours within the entire vocabulary of the document collection, we consider only those terms that occur within PRD. The size of PRD may be varied as a parameter. The rest of the procedure for obtaining the expanded query is similar to that applied for the pre-retrieval QE method, discussed in Section 5.2.1.

5.2.3 Extending the query term set

For several queries, e.g., TREC query 390 (*Orphan Drugs*), query term n-grams convey the user's information need far more accurately than the query terms taken individually. For such queries, the individual terms may have multiple associations (nearest neighbours) that are not related to the actual information need. Thus, for these queries, it might be better to look for nearest neighbours using the compositional sense of two or more words taken together. As pointed out in [103], an appropriate embedded representation of the conceptual meaning conveyed by the composition of two or more words can be obtained by simply adding the corresponding vectors in the embedding space. We use these ideas to obtain an extended query term set (EQTS) for any given query as follows.

Given a query Q consisting of m terms $\{q_1, \ldots, q_m\}$, we first construct Q_c, the set of query word bigrams.

$$Q_c = \{\langle q_1, q_2\rangle, \langle q_2, q_3\rangle, \ldots, \langle q_{m-1}, q_m\rangle\} \tag{5.4}$$

We define the embedding for a bigram $\langle q_i, q_{i+1}\rangle$ as simply $\mathbf{q_i} + \mathbf{q}_{i+1}$, where $\mathbf{q_i}$ and $\mathbf{q}_{i+1}$ are the embeddings of words q_i and q_{i+1}. Next, we define the extended query term set (EQTS) Q' as

$$Q' = Q \cup Q_c \tag{5.5}$$

We consider only the linear chain composition of q_i and q_j from left to right instead of considering all possible combinations, assuming coexisting terms (when $j - i = 1$) give more meaningful composition than terms q_i and q_j with other words appearing in between them (when $j - i > 1$).

$$C' = \bigcup_{q \in Q'} NN(\mathbf{q}) \tag{5.6}$$

The effect of compositionality can be incorporated into the methods described in Sections 5.2.1–5.2.2 by replacing C in Equations (5.1) and (5.2) by C' obtained above, and applying it to compute the expansion term weight for a term w in Equation (5.3).

5.2.4 Retrieval

In Equation (5.3), the expanded query is formed depending only on the terms similar to the original query terms (Q). In order to maintain the information contained in the original query model, the query model is linearly interpolated with the embedding based similarity weight following Equation (5.7). The actual retrieval can be done using any standard retrieval model. For our experiments, we used Language Model with two different smoothing techniques, specifically, Jelinek Mercer and Dirichlet smoothing [172, 175, 174].

$$FW'(w) = \phi P(w|Q) \; + \; (1 - \phi) FW(w) \tag{5.7}$$

In Equation (5.7), $F(w)$ represents the weight of the term w based on the vector similarity (following Equation (5.3)). To select the K candidate expansion terms, we can either use Q - the actual query terms, or Q' - the query terms along with its composed forms in Equation 5.3 (as discussed in Section 5.2.3). Note that, we are supposed to get two different set of expansion terms C' and C respectively depending on whether the composition of query terms is considered (Equation (5.6)) or not (Equation (5.1)). The interpolation parameter ϕ is used to combine the original unexpanded query with the expansion terms. In Equation (5.7), $P(w|Q)$ is the maximum likelihood estimation of the term w being a query term.

5.3 Effects of embedding variations

In this section, we describe two important factors – *viz.*, the effect of the i) *choice of training collection* and ii) *term normalization* – which can lead to variations in the embedded space of vectors thus potentially leading to differences in IR effectiveness. We also formulate and describe ways to measure the similarity between two different embeddings.

5.3.1 Measuring similarity between embeddings

Word vector embeddings are obtained through training on an underlying collection of documents. More formally, in skipgram model of *word2vec* [103], word vectors are obtained by sliding a context window pivoted at each word position and maximizing the likelihood of generating the context given the current word. Clearly, the context words surrounding the target word can be very different for different collections resulting in different representative vectors for the same word. Given two different embedded spaces $\mathbb{E}_1$ and $\mathbb{E}_2$, we now formally describe ways of measuring the similarity between them. By the property of the objective function of the word2vec algorithm, two word vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ will have similar representations if the word y shares a fair amount of context terms with the word x. Intuitively, two embedded spaces will be similar if there is considerable overlap in the set of neighborhood (top-k most similar) vectors around each word. Formally, this can be measured by the average Jaccard similarity between the top-K neighborhoods around each word in the two embedded spaces, i.e.,

$$\sigma_k(\mathbb{E}_1, \mathbb{E}_2) = \frac{1}{|\mathbb{E}_1 \cap \mathbb{E}_2|} \sum_{\mathbf{x} \in \mathbb{E}_1 \cap \mathbb{E}_2} \frac{|N_k^{\mathbb{E}_1}(\mathbf{x}) \cap N_k^{\mathbb{E}_2}(\mathbf{x})|}{|N_k^{\mathbb{E}_1}(\mathbf{x}) \cup N_k^{\mathbb{E}_2}(\mathbf{x})|}, \tag{5.8}$$

where $N_k^{\mathbb{E}}(\mathbf{x})$ denotes the neighborhood set of top-k most similar vectors around word x in embedding $\mathbb{E}$. The Jaccard similarity does not take into account the relative ranks of the neighborhood word vectors. A way to incorporate the rank information then is to measure the average Spearman's correlation coefficient ρ (normalized within the range of $[0, 1]$ to be consistent with Equation 5.8), between the top-K neighborhoods, as follows.

$$\rho_k(\mathbb{E}_1, \mathbb{E}_2) = \frac{1}{|\mathbb{E}_1 \cap \mathbb{E}_2|} \sum_{\mathbf{x} \in \mathbb{E}_1 \cap \mathbb{E}_2} \frac{1}{2}\Big(1 + \rho\big(N_k^{\mathbb{E}_1}(\mathbf{x}), N_k^{\mathbb{E}_2}(\mathbf{x})\big)\Big) \tag{5.9}$$

For both metrics, a higher value indicates higher similarity between two embedded spaces. In section 5.5, we use the metrics of Equations 5.8 and 5.9 to measure the similarity between embeddings learned over different collections and under different settings, and investigate the correlation between the differences in embedding spaces with differences in the performance of the downstream IR tasks.

5.3.2 Collection choice - target vs. external collection

Some reported studies on word embedding based approaches in IR use word vectors trained on respective document collections to improve retrieval performance [62, 56], while there are others that have used pre-trained word vectors [169]. Since the collection on which an embedding method is trained can play an important role in learning the relatedness between terms, an obvious choice is to use the underlying document collection of the retrieval system to learn the word vectors. The word vectors trained on the target retrieval collection are likely to capture the underlying term semantics specific to that collection, which can lead to improvements in IR effectiveness. The alternative is to use pre-trained vectors on large, external document collections that are often publicly available (e.g. word2vec, fastText, provide such pre-trained vectors for different languages). However, the term semantics learned with such generic collections, e.g. *Wikipedia*, may not capture the relevant associations between terms

required to improve IR effectiveness on a particular search collection, e.g. news-wire. This leads to our next research question, depending on the choice of collections while training the embedding model.

(RQ_3) Does it help to use word vectors trained over the target test collection instead of pre-trained word vectors on external corpora, as the former may better capture semantic relations of the target collection?

5.3.3 Choices for term normalization

Term normalization (e.g. stemming) plays an important role in IR. An intuitive pre-processing step is thus to apply stemming before training and directly learn the word vectors for stemmed roots. During retrieval, the stemmed query terms can then be exactly matched with the embedded word vectors. However, after training word vectors on a document collection that is not pre-processed, or when working with pre-trained word vectors trained on unprocessed collections, one needs to consider how to match the embedded word vectors with the index units of an IR collection (typically stemmed). We observed that the reported studies usually lack clarity regarding this matching step and this has led to differences in experimental setup when implementing the same retrieval methodology by different research studies. For instance, the work in [62] used pre-processing before training word vectors on the TREC Robust collection. However, when reproducing their results for a baseline, such pre-processing was not applied in [169]. We propose an indexing unit composition (IUC) method to resolve this normalization difference between the index terms and the word vectors by stemming the trained word vectors and composing (vector sum) the ones that yield identical stems (representative of equivalence classes). This yields a single vector representation for each equivalence class. Another alternative to account for term normalization differences could be to use an alternative embedding model, such as fastText [21], that implicitly performs term normalization by utilizing character n-grams. These experimental choices lead us to following research questions –

(RQ_4) What effect does embeddings trained on a stemmed collection have on IR performance compared to an unstemmed collection?

(RQ_5) What effect does using approximations such as IUC or different embedding model such as fastText have on IR performance?

In rest of the chapter, we first present the empirical results of the proposed QE methods on benchmark TREC datasets in Section 5.4. After discussing the performance of the proposed methods with two differently smoothed language model, we focus on the experiments on variations of embedding applied with the QE method (Section 5.5). We experimented with different datasets with embeddings learned on *in-domain* and *out-domain* text sources.

5.4 Experiments on query expansion

5.4.1 Dataset overview

The query expansion experiments are performed on both TREC news and web collections. Specifically, we used TREC disks 1, 2 and disks 4, 5 (comprising of news articles) along with the topic sets for the TREC 1, 2, 3 and 6, 7, 8 respectively. For TREC disks 4 and 5, the Robust topic set is also used for experimentation. The proposed methods were also evaluated on TREC web track collection WT10G, GOV2 and ClueWeb09 - category B with corresponding topic sets. The characteristics of the datasets can be found in Table 2.1.

5.4.2 Indexing and word embeddings

At the time of indexing the test collections, we remove stopwords following SMART stopword-list. Porter's stemmer is used for stemming of words. The text of the same document collection on which retrieval will be performed, is used (after stopword removal and stemming) for training the embedding models. Due to vocabulary size scale issue, for experiments on ClueWeb09 collection, the embedding learned on the GOV2 collection is used. The vectors are embedded in an abstract 200 dimensional space with negative sampling using 5 word windows on continuous bag of words model. For the training, we removed any words that appear less than three times in the whole corpus. These are as per the parameter settings prescribed in [103].

5.4.3 Parameter settings

While doing the experiments, only the *title* field of TREC topics is considered as query. The smoothing parameters exclusive to the respective models (λ and μ for LM-JM and LM-Dir respectively) are varied in the range $[0.1, 0.9]$, and $\{100, 200, 500, 1000, 2000, 5000\}$. For each topic set, the retrieval parameters λ and μ are tuned separately; that is, the parameter producing the best performance while doing the baseline retrieval on the respective topic sets, is applied for both initial, as well as proposed expanded retrieval. The linear interpolation parameter ϕ (see Equation 5.7) is varied in the range $[0.3, 0.7]$. For the feedback based baseline method, RM3, the parameters are tuned for all the collections in the similar way as presented in Section 3.5.

The proposed expansion methods QE_{pre} and QE_{post} have two unique parameters: $i)$ K - the number of expansion terms having the highest weights associated with them (following Equation (5.7)), and $ii)$ ϕ - the interpolation parameter. Additionally, the feedback based method QE_{post} (Section 5.2.2) has one more parameter, the number of documents to use for feedback. To compare the best performance of the proposed methods, we explored all parameter grids to find out the best performance of the individual approaches. Specifically, the number of expansion terms K is varied in the range $\{30, \ldots, 140\}$ in steps of 10. The interpolation parameter ϕ is varied from 0.1 to 0.9 in steps of 0.1. For the post-retrieval expansion, the number of feedback document is varied in the range $\{10, \ldots, 30\}$ in steps of 10. Finally, we report the result after 5-fold cross-validation among all the results obtained. The results are reported in Table 5.2.

Method	wvec compo	TREC123	TREC678	Robust	WT10G	GOV2	CW09B
LM-JM	-	0.1967	0.2189	0.2658	0.1747	0.2313	0.1066
QE_{pre}	no	0.2072	0.2243	0.2783	0.1817	0.2413	0.1117
QE_{post}	yes	0.2153	0.2270	0.2846	0.1834	0.2459	0.1127
LM-Dir	-	0.2278	0.2246	0.2875	0.2192	0.2973	0.1773
QE_{pre}	no	0.2311	0.2266	0.2893	0.2215	0.2985	0.1801
QE_{post}	yes	0.2406	0.2278	0.2936	0.2232	0.2995	0.1824

Table 5.1: Comparison of performance in terms of MAP with and without applying composition (respectively using C and C' in Equation (5.2)).

5.4.4 Results

In the preliminary set of experiments, we compare the effect of applying composition, (replacing C in Equations 5.1 and 5.2 by C' obtained by Equation 5.6) when computing the similarity between an expansion term and the query, for the pre-retrieval expansion approach QE_{pre} (see Section 5.2.1). The effect of applying compositionality is tested separately for both language models (LM-JM and LM-Dir). In terms of MAP, the relative performance of QE_{pre} with and without composition is presented in Table 5.1. It is evident from the result that applying composition indeed affects the performance positively for all the topic sets, irrespective of the retrieval models used. Thus, in subsequent experiments with both proposed QE methods, we report the results after applying composition for similarity computation and expansion term selection.

Table 5.2 shows the performance of QE_{pre} and QE_{post}, compared with the baseline LM models and feedback model RM3. From the table, it can be seen that both the QE methods based on word embeddings always outperform both the LM baselines (almost always significantly) in terms of all metrics. Among the QE methods, embedding based expansion with feedback (QE_{post}) is always seen superior than its pre-retrieval counter-part. The performance of QE_{post} is seen to be superior than the RM3 (often significantly) for most of the topic sets in terms precision at rank 5 ($P@5$).

A query-by-query comparison between the baseline LM and post-retrieval method QE_{post} is presented in Figure 5.1. Each vertical bar in the figure corresponds to a query, and the height of the bar is the difference in AP for the baseline (LM-JM) and QE_{post} two methods for that query. Bars above X-axis corresponds to queries for which, the performance is getting improved by the proposed expansion method. The figure show that, as an expansion technique, the post-retrieval method (QE_{post}) is generally safe: it yields improvements for most queries (bars above the X axis), and hurts performance for only a few queries (bars below the X axis).

In Figure 5.2, a query-by-query comparison between the post-retrieval QE (QE_{post}) and RM3 is presented. From this figure, we notice an interesting trend of the proposed method. For half of the topics of TREC 678 and GOV2, we observe a relative improvement of QE_{post} over RM3 which implies that QE_{post} is almost equally effective as RM3 for average topics. However, for the topics of CW09B, we see a greater number of topics for which the performance of QE_{post} is superior than RM3. These experiments provide some answers to research

Query	Method	LM-JM			LM-Dir		
		MAP	P@5	Recall	MAP	P@5	Recall
TREC 123	Baseline	0.1967	0.4213	0.4477	0.2278	0.5213	0.4942
	QE_{pre}	0.2153*	0.4867*	0.4755*	0.2406*	0.5293	0.5052
	QE_{post}	0.2344*	**0.5013**^{*r}	0.4882*	0.2456*	**0.5427**^{*pr}	0.5163*
	RM3	**0.2507**^{*ps}	0.4653*	**0.5154**^{*ps}	**0.2798**^{*ps}	0.5320	**0.5521**^{*ps}
TREC 678	Baseline	0.2189	0.4160	0.5300	0.2246	0.4320	0.5258
	QE_{pre}	0.2270*	0.4373	0.5435*	0.2278	0.4493	0.4000
	QE_{post}	0.2407*	**0.4693**^{*pr}	0.5595*	0.2303	**0.4653***	0.5417*
	RM3	**0.2480**^{*p}	0.4440*	**0.5647**^{*p}	**0.2529**^{*ps}	0.4640*	**0.5720**^{*ps}
Robust	Baseline	0.2658	0.4364	0.7881	0.2875	0.5354	0.7956
	QE_{pre}	0.2846*	0.4889*	0.8080	0.2936	0.5172	0.8000
	QE_{post}	0.2959*	**0.5030***	0.8336*	0.2940	0.5354	0.8096*
	RM3	**0.3309**^{*ps}	0.4929*	**0.8596**^{*p}	**0.3379**^{*ps}	**0.5374**	**0.8594**^{*ps}
WT10G	Baseline	0.1747	0.3152	0.6377	0.2192	0.3537	0.7086
	QE_{pre}	0.1834	0.3172	0.6465	0.2232	**0.3976**^{*sr}	**0.7174**
	QE_{post}	0.1939*	**0.3414**^{*p}	0.6646*	0.2185	0.3596	0.7157
	RM3	**0.2094**^{*p}	0.3394*	**0.6743**^{*p}	**0.2295***	0.3697	0.7113
GOV2	Baseline	0.2313	0.4698	0.5915	0.2973	0.5799	0.6820
	QE_{pre}	0.2457	0.5087	0.6007	0.2995	0.5893	0.6847
	QE_{post}	0.2636*	**0.5705**^{*r}	0.6072*	0.3025*	**0.6134**^{*pr}	0.6812
	RM3	**0.2638**^{*p}	0.4134	**0.6359**^{*p}	**0.3269**^{*ps}	0.6040*	**0.6995**^{*s}
CW09B	Baseline	0.1138	0.1687	0.5937	0.1773	0.2303	0.7223
	QE_{pre}	0.1199*	0.2020	0.6280*	0.1824*	0.2434	**0.7430**^{*sr}
	QE_{post}	**0.1697**^{*pr}	**0.2727**^{*pr}	**0.6709**^{*pr}	0.1792	**0.2556***	0.7144
	RM3	0.1540*	0.2505*	0.6480*	**0.1882***	0.2545*	0.7304*

Table 5.2: Different evaluation metrics for baseline retrieval and various embedding based QE strategies including RM3. A * in the QE rows denotes a significant improvement over the baseline. A p, s and r in the QE metrics with optimal performance denotes a significant improvement over the pre-retrieval, post-retrieval and RM3 based QE techniques respectively.

questions (RQ_1) and (RQ_2) listed in Section 5.2.

A dataset wise comparison of performance of the proposed QE methods with the GLM (proposed in Chapter 4) can be easily exhibited from Table 4.1 and 5.2. For the news topics (TREC123, TREC678 and robust), both the proposed QE methods (QE_{pre} and QE_{post}) are seen to be better than GLM in terms of all evaluation metrics. However, for the web collection WT10G, GLM is seen to be better performing in terms of early precision (P@5) than the QE techniques. The reason is the noisy nature of the web corpora in which the chances of query drifting is severe when performing query expansion.

5.4.5 Discussion

Query expansion intuitively calls for finding terms which are both "similar" to the query, and which occur frequently in (pseudo) relevant documents. In the proposed embedding based QE techniques, only the terms which are semantically similar to the query terms in the collection-level abstract space are considered as the expansion terms. More precisely, in the pre-retrieval QE method QE_{pre}, expansion terms are selected from the entire vocabulary,

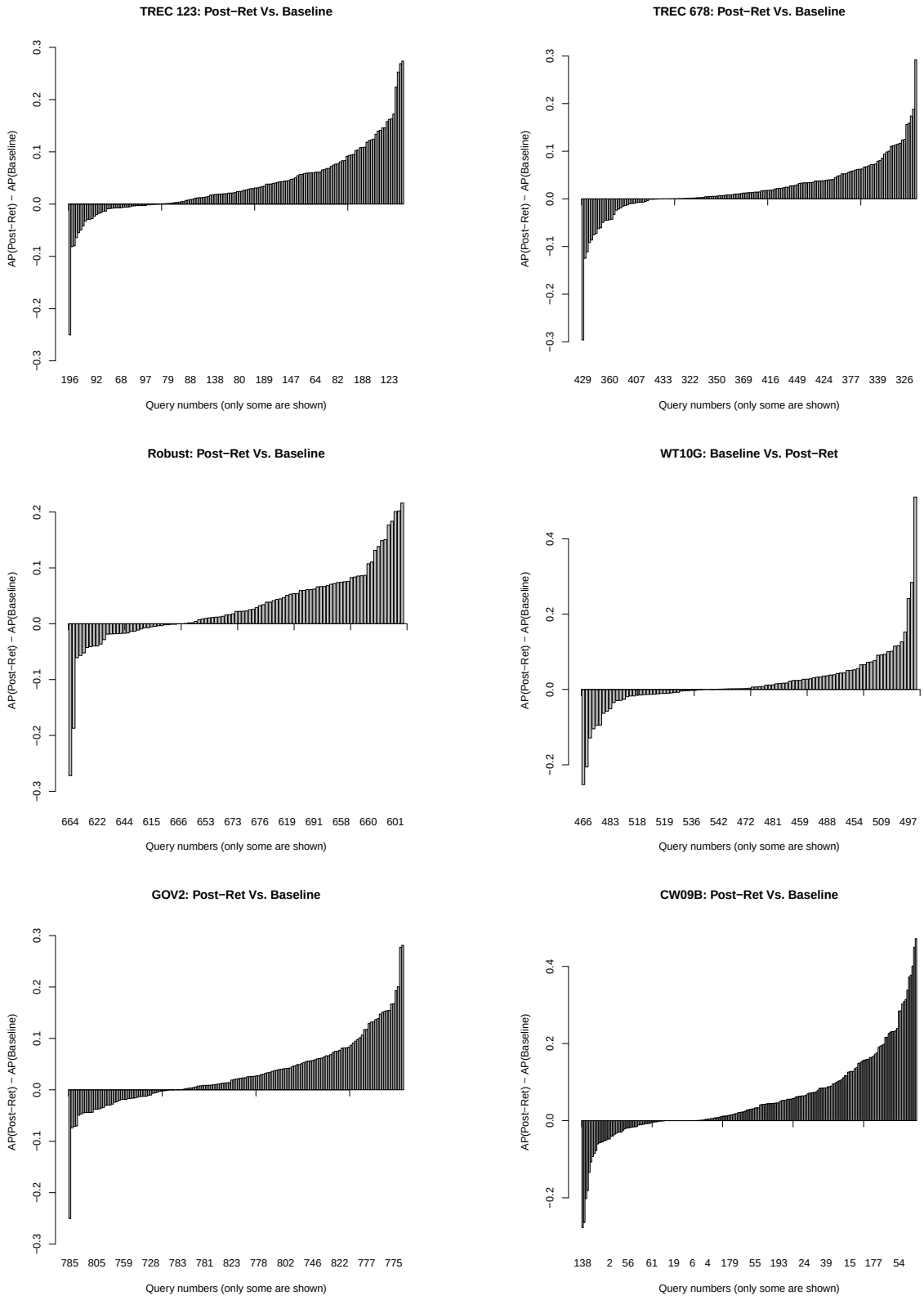


Figure 5.1: Comparison of baseline and post-retrieval QE with respect to AP for individual queries.

