IMPROVING DOCUMENT RANKING WITH QUERY EXPANSION BASED ON BERT WORD EMBEDDINGS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DOĞUHAN YEKE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JULY 2020

Approval of the thesis:

IMPROVING DOCUMENT RANKING WITH QUERY EXPANSION BASED ON BERT WORD EMBEDDINGS

submitted by **DOĞUHAN YEKE** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Date: 13.07.2020

Name, Surname:	Doğuhan Yeke
Signature :	

ABSTRACT

IMPROVING DOCUMENT RANKING WITH QUERY EXPANSION BASED ON BERT WORD EMBEDDINGS

Yeke, Doğuhan
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Nihan Kesim Çiçekli

July 2020, 56 pages

In this thesis, we present a query expansion approach based on contextualized word embeddings for improving document ranking performance. We employ Bidirectional Encoder Representations from Transformers(BERT) word embeddings to expand the original query with semantically similar terms. After deciding the best method for extracting word embeddings from BERT, we extend our query with the best candidate terms. As our primary goal, we show how BERT performs over the Word2Vec model, known as the most common procedure for representing terms in the vector space. After that, by leveraging the relevance judgment list, we show positive contributions of integrating tf-idf and term co-occurrence properties of terms to our query expansion system. Our experiments demonstrate that BERT outperforms Word2Vec in well-known evaluation metrics. In addition, we also conduct several experiments that address the most popular issues in information retrieval systems.

Keywords: Query Expansion, BERT, Document Ranking, Relevance Feedback

ÖZ

BERT WORD EMBEDDİNGS'İ TEMEL ALAN SORGU GENİŞLETME İLE BELGE SIRALAMASINI GELİŞTİRME

Yeke, Doğuhan
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Nihan Kesim Çiçekli

Temmuz 2020 , 56 sayfa

Bu tezde, belge sıralaması performansını iyileştirmek için bağlamsal kelime düğünlerine dayanan bir sorgu genişletme yaklaşımı sunuyoruz. Orijinal sorguyu anlamsal olarak benzer terimlerle genişletmek için Transformers (BERT) kelime düğünlerinden Çift Yönlü Enkoder Temsilleri kullanıyoruz. BERT'den kelime düğünlerini çıkarmanın en iyi yöntemine karar verdikten sonra, sorgumuzu en iyi aday terimleriyle genişletiyoruz. Birincil hedefimiz olarak, BERT'nin vektör uzayındaki terimleri temsil etmek için en yaygın prosedür olarak bilinen Word2Vec modeli üzerinde nasıl performans gösterdiğini gösteriyoruz. Bundan sonra, alaka düzeyi karar listesinden yararlanarak, terimlerin tf-idf ve terim birlikte ortaya çıkma özelliklerini sorgu genişletme sistemimize entegre etmenin olumlu katkılarını gösteririz. Deneylerimiz, BERT'nin iyi bilinen değerlendirme metriklerinde Word2Vec'ten daha iyi performans gösterdiğini göstermektedir. Ayrıca, bilgi erişim sistemlerindeki en popüler sorunları ele alan çeşitli deneyler de yapıyoruz.

Anahtar Kelimeler: Sorgu Genişletme, BERT Modeli, Doküman Sıralama, Alaka Geri Bildirimi

To my mother

ACKNOWLEDGMENTS

First, I would like to express my respect and admiration to my supervisor Prof.Dr. Nihan Kesim Çiçekli for her great guidance, positiveness and vision. This research would have never been accomplished without her knowledge and support.

Second, I would like to thank professors in our department for their help in giving advice and providing necessary technical stuff.

Finally, I would like to dedicate this thesis to my family for their love, courage and unending support. I know their support will always be with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous-Bag-of-Words
DTW	Dynamic Term Weighting
ELMo	Embeddings from Language Models
GloVe	Global Vectors for Word Representation
JSON	JavaScript Object Notation
MAP	Mean Average Precision
NER	Named Entity Recognition
NDCG	Normalized Discounted Cumulative Gain
QE	Query Expansion
PRF	Pseudo-Relevance Feedback
RJ	Relevance Judgment
SVD	Singular Value Decomposition
TC	Term Co-occurrence
Tf-Idf	Term Frequency-Inverse Document Frequency
TREC	Text Retrieval Conference