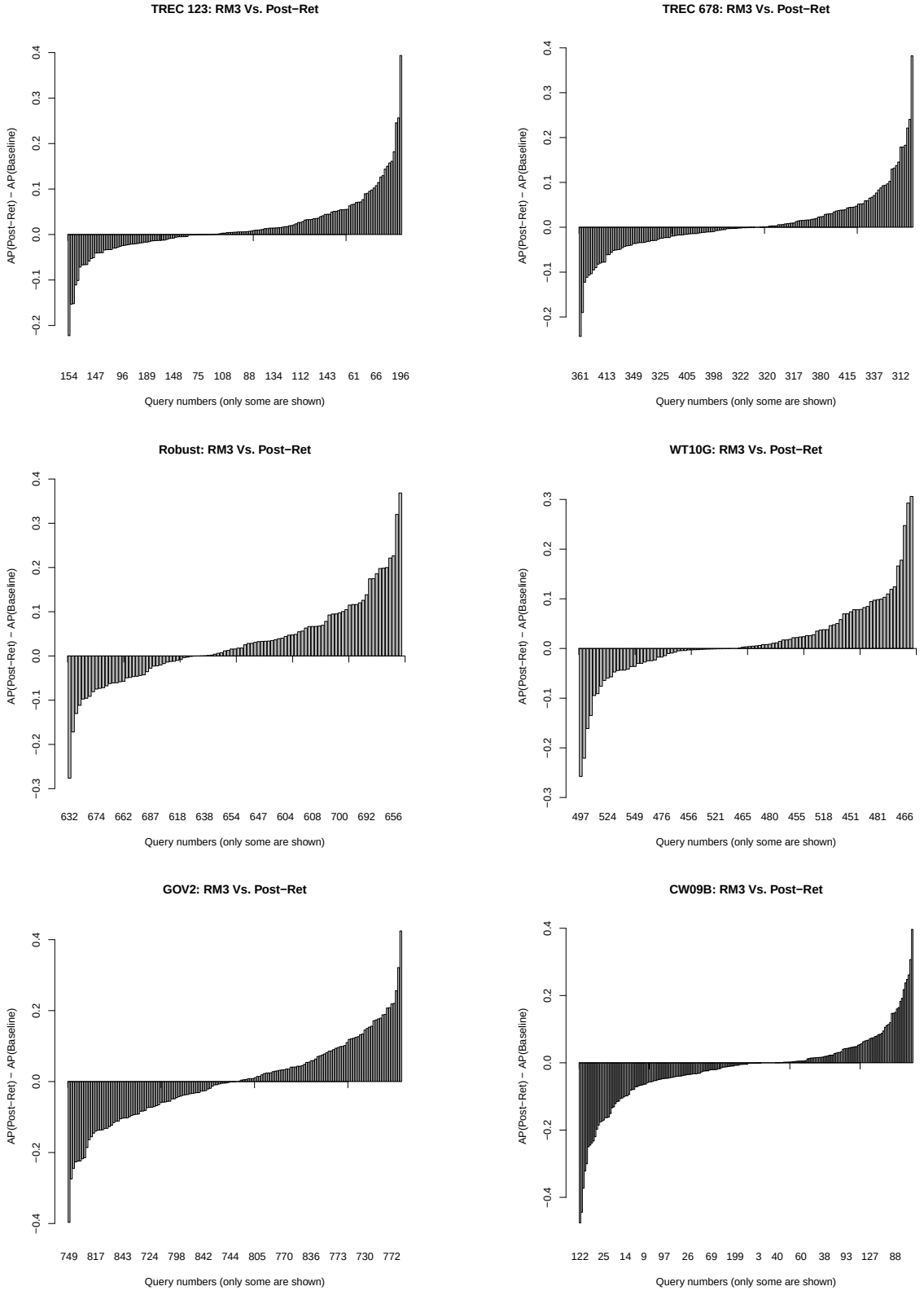


Figure 5.2: Comparison of RM3 and post-retrieval QE with respect to AP for individual queries.

based on the similarity with query terms (or, composed query terms). In the post-retrieval expansion model QE_{post}, the vocabulary is restricted to terms occurring in top-ranked documents (as retrieved using the initial query). Thus, candidate expansion terms are obtained only from documents with a relatively high vocabulary overlap with the query. This mitigates the risk of query drift to an extent. The post-retrieval approach (QE_{post}) thus performs better as compared to its pre-retrieval counter-part QE_{pre}. Moreover, due to the reduction in the size of the search space, the expansion process for QE_{post} is also significantly faster than its pre-retrieval counter-part. Due to the generalization that occurs when training the vector representation of the words on the *entire* corpus, however, the effect of query drift cannot be completely nullified.

Specifically, the techniques fail to capture the other features (such as co-occurrence frequency of a candidate expansion term with query terms) of state-of-the-art traditional expansion methods. Not surprisingly, as seen in Table 5.2, the performance of RM3 is often significantly better than the proposed methods. This indicates that co-occurrence statistics are more powerful than the similarity in the abstract embedding space. As suggested above, the reason RM3 works better than the embedding based QE, may be attributed to the generalization of context that occurs while training the embedding model. The embedding model we used, *word2vec*, does not capture the document level or collection level difference in context due to the generalization. In spite of this, we have seen a consistent (often significant) improvement of precision at rank 5 ($P@5$) over RM3 by QE_{post} for almost all topics. Another observation we can make from Table 5.2 is, the percentage of improvement over the initial retrieval is notable for both QE_{pre} and QE_{post} when the retrieval is performed using LM-JM as contrasted to LM-Dir.

Even though the simplicity of the methods puts them at an obvious disadvantage, the same simplicity has an advantage that will become clear in the next section: because both techniques are fairly straightforward, there are fewer confounding factors when studying how retrieval effectiveness changes with variations in the underlying embedding method.

5.5 Performance variations with different embeddings

In this section, we will discuss about the empirical effect on performance of the proposed QE method due to variability in the applied embedding model. The objective of this experiments is firstly to investigate what parameters of word2vec can lead to significant differences in the embedded space. We first conduct a set of experiments to measure the inter-embedding similarities between a set of embedded vector spaces with different settings. We then study the propagation effects of differences in embedded spaces on IR effectiveness.

5.5.1 Dataset and embedding settings

We use three different standard IR collections and learn word embeddings over them under different settings. Each embedding setting corresponds to a permutation of the following three components.

- Training algorithm: either **word2vec** or **fastText**.

Collection	Normalization	Name	Collection	Normalization	Name
Wikipedia	Unprocessed	Wk-U	WT10G	Unprocessed	WT-U
Wikipedia	Composed (Post)	Wk-C	WT10G	Composed (Post)	WT-C
Wikipedia	Stemmed (Pre)	Wk-S	WT10G	Stemmed (Pre)	WT-S
TREC Robust	Unprocessed	Rb-U	GOV2	Unprocessed	Gv-U
TREC Robust	Composed (Post)	Rb-C	GOV2	Composed (Post)	Gv-C
TREC Robust	Stemmed (Pre)	Rb-S	GOV2	Stemmed (Pre)	Gv-S

Table 5.3: Names for different experiment settings.

- Training collection: on which embeddings are trained, either **target** or **external**.
- Term normalization applied before/after learning word embeddings.

As concluded in [183], the choice of parameter settings applied while training the embedding model, specifically dimension, size of window and models (cbow or skipgram) can hardly affect the performance[2]. Hence, without varying further, we experimented with the following settings for both *word2vec* and *fastText*. For both word2vec and fastText, cbow model is used with context window size set to 10 and 5 respectively. The dimension of the vectors is set to 200 and 100 for word2vec and fastText respectively. All the other parameters are set to their default values. For the training collection option, **target** indicates the collection on which retrieval is to be performed. We use TREC Robust, i.e. TREC disk 4, 5 excluding CR with topic ids 301-450 and 601-700 (indicated by **Rb**), WT10G (**Wt**) and GOV2 (**Gv**) collections for this purpose. The statistical overview of the test collections can be found in Table 2.1. The **external** option indicates that the embedding is learned on an external collection. As the external collection, we use the entire Wikipedia dump[3] (November 2013) that was used in the Tweet contextualization track of CLEF-2013 [17]. The collection contains a total of $3,902,345$ documents in xml format. While creating the text dump to train the embedding model, all structural metadata (such as html tags, urls etc.) are removed.

To study the effect of term normalization on performance, we experimented with the following three variations.

- **U -** No normalization or pre-processing (specifically stemming) of the words (other than case-folding and stopword removal) during word vector training.
- **C -** No pre-processing during word vector training, followed by a post-processing step of composition (vector-sum) of the words that yield identical stems.
- **S -** Pre-processing (specifically stemming with Porter stemmer [119]) each word of a collection before training word embedding.

Table 5.3 summarizes different settings used for learning different embedding variants.

5.5.2 Results and discussion

Next, to study the effect of different embedding variants on retrieval performance, we employ the embedded vectors for query expansion (QE) as proposed in Section 5.2.1. Specifically,

[2]Our initial study also confirms the observations noted in [183]
[3]Available from http://qa.termwatch.es/data

Equations (5.1) and (5.2) is used with C' of Equation (5.6).

The cosine-similarity values of these nearest neighbors are used as their weights in Q'. We used Lucene for indexing and retrieval and implementing the above model. The pre-retrieval QE approach has been applied here with differently made embeddings.

Word2vec	Rb-U	WT-U	Gv-U
Wk-U	0.5828 / 0.0911	0.5643 / 0.0600	0.6040 / 0.1319

Word2vec	Rb-C	Rb-S	Wk-C	Wk-S
Rb-C		0.6057 / 0.1396	0.5773 / 0.0827	0.5757 / 0.0794
Rb-S			0.5881 / 0.1085	0.5997 / 0.1282
Wk-C				0.7022 / 0.3161

Word2vec	WT-C	WT-S	Wk-C	Wk-S
WT-C		0.8315 / 0.4853	0.5919 / 0.1249	0.6063 / 0.1521
WT-S			0.5993 / 0.1355	0.6180 / 0.1694
Wk-C				0.7002 / 0.3170

Word2vec	Gv-C	Gv-S	Wk-C	Wk-S
Gv-C		0.7412 / 0.3388	0.5903 / 0.1237	0.6101 / 0.1451
Gv-S			0.5804 / 0.0970	0.5927 / 0.1190
Wk-C				0.6989 / 0.3094

FastText	Rb-U	WT-U	Gv-U
Wk-U	0.5899 / 0.1295	0.6286 / 0.1889	0.5744 / 0.1787

FastText	Rb-C	Rb-S	Wk-C	Wk-S
Rb-C		0.7659 / 0.4002	0.5994 / 0.1182	0.5837 / 0.0911
Rb-S			0.5912 / 0.1054	0.5940 / 0.1071
Wk-C				0.7532 / 0.3924

FastText	WT-C	WT-S	Wk-C	Wk-S
WT-C		0.5595 / 0.0669	0.5611 / 0.0455	0.5539 / 0.0315
WT-S			0.5914 / 0.1247	0.6083 / 0.1549
Wk-C				0.7339 / 0.3714

FastText	Gv-C	Gv-S	Wk-C	Wk-S
Gv-C		0.7484 / 0.3086	0.5484 / 0.0349	0.6572 / 0.2304
Gv-S			0.5832 / 0.0431	0.5505 / 0.0380
Wk-C				0.7411 / 0.3892

Table 5.4: σ_k and ρ_k (below and above diagonal of each cell, respectively) values between different embedding spaces. A value of $k = 60$ is used to compute the similarities. Since it is not possible to compute the inter-embedding similarities between an unstemmed space and a stemmed one without the application of IUC, some comparisons do not exist in the table, e.g.between the pair WT-C and WT-U.

Table 5.4 reports the similarity metrics ρ_k and σ_k (Equations 5.8 and 5.9) between embeddings learned using different collections and settings. Note how the embeddings learned from different collections differ significantly as measured by the two parameters. For e.g., the σ_k values between Wk-U and the three collections (Rb-U,WT-U, Gv-U) lie between 0.56 and 0.60 indicating that the semantic concepts associated with the terms as captured by different collections vary significantly. Further, even when comparing the embeddings learnt over same document collection, we observe that normalization choices lead to very different embedding spaces. For example, word2vec embeddings for Rb-C and Rb-S have a σ_k of 0.6057 indicating that even for the same underlying corpus, different normalization approaches can result in very different embedding spaces. Similar observations can be made for other settings and together, they lend significant weight to our initial hypothesis that *embedding spaces can differ appreciably with variations in term normalization, stemming, etc. even if the underlying corpus remains same.*

Next, we report the results of the word vector based QE approach on the three test collections in Table 5.5. We observe from the table that for word2vec embeddings, using the target corpus and some form of normalization (either composition (C) or stemming (S)) for learning embeddings, in general, helps in achieving significantly better performance than using embeddings learned from an external, un-normalized corpus. Further, note that from Table 5.4, the space of embedded word vectors trained on the unstemmed version of an external col-

lection exhibits lower similarity to the target corpus than that trained on a processed version (composed or stemmed). For example, ρ(**Wk-U**, **Rb-U**) is lower than ρ(**Wk-S**, **Rb-S**) (same applies for σ values). This indicates that the retrieval performance obtained using embeddings learned from unprocessed collection should be lower than that obtained with the processed collection. This can be verified from Table 5.5, which shows that word2vec QE MAP values for TREC-Rb with unprocessed Wikipedia (0.2280) is lower than that of composed and stemmed Wikipedia (0.2477 and 0.2496 respectively).

An intriguing observation is that this trend of getting better IR effectiveness with word2vec embeddings learned from normalized, target collections does not held when fastText is used for learning embeddings. As we observe from Table 5.5, IR perfromance is better using fastText embeddings learned using un-normalized, external collection. One exception here is the GOV2 colelction, where higher IR performance is achieved using target collection. We speculate that the reason for this observed behavior could be the way embeddings are learned by fastText. Unlike word2vec that works at individual word level, fastText first learns the embedded representations of character n-grams before combining them to obtain vectors for words. Therefore, it performs term normalization in an *implicit* manner and applying stemming before running fastText has the effect of reducing the number of character n-grams and altering their contexts, potentially leading to noisier representations of words.

Finally, for both word2vec and fastText, the post-processing step of composing unprocessed (whole) words into indexing units (stems) yields results that lie between the two extremes of learning the embeddings on unprocessed and stemmed collections. Given that in many cases training embedings on target corpus may not be feasible due to time and resource constrains, our experiments suggests that in such cases, it may be a better trade-off to use easily available pre-trained word vectors trained on large unprocessed collections (such as Google-News) and applying compositions such as proposed in this chapter (IUC) to make use of advantages offered by term-normalization.

5.6 Summary

In this chapter, we introduced some query expansion methods based on word embedding technique. Experiments on standard text collections show that the proposed methods are performing better than unexpanded baseline model. However, they are significantly inferior than the feedback based expansion technique, such as RM3, which uses only co-occurrence based statistics to select terms and assign corresponding weights.

We also investigated the impact of term normalization and collection choices in learning word embeddings and their effect on ad hoc retrieval performance. We proposed two metrics for measuring the similarities between the embedded spaces of word vectors obtained under different settings.

Settings			MAP		
Method	**Domain**	**Normalization**	**Rb**	**WT10G**	**GOV2**
No QE			0.2355	0.1747	0.2313
Word2vec	External	Unprocessed (U)	0.2280	0.1779*	0.2397^{*T}
Word2vec	External	Composed (C)	0.2477^{*U}	0.1811^{*U}	0.2418*
Word2vec	External	Stemmed (S)	0.2496^{*UC}	0.1833^{*U}	**0.2466**^{*UC}
Word2vec	Target	Unprocessed (U)	0.2272*	0.1791*	0.2354*
Word2vec	Target	Composed (C)	0.2486^{*U}	**0.1850**^{*UE}	0.2443^{*U}
Word2vec	Target	Stemmed (S)	**0.2520**^{*UCE}	0.1834^{*U}	0.2457^{*U}
FastText	External	Unprocessed (U)	**0.2502**^{*CST}	**0.1822**^{*CST}	0.2463^{*S}
FastText	External	Composed (C)	0.2445*	0.1799*	0.2422*
FastText	External	Stemmed (S)	0.2432*	0.1773*	0.2401*
FastText	Target	Unprocessed (U)	0.2452*	0.1763*	**0.2501**^{*SE}
FastText	Target	Composed (C)	0.2473^{*U}	0.1774^{*S}	0.2489^{*E}
FastText	Target	Stemmed (S)	0.2463*	0.1760*	0.2458^{*E}

Table 5.5: Embedding based QE effectiveness with various settings on standard IR collections. A '$*$' indicates significance (paired t-test with 95% confidence) w.r.t 'No-QE'; U, C and S indicate significance w.r.t *U*nprocessed, *C*omposed and *S*temmed, respectively. For any method-normalization pair, E and T respectively indicates significant differences between the *E*xternal and *T*arget collections.

Chapter 6

Word embedding based pseudo relevance feedback *

6.1 Introduction

Standard information retrieval (IR) models for text search, as discussed in Section 2.2, are based on the mutual term independence assumption. Specifically, the occurrence of any term in the document is not considered to be dependent on the presence or absence of any other term. Incorporating representation of term dependencies within the framework of IR is generally expected to improve retrieval effectiveness. The traditional methods to incorporate terms dependencies range from representing terms in a reduced dimensional space by algebraic or probabilistic approaches [50, 83] to making use of generative models for term dependence that are based on word translation models or latent topic models [18, 158]. However, none of these methods provide a way of representing semantic relations involving multi-word *concepts*, such as the semantic equivalence between the term 'osteoporosis' and the concept expressed together by the terms 'bone' and 'decay'.

As discussed in Section 2.4, the recently developed theory of word embeddings [103], where a word, instead of being treated as a categorical variable, is transformed into a vector of real numbers, opens up a new avenue for exploring the benefits of leveraging term compositionality in the context of IR. One of the most powerful features of word embeddings is that the addition of word vectors corresponds to a semantic composition of the terms. Thus, addition of the vectors for 'bone' and 'decay' yields a vector that is in close proximity (within the embedded vector space) to the vector for 'osteoporosis'. Most of the existing research exploring the use of word embeddings for IR has involved improving the effectiveness of initial retrieval via improved document representations that incorporate semantic similarities between terms [157, 92]. For instance, the work in [157] represents documents and queries as composed vectors of their constituent words in order to compute the semantic similarity between them. The compositional characteristic of the word vectors has also been used in [178] to learn the weights of query terms during retrieval. The word embedding based query expansion method, proposed in Chapter 5, is an ad-hoc procedure where the feedback information

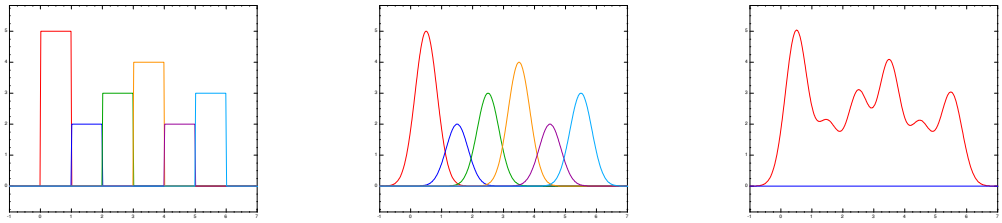
*Some material from [133] has been reused in this chapter.

is seen to be incompetent.

In this chapter, we systematically examines the use of word vector embeddings for relevance feedback (RF) and query expansion (QE). This is an interesting direction for study because the additional information about the semantic relations between potential expansion terms (as captured by the distances between the corresponding vector embeddings) may be utilized to further improve retrieval effectiveness. In fact, this constitutes the key idea behind our proposed feedback method. The existing studies, which explore word embeddings for QE are somewhat ad-hoc in nature. For example, [66, 67] simply use the k nearest neighbours of a query word vector as additional query terms for the purpose of medical and advertisement search respectively. A major limitation of these approaches is that QE is done prior to the initial retrieval. As a result, these methods have no way of utilizing information which has been shown, in general, to be useful for RF, e.g. the co-occurrence of terms in the query with those in the top ranked documents, the document similarity scores, etc. (see [38] for an axiomatic analysis of RF). The results reported in Chapter 5 are also shown to be inferior than feedback based method, specifically RM3 [86].

Despite using statistical term co-occurrences, standard RF models cannot take into consideration term composionality beyond phrases and term proximity. The goal of this study is to develop a formal RF model that provides an impeccable way to incorporate both semantic relationships and compositionality between terms along with term co-occurrence information. The key idea of the proposed method is that the embeddings of each of the query word to be treated as data points, around which kernel functions can be placed in order to construct a kernel density estimator of the underlying probability density function that generates the observed query word vectors. We argue that the proposed model provides a framework to utilize term compositionality during QE. Intuitively speaking, the (Gaussian) kernels placed around the data points (query word embeddings) ensure that a word which is semantically related to a query term (i.e. close to a query term vector in the embedding space), is assigned a high likelihood by the estimated density function. The terms having the highest probabilities may then be selected as expansion terms. A composed word vector (denoting the concept represented by the addition of two or more query term vectors) can be treated as an additional data point to control the shape of the density function in its local neighborhood. The implication of this is, terms in top ranked documents that are semantically related to the composed concept will be assigned a high probability. For example, the word 'china' will get a high probability due to the contribution from the composed word vectors of 'hong' and 'kong'; however separately, the terms may be associated with uncorrelated words (e.g. 'king' as in 'king kong').

The empirical experiments of the proposed method is conducted over several TREC benchmark datasets, specifically TREC123, TREC678 and Robust topic sets on adhoc news collections and WT10G web collection. The results show that the proposed feedback method significantly outperforms the standard relevance model (RLM). The remainder of this chapter is organized as follows. Section 6.2 reviews some related research. Section 6.3 provides a basic background of Kernel Density Estimation. Section 6.4 describes the proposed word embedding based relevance feedback method. Section 6.5 discusses how word compositionality



(a) A sample histogram (b) Gaussian at the data (c) Gaussian at the data

Figure 6.1: KDE operates by smoothing a histogram by combining Gaussians centred around the data points.

can be integrated into the proposed model. Section 6.6 presents experimental results. The chapter is concluded in Section 6.7.

6.2 Related work

Similar to our work on application of kernel density estimation, the work in [57] applies KDE in microblog retrieval for reranking the set of initially retrieved tweets based on the time-stamps of the documents. In our work, we apply KDE on the word vectors rather than on the time scale. Other work on applications of word vectors for modeling term dependence in indexed document representations is reported in Chapter 4 ([62]) for monolingual IR and [157] for cross lingual IR. Representing relative semantic distance between words derived from manually compiled resources, such as the 'WordNet', has been applied to weight query terms [113]. Contrastingly, in our work the relative distances between words is computed using vector space distance measures. Another approach to incorporate of semantic distances between terms is by the application of topic models, assuming that words in the same topic are semantically close to each other [167, 61]. However, neither of these two reported studies addresses word compositionality. Compositionality of query terms is investigated in [94], whereas [178] learns a regression model to predict optimal weights of query terms trained on available relevance judgments. A different approach to utilize the context of terms is proposed in [150], which applies principles of quantum mechanics to learn semantic representations for words and phrases to perform QE. To capture term dependencies in RF, a Markov random field based approach was reported in [102]. Similar to our work, the authors of [102] used a set of proximity-based features along with term-occurrence statistics. A limitation of [102] is that it cannot address word compositionality like the KDE RF models.

6.3 Brief introduction to Kernel Density Estimation (KDE)

In this section, we briefly describe kernel density estimation (KDE), a statistical technique for probability distribution estimation, which we utilize in our proposed RF model. Another basis of the proposed model is the RLM based feedback method [91, 86], the background of which can be found in Chapter 2.3.

Kernel density estimation (KDE) is a widely used non-parametric method of estimating the probability density function (PDF) of a continuous random variable. It is non-parametric

because it does not assume any underlying distribution for the variable. Essentially, at every pivot point, a *kernel function* is created with the datum at its centre — this ensures that the kernel is symmetric around the datum. The PDF is then estimated by adding all of these kernel functions and dividing by the number of data. Formally, kernel density estimation can be defined as follows:

> Let $X = \{x_1, \ldots, x_n\}(x_i \in \mathbb{R}^d)$ are independent, identically distributed (IID) random samples drawn from some unknown distribution P with density function f. The shape of the density function f from which these points are sampled can then be estimated using KDE following Equation (6.1).
>
> $$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{6.1}$$
>
> In Equation 6.1, $\hat{f}(x)$ is the estimated value of $f(x)$, the true density function, and $K : \mathbb{R}^d \to \mathbb{R}$ is a kernel function scaled by a bandwidth parameter $h > 0$.

Equation 6.1 can be generalized to a weighted version of KDE in which the local kernel function around the i^{th}data point is weighted by α_i, where $\sum_i \alpha_i = 1$. The weighted version of KDE can be mathematically expressed in the following way:

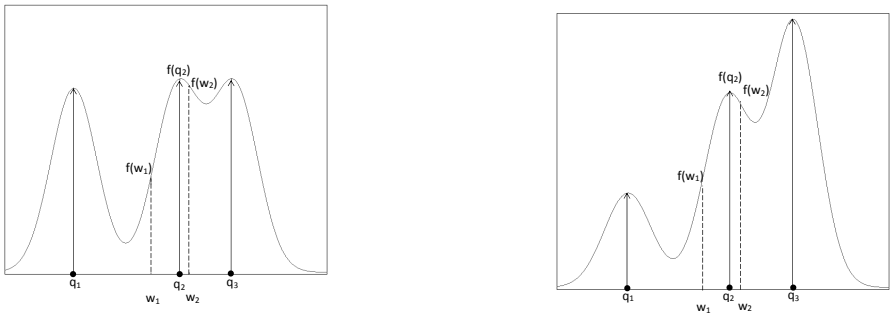
$$\hat{f}(x, \alpha) = \frac{1}{nh^d} \sum_{i=1}^{n} \alpha_i K(\frac{x - x_i}{h}) \tag{6.2}$$

It can be seen from Equation 6.1 and 6.2 that KDE is a data driven method that can be thought of as estimating the density function with the help of a *soft* or *smoothed* histograms. This is conceptually illustrated in Figure 6.1. A histogram or discrete frequency distribution (Figure 6.1a) can be approximated by a sum of Gaussian kernels, each centred around one of the data points, as shown in Figure 6.1b. For this particular example, the mixture distribution (Figure 6.1c) is obtained by applying Equation 6.1, with the kernel K defined to be a Gaussian function centred around each x_i.

In our work, as briefly discussed in Section 6.1, we treat the query terms as the observed sample points, and use KDE to estimate the latent distribution that generates the query terms. Further, the weights α_i in Equation 6.2 are set according to the co-occurrence statistics computed over the feedback documents. In the context of our work, described in Section 6.4, we establish that the KDE-based RF method is a generalized way to estimate the RLM (discussed in Section 2.3). The advantage is that while the RLM can only take into account the statistical co-occurrence of terms computed at the level of documents, the proposed KDE-based approach can also incorporate semantic relationship between the terms with the help of word vector embedding.

6.4 KDE based relevance feedback

In this section, we first establish the working principle of our proposed model as a generalization to the RLM, following which, we describe the general schematics of the proposed method, and its relation with the RLM.



(a) Unweighted one dimensional KDE

(b) Weighted one dimensional KDE

Figure 6.2: Relevance model density estimation with one dimensional weighted and unweighted KDE.

6.4.1 One dimensional KDE

The objective of this work is the same as that of RLM, which is to estimate the probability density function $P(w|R)$, where R is the generative model of relevance. Since the only observed variables of this density function are the query terms, we approximate this distribution with $P(w|Q)$ (similar to RLM [91], discussed in Section 2.3). However, RLM (see Equation 2.10) only takes into account the document-level co-occurrence of terms, ignoring all semantic relationships between the terms. Our goal is to use semantic information as further evidence for better estimation of $P(w|Q)$.

In order to go beyond binary-term-independence based co-occurrence statistics based retrieval, we use word embedding scheme, specifically *word2vec* [103]. As discussed in Section 2.4, if a word w and a query term q_i are semantically related, they will occur in similar contexts; consequently, the vector embeddings of w and q_i will be in close proximity.

As the query terms are embedded as vectors, the probability density function $P(w|R)$ can therefore be estimated with kernel density estimation. For a careful observation of the workflow, imagine the existence of a continuous probability density function $f(w)$ from which the discrete probabilities $P(w|R)$ of the RLM (Equations 2.8 and 2.9) are sampled. The shape of this *relevance density function* is controlled by a set of pivot points consisting of the embedding of the query terms. The overall idea is illustrated in Figure 6.2, which shows a sample query with three terms $\{q_1, q_2, q_3\}$. The existence of an embedded vector for each of the query terms makes it possible to define the notion of distances between them. This enables the relevance model probability distribution function to be visualized as a function pivoted around the query term vectors projected on a one dimensional line, as shown in Figure 6.2a.

In reality, each query terms are not equally important. However, the one dimensional KDE of Figure 6.2a considers each query term equally. The weighted version of the KDE for the relevance function is shown in Figure 6.2b, where in addition to the query points themselves, the shape of the function is also controlled by a set of weights $\alpha_1, \ldots, \alpha_k$, each weight $\alpha_i \in \mathbb{R}$ associated with the query term q_i. Intuitively speaking, these weights provide a mechanism to better control the shape of the density function. In general, for a query $Q = \{q_1, \ldots, q_k\}$, comprising k terms, the probability density function for relevance, denoted by $f(w, \alpha)$, is estimated by KDE, as shown in Equation 6.3. In particular, to simplify the model description, we define the kernel function as a Gaussian one, as a result of which, the scaled kernel around

the query vector data point, q_i, is defined as $K(\frac{w-q_i}{h}) = \mathcal{N}(\frac{w}{h}; \frac{q_i}{h}, \sigma)$, a notation used to refer to a normal distribution of the scaled variable $\frac{w}{h}$ parameterized by the mean $\frac{q_i}{h}$ and standard deviation σ.

$$\begin{aligned} f(w, \alpha) &= \frac{1}{kh} \sum_{i=1}^{k} \alpha_i K(\frac{w - q_i}{h}) \approx \sum_{i=1}^{k} \alpha_i \mathcal{N}(\frac{w}{h}; \frac{q_i}{h}, \sigma) \\ &= \sum_{i=1}^{k} \alpha_i \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(\mathbf{w} - \mathbf{q_i})^T(\mathbf{w} - \mathbf{q_i})}{2\sigma^2 h^2}) \end{aligned} \tag{6.3}$$

It is important to note that the vector embedding of the words of the collection makes it possible to define and compute the value of $w - q_i$ as a distance measure, e.g. the squared Euclidean distance, $(\mathbf{w} - \mathbf{q_i})^T(\mathbf{w} - \mathbf{q_i})$, between two word vectors w and q_i[2].

To see the relation of one dimensional KDE with the standard RLM, imagine that the document boundaries of the top retrieved documents are ignored, i.e., we consider the union of the set of feedback documents as a single document model (unlike the models described in Section 2.3). That is, instead of considering separate document models for each of the pseudo relevant documents, we take the union of the terms present in the top ranked documents as a urn with terms from those documents. It is easy to see that this leads us to a slightly different interpretation of the IID sampling based RLM, as shown in Equation 6.4 (compare it with Equation 2.8).

$$P(w|R) = P(w|\mathcal{M}) \prod_{i=1}^{k} P(q_i|\mathcal{M}) \tag{6.4}$$

In Equation 6.4, $\mathcal{M}$ represents the set of terms of the union of M top ranked documents retrieved for query Q. $P(w|\mathcal{M})$ and $P(q|\mathcal{M})$ are computed by MLE. Note that for a term w to have a high likelihood of being sampled from this relevance model, it needs to occur frequently in the set of top ranked documents. This is because to maximize $P(w|R)$, both $P(w|\mathcal{M})$, i.e., the normalized frequency of term w in the set of top ranked documents, and $P(q|\mathcal{M})$, i.e. the normalized frequency counts of the query terms in the set of top ranked documents, should be maximized.

With appropriate choice of the weight vector α, the density function, estimated from Equation 6.3, conforms to this characteristic, i.e., frequently co-occurring terms are assigned higher likelihoods values. Defining $\alpha_i = P(w|\mathcal{M})P(q_i|\mathcal{M})$ and substituting in Equation 6.3 yields Equation 6.5.

$$f(w) = \sum_{i=1}^{k} P(w|\mathcal{M})P(q_i|\mathcal{M}) \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(\mathbf{w} - \mathbf{q_i})^T(\mathbf{w} - \mathbf{q_i})}{2\sigma^2 h^2}) \tag{6.5}$$

For a term non-query term w, the closer w is to a query point, the higher is the value of this density function. This can be observed in Figure 6.2a which shows two terms w_1 and w_2 in the neighbourhood of a query term q_2. As w_2 is closer to q_2 than w_1, the value of the density function at w_2, i.e. $f(w_2)$ will be higher than that at w_1, i.e. $f(w_1)$.

Thus, while the traditional relevance model of Equation 2.8 (simplified to Equation 6.4) can only take into account the statistical co-occurrence of terms, the KDE-based relevance

[2] $\mathbf{w}$ (bold face) denotes the vector embedding of a word w.

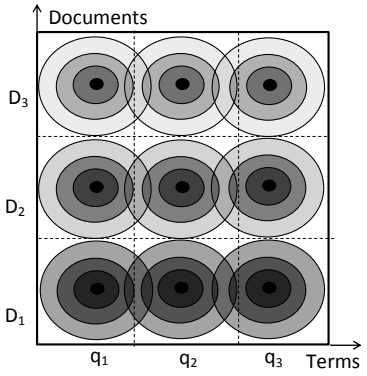


Figure 6.3: Relevance model density estimation with two dimensional KDE.

model can estimate $P(w|R)$ values by additionally considering the semantic relationships between the terms with the help of vector embeddings. The semantic relationship between words provides more reliable evidence than co-occurrence statistics because of the local contexts involved in deducing the term relationships. This makes the KDE-based model more general than the standard RLMs.

6.4.2 Two dimensional KDE

While developing the one dimensional KDE model, the set of all top ranked documents was considered as a single document model. We now generalize the one dimensional KDE to two dimensions so as to weight the contributions obtained from different documents in the ranked list separately. This idea is graphically shown in Figure 6.3. For a query consisting of k terms, $Q = \{q_1, \ldots, q_k\}$, instead of k pivot points, as in the one dimensional KDE, we now consider kM points, where M is the number of feedback documents. Each pivot point represents a query term q_i $(i = 1, \ldots, k)$ occurring in a document D_j $(j = 1, \ldots, M)$; the points being shown in a grid layout in Figure 6.3. The y-axis, in this case, represents the normalized term frequency of query terms in respective feedback documents. The x-axis corresponds to the query term vectors without any order, similar to one dimensional KDE (Figures 6.2a and 6.2b). The shape of the density function, which is now a mixture of two dimensional Gaussians, is shown as contour lines.

It can be seen from Figure 6.3 that the value of the density function is maximum at the pivot points themselves and gradually decreases with increasing neighbourhood size. This is shown with different shades of grey in Figure 6.3, where a darker shade denotes a higher value for the density function and a lighter one denotes a smaller value. In this model, unlike the one dimensional counterpart, the shape of the density function, not only depends on the distance of a term from a query word vector (along the x-axis as in the one dimensional model), but also on the likelihood of sampling that query term from a particular feedback document (along the y-axis), as seen from the different shades of the contour lines in Figure 6.3. The value of the density function decreases slowly for the document D_1, the top ranked one, as can be seen from the three contour regions of darker shades centred around the query points. The value of the density function decreases more rapidly for document D_3, as can be seen from the shades of gray lighter than those of D_2 and D_1 making up the contour regions.

Formally speaking, the kernel functions around the data points, that are used to estimate the density function, are bivariate normal distributions. A pivot point $\mathbf{x_{ij}} = (q_i, D_j)$ in this two dimensional space encapsulates the word vector for the i^{th} query word q_i and its normalized term frequency in feedback document D_j, i.e. $P(q_i|D_j)$. For simplicity, we define the covariance matrix Σ as a diagonal matrix with equal covariance σ for both the dimensions (notice in Figure 6.3 that the contours are circles instead of ellipses).

$$\begin{aligned} f(\mathbf{x}, \alpha) &= \frac{1}{kMh^2} \sum_{i=1}^{k} \sum_{j=1}^{M} \alpha_{ij} K(\frac{\mathbf{x} - \mathbf{x_{ij}}}{h}) \approx \sum_{i=1}^{k} \sum_{j=1}^{M} \alpha_{ij} \mathcal{N}(\frac{\mathbf{x}}{h}; \frac{\mathbf{x_{ij}}}{h}, \sigma) \\ &= \sum_{i=1}^{k} \sum_{j=1}^{M} \alpha_{ij} \frac{1}{2\pi\sigma^2} \exp(-\frac{1}{2\sigma^2 h^2} (\mathbf{x} - \mathbf{x_{ij}})^T (\mathbf{x} - \mathbf{x_{ij}})) \end{aligned} \tag{6.6}$$

With this simplifying assumption about the bivariate normals, the density function estimation for a point $\mathbf{x} = (w, D_j)$ is shown in Equation 6.6. Substituting the values of $\mathbf{x}$ and the pivot points $\mathbf{x_{ij}}$ in Equation 6.6, we get Equation 6.7.

$$f(\mathbf{x}, \alpha) = \sum_{i=1}^{k} \sum_{j=1}^{M} \frac{\alpha_{ij}}{2\pi\sigma^2} \exp(\frac{(\mathbf{w} - \mathbf{q_i})^2 + (P(w|D_j) - P(q_i|D_j))^2}{-2\sigma^2 h^2}) \tag{6.7}$$

In Equation 6.7, the values for the weights for the kernels, α_{ij}, are set to $P(w|D_j)P(q_i|D_j)$, analogous to the weights set for Equation 6.5. From Equation 6.7, we can observe that a term w in document D_j, denoted as the two dimensional vector $\mathbf{x}$, will get a high value of the density function if:

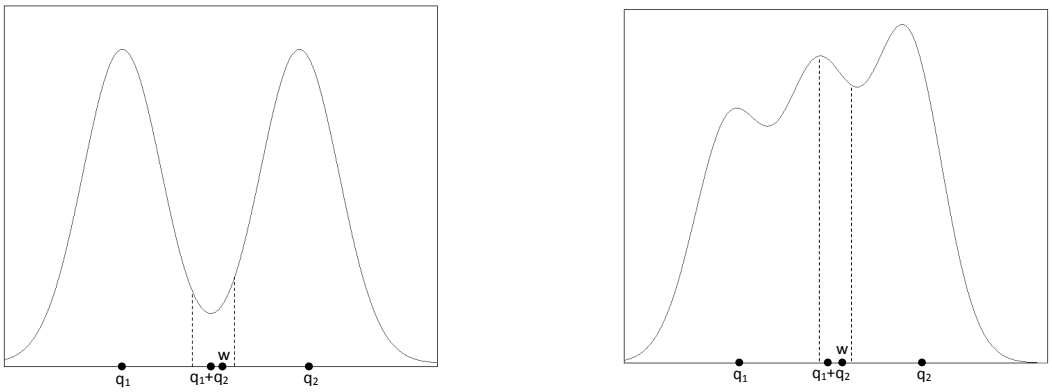
1. the embedded vector of w is close to the query terms q_i, $i = 1 \ldots, k$, or in other words, w is semantically related to the query terms; and
2. w frequently co-occurs with the query terms in each top ranked document D_j, $j = 1 \ldots, M$.

To see the relationship of the 2 dimensional KDE with the RLM, note that to maximize $P(w|R)$ in Equations 2.8 and 2.9, the MLE estimates $P(q|D)$ and $P(w|D)$ should be maximized. Maximizing both would amount to minimizing their relative difference, i.e. $P(w|D) - P(q|D)$ thus contributing to maximizing $f(\mathbf{x}, \alpha)$ in Equation 6.7.

Further, in comparison to the RLM, the two dimensional KDE method of RF is more general because in addition to the term sampling probabilities from individual documents, it can also accommodate the distances between the word vector, or in other words, the semantic similarity between a word in the set of top documents and the query terms. As the two dimensional KDE-based RF method is a more fundamental model than the one dimensional counterpart, the experimental results of only the two dimensional model is presented in this chapter.

6.5 Word compositionality

After introducing the KDE-based RF model and establishing it as a generalization of the RLM in Section 6.4, in this section, we discuss how the KDE-based RF model makes provision for the incorporation of word compositionality.



(a) Without composition

(b) With composition

Figure 6.4: Illustration of query term composition in KDE feedback for a two term query.

6.5.1 Compositions in KDE based RF

As discussed in Section 2.4, the conceptual meaning of the *composition* of two or more words can be achieved by performing an addition operation over the embedded vectors [103]. In this section, we formally present how the compositionality of term-concepts are integrated in the proposed model to better capture the semantic relationships.

Extending the set of pivot points

The proposed KDE-based RF, provides a natural way to incorporate compositionality. To elaborate on this, recall that the density estimation of the relevance model is done with a set of pivot points corresponding to the query word vectors (see Equations 6.5 and 6.7). Now, this set of pivot points can be extended to include the compositions between the query terms. The idea is illustrated in Figure 6.4. Figure 6.4a shows the shape of the one dimensional KDE function $f(w)$ for a query comprised of two terms q_1 and q_2. It can be seen how the shape of the function changes after including the pivot point $\mathbf{q_1} + \mathbf{q_2}$ in the density estimation, due to the additional kernel function around the composed word in Figure 6.4b. In fact, the sampling likelihood of a term w in a neighbourhood around the composed point $\mathbf{q_1} + \mathbf{q_2}$ increases with the addition of the composed point. This is useful when the meaning of the composed term is more focused than its constituent terms. For example, addition of the vectors for 'bone' and 'disease' yields a vector that is likely to be in close proximity to the vector for the word 'osteoporosis'. It is easy to see that the same arguments apply for the two dimensional KDE as well. The weight of the new pivot point can then be set to the average of the weights of the constituent points, composition of which formed the new point.

Deriving composed words

To get the composition of vectors, we employ the same procedure that we applied for query expansion in Section 5. Given a query of k terms, $Q = \{q_1, \ldots, q_k\}$, we obtain the set Q_c, comprised of $k-1$ composed query points, as shown in Equation 5.4. The query used for KDE RF in Equations 6.3 and 6.7 is then follow Equation 5.5, with the composed vectors of the form $\mathbf{q_i}+\mathbf{q_{i+1}}$ acting as additional data points for the density estimation. We consider only the

linear chain composition from left to right instead of considering all possible combinations, because the query terms are formulated in such a way that it is mostly unlikely that q_i and q_j $(j - i > 1)$ would refer to a meaningful composition with other words appearing in between them.