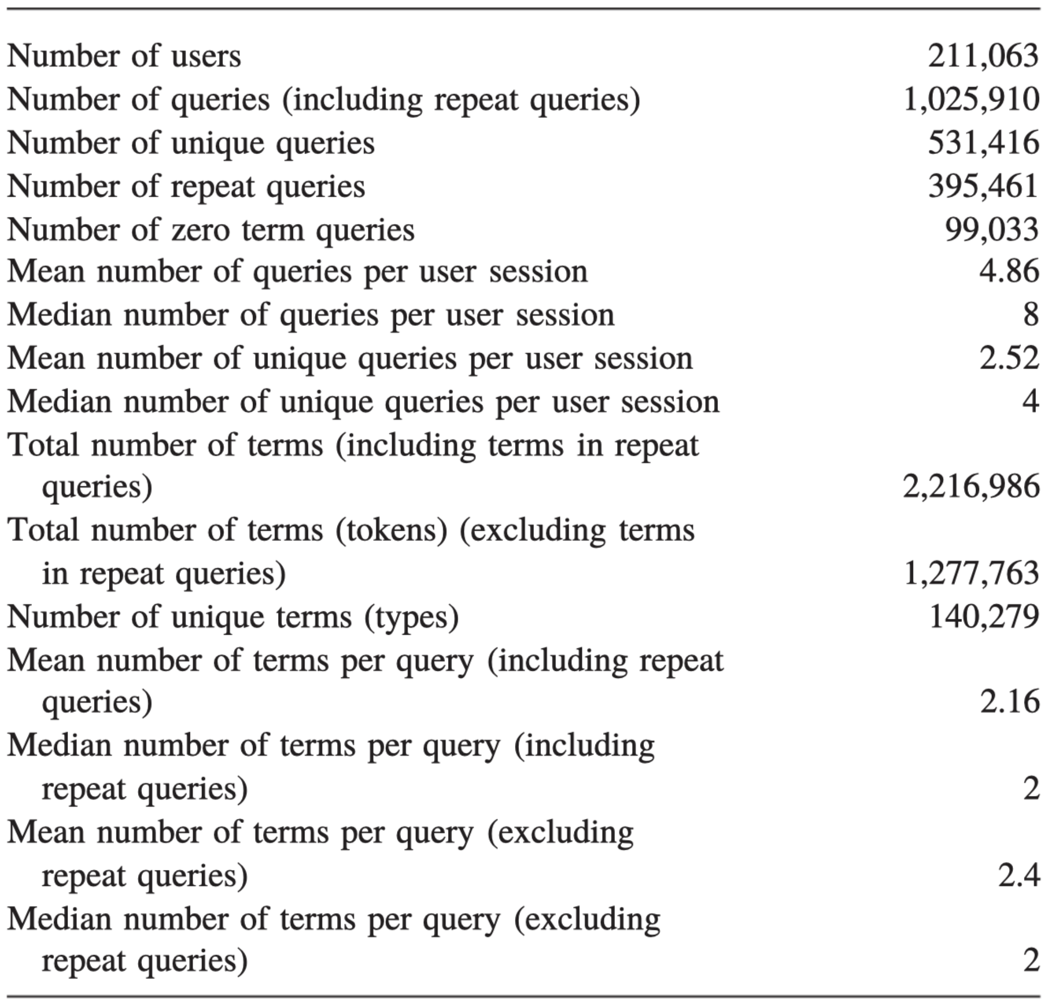
CHAPTER 1

INTRODUCTION

As the number of documents on the web continues to grow exponentially every day [8], finding necessary documents for the users in a reasonable time becomes challenging more than ever. The search should be intelligent to understand the query's context correctly. In other words, the search should be fast and efficient to satisfy the users' information need [10]. When the user provides keywords to an information retrieval system, it matches these keywords with the ones used for indexing the documents. Therefore, the query issued by the user directly affects the document ranking performance. When the user provides non-ambiguous and topic-specific keywords in their search, user satisfaction is met since the system can quickly return relevant documents to the user. However, due to the nature of natural languages, the query issued by the user can be ambiguous. In that case, returning the relevant documents to the user's information need becomes compelling.

The user needs additional tools for finding relevant documents. There is a detailed study for investigating the user behaviors in the Excite search engine [1] [34]. The analysis is comprehensively done on 200,000 users. Details about the behavior of the users are shared in Figure 1.1. By looking at the statistics, we can observe that the average query size written by the users on the web is approximately 2.4 words. As another example, there is an essential finding for representing users' behaviors in different countries [3]. According to the statistics shown in Figure 1.2, the average query size issued by the users is generally one or two words. Therefore, finding the necessary documents by only stating a few words in the vast corpus is challenging. Hence, the user needs assistance from the search engines to reach the relevant documents.

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



Number of users	211,063
Number of queries (including repeat queries)	1,025,910
Number of unique queries	531,416
Number of repeat queries	395,461
Number of zero term queries	99,033
Mean number of queries per user session	4.86
Median number of queries per user session	8
Mean number of unique queries per user session	2.52
Median number of unique queries per user session	4
Total number of terms (including terms in repeat queries)	2,216,986
Total number of terms (tokens) (excluding terms in repeat queries)	1,277,763
Number of unique terms (types)	140,279
Mean number of terms per query (including repeat queries)	2.16
Median number of terms per query (including repeat queries)	2
Mean number of terms per query (excluding repeat queries)	2.4
Median number of terms per query (excluding repeat queries)	2

Figure 1.1: Statistics of user behaviours in Excite search engine, adapted from Spink et al. [34]

.

After observing these statistics, it is obvious that users need complementary systems to reach the necessary documents. Most of the time, users may try to overcome this problem by doing several searches or paraphrasing their query [13]. However, they do not know the basic key terms to reach the relevant documents. Besides, leaving this job to users decreases user satisfaction [10]. Therefore, there must be an automatic system in search engines to solve this difficulty [7]. In order to solve this common problem, the researchers created different techniques which are:

- Stemming

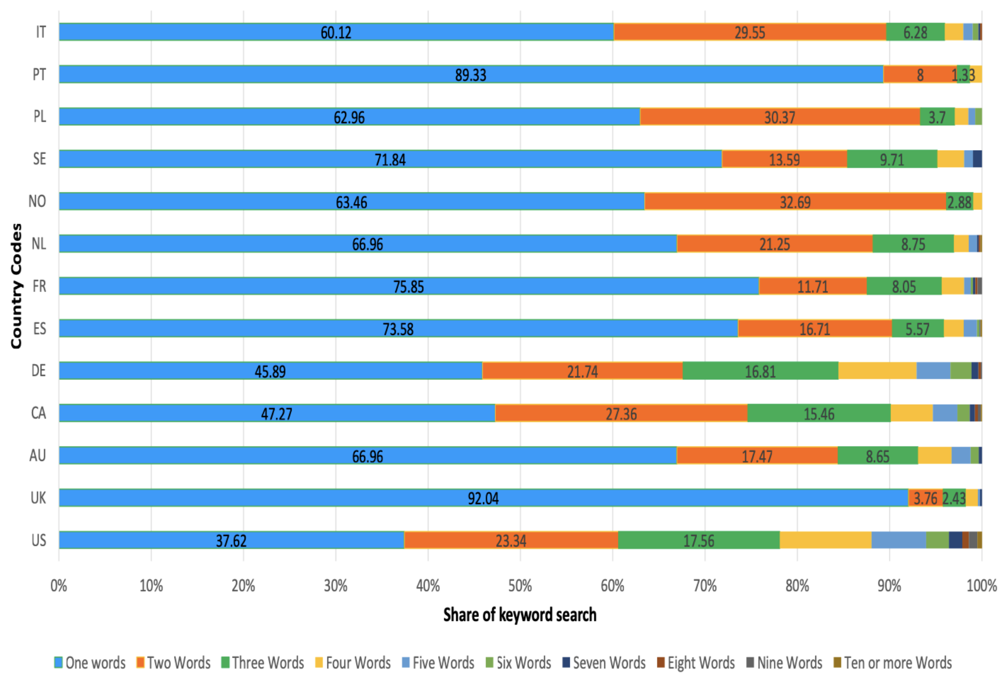


Figure 1.2: Query size statistics for each country, adapted from Azad and Deepak [3]

.

- Full-text Indexing
- Query Expansion
- Translation based model

Although each method has certain advantages and disadvantages, in this thesis, we focus on the Query Expansion technique, which is seen as the most successful method for solving the vocabulary mismatch problem among all the methods.

Query expansion is reformulating the query to improve document ranking performance by adding new terms. However, finding the correct terms to expand the query is a highly challenging task. Since the query size is inadequate and words can be equivocal, comprehending the context of the query and intention of the user is near impossible. For example, when the user enters a query "bank", retrieving the desired documents for that specific user in the large size of the web is a perfect storm. The word "bank" can be a financial organization or land alongside or sloping down to a river or lake. We need to look for the word's surroundings to learn the context of the words. Therefore, in order to reformulate the query, query understanding is essential.

1.1 Motivation and Problem Definition

In this thesis, we focus on the vocabulary mismatch problem, which occurs when the keywords issued by the users do not match with indexed terms. Query expansion system bridges the gap between query terms and relevant documents. We aim to improve the document ranking performance of search results by constructing a query expansion system. We achieve this by employing Bidirectional Encoder Representations from Transformers (BERT) [11] contextualized word embeddings for finding the best candidate terms.

Our motivation is to use BERT for constructing word embeddings. Until today, Word2Vec was the most popular model for representing word embeddings. In recent days, however, deep learning models become prevalent in natural language processing. Although BERT is successfully adapted in different studies, its usage in query expansion has not been examined. In our query expansion system, we aim to use BERT's word embeddings on finding relevant terms. We also analyze the performance of BERT over the Word2Vec model.

We compare our query expansion system with a recent work [2], which employs Word2Vec [21] model instead of BERT. In this thesis, we set the same environment and conduct several experiments to measure the BERT and Word2Vec model's performance difference. According to our findings, the BERT-based query expansion system performs better compared to using Word2Vec static word embeddings. Besides, integrating relevance judgment information to our system, we show the contributions of using tf-idf and term co-occurrence in term weighting step. In the end, we share the results of all methods that we discuss throughout the study.

1.2 Contributions and Novelties

This thesis contributes to the literature of query expansion systems from a few perspectives.

Our contributions are as follows:

- extracting word embedding representations from BERT stacked layers.
- showing BERT's performance over the Word2Vec model on word embeddings.
- improving document ranking performance in the TREC 2018 Common Core Track.

1.3 The Outline of the Thesis

The rest of this thesis organized as follows;

Chapter 2 summarizes the relevant literature on query expansion systems by examining different word embeddings models and query expansion techniques.

Chapter 3 shows our base work, which uses BERT in word embedding representation to construct a query expansion mechanism. We also provide the details of extracting word embeddings from BERT in this chapter. Following that, we compare the performance of BERT with Word2Vec model.

Chapter 4 represents our extended work, which integrates relevance judgments to our system.

Chapter 5 states our experiments in comparing different methods. We present our results on the TREC 2018 Common Core Track using different evaluation metrics.

Chapter 6 presents the conclusion of our work and reveals future work.