Relevance feedback with concepts

It should be noted that we do not need to find meaningful word compositions from either the whole collection, which is computationally expensive, or even from the set of feedback documents. Instead, the inherent non-parametric (data driven) characteristic of the KDE feedback ensures that the desired word composition effect can be achieved from the query side in a computationally efficient way. It is easy to see that even without explicitly extracting a term w, say 'osteoporosis', which is closest in meaning with respect to the concept represented by the composition, say 'bone decay' ($\mathbf{q_1} + \mathbf{q_2}$), the KDE feedback model is able to assign a high likelihood of relevance to such terms which are close to the composed word vector (see Equations 6.3 and 6.7, and Figure 6.4). If the composition of two query terms does not refer to a concept that differs in meaning from the constituents terms, say q_i and q_{i+1}, the additional pivot point will be close to either of the two points, $\mathbf{q_i}$ or $\mathbf{q_{i+1}}$. The shape of the density function will not change significantly in such a case. Thus, the KDE-based RF model provide a natural way to achieve concept based RF without the computational overhead of checking meaningful compositions in the collection or pseudo-relevant documents.

We summarize our proposed KDE-based RF method and the utilization of word compositions within its framework in Algorithm 2. Following the exposition in [86], we linearly interpolate the derived density function with the underlying query model (similar to Equation 2.10) to compute the overall mixture model used for our KDE-based RF based experiments.

6.6 Evaluation

The objective of our experiments is to investigate whether semantic information (relative semantic similarities between terms), when used in conjunction with co-occurrence statistics within the top ranked documents, can improve IR effectiveness. To explore the potential of the models, we report results obtained after a second-round retrieval using an expanded query. For these experiments, the expansion terms are selected using the $P(w|R)$ values obtained after linearly interpolating query model with estimated KDE (Equation (6.7)). These experimental results are reported in the following section.

6.6.1 Experimental setup

Datasets

Our experiments were conducted on several standard TREC adhoc test collections. Specifically, we used TREC disks 1, 2 and disks 4, 5 (comprising of news articles) along with the topic sets for the TREC 1, 2, 3 and 6, 7, 8 respectively. For TREC disks 4 and 5, the Robust topic set is also used for experimentation. To see the effectiveness of the proposed method on the web

Algorithm 2: 2-dim KDE Relevance Feedback

Data: Query $Q = \{q_1, \ldots, q_k\}$
Data: Set of pseudo-relevant documents $\mathcal{M} = \{D_j\}_{j=1}^{M}$
begin
  `// Modify the query` Q `into` Q_c
  $Q_c = \bigcup_{i=1}^{k-1}(\mathbf{q_i} + \mathbf{q_{i+1}})$
  $PRD \leftarrow$ All unique terms from $\mathcal{M}$
  `// Iterate over each candidate expansion term`
  for *each* $w \in PRD$ **do**
    $f(w) = 0$
    $\mathbf{w} \leftarrow LookupVec(w)$ `// get vector for` w
    `// Iterate over each pseudo-rel doc`
    for $m = 1, \ldots, M$ **do**
      `// Iterate over query terms`
      for $q \in Q_c$ **do**
        $\mathbf{q} \leftarrow LookupVec(q)$ `// vector for` q
        $\alpha \leftarrow P(w|D_m)P(q|D_m)$
        $dist_1 \leftarrow EuclideanDist(\mathbf{w}, \mathbf{q})$
        $dist_2 \leftarrow P(w|D_m) - P(q|D_m)$
        $f(w) \leftarrow f(w) + \frac{\alpha}{2\pi\sigma^2}\exp(\frac{dist_1^2 + dist_2^2}{-2\sigma^2h^2})$
  $EQ \leftarrow$ top N terms with highest $f(w)$
  Linearly interpolate query model with the terms of EQ
  Re-retrieve

collection, we experimented on TREC Web track collection WT10G, comprising web pages, along with the TREC 9 and 10 Web track topics. The dataset characteristics can be found in Table 2.1.

Baselines and implementation tools

As discussed in Section 2.3, language model based feedback method, specifically RLM [91] can be used with any retrieval methods. However, as the basics of RLM is based on language model, using LM based retrieval with RLM based qe methods is a popular practice among researchers. One of the fundamental aspect of the proposed KDE-based feedback method is based on RLM. Hence, while experimenting with the proposed method, we use language model based retrieval with both Jelinek-Mercer and Dirichlet smoothing. For the web collections WT10G, the cleaned index is used for both initial retrieval as well as for relevance feedback (see Section 3.6 for the explanation in favour of this choice). Since we argued in Section 6.4.2 that the two dimensional KDE-based RF is a generalization of the RLM, we used RLM as another baseline. More specifically, we used RM3, i.e. query mixture model with IID sampling (see Equation 2.10), as our RLM baseline method, since RM3 is reported to outperform the other variants of RLM [86, 97].

Parameter settings

There are two parameters specific to the KDE feedback models, h - the bandwidth, and σ - standard deviation of the Gaussians. Instead of exploring the full grid for these two parameters, in order to restrict the parameter settings to a tractable number of configurations, we

set the bandwidth parameter h to 1, and then varied the parameter σ to optimize the retrieval effectiveness in terms of mean average precision (MAP). Specifically, σ is varied from 0.1 to 0.9 in steps of 0.1. The intuition behind setting the value of h to 1 is that the distance between two word vectors always lies between 0 and 1. Thus it would be a reasonable choice to set h to 1, i.e., for each pivot point (query term), the points with distance of at most 0.5 $(h/2)$ from each pivot point were estimated to be similar to the query term. Along with the parameters specific to the proposed model, there are three general parameters for RLM based feedback models, $i)$ M - the number of feedback documents, $ii)$ N - the number of expansion terms, and $iii)$ ϕ - the prior weights of the query model. These parameters are set in accordance with the experiments reported in Chapter 3 and presented in Table 3.4. On a brief, we fixed M - the number of feedback documents to 10. The number of expansion terms N is varied in the range $\{50, \ldots, 90\}$ and $\{30, \ldots, 70\}$ (in steps of 10) respectively for LM-JM and LM-Dir based retrieval. The query mixing parameter ϕ is varied from 0.3 to 0.7 in steps of 0.1 respectively. The parameter associated with the retrieval models, specifically λ and μ are varied respectively in the range $\{0.7 - 0.9\}$ and $\{1000, 1500, 2000\}$. Therefore, we experiment on different values for four parameters, specifically $i)$ λ or μ, $ii)$ N, $iii)$ ϕ and $iv)$ σ. The reported results are attained after performing 5-fold cross validation on different permutations of the above parameters varied in the previously discussed range. The tuning of σ is elaborated, separately for each of the topic sets, in the next section.

For KDE feedback experiments, word vectors are embedded with the help of *word2vec* [103], trained on pre-processed content, i.e. stop words removed and stemmed with the Porter stemmer, extracted from the respective document collections. The number of dimensions for the word vectors was set to 200 and the neural network architecture used was the 'continuous bag-of-words' (cbow) model. Training was conducted with negative sampling. These were as per the parameter settings of [103].

6.6.2 Results

In this section, we report the results of the feedback based query expansion experiments conducted on each dataset with different topics. First we present the effect of parameter variation (specifically σ) in the proposed model. The retrieval results on TREC news and web topics are presented later in this section.

Effect of varying sigma in retrieval performance

The spread of the Gaussian kernels, σ, is an important parameter to tune in the KDE feedback models. Values of the kernel width which are too small results in a sparse multi-modal density function which fails to capture the semantic relations between query terms and the words in the top ranked documents. A higher width of the Gaussians smooths out the effect and the density estimates become more reliable. Similarly, values of the kernel widths which are too high tend to over-smooth the peaks of the distribution centred around the query word vectors for the proposed KDE model.

Sensitivity of KDE based feedback to the parameter σ on the individual topic sets are shown in Figure 6.5. The results reported in Figure 6.5 for all the topic sets, the feedback

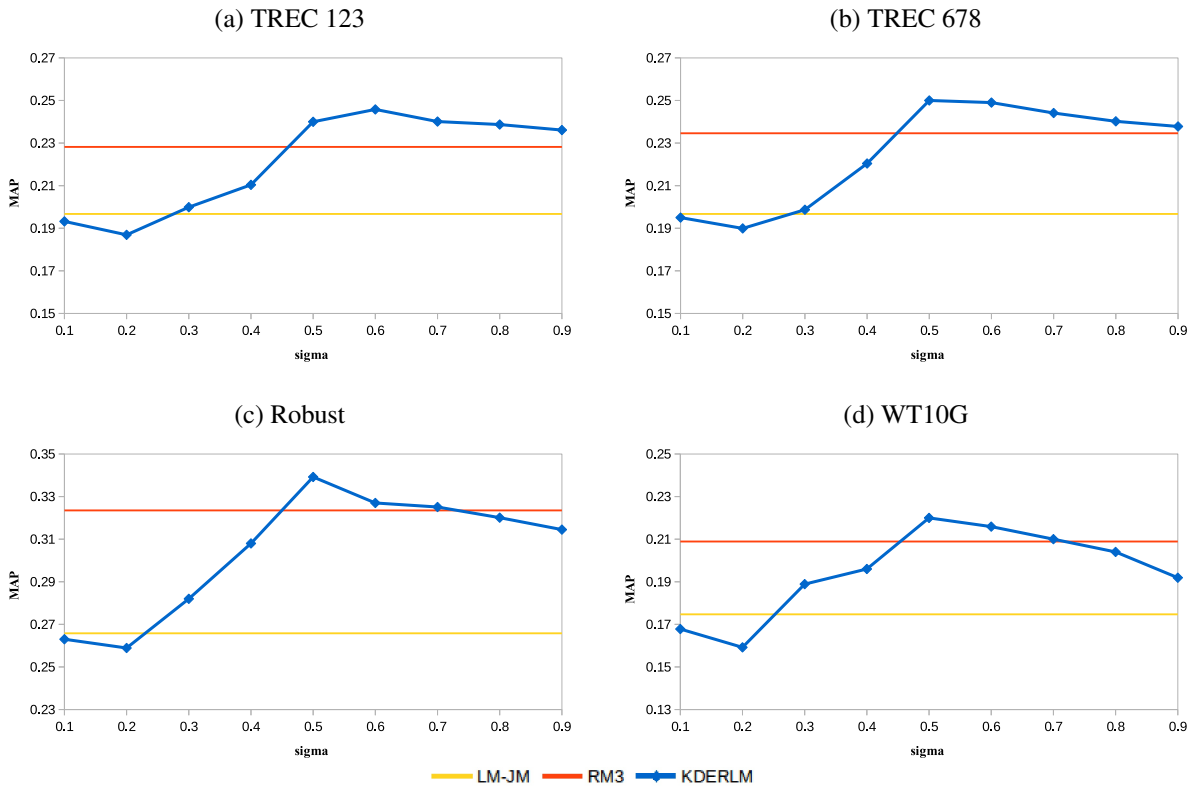


Figure 6.5: Effect of varying σ (h set to 1) for KDE feedback model.

parameters, i.e. M, N and ϕ are set to 10, 70 and 0.4 respectively. The retrieval is performed using LM-JM for all the topic sets both, for RM3 and KDERLM. From Figure 6.5, it can be seen that the best results with the proposed KDE based feedback are obtained when the value of σ is set around 0.5 to 0.7. For TREC 123, the optimal performance is achieved when σ is set to 0.6 (see Figure 6.5a). For the purpose of comparison, the figure also shows the MAP values obtained with the LM-JM and RM3 as constant lines, since these two methods do not depend on the σ parameter. For small values of σ less than 0.5, the results obtained with KDE feedback are worse than the MAP values obtained with RM3. With values of σ above 0.5, the feedback model outperforms the RM3 counterpart. However, we see an exception for robust and WT10G topic sets; for these two topic sets, the performance of the KDE based model falls short of RM3 when σ is set beyond 0.7.

Retrieval using language model

Results for all the topic sets using both language models (LM-JM and LM-Dir) are shown in Table 6.1. The following interesting observations can be made from Table 6.1.

Firstly, the KDE feedback model always outperforms RM3 and significantly outperforms[3] the baseline LM-JM with respect to all three evaluation metrics (MAP, P@5 and Recall). The performance is significantly better than the RLM based model with respect to MAP for all topics when the retrieval is performed using LM-JM. This formally shows that the probability estimates, i.e. the $P(w|R)$ values, computed with the help of the KDE models act as better

[3] paired t-test with 95% confidence measure

Query	Method	LM-JM			LM-Dir		
		MAP	P@5	Recall	MAP	P@5	Recall
TREC 123	LM	0.1967	0.4213	0.4477	0.2278	0.5213	0.4942
	RM3	0.2507*	0.4653*	0.5154*	0.2798*	**0.5320***	0.5521*
	KDERLM	**0.2661***†	**0.4798***	**0.5373***†	**0.2809**	0.5287	**0.5564***
TREC 678	LM	0.2189	0.4160	0.5300	0.2246	0.4320	0.5258
	RM3	0.2351*	0.4547*	0.5366	0.2529*	0.4640*	0.5720*
	KDERLM	**0.2536***†	**0.4601***†	**0.5520***†	**0.2566***†	**0.4739***†	**0.5742***
Robust	LM	0.2658	0.4364	0.7881	0.2875	0.5354	0.7956
	RM3	0.3309*	0.4929*	**0.8596***	0.3379*	**0.5374**	0.8594*
	KDERLM	**0.3420***†	**0.5043***	**0.8596***	**0.3426***†	0.5253	**0.8601***
WT10G	LM	0.1747	0.3152	0.6377	0.2192	0.3537	0.7086
	RM3	0.2094*	0.3394*	0.6743*	**0.2295***	0.3697*	0.7113
	KDERLM	**0.2221***†	**0.3419***	**0.6910***†	0.2293*	**0.3712***	**0.7197***

Table 6.1: Results of KDE feedback methods with QE when the retrievals are performed using LM-JelinekMercer and LM-Dirichlet. All the parameters of the KDE-based feedback method, as well as the the parameters of the baseline methods are tuning using 5-fold cross-validation. $*$ and † denote significance with respect to LM and RM3 respectively.

term selection scores than those computed with RLM, and empirically validates the hypothesis that the semantic relationships between the words, represented by the distances between the word vectors, play an important role in improving the RLM model. Thus utilizing the word vector distances as a part of density estimation can result in selection of better terms for QE.

Secondly, it is seen that the performance improvement for retrieval using LM-JM is greater than LM-Dir. This is because LM-Dir is seen to be more a powerful retrieval model than LM-JM (as seen in Chapter 3) and starting from a weak baselines produces bigger improvements (this is seen to be true for both RM3 and KDERLM). Nonetheless, in case of retrieval using LM-Dir, KDERLM is mostly seen to be producing the best performance with respect to MAP (mostly significantly). Only case where RM3 is better than KDERLM (i.e. for WT10G), the difference of MAP between KDERLM and RM3 is insignificantly small (0.0002). Except for TREC 123 and Robust topic sets, precision at rank 5 is always seen to be better than RM3.

Irrespective of the retrieval model followed, the recall has improved significantly than the baseline retrieval and always seen to the optimal for KDERLM. This shows that integrating the semantic similarity component in the RLM model pulls up relevant documents that are not seen earlier.

6.7 Summary

In this chapter, we have proposed a non-parametric statistical framework for making use of the embedded word vectors for RF. Instead of taking an ad-hoc approach to weight the co-occurrence statistics between terms of the top ranked documents and the query terms, we systematically establish this weighted co-occurrence computation with the help of kernel

density estimation (KDE). In particular, the terms of a query are considered to be the pivot points controlling the shape of the estimated probability density function. We argue that it is easy to incorporate the notion of word compositionality (the composed word referring to a different concept than those of the constituents) within our model by term wise vector addition of the query word vectors.

We evaluated our proposed feedback approach on a number of standard IR test collection, containing both news as well as web documents. Experiments demonstrate that the proposed KDE-based RF method significantly outperforms the relevance model based feedback methods, which only use statistical co-occurrences between query terms and words within top ranked documents. From the results, it can be concluded, that representing the words as vectors can effectively capture the semantic relationships between terms, and hence can act as sources of useful information for RF in addition to term co-occurrences. For RF, this semantic relationship between the terms, when systematically used inside the framework of a density estimation model, proves beneficial in modeling term relations effectively.

As a part of future work,

Chapter 7

Query performance prediction using word embedding *

7.1 Introduction

Query performance prediction (QPP) is the task of automatically estimating (without any human inputs) the quality of the search results for a query. If a query is predicted to be "difficult", i.e., the search results are estimated to be of poor-quality, this prediction may be used to selectively suggest further actions, such as query reformulation or relevance feedback, in an attempt to improve retrieval performance. Of course, retrieval quality depends on a very large number of factors. Predicting query performance, or equivalently, estimating the difficulty of a query, is therefore a challenging problem.

Researchers have attempted to identify various features that may indicate query difficulty. For example, a query may be difficult because it is ambiguous. Classic examples of ambiguous queries include 'python' (with the programming language or the snake as possible senses) and 'Paris Hilton' (referring to either the celebrity or the Hilton group of hotels in Paris). Although such queries are ambiguous, the underlying information need in the mind of the user issuing the query typically pertains to one of the possible senses of the query. The top ranked documents retrieved by an IR system may contain documents about different possible senses of the query. However, only a few among these documents, i.e., the ones which are related to the intended sense of the query, will be relevant. This can lead to poor IR effectiveness. Retrieval quality is also determined by the heterogeneity of the document collection on which the search is performed. If a user submits the query 'python' with the programming language sense of the term in mind, and the target collection consists of documents from the Computer Science domain only, search results could well be perfectly satisfactory.

Thus, the factors that affect retrieval performance for a query may be related to the actual expression of the user's information need (e.g., query term ambiguity), or to the properties of the target collection (e.g., heterogeneity). Carmel et al. (2006) present a general model for the broad classes of factors affecting retrieval performance. This model is shown in Figure 7.1 (reproduced from [29]). In the figure, Q represents a set of queries corresponding to an

*Some material from [134] has been reused in this chapter.

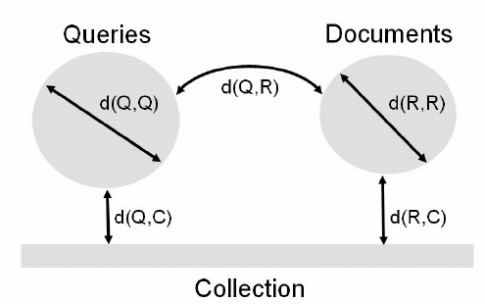


Figure 7.1: Broad classes of factors on which retrieval effectiveness usually depends (reproduced from [29]).

information need, R the set of relevant documents, C the target document collection, and $d(\cdot,\cdot)$ represents various distance measures that may be interpreted as follows.

1. $d(Q,Q)$ represents query specificity. Intuitively, the space of possible queries corresponding to a specific and well-defined information need is relatively narrow. This corresponds to a small value of $d(Q,Q)$. On the other hand, a vague or under-specified information need may be represented by diverse queries, which result in a larger value of $d(Q,Q)$. Thus, a high value of $d(Q,Q)$ suggests that retrieval results may be unsatisfactory.

2. $d(Q,C)$ measures how discriminative the terms in query Q are with respect to the collection C. Presence of informative (discriminative) terms in the query is likely to yield high retrieval effectiveness.

3. $d(R,C)$ estimates how well-separated the relevant documents are from the rest of the collection. A high value is indicative of potentially effective retrieval performance.

4. $d(R,R)$ is a measure of the topical diversity of the relevant information. A high value indicates that relevant document are diverse, and that a system may find it difficult to retrieve all the different aspects of the relevant information.

5. $d(Q,R)$ roughly corresponds to the semantic distance between the expression of the information need and the relevant content. A high value suggests that there may be a significant vocabulary gap between a query and its relevant documents. For such queries, retrieval quality is likely to be poor.

Out of the factors affecting query performance enumerated above, our work in this chapter focuses on the first one, namely $d(Q,Q)$ or *query specificity*. Query specificity is traditionally estimated by aggregating various collection-level statistics over the query terms (and possibly other terms from the top ranked documents). Examples of such collection statistics include the average or maximum of the inverse document frequency (IDF) values of query terms [75]. The rationale behind these approaches is that terms with high IDF are relatively rare in the collection, leading an IR model to easily distinguish the documents containing these terms from the rest of the collection.

Query specificity is also inversely related to the ambiguity of query terms. Thus, some existing QPP approaches attempt to quantify query ambiguity in order to estimate query difficulty [75]. These approaches generally use static resources like WordNet [104]. Specifically, an aggregate of the number of possible senses of the query terms from WordNet has been used as a measure of query ambiguity. The limitation of such WordNet-based approaches is that they are unable to capture collection-specific semantic relationships between the query terms. For example, it is not possible for a WordNet-based approach to predict that the word 'Hilton' may be less ambiguous in a collection of hotel reviews, as a result of which the query 'Paris Hilton' is less likely to be difficult in such a collection.

In this work, we propose to estimate $d(Q, Q)$ by leveraging the vector embedding of words, which potentially captures collection specific term-semantics. The embedded space of word vectors is then used to quantify the ambiguity of a query. Specifically speaking, we assume that each sense of a query term roughly corresponds to a cluster of word vectors around the neighborhood of the query term vector. Intuitively speaking, the number of clusters around the neighborhood of a query term is thus a potential indicator of the specificity of the term, i.e. lower the number of clusters higher is the specificity of the term.

Quantifying $d(Q, Q)$ based on this idea has an advantage over approaches that use static resources like WordNet. The approach dynamically adapts itself according to the diversity of the collection. The word vector representations, being collection specific, can potentially identify that 'Paris Hilton' is not a difficult query for a collection of hotel reviews. In such a collection, it is expected that the neighbouring vectors of the embedding for 'Hilton' would correspond to the hotel chain instead of the celebrity.