CHAPTER 2

RELATED WORK

The history of Query Expansion(QE) systems dates back to 1960s [20]. Since then, the researchers' quests to discover new methods in query expansion never slowed down [3]. Different approaches have been proposed for semantical query expansion. Among earlier attempts, the ontology-based approach is considered a successful example of semantically extending the query [4]. In recent years, however, the query expansion literature is mostly directed into word embedding techniques. In this chapter, we first present the relevant literature on word embeddings by analyzing different models. We also provide the steps of a generic query expansion system.

2.1 Word Embeddings

Word embedding is a way of representing the term in a continuous vector space [22]. If we had a small size of a dictionary, we could create a vector with the dictionary's size and allocate a unique index to represent that term. However, in almost all languages, the words in the dictionary are so many that words can not be represented using a one-hot vector. Therefore, different models are created to represent words. The most successful ones are Word2Vec [21], Glove [29], Relevance-based embeddings, ELMo [30] and BERT [11].

2.1.1 Word2Vec

In Word2Vec, there are two iteration-based models, Continuous-Bag-of-Words (CBOW) and Skip-Gram for finding the vector representations [21]. While the former uses surrounding words to estimate the meaning of the chosen word, the latter uses the center word to estimate the surroundings of that word. These models inspired researchers to create different studies [32, 12]. Recently, Word2Vec with CBOW and Skip-Gram models opt for query expansion [2]. The findings show that word embedding based query expansion shows promising results. Similarly, there is research that uses the Word2Vec CBOW model and pseudo-relevance feedback methods in query expansion. Integrating the candidates using word embeddings with a pseudo-relevance feedback technique, they get prospering results [18].

While Word2Vec is a learning-based algorithm, there is another popular algorithm for representing words, which is the statistic-based GloVe algorithm.

2.1.2 GloVe

GloVe is an unsupervised learning approach to represent words in a vector space. GloVe uses the statistics of word occurrences in the corpus in its training process. In order to create vector spaces, GloVe [29] uses a co-occurrence matrix, which is a word-document matrix.

Besides the use of GloVe in a named entity recognition and similar tasks, GloVe is also successfully used in query expansion. A recent study in the query expansion research demonstrates the effectiveness of GloVe for word similarity tasks [12]. However, GloVe has several disadvantages. GloVe is a little bit costly in terms of space and time. In order to construct its co-occurrence matrix of all words, it should allocate much storage. It also takes so much time to reconstruct the matrix when the window size changes. Therefore, researchers keep finding more efficient models. After machine learning-based solutions(Word2Vec) and statistic-based solutions (GloVe), some researchers directed the relevance-based embeddings.

2.1.3 Relevance-based Embeddings

Both Word2Vec and GloVe capture term proximity for creating word embeddings. However, instead of using term proximity, adapting relevance can better establish semantic similarity between words. There is a recent study that compares relevance-based embeddings over Word2Vec and GloVe [38]. They constructed a simple neural network to construct word embeddings. According to their findings, relevance-based embeddings can be good at representing words semantically. However, the researchers started to construct a more complex neural network to represent word embeddings better. The first attempt was ELMo [30].

2.1.4 ELMo

Since the demand for word embeddings increases, the researchers adapt deep contextualized models to find these vectors. First, an Embeddings from Language Models (ELMo) is created to get more context information about the input [30]. ELMo uses bidirectional LSTMs and language modeling in its architecture. In ELMo's language modeling, the word is tried to be generated by using previous words. According to the findings, ELMo's word representations outperform the previous model dramatically. With the creation of ELMo, new deep learning models have been released. Each new model performs better than the previous one. So far, the most successful one is BERT.

2.1.5 BERT

After ELMo, BERT [11] and other models are formed. In ELMo, representation could not take advantage of both the left and right contexts of a word simultaneously. BERT differs from ELMo in the use of language modeling. In BERT, the masked language is adapted, masking one word and predicting the word with its surroundings.

In BERT, there are two main approaches, fine-tuning approach, and feature-based approach. The former has been studied widely, and different models have been presented in the literature. According to the ad hoc task, the researchers adopt different models such as BERTforMaskedLM, BERTforNextSentencePrediction, BERTforSequenceClassification [1]. In this approach, according to the need, one classification layer is added to the top of the pre-trained model. Then, the parameters are trained for the task. In the latter, the features are extracted from the model to use them in a

[1] `https://huggingface.co/transformers/model_doc/bert.html`

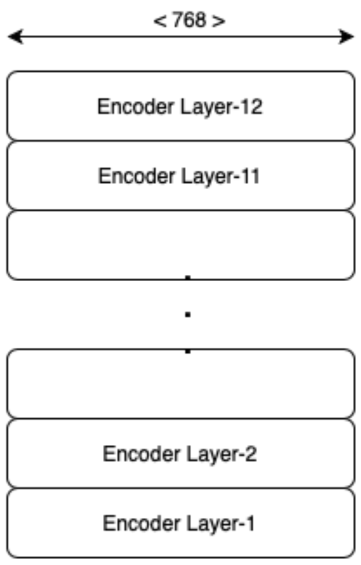


Figure 2.1: BERT Transformer Layers in Base Model

task. In this approach, two types of pre-trained BERT models, BERT-Base and BERT-Large, are used. Among these models, BERT-Large is more compute-intensive than BERT-Base. However, BERT-Base is sufficient to see the power of BERT.