Research Objective. The objective of this chapter is to improve the state-of-the-art query performance prediction (QPP) effectiveness. Devising effective QPP is a step towards developing smarter search systems that, in principle, would be able to carry out alternate actions for difficult search queries, including asking the user a list of possible query reformulations, or presenting different facets of search results, to recommending documents that other people have found useful on related topics.

The rest of this chapter is organized as follows. In Section 7.2, we start by reviewing existing literature on query performance prediction and differentiate our work from existing approaches. In Section 7.3, we formally describe our word-embedding based predictor. The evaluation setup and the results of our experiments with the proposed method are presented in Section 7.4 and 7.5, respectively. Finally, in Section 7.6, we conclude the chapter with directions for future work.

7.2 Background and related work

Predicting the performance of a query has been an active research area in the information retrieval community over the last decade with many positive outcomes. According to the broad classes of features useful for QPP (see Figure 7.1), QPP approaches can be divided into two categories - a) *pre-retrieval* approaches, which only make use of $d(Q, Q)$ and $d(Q, C)$; and b) *post-retrieval* approaches, which include the other three. We start this section with a somewhat formal introduction to QPP in Section 7.2.1, and then provide a survey of QPP

approaches belonging to the pre-retrieval and post-retrieval categories in Sections 7.2.2 and 7.2.3 respectively.

7.2.1 Formal description

Following the exposition of the study in [73], a query performance predictor $\mathcal{P}$ can be formally represented as a function as shown in Equation 7.1,

$$\mathcal{P} = f_{pred}(Q, C, E, R) \rightarrow \mathbb{R}, \tag{7.1}$$

where Q is a given query, C is the target corpus of documents, E is any external resource (e.g. Wikipedia and WordNet), and R denotes a retrieval function that returns a ranked list of documents from C in response to Q. Depending on how C, E or R are used by f_{pred} in Equation 7.1, performance predictors can be divided into two categories [28].

- Pre-retrieval approaches, which only make use of query term statistics (e.g., specificity of the query terms, average number of senses from WordNet [104]) to estimate query difficulty [81, 75]. These methods do not have to rely on executing the query on an IR system to compute f_{pred}. For such predictors, Equation 7.1 can be rewritten as follows:

  $$\mathcal{P}_{pre} = f_{pred}(Q, C, E, \varnothing) \rightarrow \mathbb{R}. \tag{7.2}$$

  That is, R is set to $\varnothing$, as no retrieval function is used in these approaches; only the query terms (Q), collection statistics (C), and possibly some external resource(s) (E) are used to estimate query difficulty.
- Post-retrieval approaches, which can additionally harness information obtained from the top ranked documents retrieved for the given query (e.g score of top documents, robustness, and clarity of retrieval) [44, 29, 181, 145]. The following notation may be used for a post-retrieval predictor:

  $$\mathcal{P}_{post} = f_{pred}(Q, C, E, R) \rightarrow \mathbb{R}. \tag{7.3}$$

  That is, in these approaches, a list of documents retrieved by the method R is used in addition to the information used by pre-retrieval predictors.

In terms of the broad classes of distance measures between the different features useful for query performance prediction (schematically shown in Figure 7.1), all pre-retrieval predictors are based on some function of $d(Q, C)$ and $d(Q, Q)$. In contrast, a post-retrieval predictor is usually based on some function of $d(Q, R)$, $d(R, R)$ and $d(R, C)$ in addition to $d(Q, C)$ and $d(Q, Q)$.

7.2.2 Pre-retrieval performance predictors

Pre-retrieval performance predictors can be broadly classified into the following types according to the query term features used for computing the predictor function - a) *specificity*, b) *rank sensitivity*, c) *term relatedness*, and d) *ambiguity* [73]. Following is a brief description of each.

Specificity

The *specificity* based predictors assume make use of the assumption that discriminative terms are better able to extract out the relevant content from the whole collection. The predictor in this case is a function of $d(Q, C)$, i.e. how specific or discriminative are the query terms. As estimates of $d(Q, C)$ existing approaches usually rely on both inverse document frequency (IDF) and inverse collection frequencies (ICTF) of terms. For example, the study in [44] computes the specificity of a query as the average of the IDF values over the constituent terms. In some studies, e.g. [142], the term with the maximum IDF (*MaxIDF*) is used for estimating query specificity. Some studies, e.g. [79], compute the standard deviation of IDF from the average of ICTF values (*AvgICTF*) as the specificity-based predictor function.

The work reported in [177] proposed a number of predictors that use the collection based specificity to predict query difficulty. More specifically, three predictors, which respectively employ a) the sum of collection and each query term similarity (*SumSCQ*), b) normalized query and collection similarity (*AvgSCQ*), and c) the maximum of collection and query term similarity of the query term (*MaxSCQ*), were proposed as QPP approaches.

Rank sensitivity

In *rank sensitivity* based predictors, proposed in [177], the intention is to estimate the difficulty of a query by term weight variation across the collection. It is reasoned that if the query term weight distribution is even across all the documents of the collection, it would be difficult to distinguish the relevant documents from those non-relevant documents containing the query terms [177]. In other words, more the deviation of the query term weights, the easier is the query. The authors propose three methods, based on this conjecture, a) the sum of query term weight variations (*SumVAR*), b) normalized sum of query term weight variations (*AvgVAR*), and c) maximum of query term variations (*MaxVAR*).

Term relatedness

In contrast to the specificity and the rank sensitivity approaches that apply the estimates of $d(Q, C)$ values, the term relatedness approaches make use of the $d(Q, Q)$ estimates. More specifically, *term relatedness* based predictors, such as *AvLesk* [16], *AvPath* [120] and *AvVP* [115] use external resources (usually WordNet [104]) or co-occurrence statistics to determine the semantic distances between the query terms. The key idea behind using estimates of $d(Q, Q)$ is that semantically related queries are likely to be specific of a focused information need. On the other hand semantically unrelated query terms are likely to be indicative of queries that are underspecified.

One of the shortcomings of this predictor is that it cannot be applied for single term queries, for which the term relatedness value will be zero. Moreover, Wordnet based semantic distances are static in nature and cannot be adapted according to the domain of the collection. Although co-occurrence statistics based approaches make provision for domain adaptation, they fail to take into account the local context of words (e.g. the local context of word

embedding approaches). Our approach of computing $d(Q, Q)$ overcomes these shortcomings.

Ambiguity

The presence of an ambiguous term in the query is likely to make the information need imprecise. Since an ambiguous term has multiple senses associated with it and only one of them is likely to be relevant to the information need, it is likely that documents pertaining to each such sense would appear within the top retrieved list, as a result of which, it is difficult for a retrieval function to perform effectively [141]. It is thus reasonable to presume that an ambiguous query is usually a difficult one [28, 107].

Since our proposed QPP approach is based on the quantified ambiguity of the query, we will take a look at the predictors in this category more closely. The study [81] proposes a QPP approach (named *AvQC*), which involves computing the average inter-document similarity of all pairs of documents that contain at least one query term. A variant of AvQC, named *AvQCG*, is also proposed, which additionally includes document pairs containing all query terms in the average computation. Cosine similarity metric was used for calculating these similarities between document pairs. The rational here is that more the inter-document similarity, more coherent are the documents, which in turn implies an increase in the likelihood of the retrieval effectiveness [81].

A major drawback of this inter-document based QPP method is that it is not time scalable due to its quadratic time complexity of iterating over ${}^{N}C_{2}$ possible document pairs, where N is the number of documents containing at least one query term. For a large collection, a heuristic to overcome this problem is to sample a subset of documents pairs for the computation [74].

Some existing approaches use WordNet [104] to find out the ambiguity in a query [107]. In their proposed method, named Averaged Polysemy (*AvP*), all possible non overlapping combination of the query terms (up to word windows of length 5) are considered to check the number of senses associated with the query. The number of senses found is then averaged over all terms considered. The predictor function is then defined to be inversely related to this estimated ambiguity value of the query. In another variant, named Averaged Noun Polysemy (*AvNP*), only the noun senses are considered, unlike all the senses used in *AvP*.

A performance comparison of these pre-retrieval predictors can be found in [75]. We refer the readers to [28, 73] for a comprehensive details of different pre-retrieval performance predictors.

7.2.3 Post-retrieval performance predictors

In this section, we briefly describe some of the well known post-retrieval query performance predictors. Unlike pre-retrieval predictors, the performance of post-retrieval methods depends on the set of top-ranked documents (i.e. the set R of Equation 7.3) which in turn depends on the underlying retrieval model. As classified in [144], post-retrieval predictors can be divided into three types, based on the following criteria. 1. The clarity (a measure of distinctiveness) of the top-retrieved documents compared to the whole collection [44, 45, 5].

2. Quantifying a notion of the robustness of the top retrieved documents [28, 168, 156, 180, 10]. 3. A function (typically variance) of the retrieval scores of the top-ranked document list [145, 55, 181, 153]. Earlier QPP approaches, e.g. clarity score (CS) [44], and weighted information gain (WIG) [181]), involve computation of the estimation score with the help of the vocabulary overlap between the query Q, the top ranked documents R and the collection C (see Equation 7.3). The approach proposed in [23] applied relevance feedback information for performance predictions. Reference list based post-retrieval performance predictors are proposed in [144, 129, 130].

The Normalized Query Commitment (NQC) [143, 145] predictor measures the standard deviation of retrieval scores over the set R, i.e. the top documents retrieved for the query. The study in [145] reports that the variation within the retrieval scores has a correlation with the query performance. A high deviation is an indication that a retrieval system is able to effectively separate out the top k set from the rest of the ranked list, potentially indicating that the relevant and the non-relevant documents are well separated out. On the other hand, a low variance indicates that it is difficult for a retrieval system to separate out the top-k ones on the basis of the retrieval status values (similarity scores).

In spite of being a simple and computationally fast approach, NQC has been reported to perform well for query performance prediction, significantly outperforming the more computationally expensive predictors, e.g. the CS and WIG [145].

A somewhat similar approach (also based on variances of retrieval scores) is presented in [47]. Term overlap between multiple query expansion methods were used for QPP in [112, 72]. In recent studies, a wide range of supervised methods have been applied for the QPP task, e.g., the study in [123] applied learning to rank for effective estimation of the predictor function. For a comprehensive study of post retrieval predictors, see [28, 73].

7.2.4 Combination of predictors

It is a common practice to combine different predictors together for more effective QPP. Pre-retrieval predictors are reported to be combined using linear regression [74]. However, Kurland et al. has shown that a combination of pre-retrieval and post-retrieval predictors is more effective in practice [90]. The study in [90] reports that this combined approach significantly outperforms the stand-alone performance of both these predictors, and also outperforms a range of different pre-retrieval combinations. Similar observations are reported in [123, 168]. Formally speaking, a pre-retrieval and a post-retrieval predictor can be linearly combined as shown in Equation 7.4.

$$\mathcal{P}^{*}_{comb}(Q) = \alpha\mathcal{P}_{pre}(Q) + (1-\alpha)\mathcal{P}_{post}(Q) \tag{7.4}$$

In Equation 7.4, $\mathcal{P}_{pre}$ and $\mathcal{P}_{post}$ respectively denote pre-retrieval and post-retrieval performance predictors (Equation 7.2 and 7.3), and $\alpha \in [0,1]$ denotes a linear combination parameter.

In the context of the related literature, the focus of our work is to develop a word embedding based pre-retrieval predictor function that uses the embedded word vectors in comput-

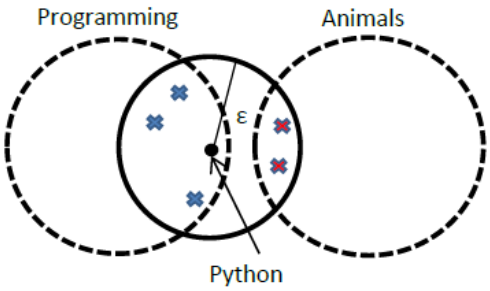


Figure 7.2: Illustrative diagram of the neighborhood of an ambiguous word with multiple senses.

ing $d(Q, Q)$ (see Figure 7.1). We then seek to linearly combine our word embedding based pre-retrieval predictor with a standard post-retrieval prediction function, namely the NQC.

To the best of our knowledge, word embedding techniques have not been used so far to predict the performance of a query without human assessment. Utilizing the semantic similarity based feature of Word embedding (that two words are embedded close to each other if they share contextual terms), in this work, we present a novel query performance predictor that is based on embedded vectors of words.

7.3 Word embedding based query performance prediction

In this section, we describe our proposed method for predicting query performance using word embeddings. Recall from Section 7.1 that our proposed approach is based on quantifying the ambiguity of a query. We first describe how embedded vectors can be leveraged to quantify query ambiguity. Later in this section, we present how to combine the estimated ambiguity of a query with a post-retrieval predictor.

7.3.1 Analysis of local neighbourhood in embedded space

A word embedding algorithm, e.g., word2vec [103], maps each word w in a document collection to a real-valued vector $\mathbf{w} \in \mathbb{R}^d$ (d being an integer) in such a way that if two words are semantically related (in the sense that they occur in similar contexts), then their embedded vectors are also similar to each other (in terms of inner product). The local neighbourhood around a word vector $\mathbf{w}$ thus comprises a set of words that are semantically related to it.

Now consider an ambiguous word such as 'python', which is associated with (at least) two very different senses - one related to the sense 'programming', and the other to that of 'animal'. Words associated with both these senses are expected to appear in the local neighbourhood around the vector for the word 'python'. Further, it is likely that the words in this neighbourhood would form two distinct clusters. This is because words in the neighbourhood associated with the 'animal' sense, such as 'snake', are likely to be of low similarity (high distance) with a word associated with the programming sense, e.g. 'jupyter'. In contrast, if each neighbouring term around the local neighbourhood of a word is similar (low distance) to every other term in the neighbourhood, it is likely that the word itself is unambiguous. This idea is schematically illustrated in Figure 7.2, which shows two distinct clusters in the

ϵ-neighbourhood[2] of the word 'python' containing word vectors related to the 'programming' and the 'animal' sense shown with blue and red crosses respectively.

As discussed in Section 7.1, from a retrieval point of view, the ambiguity of a term is determined exclusively by its occurrences within the target corpus. For example, the term 'python' would most likely be unambiguous if the target collection consisted only of zoological reports. A static lexical resource such as WordNet [104] would still characterise such a term as ambiguous, based on the number of senses generally associated with the term. In contrast, when word embeddings are generated from the target document collection, our approach is expected to have the flexibility to capture the nature of terms based on their actual occurrences in the corpus.

7.3.2 Local neighbourhood as a Gaussian Mixture Model (GMM)

We now formalise the intuition outlined in the previous section into a probabilistic model that can be used to quantify the ambiguity of query terms. Let $\mathbf{q}$ be the vector corresponding to a query term $q \in Q$ (Q being the whole query comprised of a number of terms). Let $N_\epsilon(\mathbf{q})$ denote the ϵ-neighborhood of $\mathbf{q}$, i.e.

$$N_\epsilon(\mathbf{q}) = \{\mathbf{x} : \cos^{-1}\left(\frac{\mathbf{x}\cdot\mathbf{q}}{|\mathbf{x}||\mathbf{q}|}\right) < \epsilon\}. \tag{7.5}$$

As per the word2vec objective function, the most useful notion of distance in the embedded space of word vectors is the inverse of the cosine similarity or the angle between a pair of vectors as shown in Equation 7.5.

We then consider these word vectors in each $N_\epsilon(\mathbf{q})$ as observations drawn from a Gaussian Mixture Model (GMM) of K components. Then for $\mathbf{x} \in N_\epsilon(\mathbf{q})$,

$$P(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{7.6}$$

where $\theta = \{\theta_k\}_{k=1,\ldots,K}$ is the set of parameters for the K components comprising π_k, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$ (the prior probability of selecting the kth component, the mean vector, and the covariance matrix for component k respectively).

Each Gaussian component in the neighbourhood of a query term potentially corresponds to a sense of the query term. Given the set of neighbourhood word vectors of a query term vector $\mathbf{q}$, the parameter values for the K components (i.e. the values of π_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $k = 1, \ldots, K$) can be estimated by expectation maximization (EM).

7.3.3 Quantifcation of query ambiguity post GMM estimation

Figure 7.3 shows an example scenario with the ϵ-neighborhoods of two query terms, one for an ambiguous term (left) and the other for an unambiguous term (right). Each histogram alongside the ϵ-neighborhoods shows the absolute number of points sampled from each component [3], indicative of the prior distribution $\boldsymbol{\pi}$. A skewed $\boldsymbol{\pi}$ distribution indicates a higher

[2] The ϵ-neighborhood of a word vector $\mathbf{w}$ comprises set of word vectors that are at most ϵ distance from $\mathbf{w}$.

[3] For illustrative purpose, the figure plots the absolute number of words in each Gaussian component instead of the normalized probabity values.

likelihood of selecting the most prevalent sense of a term by uniform sampling from $\boldsymbol{\pi}$, as shown in the right side histogram of Figure 7.3. On the other hand, the left side histogram of Figure 7.3 shows a more uniform $\boldsymbol{\pi}$ distribution, which indicates that it is less unlikely to select the most prevalent sense of a term.

Post EM computation of the GMM parameters, following the intuition presented in Section 7.3.1, the intention is to compute the probability of generating words from the most dominant sense of a query term, which potentially correlates well with the inherent query specificity or clarity and is inversely related to the query ambiguity. This probability is expected to be high in cases where the true number of components (senses) is a small number i.e. the query is relatively less ambiguous. Consequently, in such cases, the priors for only a small number of components are high, or in other words, the variance of the prior values is high. For example, the right-hand side histogram of Figure 7.3 shows an example scenario of a query term that is associated with one dominating sense, where it can be seen that the variance of the priors is high. We proceed as follows to compute the probability of generating words from the most dominant sense of a query term.

Let $m \in \{1 \dots K\}$ denote the most likely sense of the term q, i.e.

$$m = \underset{k=1}{\overset{K}{\operatorname{argmax}}}\, \pi_k. \tag{7.7}$$

For a query term q, we define $P_{clarity}(q)$ as the probability that the query term vector $\mathbf{q}$ is generated by the m-th component of the GMM (i.e., the component corresponding to the most likely sense of q). We call this probability $P_{clarity}$ because it is likely to be indicative of the specificity of a query term.

Speaking more precisely, we compute $P_{clarity}$ as shown in Equation 7.8.

$$\begin{aligned} P_{clarity}(q) =& \pi_m P(\mathbf{q}|\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)) \\ =& \pi_m |2\pi\boldsymbol{\Sigma}_m|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}((\mathbf{q}-\boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1}(\mathbf{q}-\boldsymbol{\mu}_m)\right) \end{aligned} \tag{7.8}$$

A careful inspection of Equation 7.8 reveals that there are two factors on which the $P_{clarity}$ value depends. These are i) π_m, which is the prior probability of choosing the most dominating sense of the query term q; and ii) $P(\mathbf{q}|\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m))$, the posterior probability of sampling the query vector $\mathbf{q}$ from the selected component of the Gaussian mixture. Informally speaking, the first component is high if the histogram of component membership of the estimated GMM is skewed.

The second component informally measures how close is a word with the most dominating sense in the neighbourhood to the query tem q. More specifically, the value of this second component, i.e. $P(\mathbf{q}|\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m))$, indicates how close is the query term vector to the mean vector $\boldsymbol{\mu}_m$ of the most prevalent sense. For an illustration, see the histogram on the right of Figure 7.3 which indicates a high posterior likelihood of sampling $\mathbf{q}$ from the most dominating component (the one central to the neighborhood). On the other hand, this posterior probability is low in the left plot of Figure 7.3, where the query term vector $\mathbf{q}$ is relatively far apart from the mean vector with the highest π_m value. The two components of Equation 7.8 together provides a predictor function for our QPP approach, which informally speaking,

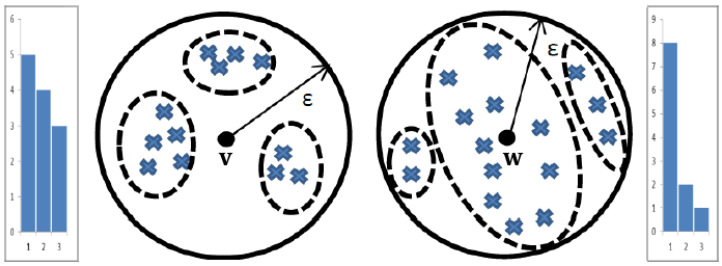


Figure 7.3: Illustrative diagram of the neighborhood of an ambiguous word with multiple senses.

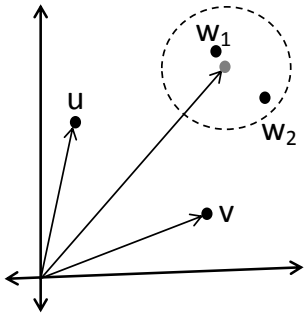


Figure 7.4: Word vector composition in an abstract two dimensional space (reproduced from [133]).

favours queries where neighbouring word vectors form a single dominating cluster and are close to this cluster centre.

Since $P_{clarity}$ likely corresponds to the specificity of a term, we estimate the ambiguity of q as the probability of the complementary event.

$$P_{ambiguity}(q) = 1 - P_{clarity}(q) = 1 - \pi_m P(\mathbf{q}|\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)) \tag{7.9}$$

The overall ambiguity of a multi-term query Q is then estimated by aggregating the individual probabilities for each query term as shown in Equation 7.10.

$$P_{ambiguity}(Q) = \prod_{q \in Q} P_{ambiguity}(q) \tag{7.10}$$

7.3.4 Composing query term vectors

For a multi-term query it is usually the case that each individual query term is associated with multiple senses, e.g., the word 'python', in isolation, may be associated with the 'animal' or the 'programming' sense. However, in conjunction with multiple query terms (e.g., *programming*), the underlying information need becomes more focused and less ambiguous. This is particularly true when the query terms constitute a phrasal concept (e.g., 'python programming').