In the architecture of the BERT-Base-Uncased model, there are 12 layers shown in Figure 2.1, and all these layers hold information about the word in the size of 768. Therefore, while extracting the features of word, BERT uses different techniques which are:

- Second-to-Last Hidden
- Last Hidden
- Weighted Sum Last Four Hidden
- Concat Last Four Hidden
- Weighted Sum All 12 Layers

Second-to-Last and Last Hidden methods use specific layers. Weighted Sum Last Four Hidden and Weighted Sum All 12 Layers take the average of specific hidden layers. Concat Last Four Hidden concatenates different parts of layers to represent the vector space of words. According to their results, Concat Last Four Hidden Layers shows the best performance on the Named Entity Recognition(NER) task. In their study, they also note that these results can change depending on the task. Therefore, to obtain the best method, one should try each method to see which method fits best for their problem.

In the literature, BERT's success is mostly based on its fine-tuning approach. BERT has already state-of-art results in different ad hoc tasks such as question answering [37], classification of documents [1], text summarization [39] and passage ranking [24]. Compared to these studies, far less attention has been paid to using BERT for query expansion. Among studies conducted on BERT embeddings and query expansion, there is a recent research [26]. However, different from our approach, they use classic query expansion methods to generate better queries for BERT-based rankers, not use BERT to expand the query. In this thesis, we show the success of BERT word embeddings in the query expansion task.

2.2 Query Expansion Steps

In a query expansion system, there is a chain of actions, as shown in Figure 2.2. The data pre-processing step applies the necessary filters to the raw data. After that, in the term extraction step, the candidate terms are formed. We rank candidate terms in the next step. In the term selection step, we decide the number of candidate terms chosen for adding to the expanded query. In the last step, we assign the weight of all terms in the expanded query.

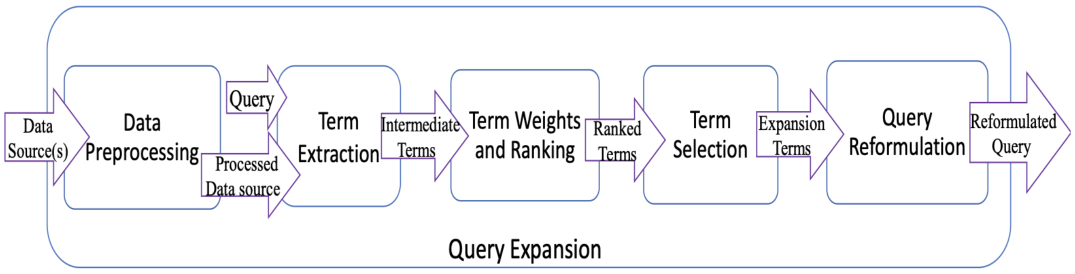


Figure 2.2: General Query Expansion System, taken from Azad and Deepak [3]

.

2.2.1 Data Sources

The first step is to choose the candidate pool and data processing methods. Data preprocessing methods may change depending on the candidate pool choice. There are many ways to choose the data source, varying from adapting external resources to top retrieved documents. Generally, these can be categorized as: documents, hand-built data sources, external data sources, and hybrid data sources.

- Documents: The primary choice of data source adapted by the researchers is to use documents in the corpus. In a query-document environment, choosing the candidate terms that frequently appear in the documents seems logical. The words in the documents may be well-representative for the relevant documents, so adding these terms to the query can significantly increase the retrieval system's performance. There are successful implementations of this idea [6].

- Hand-Built Data Sources: The second method for the choice of a data source is to create different mechanisms for hand-built sources. These textual resources can be a dictionary, an ontology, a thesaurus, or Wikipedia. The researchers investigated these resources to reveal meaningful information. WordNet [23] is the most popular one among these sources. WordNet and other systems created from human-made sources are used successfully by several researchers [19, 28, 35].

- External Data Sources: These categories involve all other resources apart from the first two data source options. These can be web and user logs. The public resource for user logs is AOL query log [2]. This dataset is predominantly used for personalization and adjustment from user session profiles.

- Hybrid Data Sources: Combining any combination of previous approaches can create a hybrid data source. This data source choice is also successfully implemented by different researchers [9]. Using different data sources certainly brings a more significant impact and more successful results.

[2] https://jeffhuang.com/search_query_logs.html

2.2.2 Data Pre-processing and Term Extraction

Data pre-processing involves the process of converting raw data to meaningful data, data pre-processing. To prevent the system from "garbage-in, garbage-out", we need a data pre-processing step. In the data pre-processing step, the most common approaches are:

- Removing stop words
- Removing rare terms
- Removing punctuation
- Making terms lower-cased
- Stemming terms
- Tokenization

The most common filtering of removing stop words, removing punctuation, making terms lower-cased, and tokenization methods [2, 32]. However, the remaining filters depend on the task. For example, while some studies remove rare words [2], some studies do not apply this filtering to their data. In some studies, they see rare words as valuable data since methods like tf-idf give more importance to rare words than very repetitive words. For stemming, there are different algorithms, Porter [31] and Krovetz [17]. Although Porter stemming is more popular, Krovetz algorithm is also applied in some studies [32].