In the context of Word embedding, the conceptual meaning of two or more words taken together can be realized by adding (called composition) the constituent word vectors [103]. Thus, if $\mathbf{w}$ is the addition of the word vectors $\mathbf{u}$ and $\mathbf{v}$, then $\mathbf{w}$ is likely to represent the concept corresponding to the composition of the words and is expected to be close to other words representing the same concept as u and v taken together, e.g. 'python' + 'programming' is expected to be close to 'jupyter'.

Figure 7.4 illustrates this idea in a two dimensional embedding space. In the figure, $\mathbf{w}$ represents the vector sum of $\mathbf{u}$ and $\mathbf{v}$. While $\mathbf{w}$ may not represent an actual vocabulary word, its neighbourhood is expected to contain the embedding of words like w_1 and w_2 that occur in similar contexts as u and v together. From Figure 7.4, we may also infer that the word w_1 is more similar to the composition than the word w_2, as $\mathbf{w_1}$ is close to $\mathbf{w}$ than $\mathbf{w_2}$. Word vector composition has been shown to improve the quality of relevance feedback [133].

To incorporate this concept-based disambiguation in our QPP approach, in addition to aggregating GMM posteriors over individual query terms (as per Equation 7.10), we also include pairwise composition of successive query term vectors, i.e. $\mathbf{q}' = \mathbf{q_i} + \mathbf{q_{i+1}}$, in the aggregation process. Formally, for a given multi-term query with n word vectors, say $Q = \{\mathbf{q_i}_{i=1}^{n}$, we generate an 'expanded' query Q_e as

$$Q_e = Q \cup \{\mathbf{q_i} + \mathbf{q_{i+1}}\}_{i=1}^{n}. \tag{7.11}$$

$P_{clarity}(q_i, q_{i+1})$ may be defined as in Equation (7.9), by using the vector sum of $\mathbf{q_i}$ and $\mathbf{q_{i+1}}$ in place of $\mathbf{q}$. We plug in Q_e instead of Q in Equation (7.10) to aggregate contributions over query term vectors along with the composed ones. Note that we avoid considering all possible terms pairs $(\mathbf{q_i}, \mathbf{q_j})_{i \neq j}$ because successive term pairs are more likely to form phrasal concepts.

7.3.5 Combination with post retrieval estimate

The $P_{ambiguity}(Q)$ (Equation 7.10) only considers the query terms for approximating the ambiguity (i.e. difficulty) of the query. However, it does not take into account the post-retrieval features that are reported to be useful for performance prediction of a query [44, 29, 181, 145]. As stated in Section 7.2, the combination of pre and post retrieval methods can outperform both the predictors [168, 123, 90].

Following Equation 7.4 (see Section 7.2), the proposed predictor $P_{ambiguity}(Q)$ is combined with an effective post-retrieval QPP method, namely Normalized Query Commitment (NQC) [145]. NQC relies on the idea that a query is likely to yield better retrieval effectiveness if the retrieval status values are skewed, or in other words, variance of the similarity scores is high. The final predictor is obtained after combining the proposed predictor with NQC using linear interpolation, as shown in Equation 7.12.

$$P^{*}_{ambiguity}(Q) = \alpha P_{ambiguity}(Q) + (1 - \alpha) NQC \tag{7.12}$$

The linear interpolation parameter α, in Equation 7.12, is the prior weight of selecting the embedding based pre-retrieval predictor. In our work, we train the parameter α on the development topic sets, and apply the tuned value on the respective test topic sets.

7.4 Evaluation

The objective of this study is to investigate whether word vector embedding of query terms can effectively capture query ambiguity, which in turn may be useful for predicting the retrieval performance of the query. To empirically validate our proposed method, we apply our

TREC disks	Query set	Query fields	Query ids	Dev set	Test set
Disks 1 & 2	TREC 2	title	101-150	✓	
	TREC 3	title	151-200		✓
Disks 4 & 5 excluding CR	TREC 6	title	301-350	✓	
	TREC 7	title	351-400		✓
	TREC 8	title	401-450		✓
WT10G	TREC 9	title	451-500	✓	
	TREC 10	title	501-550		✓

Table 7.1: Overview of datasets used for experiments on QPP

Predictor	Pre-retrieval Category			
	Specificity	Term relatedness	Rank Sensitivity	Ambiguity
SumSCQ [177]	✓			
AvgSCQ [177]	✓			
MaxSCQ [177]	✓			
AvIDF [44]	✓			
MaxIDF [142]	✓			
SumVar [177]		✓		
AvgVAR [177]		✓		
MaxVar [177]		✓		
AvLesk [16]			✓	
AvQC [81]				✓
AvQCG [81]				✓
AvP [107]				✓
AvNP [107]				✓

Table 7.2: Baseline Predictors Overview

proposed QPP method on a range of different benchmark test collections commonly used for ad hoc IR experiments. We start this section by describing the dataset characteristics and the baseline QPP methods. We then describe tuning of the QPP parameters.

7.4.1 Experimental setup

We conduct our experiments on the standard TREC ad hoc tasks and web tasks datasets (Table 7.1 presents an overview). Indexing and retrieval on these collections are conducted using Lucene[4]. Stopword removal and stemming is known to positively influence the performance of retrieval. Hence, each word was stemmed using Porter Stemmer, and stopwords were removed using the SMART stopword list[5]) before training the word2vec model.

Since our proposed QPP approach is pre-retrieval based, for a fair comparison we choose as baselines the best pre-retrieval predictors as per the reported results in [73]. Table 7.2 enlists the baseline methods for our experiments (see Section 7.2 for a brief survey of these methods). All the QPP approaches in our experiments, i.e. the proposed method and the baselines, are implemented within the Lucene framework.

[4]http://lucene.apache.org/core/

[5]http://www.lextek.com/manuals/onix/stopwords2.html

As discussed in Section 7.2, a combination of pre-retrieval and post-retrieval QPP approaches can outperform the ones which employ only the former or the latter. Consequently, in addition to using each standard pre-retrieval QPP based approach as a baseline, following the notion of Equation 7.4, we combine each pre-retrieval predictor a post-retrieval one. To eanble fair comparison, we apply the same post-retrieval predictor in combination with different pre-retrieval approaches. In particular, we use the NQC as the post-retrieval predictor.

We adopt the notation of denoting a combined predictor by inserting an asterisk over the corresponding pre-retrieval predictor's name. For example, MaxIDF* indicates the hybrid predictor that combines MaxIDF with the post-retrieval predictor NQC. The objective of the experiments is to show that our proposed QPP method combined with NQC, i.e., $P^*_{ambiguity}(Q)$ of Equation 7.12, can outperform NQC alone and other pre-retrieval predictors in combination with NQC.

We evaluate the QPP approaches in our experiments by measuring the correlation of the predicted ordering of the queries in each topic-set with the ground-truth ordering of the queries sorted by their average precision (AP) values. Following the common practice in the community [145, 29, 75, 44, 81], to evaluate QPP methods, we use Pearson's ρ and Kendall's τ [88] as the rank correlation coefficient. Both these rank correlation methods reports a value in the range $\{1, -1\}$, higher values indicating better correlation. The significance in difference is computed by the overlap in confidence interval (of the correlation coefficients) of any two methods. If there is an overlap in the interval, the performance difference is not significant.

7.4.2 Parameter setting

In order to compute the word vector based predictor function, word vectors were trained individually on each document collection (see Table 3.2). The dimensionality of the word vectors for estimating the $P_{ambiguity}(Q)$ scores was set to 200 using the *cbow* model of *word2vec* with negative sampling [103].

For our experiments, the underlying retrieval models employed to compute the NQC scores are Language Model with Jelinek Mercer smoothing (LM-JM) [175, 82], and BM25 [128]. As per standard settings in the reported literature [82], the LM-JM smoothing parameter (λ) was set to 0.6, whereas for BM25, k_1 and b, were set to 1.2 and 0.75 respectively. The number of top ranked documents used to compute the NQC scores was set to 100 as per [145].

The GMM based predictor function relies on the ϵ-neighbourhood set of words for each query term. We set the value of $\epsilon = 0.1$ in our experiments, by varying its value within a range of $[0, 0.5]$ in our initial experimental investigation. Another parameter to the GMM-based predictor is the number of components (K) of the underlying GMM that is to be used to estimate the posterior of the observed vectors in the neighbourhood around a query term. A problem with selecting a constant value of K for all queries is that the number of terms in a query can vary considerably, making it less likely that a constant value of K will be effective for all queries. Instead, we make this parameter depend on the number of vectors that we find around the ϵ-neighbourhood. Note that for different query terms, the ϵ-neighbourhood

Method	TREC 2				TREC 6				TREC 9			
	Opt. Params.		QPP Evaluation		Opt. Params.		QPP Evaluation		Opt. Params.		QPP Evaluation	
	α	γ	ρ	τ	α	γ	ρ	τ	α	γ	ρ	τ
MaxIDF*	0.1	N/A	0.3046	0.2212	0.1	N/A	0.7238	**0.4824**	0.1	N/A	0.2157	0.2336
AvgIDF*	0.1	N/A	0.3422	0.2473	0.1	N/A	**0.7645**	**0.4824**	0.1	N/A	0.4024	0.3675
SumSCQ*	0.1	N/A	0.0371	0.1478	0.1	N/A	-0.0817	0.0873	0.1	N/A	0.0595	0.0800
AvgSCQ*	0.1	N/A	0.3682	0.1771	0.1	N/A	0.2015	0.1788	0.1	N/A	0.3902	0.3348
MaxSCQ*	0.1	N/A	0.4288	0.3224	0.1	N/A	0.2463	0.2963	0.1	N/A	0.4443	**0.4198**
SumVAR*	0.1	N/A	0.1209	0.2065	0.1	N/A	0.2642	0.2424	0.1	N/A	0.1519	0.1339
AvgVAR*	0.1	N/A	0.4517	0.2408	0.1	N/A	0.5720	0.3992	0.1	N/A	0.3642	0.3299
MaxVAR*	0.1	N/A	0.4288	0.3224	0.1	N/A	0.5466	0.4449	0.1	N/A	0.3574	0.3463
AvP*	0.1	N/A	0.4465	0.2686	0.1	N/A	0.6283	0.4433	0.1	N/A	0.4359	0.2646
AvNP*	0.1	N/A	0.4238	0.2098	0.1	N/A	0.5765	0.4008	0.1	N/A	0.4920	0.2483
AvLesk*	0.2	N/A	0.4139	0.2522	0.2	N/A	0.6326	0.4727	0.2	N/A	0.3843	0.2401
AvQC*	0.1	N/A	0.4897	0.2898	0.1	N/A	0.3166	0.2686	0.1	N/A	0.2870	0.2441
AvQCG*	0.3	N/A	0.4896	0.2816	0.1	N/A	0.5263	0.3584	0.1	N/A	0.4311	0.2800
$P^*_{ambiguity}(Q)$	0.2	13	**0.5448**	**0.3437**	0.3	8	0.7417	**0.4824**	0.2	8	**0.5676**	0.3577

Table 7.3: Optimal parameter settings on the training topic sets for the hybrid approaches.

can contain a variable number of terms. For choosing K, we define a parameter γ defined as

$$\gamma = \frac{|N_\epsilon(\mathbf{q})|}{K} \tag{7.13}$$

to represent the average number of words in each cluster. This approach admits a variable number of components (K) in the GMM depending on the number of semantically similar words found in its ϵ-neighbourhood. Each component (sense) of the GMM comprises γ words on an average.

For the development topic sets, we varied γ (which is the average number of words in each component of the GMM) in the range from 5 to 15 in unit steps. To simplify the GMM estimation, we compute $P_{ambiguity}(Q)$ (Equation 7.8) score of a query by considering a GMM with uniform prior (π) and unitary covariance matrix (Σ) which results in an isotropic Gaussian Mixture. Note that an isotropic unitary Gaussian Mixture with uniform priors is equivalent to K-Means clustering of the constituent points (word vectors) [19, Chapter 9].

Another parameter of our proposed method is the linear combination weight α (see Equation 7.12) which denotes the importance of the word embedding based predictor. The parameter α is varied in the range of 0 to 1 in steps of 0.1. A value of $\alpha = 0$ degenerates the prediction measure $P^*_{ambiguity}(Q)$ to the NQC measure, whereas a value of $\alpha = 1$ solely uses the pre-retrieval word vector based prediction.

Both the parameters α and K are trained separately on each development topic set, i.e. TREC 2, TREC 6 and TREC 9 (see Table 7.1). The reason for using three different topic sets for tuning the parameters is due to the fact that the underlying document collections are different for these topic sets. The particular values of α and K that yield the optimal predictions on the development sets are then applied on the topics of the corresponding test sets. The interpolation parameter (α in Equation 7.4) for all hybrid QPP approaches are also tuned on the corresponding training topic sets, namely TREC 2, TREC 6 and TREC 9.

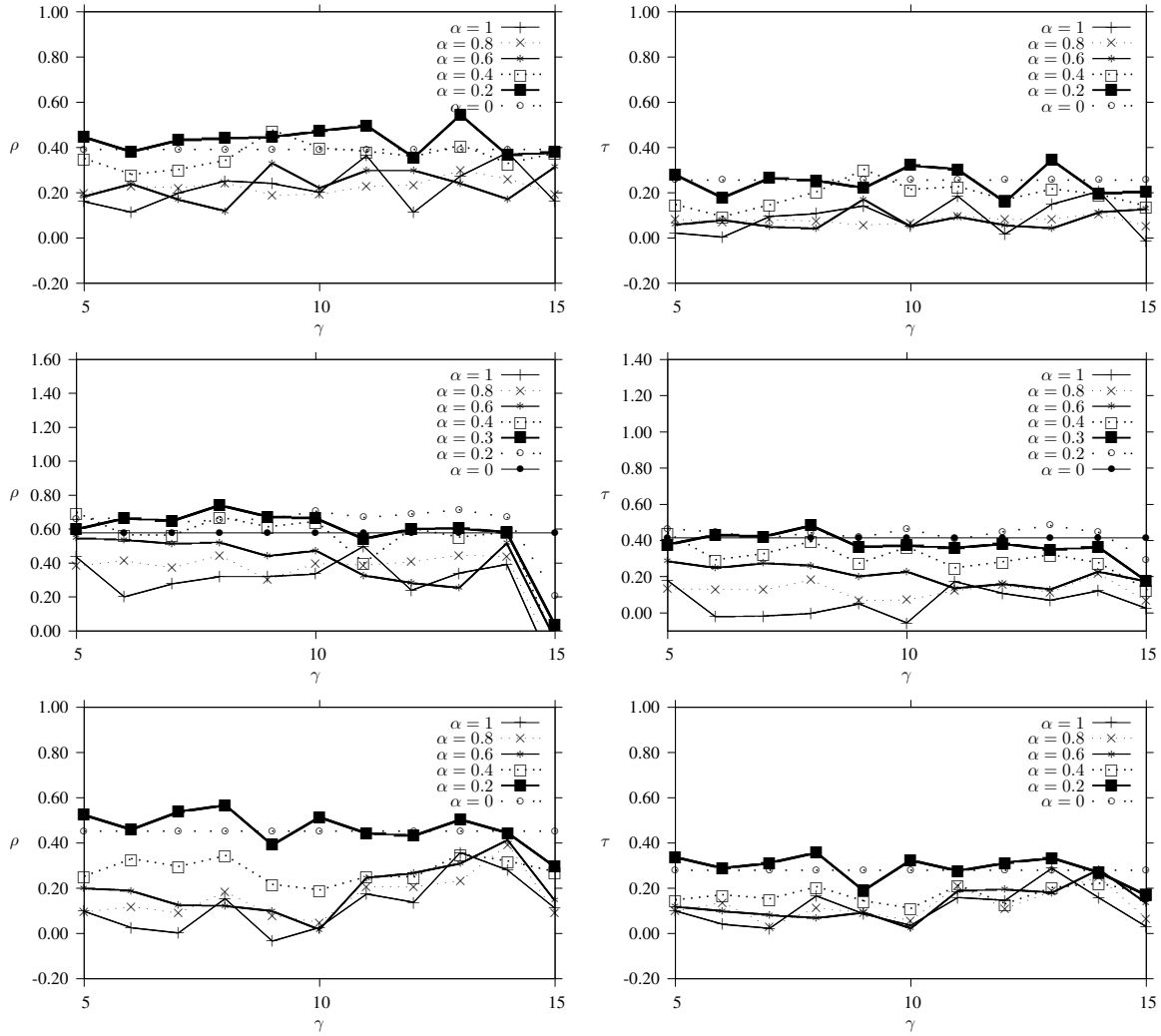


Figure 7.5: QPP sensitivity measured with Pearson's ρ (left) and Kendall's τ (right), for different values of α (Equation 7.12) and γ of Equation 7.13, on development topic sets TREC 2 (top), TREC 6 (middle) and TREC 9 (bottom).

7.5 Results and discussion

Figure 7.5 shows the QPP effectiveness obtained with different parameter settings of $P^*_{ambiguity}(Q)$ on the development topic sets. The parameters varied were α (interpolation parameter for NQC combination) and γ (the average number of points in each cluster or GMM component). Optimal results on these topic sets are also summarized in Table 7.3.

The general trends observed from the plots of Figure 7.5 and the data reported in Table 7.3 are as follows. Firstly, the proposed hybrid approach $P^*_{ambiguity}(Q)$ outperforms NQC (corresponding to the line plots for $\alpha = 0$) for values of α in the range of $[0.2, 0.3]$. This indicates that the post-retrieval predictor NQC can be considerably improved by augmenting it with information from the semantic similarities between words, which a post-retrieval predictor alone cannot capture.

Secondly, from Table 7.3 it is observed that the combination of the word vector based approach with NQC outperforms combinations of NQC with all other existing pre-retrieval approaches, except for TREC 6 (where combination of NQC with AvgIDF is best). The likely

	TREC 3		TREC 7		TREC 8		TREC 10	
	ρ	τ	ρ	τ	ρ	τ	ρ	τ
AvgIDF	0.2353	0.2490	0.4262	0.3796	0.5910	0.3518	0.3736	0.2359
MaxIDF	0.2285	0.2772	0.3524	0.2662	0.4938	0.2996	0.2219	0.1233
SumSCQ	-0.1289	-0.1380	0.0555	0.0612	0.0660	0.0612	0.2442	0.1886
AvgSCQ	0.3938	0.2963	0.4925	0.4302	0.4478	0.3159	0.2799	0.1886
MaxSCQ	0.3269	0.2107	0.4564	0.3949	0.5532	0.4384	0.3900	0.2947
SumVAR	0.0271	0.0008	0.2099	0.2522	0.5889	0.3976	0.3999	0.2637
AvgVAR	0.4042	0.3502	0.4372	0.3992	0.6475	0.4200	0.3585	0.2653
MaxVAR	0.3722	0.2655	0.4004	0.4018	0.6420	0.4371	0.4261	0.3159
AvP	0.1379	0.0567	0.3080	0.2776	0.2323	0.0994	0.0953	0.0215
AvNP	0.0290	-0.0190	0.3452	0.2975	0.2478	0.1623	0.0987	0.0261
AvgLesk	0.0297	-0.0375	0.2696	0.2509	0.0746	0.1014	0.2221	0.0896
AvQC	0.0920	0.0907	0.1032	0.0574	0.0560	0.0336	0.0472	0.0156
AvQCG	0.0932	0.0907	0.1135	0.0575	0.0563	0.0346	0.0473	0.0157
$P_{ambiguity}(Q)$	0.3222	0.2457	0.2043	0.2132	0.2132	0.1559	0.1544	0.0033
NQC	0.3146	0.1559	0.4580	0.3502	0.6717	0.4335	0.4676	0.2686
AvgIDF*	0.2782	0.2620	0.4551	0.4253	0.6315	0.3878	0.4119	0.2588
MaxIDF*	0.2862	0.3159	0.4204	0.3094	0.5815	0.3714	0.2157	0.2336
SumSCQ*	-0.1280	-0.1363	0.0602	0.0661	0.0728	0.0727	0.2455	0.1902
AvgSCQ*	0.4013	0.3061	0.5078	0.4351	0.4668	0.3241	0.2859	0.1967
MaxSCQ*	0.3347	0.2065	0.4706	0.4171	0.5739	0.4612	0.3945	0.2980
SumVAR*	0.0406	0.0155	0.2452	0.2718	0.6223	0.4253	0.4146	0.2718
AvgVAR*	0.4599	**0.3649**	0.4951	0.4433	0.6850	0.4543	0.4174	0.2996
MaxVAR*	0.4077	0.3127	0.4441	**0.4449**	0.6742	0.4659	0.4567	0.3518
AvP*	0.3129	0.1918	0.4784	0.4286	0.5221	0.2735	0.3281	0.1853
AvNP*	0.2327	0.1282	0.5141	0.4090	0.3697	0.1478	0.2634	0.1102
AvgLesk*	0.1794	0.1069	0.4095	0.3453	0.3457	0.3078	0.3292	0.1673
AvQC*	0.3224	0.1641	0.4047	0.2996	0.6393	0.4629	0.2870	0.2441
AvQCG*	0.3360	0.1771	0.4583	0.3518	0.6903	0.4335	0.4311	0.2800
$P^*_{ambiguity}(Q)$	**0.4741**	0.3648	**0.5362**	0.4344	**0.7070**	**0.4798**	**0.4936**	**0.3524**

Table 7.4: Comparisons of the word embedding based QPP method against various baselines on the test topic sets. NQC used LM-JM retrieval scores λ set to 0.6.

reason for this is that the word vector approach utilizes term relationships with the help of the semantic similarities between the words, whereas the existing ones either treat each individual query term independently (e.g. AvgIDF*), or learn the semantic relationships in a static way (e.g. AvLesk*). Although AvgIDF* outperforms $P^*_{ambiguity}(Q)$ on TREC 6, its effectiveness for the other topic sets is not satisfactory. The effectiveness of $P^*_{ambiguity}(Q)$ is relatively stable and satisfactory across the topic sets.

It can also be seen that the word embedding based QPP outperforms WordNet based QPP (AvLesk*, AvP* and AvNP*) because the word embedding based approach being trained on respective collections is better able to capture collection specific term semantics in comparison to the static semantics of a knowledge base.

After obtaining the optimal parameter settings on each development topic set, we applied these settings on each of its corresponding test set, the results being shown in Table 7.4. In this set of experiments, we test the performance of each pre-retrieval predictor in stand-alone and in combination with NQC (with LM-JM). We observe that the performance of the pre-retrieval approaches on their own (baselines and the word vector based one) are not satisfactory as

seen from the upper half of Table 7.4.

Among the pre-retrieval only QPP methods, the average idf typically performs the best. This shows that the specificity of query terms is an important criteria for QPP, which our proposed word vector based method $P_{ambiguity}(Q)$ does not make use of. However, the word-vector based approach in combination with NQC, i.e. $P^*_{ambiguity}(Q)$, outperforms (in almost all cases) combinations of the specificity based predictors, such as MaxIDF and AvgIDF, with NQC. This shows that term semantics in combination with specificity information derived from the top retrieved documents can outperform approaches that do not use term semantics.

For TREC 3 and TREC 7 topic sets, AvgVAR* and MaxVAR* attain the best τ values respectively, the τ correlation values being close to our proposed method $P^*_{ambiguity}(Q)$. Compared to all the other baseline predictors, it can be observed that performance of $P^*_{ambiguity}$ is the most consistent one.

Additionally, in order to investigate the performance of our proposed method with a different retrieval model based NQC scores, we report more results with BM25 based NQC scores in Figure 7.6. For this set of experiments, we followed the same parameter tuning methodology, i.e. train on TREC 2, TREC 6 and TREC 9 topic sets and test on the respective test topic sets (see Table 7.1). As seen from the experiments on LM-JM retrieval model (also from [168, 90, 123]), the hybrid approach of pre-retrieval predictors with a post-retrieval one (in this case NQC) outperforms the effectiveness of stand-alone pre-retrieval methods, in this set of experiments of investigating the effectiveness of our predictor with a different retrieval model based NQC scores (BM25 in this case), we only report evaluation measures for the hybrid combinations.

From Figure 7.6, we observe that the hybrid of the various QPP approaches with BM25 based NQC exhibits similar trends in observations with respect to both the evaluation metrics.

In Figure 7.6 (seen best in colour), the Pearson's and Kendall's correlations are presented as blue and red bars, respectively. We observe that the performance of the proposed embedding based predictor is in general the best in case of news collections, namely TREC 2 to TREC 8. For web collection topics (TREC 9 and 10), we see that the embedding base approach did not achieve the optimal performance. Specifically, for TREC 9, AvLesk* [16] performs marginally better than $P^*_{ambiguity}$ in terms of Pearson's correlation (ρ). However, in terms of Kendall's correlation (τ), $P^*_{ambiguity}$ achieves the best performance. For the other topic set of web collection, i.e. TREC 10, MaxVar [177] attains the best correlations.

It is worth mentioning that the performance of the other predictors are somewhat inconsistent in the sense that they perform satisfactorily for some topic sets but fail to achieve satisfactory effectiveness on others. For example, AvNP* yields the second best Pearson's correlation for TREC 7 but performs relatively poorly on other topic sets. Similar observations can be made for MaxVar*; further, its performance varies considerably for TREC 9 and TREC 10 despite the underlying document collection being the same. Our proposed word embedding based approach, on the other hand, almost always considerably outperforms the baseline methods and that too relatively more consistently than the other ones.

7.6 Summary

Estimation of query performance without using relevance judgment is an essential research area in information retrieval. Among other approaches, quantifying the ambiguity of query for this estimation is a common practice in the IR research community. In this chapter, we described a novel word embedding based query performance predictor, that estimates the ambiguity of a query. The motivation behind the word embedding based predictor is that the word vectors in the local neighborhood of the query word vectors for an ambiguous query are likely to contain terms associated with different senses. Consequently, identifying the number of senses and the semantic similarity of the query terms with the most prevalent sense is likely to be useful as a predictor function.

Being a pre-retrieval predictor in nature, the embedding based predictor does not utilize any post-retrieval information, such as the similarity score distribution of the top-ranked documents. To assimilate the post-retrieval knowledge, we further propose a hybrid framework to combine the embedding based predictor with a post-retrieval predictor. In particular, for our experiments with hybrid QPP approaches, we considered NQC, which is a relatively simple to implement and effective post-retrieval performance predictor. Experiments conducted on a number of benchmark TREC datasets demonstrate that the hybrid of the word vector based approach with NQC almost always outperforms combinations of other pre-retrieval predictors with NQC.

Note that our proposed predictor is based on using embeddings to approximate the ambiguity of the query ($d(Q,Q)$ in Figure 7.1), but the experimental results only report the combined effect of all factors on QPP. It would be interesting to isolate the role of $d(Q,Q)$, and to more directly study how well our predictor quantifies $d(Q,Q)$ (for example, by analyzing a small sample of individual queries in detail).

Another interesting direction for future work would be to use the embeddings of words in the retrieved documents to predict the performance on the basis of ambiguity of the retrieved documents ($d(R,R)$ in Figure 7.1). As a generalized predictor, we would also like to utilize the embedded vectors of the top retrieved documents in combination with vectors of neighbouring terms of the query. Finally, we would like to explore the performance of the proposed prediction approach in combination with other post-retrieval methods.

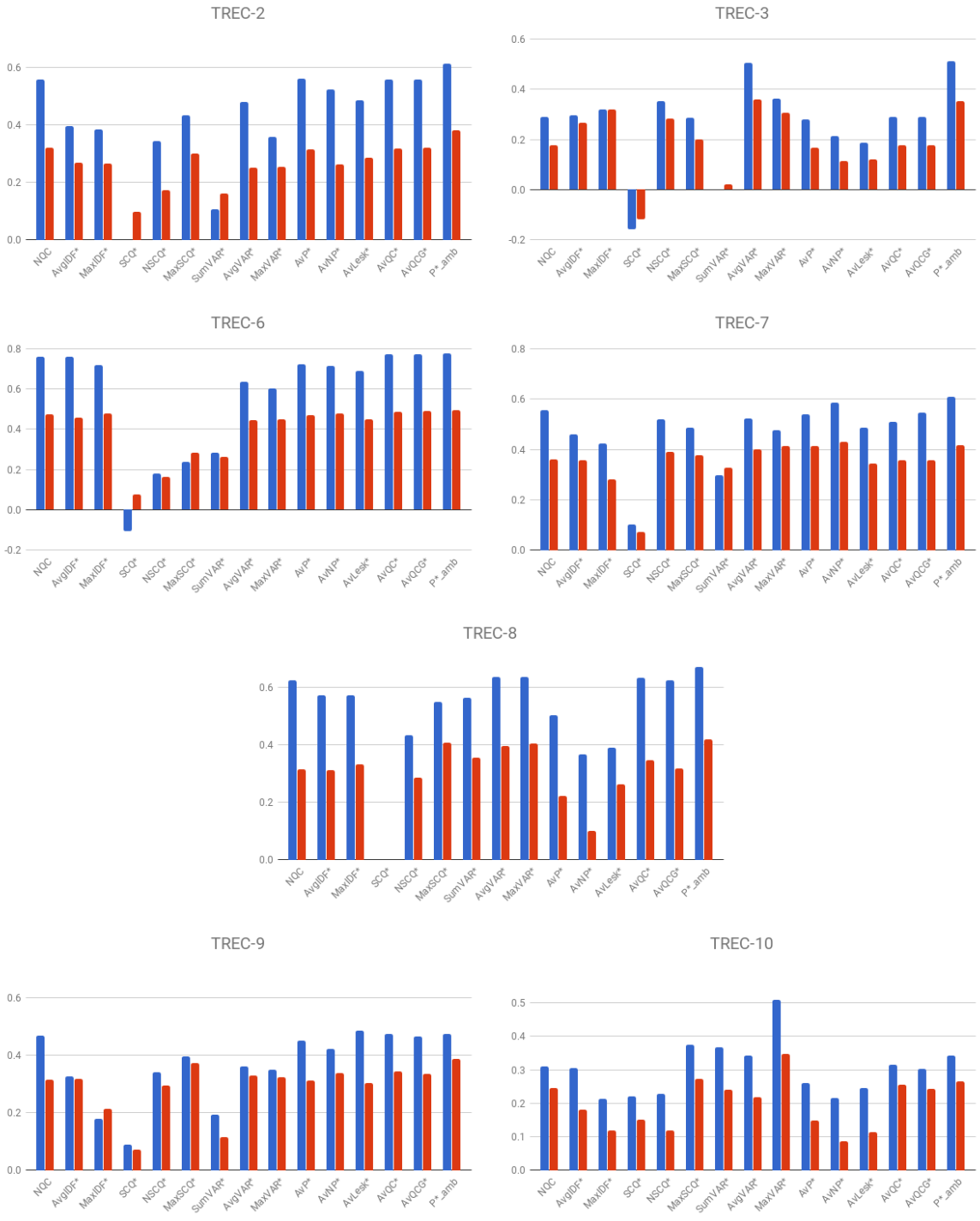


Figure 7.6: The performance of the baseline methods as well as the proposed methods in terms of performance prediction of BM25 retrieval model. In the plot, Y axis represents the correlation values obtained between the true AP of the retrieval model, and the prediction score. The blue bar and the red bar represent the Pearson's Rho (ρ) and Kendall's Tau (τ) correlation coefficient respectively.

Chapter 8

Conclusions and future work

The objective of the research presented in this thesis is to improve the effectiveness of information retrieval approaches for different tasks using word embeddings. Particularly, we focused on improving baseline and query expansion performance utilizing embedding of vocabulary terms. Additionally, we presented a novel embedding based query performance predictor that successfully beat almost all of the state-of-the-art predictors having similar nature of operation. A chapter-wise findings and conclusions are stated below:

1. *Chapter 3:* As a preliminary experiments, we setup the basic preprocessing groundwork to follow for web collections. Experimental results concluded with the following observations:

    - The performance of all retrieval models are susceptible to indexing time preprocessing applied. Most of the retrieval models are observed to be performing substantially better when the clean index is used.
    - Jelinek-Mercer smoothed language modeling exhibits mixed behaviour with regard to the presence of noise in the corpus. For collections with substantial amount of noise (e.g. ClueWeb09B), cleaning results in improved performance. However, for relatively smaller collections (e.g. WT10G), removing the noise affects the performance negatively. This can be explained using the fact that Jelinek-Mercer smoothing does not take document length into account.
    - As reported in earlier studies, Dirichlet smoothing works better than Jelinek Mercer smoothing. This is particularly true for Web corpora.
    - The noise present in the web documents is seen to having adverse effect on pseudo relevance feedback based query expansion and, using the clean content is seen to be beneficial.

2. *Chapter 4:* A generalized version of the language model for IR that combines semantic similarity with statistical similarity, is proposed in this chapter. The proposed model is based on the traditional language model with the additional feature of considering two additional cases of generating a term, i) from a document, or ii) from the collection and then changing it to another term after passing it through a noisy channel. The term

transformation probabilities of the noisy channel, in turn, are computed by making use of the distances between the word vectors embedded in an abstract space. We argue that this model has the following two fold advantage:

a) it is able to estimate how well a term fits in the context of a document (by transformation via document sampling);

b) it is able to decrease the vocabulary gap by adding other useful terms to a document (transformation via collection sampling).

Experimented on several TREC collections showed that our method significantly outperforms the standard language modeling based baseline. Possible future work will be to investigate compositionality of terms from the vector embeddings of words.

3. *Chapter 5:* In this chapter, some query expansion methods based on word embedding technique are introduced. Expansion term selection and weighting are based on embedding similarity. Given a query, the pre-retrieval expansion method QE_{pre} searches for semantically similar terms in the whole vocabulary, while its post-retrieval counterpart QE_{post} explores a set of initially retrieved documents to select expansion terms. Experiments on standard test collections show that both the proposed methods are performing better than unexpanded baseline model. The proposed methods are however seen to be inferior than feedback based expansion technique RM3, which uses only co-occurrence based statistics to select terms and assign corresponding weights. A possible future work, in this direction, is to apply the embeddings in combination with co-occurrence based techniques (e.g. RM3). In this work, we restrict the use of embeddings only to select similar words in the embedded space. Thus another possible future scope is to use the embeddings exhaustively for utilizing other aspects of the embedded forms.

   We use the proposed QE_{pre} expansion framework to observe how selection of term normalization and collection choice applied during word embeddings learning affect retrieval performance, We formalize two metrics for measuring the similarities between the embedded spaces of word vectors obtained under different settings. Using embeddings trained over different collections and under different settings for the query expansion task, we found that:

   (a) small differences in settings can lead to considerable differences in the embedded spaces of word vectors;

   (b) these differences can lead to considerable variations in the effectiveness of downstream task of ad hoc IR,

   (c) there is no 'clear winner' among the term normalization alternatives, since we observed that word2vec generally works well on a stemmed collection, whereas fastText on an unprocessed collection, and this can be attributed to the inherent characteristics of the embedding algorithms;

(d) our proposed post-processing term normalization by composing vectors of words yielding the same stem produces more stable results (lying between the two extremes of unprocessed and stemmed) across the different word embedding algorithms;

(e) composition based post term normalization can be a good choice when working with pre-trained word vectors trained with word2vec, since it outperforms the results with unprocessed word vectors on three standard IR collections;

(f) no 'clear winner' among the word vector training algorithms, since the results produced by word2vec are slightly better than those obtained with fastText on TREC-Rb and WT10G, whereas fastText word vectors produces better results on GOV2.

In future, we plan to solidify these observations to offer general best practices for a range of different neural IR methods.

4. *Chapter 6:* In a third attempt on improving retrieval effectiveness with word embedding, in this chapter, a non-parametric statistical framework is proposed which makes use of embedded word vectors for relevance feedback. The proposed model combines co-occurrence statistics with semantic similarity between terms of the top ranked documents and the query terms with the help of kernel density estimation (KDE). Particular, the query terms are considered as the pivot points controlling the shape of the estimated density function. Concept level term compositionality is also incorporated in the model by term wise vector addition of query word vectors.

   The proposed method was evaluated on standard IR test collection, with both news as well as web documents. Experiments on standard IR test collections (with both news and web documents) demonstrate that the proposed RF method based on KDE significantly outperforms RM3, which only use statistical co-occurrences between query terms and words within top ranked documents. Thus, it can be concluded that using word vectors can act as sources of useful information for relevance feedback in addition to term co-occurrences.

   As a part of future work, we would like to explore potential applications of larger units of embeddings, such as those of sentences and paragraphs, for further improving retrieval effectiveness.

5. *Chapter 7:*

   In this chapter, a novel word embedding based query performance predictor is proposed, which is based on estimating the ambiguity of a query. Specifically, the query difficulty is quantified by the number of different senses each query term is associated with. The key hypothesis behind the embedding based predictor is that, for an ambiguous query, the vectors in local neighborhood of the query word vectors are expected to contain terms that are associated with different senses. Therefore, a term with multiple senses is likely to be an ambiguous term for which, retrieval performance is supposed to be poor for simple retrieval models. Based on the above hypothesis, the proposed

predictor is pre-retrieval predictor in nature, and does not utilize any post-retrieval information. The proposed model is further extended as a hybrid model to incorporate the post-retrieval knowledge. The hybrid framework has the flexibility to be used with any post-retrieval predictors. For the experiments reported in this chapter, we considered NQC, which is a relatively simple to implement and effective post-retrieval performance predictor. The efficacy of the proposed method was tested on a number of benchmark TREC datasets which demonstrated that the hybrid method, based on word embeddings, has the potential to almost always outperform NQC and combinations of other pre-retrieval predictors with NQC.

In future, we would like to explore the performance of the proposed prediction approach in combination with other post-retrieval methods. Till now, the proposed predictor uses embeddings for approximating ambiguity of the query ($d(Q, Q)$ in Figure 7.1). Additionally, we would like to use the embeddings of words in the retrieved documents to predict the performance on the basis of ambiguity of the retrieved documents ($d(R, R)$ in Figure 7.1). As a generalized predictor, we would also like to utilize the embedded vectors of the top retrieved documents in combination with vectors of neighboring terms of the query.

Bibliography

[1] Qingyao Ai, Liu Yang, Jiafeng Guo, and W. Bruce Croft. Analysis of the paragraph vector model for information retrieval. In *Proc. of ICTIR'16*, pages 133–142, 2016.

[2] Qingyao Ai, Liu Yang, Jiafeng Guo, and W. Bruce Croft. Improving language estimation with the paragraph vector model for ad-hoc retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 869–872, New York, NY, USA, 2016. ACM.

[3] Mohannad ALMasri, Catherine Berrut, and Jean-Pierre Chevallet. A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval*, pages 709–715, Cham, 2016. Springer International Publishing.

[4] Giambattista Amati. *Probability models for information retrieval based on divergence from randomness*. PhD thesis, University of Glasgow, 2003.

[5] Giambattista Amati, Claudio Carpineto, and Giovanni Romano. Query difficulty, robustness, and selective application of query expansion. In *Advances in Information Retrieval, 26th European Conference on IR Research, ECIR 2004, Sunderland, UK, April 5-7, 2004, Proceedings*, pages 127–137, 2004.

[6] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, October 2002.

[7] Nawal Ould Amer, Philippe Mulhem, and Mathias Géry. Toward word embedding for personalized information retrieval. In *Neu-IR: The SIGIR 2016 Workshop on Neural Information Retrieval*, 2016.

[8] Yael Anava, Anna Shtok, Oren Kurland, and Ella Rabinovich. A probabilistic fusion framework. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1463–1472, New York, NY, USA, 2016. ACM.

[9] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 601–610, 2009.

[10] Javed A. Aslam and Virgiliu Pavlu. Query hardness estimation using jensen-shannon divergence among multiple scoring functions. In *Advances in Information Retrieval, 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007, Proceedings*, pages 198–209, 2007.

[11] Hosein Azarbonyad, Mostafa Dehghani, Maarten Marx, and Jaap Kamps. Time-aware authorship attribution for short text streams. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 727–730, New York, NY, USA, 2015. ACM.

[12] Javad Azimi, Adnan Alam, and Ruofei Zhang. Ads keyword rewriting using search engine results. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 3–4, New York, NY, USA, 2015. ACM.

[13] Leif Azzopardi, Yashar Moshfeghi, Martin Halvey, Rami S. Alkhawaldeh, Krisztian Balog, Emanuele Di Buccio, Diego Ceccarelli, Juan M. Fernández-Luna, Charlie Hull, Jake Mannix, and Sauparna Palchowdhury. Lucene4ir: Developing information retrieval evaluation resources using lucene. *SIGIR Forum*, 50(2):58–75, February 2017.

[14] Peter Bailey, Nick Craswell, and David Hawking. Engineering a multi-purpose test collection for web retrieval experiments. *Inf. Process. Manage.*, 39(6):853–871, November 2003.

[15] Saeid Balaneshin-kordan and Alexander Kotov. Sequential query expansion using concept graph. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 155–164, New York, NY, USA, 2016. ACM.

[16] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 805–810, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[17] P. Bellot, V. Moriceau, J. Mothe, E. SanJuan, and X. Tannier. Overview of INEX tweet contextualization 2013 track. In *Proceedings of CLEF '13*, 2013.

[18] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *SIGIR '99*, pages 222–229, 1999.

[19] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[20] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[21] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[22] Leonid Boytsov, David Novak, Yury Malkov, and Eric Nyberg. Off the beaten path: Let's replace term-based retrieval with k-nn search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1099–1108, New York, NY, USA, 2016. ACM.

[23] Olga Butman, Anna Shtok, Oren Kurland, and David Carmel. Query-performance prediction using minimal relevance feedback. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, ICTIR '13, pages 7:14–7:21, New York, NY, USA, 2013. ACM.

[24] Stefan Büttcher, Charles LA Clarke, and Ian Soboroff. The trec 2006 terabyte track.

[25] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting content structure for web pages based on visual representation. In *Proceedings of the 5th Asia-Pacific Web Conference on Web Technologies and Applications*, APWeb'03, pages 406–417, Berlin, Heidelberg, 2003. Springer-Verlag.

[26] Fei Cai and Maarten de Rijke. Learning from homologous queries and semantically related terms for query auto completion. *Information Processing & Management*, 52(4):628 – 643, 2016.

[27] Guihong Cao, Jian-Yun Nie, and Jing Bai. Integrating word relationships into language models. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 298–305, New York, NY, USA, 2005. ACM.

[28] David Carmel and Elad Yom-Tov. *Estimating the Query Difficulty for Information Retrieval*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2010.

[29] David Carmel, Elad Yom-Tov, Adam Darlow, and Dan Pelleg. What makes a query difficult? In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 390–397, New York, NY, USA, 2006. ACM.

[30] Carlos Castillo and Brian D. Davison. Adversarial web search. *Found. Trends Inf. Retr.*, 4(5):377–486, May 2011.

[31] Carlos Castillo, Debora Donato, Luca Becchetti, Paolo Boldi, Stefano Leonardi, Massimo Santini, and Sebastiano Vigna. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, December 2006.

[32] Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. Overview of the TREC 2009 web track. In *Proceedings of The Eighteenth Text REtrieval Conference, TREC 2009, Gaithersburg, Maryland, USA, November 17-20, 2009*, 2009.