2.2.3 Term Weights and Ranking

After choosing the data source and carrying out the data pre-processing, candidate terms are selected. At this step, the inputs are query and filtered texts from the previous step. As an output, the candidate terms and their relevance score to the query appear. The methods for term weighting and ranking can be categorized [7] as follows:

- One-to-One Association: chooses the term that is related to at least one term in the query.
- One-to-Many Association: chooses the term that is related to the whole query.
- Feature Distribution of Top-Ranked Documents: chooses top-weighted candidates from the returned documents of the initial query.
- Query Language Modeling: chooses the term with the highest probability in a statistical model done on queries.

The correlation between the terms are done with cosine similarity. Given two vector representations of two words as A and B, the cosine similarity can be formulated as:

$$\text{Cosine Similarity} = \frac{\text{A.B}}{|\text{A}| * |\text{B}|} \tag{2.1}$$

Where A.B shows the dot product of two vectors, and |A| and |B| are the magnitudes of A and B, respectively.

2.2.4 Term Selection

Although there are many candidate terms after all previous steps, adding them to the query is not realistic. Various studies are suggesting adding candidate terms ranging from one-third of the query to a few hundred. There are different studies in the literature for showing the impact of adding the different sizes of candidate terms. There is one study that shows that adding a small number of candidate terms generally performs better than adding a large number of candidate terms [33]. The opposite of that, another study shows that adding less than 20 candidate terms reduces the performance of document ranking [27].

2.2.5 Query Reformulation

In the query reformulation step, there are different approaches. While some studies adapt constant weight for their candidate terms [2], some studies assign dynamic weights to their terms. The dynamic weighting schema can depend on various properties, like using a number of co-occurrences of terms and the relations between terms [15]. The typical approach in deciding the reformulation weights is to adapt feedback documents. Although the first choice is to use explicit feedback, there are various studies that adapt pseudo-relevant feedback. Using pseudo-relevant feedback can bring several advantages in query reformulation step [33]. All these methods show successful results in their domain.

2.3 Relevance Feedback

Relevance feedback is a technique of reforming the initial query by using feedback information. There are three types of relevance feedback.

- Explicit Feedback: Assessors of the dataset give explicit feedback or relevance judgment on the query results. It shows the relevance of the documents to each query. In most of the datasets, including TREC 2018 Common Core Track, the relevance judgments are shared [3].
- Implicit Feedback: Implicit feedback consists of user behaviors like dwelling time, scrolling actions, and user's choice of viewing the document. There are significant studies to show and categorize these types of user actions [14, 16].
- Pseudo-Relevance Feedback: Pseudo-relevance feedback or blind feedback is assuming the top-n returned pages from the initial query as relevant. By choosing top-k candidate terms from these documents, the query can be extended to perform better. Various studies use pseudo-relevance feedback in their system [33].

[3] https://trec.nist.gov/data/core2018.html

CHAPTER 3

QUERY EXPANSION BASED ON BERT

Our approach is to use BERT contextualized word embeddings to expand the query for better performance in document ranking. Instead of using no-context word embedding models like Word2Vec, we try to employ BERT since BERT has already proved itself as a successful model in different areas apart from its use in query expansion in the use of query expansion as we described in Chapter 2. Before starting this research, we investigated many papers and understood that BERT has a good understanding of the context in a sentence. BERT can extract meaningful information from the words due to its bidirectional investigation of the surroundings of the word. We first find ways to extract embeddings from BERT and do some experiments on synonyms and antonyms. After these observations, we aim to use BERT word embeddings for query expansion systems. For finding related words to add the query, BERT is the proper choice for our purpose of improving document ranking with a query expansion technique. Therefore, our work can be divided into two parts: finding word embeddings from BERT and building a query expansion system with different methods like tf-idf and word co-occurrence.

We have not come across any study on query expansion with BERT embeddings in the literature; however, we found equivalent work that was carried out with Word2Vec word embeddings. Word2Vec has been the dominant approach in word embeddings so far, but being able to prove BERT success over Word2Vec would be exciting. Therefore, we have tried to find a way to extract BERT contextualized word embeddings from BERT layers, which are mentioned in Chapter 2 in detail. Then, we integrate BERT to our query expansion mechanism.

Before delving into the methods for extracting word embeddings from BERT, we want first to present our architecture. In Figure 3.1, we show the visualization of our query expansion system. BERT word embedding is included in the step of Term Extraction, where we find related candidate terms from our filtered candidate pool. After finding meaningful candidates by BERT, we sent these related candidates to our system's next steps. In the end, we want to achieve better document ranking performance, so that we could say that BERT is good at finding related terms and making a comparison with Word2Vec, which has been a standard approach for word embeddings. For an equivalent word for this purpose, we found recent research that makes a similar study with Word2Vec [2]. Like in this recent research, titles of TREC topics will be served for queries in our study.

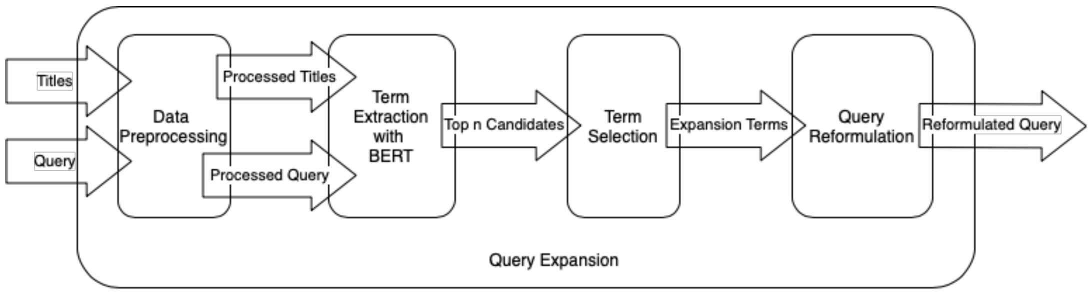


Figure 3.1: Our Query Expansion System

.

This chapter introduces the dataset that we used for our experiments. After introducing data, we explain each step of our chain of actions in query expansion. Last, in this chapter, we also share our work for extracting word embeddings from BERT, so at the end of this chapter, we can observe our system's performance on our dataset by giving the statistics of our results ranking.

3.1 Dataset

We tested our approach in TREC 2018 Common Core Track Track [1], which consists of news articles from Washington Post. The dataset has 50 topics and 595037 documents.

In order to inform the reader about the dataset, we share some sample topics from

[1] https://trec-core.github.io/2018/

Table 3.1: Sample topics from the dataset

TopicID	Topic
321	Women in Parliaments
336	Black Bear Attacks
341	Airport Security
347	Wildlife Extinction
350	Health and Computer Terminals

the dataset in Table 3.1. We use these topics as queries to find related documents. While the first column represents the topic id that we will use in the output format, the second column shows the topic itself.

Table 3.2: Sample documents from the dataset (JSON format)

Key	Value
id	0c0dbf4b0150981e4bd04115084acb53
article_url	https://www.washingtonpost.com/news/early-lead/wp/2016/12/07/...
title	Tim Tebow to take some cuts with the big-league Mets in spring training
author	Des Bieler
published_date	1481113857000
contents	Mets general manager Sandy Alderson said Tuesday of ...
type	blog
source	The Washington Post

The documents given by organizers are in JSON format. This format helps us in finding related parts quickly. We share some sample documents in Table 3.2. From these key-value pairs, we only use the id (document identifier), title, and contents (body of the document). Because of the restricted space, we had to cut the values of the article URL and contents.

For submissions, we need to follow a policy shown in Table 3.3. Every contestant needs to submit the same format to be evaluated by the community. In one sample run, the first column represents the topic identifier, while the second column is nothing but some identifier that will be used later by the community. The third column is the

Table 3.3: Sample output/run format

TopicID	Q0	DocID	Ranking	Score	Tag
321	Q0	6f01fc5c-a2b0-11e1-81b4-d0e4bf8c8fd8	1	6.04458652226206	metu-isl
321	Q0	4a7c2970fd9bf65fe09c7cf46df7b06d	2	5.973899841308594	metu-isl
321	Q0	9171debc316e5e2782e0d2404ca7d09d	3	5.973898887634277	metu-isl

document identifier. The fourth column is a score for showing relevance to the given query. The last column is just an identifier for each group that attends. Since we work under the METU-ISL lab, we chose that tag for indicating ourselves.

Table 3.4: Samples from the relevance judgment list of the dataset

TopicID	Identifier	DocID	Relevance-Score
830	0	fe1e8ffac1b3b0fa77c78a496ff3ee88	2
830	0	ff0619123a4eaf0f7d41958836388a47	0
830	0	ff86307f72f48a8fff383406f16a1a54	2

The relevance judgment is also shared. This judgment list shows the relevance of 1000 documents for each query. While some of these documents are relevant to the topics (indicated as non-zero numbers), some are not (indicated as 0), as shown in Table 3.4. We use this judgment list to give weights to candidates in Chapter 4.

For indexing the dataset and retrieval process, we use Anserini [2] tool. After giving details about the dataset, we now present our data pre-processing steps applied to this dataset.

3.2 Data preprocessing

We start by tokenizing our dataset by splitting according to space. After tokenizing the sentences from the dataset, we remove stop words using nltk module [3] since they do not reveal any meaningful information for our query expansion system. Next, we remove the words that appear less than five times. This method is prevalent in

[2] https://github.com/castorini/anserini
[3] https://www.nltk.org

different research. This step is probably taken because rare words do not improve performance, or the computation resource can not handle so much data. As a next step, we used Porter stemming [4] to convert candidate terms to their base form. Because if we add these candidate terms with the same base form with any query term, then we could not achieve information diversity. If we do not remove them, we only have duplicates of some of the query terms. Therefore, we need to filter those candidate terms. For example, if we have "connect" in our sample query, we do not want to add "connected", "connection", or "connecting". Although they are very familiar terms, the meaning is just the same. As the last step, we lowercased all terms.