[33] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Ellen M. Voorhees. Overview of the TREC 2011 web track. In *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*, 2011.

[34] Charles L. A. Clarke, Nick Craswell, and Ellen M. Voorhees. Overview of the TREC 2012 web track. In *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012*, 2012.

[35] Charles L. A. Clarke, Ian Soboroff Nick Craswell, and Gordon V. Cormack. Overview of the TREC 2010 web track. In *Proceedings of The Eighteenth Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA, 2010*, 2010.

[36] Charles LA Clarke, Nick Craswell, and Ian Soboroff. Overview of the trec 2004 terabyte track.

[37] Charles LA Clarke, Falk Scholer, and Ian Soboroff. The trec 2005 terabyte track.

[38] Stéphane Clinchant and Eric Gaussier. A theoretical analysis of pseudo-relevance feedback models. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, ICTIR '13, pages 6:6–6:13, New York, NY, USA, 2013. ACM.

[39] Stéphane Clinchant and Florent Perronnin. Aggregating continuous word embeddings for information retrieval. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality, CVSM@ACL 2013, Sofia, Bulgaria, August 9, 2013*, pages 100–109, 2013.

[40] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[41] Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Inf. Retr.*, 14(5):441–465, 2011.

[42] Nick Craswell and David Hawking. Overview of the TREC 2004 web track. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*, 2004.

[43] Nick Craswell, David Hawking, Ross Wilkinson, and Mingfang Wu. Overview of the TREC 2003 web track. In *Proceedings of The Twelfth Text REtrieval Conference, TREC 2003, Gaithersburg, Maryland, USA, November 18-21, 2003*, pages 78–92, 2003.

[44] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 299–306, New York, NY, USA, 2002. ACM.

[45] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. A framework for selective query expansion. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 236–237, New York, NY, USA, 2004. ACM.

[46] Ronan Cummins. Improved query-topic models using pseudo-relevant pólya document models. In *Proceedings of the 3rd ACM International Conference on the Theory of Information Retrieval, ICTIR 2017) Amsterdam, Netherlands, October 1-4, 2017*, 2017. To appear.

[47] Ronan Cummins, Joemon Jose, and Colm O'Riordan. Improved query performance prediction using standard deviation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 1089–1090, New York, NY, USA, 2011. ACM.

[48] Ronan Cummins, Jiaul H. Paik, and Yuanhua Lv. A pólya urn document language model for improved information retrieval. *ACM Trans. Inf. Syst.*, 33(4):21:1–21:34, 2015.

[49] Zhuyun Dai, Chenyan Xiong, and Jamie Callan. Query-biased partitioning for selective search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1119–1128, New York, NY, USA, 2016. ACM.

[50] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[51] Mostafa Dehghani, Samira Abnar, and Jaap Kamps. The healing power of poison: Helpful non-relevant documents in feedback. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2065–2068, New York, NY, USA, 2016. ACM.

[52] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 65–74, New York, NY, USA, 2017. ACM.

[53] Marco Del Tredici and Núria Bel. A word-embedding-based sense index for regular polysemy representation. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 70–78, 2015.

[54] Emanuele Di Buccio, Giorgio Maria Di Nunzio, Nicola Ferro, DK Harman, Maria Maistro, and Gianmaria Silvello. Unfolding off-the-shelf ir systems for reproducibility. In *Proc. SIGIR Workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR 2015)*, 2015.

[55] Fernando Diaz. Performance prediction using spatial autocorrelation. In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 583–590, 2007.

[56] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[57] Miles Efron, Jimmy Lin, Jiyin He, and Arjen de Vries. Temporal feedback for tweet search with non-parametric density estimation. In *Proc. of SIGIR '14*, pages 33–42, 2015.

[58] Faezeh Ensan and Ebrahim Bagheri. Document retrieval model through semantic linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 181–190, New York, NY, USA, 2017. ACM.

[59] Nicola Ferro and Gianmaria Silvello. A general linear mixed models approach to study system component effects. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 25–34, New York, NY, USA, 2016. ACM.

[60] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971, November 1987.

[61] Debasis Ganguly, Johannes Leveling, and Gareth J. F. Jones. Topical relevance model. In *AIRS '12*, pages 326–335, 2012.

[62] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 795–798, New York, NY, USA, 2015. ACM.

[63] Waseem Gharbieh, Virendra Bhavsar, and Paul Cook. A word embedding approach to identifying verb-noun idiomatic combinations. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 112–118, 2016.

[64] Krishnendu Ghosh, Plaban Kumar Bhowmick, and Pawan Goyal. Using re-ranking to boost deep learning based community question retrieval. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 807–814, New York, NY, USA, 2017. ACM.

[65] Mona Golestan Far, Scott Sanner, Mohamed Reda Bouadjenek, Gabriela Ferraro, and David Hawking. On term selection techniques for patent prior art search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 803–806, New York, NY, USA, 2015. ACM.

[66] Travis Goodwin and Sanda M. Harabagiu. UTD at TREC 2014: Query expansion for clinical decision support. In *Proc. of TREC 2014*, 2014.

[67] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. Context- and content-aware embeddings for query rewriting in sponsored search. In *Proc. of SIGIR 2015*, pages 383–392, 2015.

[68] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 55–64, New York, NY, USA, 2016. ACM.

[69] Suhit Gupta, Gail Kaiser, David Neistadt, and Peter Grimm. Dom-based content extraction of html documents. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 207–214, New York, NY, USA, 2003. ACM.

[70] Zoltan Gyongyi and Hector Garcia-Molina. Web spam taxonomy. In *First international workshop on adversarial information retrieval on the web (AIRWeb 2005)*, 2005.

[71] Donna K. Harman, editor. *Overview of TREC-1*. NIST, 1992.

[72] Maram Hasanain and Tamer Elsayed. Query performance prediction for microblog search. *Information Processing & Management*, 53(6):1320 – 1341, 2017.

[73] Claudia Hauff. Predicting the effectiveness of queries and retrieval systems. *SIGIR Forum*, 44(1):88–88, August 2010.

[74] Claudia Hauff, Leif Azzopardi, and Djoerd Hiemstra. The combination and evaluation of query performance prediction methods. In *ECIR*, volume 5478 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2009.

[75] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong. A survey of pre-retrieval query performance predictors. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 1419–1420, New York, NY, USA, 2008. ACM.

[76] David Hawking. Overview of the TREC-9 web track. In *Proceedings of The Ninth Text REtrieval Conference, TREC 2000, Gaithersburg, Maryland, USA, November 13-16, 2000*, 2000.

[77] David Hawking and Nick Craswell. Overview of the TREC-10 web track. In *Proceedings of The Tenth Text REtrieval Conference, TREC 2001, Gaithersburg, Maryland, USA, November, 2001*, 2001.

[78] Hussein Hazimeh and ChengXiang Zhai. Axiomatic analysis of smoothing methods in language models for pseudo-relevance feedback. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval, ICTIR 2015, Northampton, Massachusetts, USA, September 27-30, 2015*, pages 141–150, 2015.

[79] Ben He and Iadh Ounis. Inferring query performance using pre-retrieval predictors. In *String Processing and Information Retrieval: 11th International Conference, SPIRE 2004, Padova, Italy, October 5-8, 2004. Proceedings*, pages 43–54, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[80] Ben He and Iadh Ounis. A query-based pre-retrieval model selection approach to information retrieval. In *Coupling Approaches, Coupling Media and Coupling Languages*

for Information Retrieval, RIAO '04, pages 706–719, Paris, France, France, 2004. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

[81] Jiyin He, Martha Larson, and Maarten de Rijke. Using coherence-based measures to predict query difficulty. In *Advances in Information Retrieval , 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings*, pages 689–694, 2008.

[82] Djoerd Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, Center of Telematics and Information Technology, AE Enschede, 2000.

[83] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proc. of SIGIR'99*, pages 50–57, 1999.

[84] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, New York, NY, USA, 2013. ACM.

[85] Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

[86] Nasreen Abdul Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. Umass at TREC 2004: Novelty and HARD. In *Proc. TREC '04*, 2004.

[87] Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. Are word embedding-based features useful for sarcasm detection? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1006–1011, 2016.

[88] M. G. Kendall. *Rank Correlation Methods*. New York: Hafner Publishing Co., New York, 1955.

[89] Wessel Kraaij and Thijs Westerveld. TNO-UT at TREC-9: how different are web documents? In *Proceedings of The Ninth Text REtrieval Conference, TREC 2000, Gaithersburg, Maryland, USA, November 13-16, 2000*, 2000.

[90] Oren Kurland, Anna Shtok, Shay Hummel, Fiana Raiber, David Carmel, and Ofri Rom. Back to the roots: a probabilistic framework for query-performance prediction. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 823–832, 2012.

[91] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127, New York, NY, USA, 2001. ACM.

[92] Johannes Leveling, Debasis Ganguly, Sandipan Dandapat, and Gareth J. F. Jones. Approximate sentence retrieval for scalable and efficient example-based machine translation. In *Proc. of the COLING '12*, pages 1571–1586, 2012.

[93] Jimmy J. Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig MacDonald, and Sebastiano Vigna. Toward reproducible baselines: The open-source IR reproducibility challenge. In *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings*, pages 408–420, 2016.

[94] Christina Lioma, Jakob Grue Simonsen, Birger Larsen, and Niels Dalum Hansen. Non-compositional term dependence for information retrieval. In *Proc. of SIGIR '15*, pages 595–604, 2015.

[95] David E Losada and Leif Azzopardi. An analysis on document length retrieval trends in language modeling smoothing. *Information Retrieval*, 11(2):109–138, 2008.

[96] Petri Luukkonen, Markus Koskela, and Patrik Floréen. Lstm-based predictions for proactive information retrieval. In *Proc. of NeuIR Workshop, collocated with SIGIR*, 2016.

[97] Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1895–1898, New York, NY, USA, 2009. ACM.

[98] Saptaditya Maiti, Deba P. Mandal, and Pabitra Mitra. Tackling content spamming with a term weighting scheme. In *Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*, pages 6:1–6:5, New York, NY, USA, 2011. ACM.

[99] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[100] Jarana Manotumruksa, Craig MacDonald, and Iadh Ounis. Modelling user preferences using word embeddings for context-aware venue recommendation. In *Proc. of NeuIR Workshop, collocated with SIGIR*, 2016.

[101] Rishabh Mehrotra, Prasanta Bhattacharya, and Emine Yilmaz. Deconstructing complex search tasks: a bayesian nonparametric approach for extracting sub-tasks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 599–605, 2016.

[102] Donald Metzler and W. Bruce Croft. Latent concept expansion using markov random fields. In *Proc. of SIGIR '07*, pages 311–318, 2007.

[103] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[104] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

[105] Bhaskar Mitra. Exploring session context using distributed representations of queries and reformulations. In *Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 3–12, New York, NY, USA, 2015. ACM.

[106] Bhaskar Mitra and Nick Craswell. Query auto-completion for rare prefixes. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1755–1758, New York, NY, USA, 2015. ACM.

[107] Josiane Mothe and Ludovic Tanguy. Linguistic features to predict query difficulty. In *ACM Conference on research and Development in Information Retrieval, SIGIR, Predicting query difficulty-methods and applications workshop*, pages 7–10, 2005.

[108] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 83–92, New York, NY, USA, 2006. ACM.

[109] Kezban Dilek Onal, Ismail Sengor Altingovde, and Pinar Karagoz. Utilizing word embeddings for result diversification in tweet search. In Guido Zuccon, Shlomo Geva, Hideo Joho, Falk Scholer, Aixin Sun, and Peng Zhang, editors, *Information Retrieval Technology*, pages 366–378, Cham, 2015. Springer International Publishing.

[110] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten de Rijke, and Matthew Lease. Neural information retrieval: at the end of the early years. *Information Retrieval Journal*, Nov 2017.

[111] Jiaul H. Paik. A probabilistic model for information retrieval based on maximum value distribution. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 585–594, New York, NY, USA, 2015. ACM.

[112] Dipasree Pal, Mandar Mitra, and Samar Bhattacharya. Using multiple query expansion algorithms to predict query performance. In *Proceedings of the Fourth International Conference of Emerging Applications of Information Technology*, EAIT '14, pages 361–364, 2014.

[113] Dipasree Pal, Mandar Mitra, and Kalyankumar Datta. Improving query expansion using wordnet. *JAIST*, 65(12):2469–2478, 2014.

[114] Dae Hoon Park, Hyun Duk Kim, ChengXiang Zhai, and Lifan Guo. Retrieval of relevant opinion sentences for new products. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 393–402, New York, NY, USA, 2015. ACM.

[115] Siddharth Patwardhan and Ted Pedersen. Using wordnet-based context vectors to estimate the semantic relatedness of concepts. In *Proceedings of the EACL 2006 Workshop Making Sense of Sense-Bringing Computational Linguistics and Psycholinguistics Together*, volume 1501, pages 1–8, 2006.

[116] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[117] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281. ACM, 1998.

[118] Jay Michael Ponte. *A language modeling approach to information retrieval*. PhD thesis, University of Massachusetts, 1998.

[119] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[120] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 17–30, 1989.

[121] AFR Rahman, H Alam, and R Hartono. Content extraction from html documents. In *Proc. 1st Int. Workshop on Web Document Analysis (WDA2001)*, pages 1–4, 2001.

[122] Fiana Raiber. Adversarial content manipulation effects. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 993–993, New York, NY, USA, 2012. ACM.

[123] Fiana Raiber and Oren Kurland. Query-performance prediction: Setting the expectations straight. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 13–22, New York, NY, USA, 2014. ACM.

[124] Fiana Raiber and Oren Kurland. Kullback-leibler divergence revisited. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, pages 117–124, New York, NY, USA, 2017. ACM.

[125] Fiana Raiber, Oren Kurland, and Moshe Tennenholtz. Content-based relevance estimation on the web using inter-document similarities. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1769–1773, New York, NY, USA, 2012. ACM.

[126] Navid Rekabsaz, Mihai Lupu, and Allan Hanbury. Uncertainty in neural network word embedding: Exploration of threshold for similarity. In *Neu-IR: The SIGIR 2016 Workshop on Neural Information Retrieval*, 06 2016.

[127] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[128] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009.

[129] Haggai Roitman. An enhanced approach to query performance prediction using reference lists. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 869–872, New York, NY, USA, 2017. ACM.

[130] Haggai Roitman, Shai Erera, and Bar Weiner. Robust standard deviation estimation for query performance prediction. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, pages 245–248, New York, NY, USA, 2017. ACM.

[131] Dwaipayan Roy, Debasis Ganguly, Sumit Bhatia, Srikanta Bedathur, and Mandar Mitra. Using word embeddings for information retrieval: How collection and term normalization choices affect performance. In *Proceedings of the 27th ACM International on Conference on Information and Knowledge Management*, CIKM '18, New York, NY, USA, 2018. ACM.

[132] Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth J. F. Jones. Representing documents and queries as sets of word embedded vectors for information retrieval. 2016.

[133] Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth J.F. Jones. Word vector compositionality based relevance feedback using kernel density estimation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1281–1290, New York, NY, USA, 2016. ACM.

[134] Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth J.F. Jones. Estimating gaussian mixture models in the local neighbourhood of embedded word vectors for query performance prediction. *Information Processing and Management*, 2018. Under review.

[135] Dwaipayan Roy, Mandar Mitra, and Debasis Ganguly. To clean or not to clean: Document preprocessing and reproducibility. *ACM Journal of Data and Information Quality*, 2018. To appear.

[136] Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. Using word embeddings for automatic query expansion. In *Proc. of NeuIR Workshop, collocated with SIGIR*, 2016.

[137] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, July 2009.

[138] Bahar Salehi, Paul Cook, and Timothy Baldwin. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 977–983, 2015.

[139] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[140] Gerard Salton and Chris Buckley. Smart stopword list.

[141] Mark Sanderson. Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 142–151. Springer-Verlag New York, Inc., 1994.

[142] Falk Scholer, Hugh E. Williams, and Andrew Turpin. Query association surrogates for web search: Research articles. *JASIST*, 55(7):637–650, May 2004.

[143] Anna Shtok, Oren Kurland, and David Carmel. Predicting query performance by query-drift estimation. In *Advances in Information Retrieval Theory, Second International Conference on the Theory of Information Retrieval, ICTIR 2009, Cambridge, UK, September 10-12, 2009, Proceedings*, pages 305–312, 2009.

[144] Anna Shtok, Oren Kurland, and David Carmel. Query performance prediction using reference lists. *ACM Trans. Inf. Syst.*, 34(4):19:1–19:34, June 2016.

[145] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. Predicting query performance by query-drift estimation. *ACM Trans. Inf. Syst.*, 30(2):1–:35, May 2012.

[146] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29. ACM, 1996.

[147] Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. Document length normalization. *Inf. Process. Manage.*, 32(5):619–633, 1996.

[148] Mark D. Smucker and James Allan. An investigation of Dirichlet prior smoothings performance advantage. Technical report, CIIR, U. Mass., Amherst, 2005.

[149] Ian Soboroff. Information retrieval evaluation demo, 2013.

[150] Alessandro Sordoni, Yoshua Bengio, and Jian-Yun Nie. Learning concept embeddings for query expansion by quantum entropy minimization. In *Proc. of AAAI '14*, 2014.

[151] Nikita Spirin and Jiawei Han. Survey on web spam detection: Principles and algorithms. *SIGKDD Explor. Newsl.*, 13(2):50–64, May 2012.

[152] Fei Sun, Dandan Song, and Lejian Liao. Dom based content extraction via text density. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 245–254, New York, NY, USA, 2011. ACM.

[153] Stephen Tomlinson. Robust, web and terabyte retrieval with hummingbird searchserver at TREC 2004. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*, 2004.

[154] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 165–174, New York, NY, USA, 2016. ACM.

[155] Christophe Van Gysel, Maarten de Rijke, and Marcel Worring. Unsupervised, efficient and semantic expertise retrieval. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 1069–1079, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[156] Vishwa Vinay, Ingemar J. Cox, Natasa Milic-Frayling, and Kenneth R. Wood. On ranking the effectiveness of searches. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 398–404, 2006.

[157] Ivan Vulić and Marie-Francine Moens. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 363–372, New York, NY, USA, 2015. ACM.

[158] Xing Wei and W. Bruce Croft. LDA-based document models for ad-hoc retrieval. In *SIGIR '06*, pages 178–185, 2006.

[159] Tim Weninger, William H. Hsu, and Jiawei Han. Cetr: Content extraction via tag ratios. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 971–980, New York, NY, USA, 2010. ACM.

[160] Craig Willis. Evaluation framework - national data service - confluence, 2017.

[161] S. K. M. Wong, Wojciech Ziarko, and Patrick C. N. Wong. Generalized vector spaces model in information retrieval. In *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '85, pages 18–25, New York, NY, USA, 1985. ACM.

[162] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Modeling document novelty with neural tensor network for search result diversification. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 395–404, New York, NY, USA, 2016. ACM.

[163] Chenyan Xiong and Jamie Callan. Query expansion with freebase. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 111–120, New York, NY, USA, 2015. ACM.

[164] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proc. of SIGIR '96*, pages 4–11, 1996.

[165] Rui Yan, Yiping Song, and Hua Wu. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 55–64, New York, NY, USA, 2016. ACM.

[166] Xugang Ye, Zijie Qi, and Dan Massey. Learning relevance from click data via neural network based similarity models. In *Proceedings of the 2015 IEEE International Conference on Big Data (Big Data)*, BIG DATA '15, pages 801–806, Washington, DC, USA, 2015. IEEE Computer Society.

[167] Xing Yi and James Allan. A comparative study of utilizing topic models for information retrieval. In *Proc. of ECIR '09*, pages 29–41, 2009.

[168] Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. Learning to estimate query difficulty: Including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 512–519, New York, NY, USA, 2005. ACM.

[169] Hamed Zamani and W. Bruce Croft. Embedding-based query language models. In *Proc. of ICTIR'16*, pages 147–156, 2016.

[170] Hamed Zamani and W. Bruce Croft. Estimating embedding vectors for queries. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 123–132, New York, NY, USA, 2016. ACM.

[171] Hamed Zamani and W. Bruce Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 505–514, New York, NY, USA, 2017. ACM.

[172] ChengXiang Zhai. *Statistical Language Models for Information Retrieval - A Critical Review*, volume 2. Now Publishers Inc., Hanover, MA, USA, March 2008.

[173] Chengxiang Zhai. *Statistical Language Models for Information Retrieval*. Synthesis Lectures on HLT. Morgan and Claypool, 2009.

[174] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. SIGIR*, pages 334–342, New York, NY, USA, 2001. ACM.

[175] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, April 2004.

[176] Qi Zhang, Jihua Kang, Jin Qian, and Xuanjing Huang. Continuous word embeddings for detecting local text reuses at the semantic level. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 797–806, New York, NY, USA, 2014. ACM.

[177] Ying Zhao, Falk Scholer, and Yohannes Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Advances in Information Retrieval , 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings*, pages 52–64, 2008.

[178] Guoqing Zheng and Jamie Callan. Learning to reweight terms with distributed representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 575–584, New York, NY, USA, 2015. ACM.

[179] Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 250–259, 2015.

[180] Yun Zhou and W. Bruce Croft. Ranking robustness: a novel framework to predict query performance. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, pages 567–574, 2006.

[181] Yun Zhou and W. Bruce Croft. Query performance prediction in web search environments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 543–550, New York, NY, USA, 2007. ACM.

[182] Bin Zou, Vasileios Lampos, Shangsong Liang, Zhaochun Ren, Emine Yilmaz, and Ingemar Cox. A concept language model for ad-hoc retrieval. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 885–886, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[183] Guido Zuccon, Bevan Koopman, Peter Bruza, and Leif Azzopardi. Integrating and evaluating neural word embeddings in information retrieval. In *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, pages 12:1–12:8, New York, NY, USA, 2015. ACM.