3.3 Extracting BERT Word Embeddings

In query expansion systems, the most important and challenging part is to find contextually similar terms to the query. After pre-processing, we want to employ BERT for finding word embeddings. In addition to methods applied by BERT mentioned in Chapter 2, we formed additional methods to find the best word embedding extractor from BERT's hidden states. These methods are as follows:

- First Layer Only
- First Four Layers
- First Six Layers
- Last Six Layers
- CLS Token
- Concat First Two
- Concat First Four
- Concat Last Two
- Concat All Layers

[4] https://tartarus.org/martin/PorterStemmer/

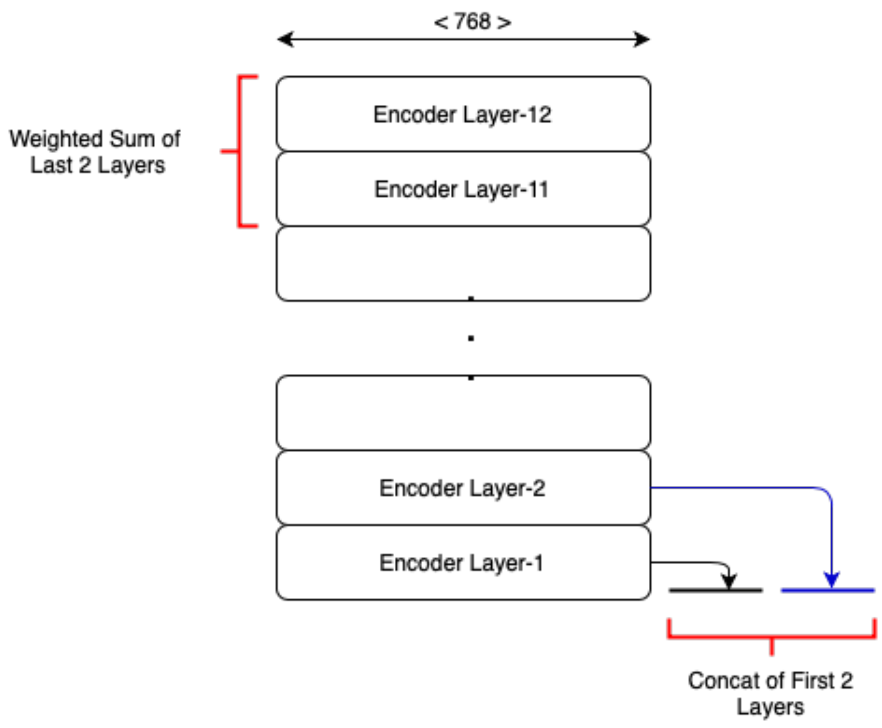


Figure 3.2: Visualization of two methods that extract word embeddings from BERT.

We can categorize these additional methods as using some layers, using special token CLS, or concatenating some layers among 12 layers. Adding these methods, we have 14 methods as total. Concat of First Two Layers and Weighted Sum of The Last 2 layers are visualized in Figure 3.2 as an example.

In order to find the best method among these, we created a test environment. Our premise is that the best method should give a high score to relevant terms and assign a low score to non-relevant terms. Given three terms (where the first two are semantically similar, and the third has a different meaning than the first two terms), we expect to see high cosine similarity score for the first two, and a low score for non-similar terms. For example, here is one sample: "virus", "pathogen", "plane". In this example, the first two are related to the epidemic; the third has no close relation with the epidemic. Similar to this example, we created 100 test inputs. We share some of the samples in Table 3.5.

After creating our test environment, we have created a scenario to test how one extractor can understand similar terms and assign a high score. This extractor should

Table 3.5: Samples from manually created dataset. Word1 and word2 are chosen as similar words while word3 has different than others.

Word1	Word2	Word3
end	finish	consume
obtain	get	yellow
look	see	sail
tiny	small	sea
hate	dislike	year

also distinguish non-similar terms and assign a low score. To formulate, let $Score_{12}$ represents the cosine similarity between term1 and term2 and $Score_{23}$ represents the cosine similarity between term2 and term3 ($Score_{13}$ can also be used as an alternative for $Score_{23}$). Our aim is to maximize the difference between $Score_{12}$ and $Score_{23}$. Besides, $Score_{12}$ should be close to one, whereas $Score_{23}$ should be close to zero since we use cosine similarity score between terms. These three properties can be formulated in the Terms Similarity Score (TSS) as follows:

$$TSS = (Score_{12} - Score_{23}) + Score_{12} + (1 - Score_{23}) \tag{3.1}$$

By taking the average TSS of all inputs, we measure the TSS performance for each method. We share our results in Table 3.6. First Layer Only gives the best score, whereas the CLS token gives the worst result. There were some thoughts that [CLS] token can represent the word like it represents the sentences. However, we observed that it might not be the case in word embeddings. Besides, Concat First Two has almost the same score as First Layer Only, so it can also be used in finding word embeddings. We also observe that the first layers are generally better representatives than the last layers. This is the case in both the method of summing layers and the method of concatenating the layers. Considering these observations, we use First Layer Only as our BERT word embedding extractor to extract word embeddings from terms.

We also wonder how BERT behaves with the words that do not appear in BERT's vo-

Table 3.6: BERT Word Embedding Scores. The score is calculated using similarity between word1-word2 and word2-word3. They are averaged over 100 inputs.

Method	Calculated Score
First Layer Only	0.4967
First Four Layers	0.4876
First Six Layers	0.4793
Last Six Layers	0.4513
Last Four Layers	0.4474
Last Two Layers	0.4402
Last Layer Only	0.4516
All Layers	0.4686
CLS Token	0.3629
Concat First Two	0.4912
Concat First Four	0.4798
Concat Last Four	0.4445
Concat Last Two	0.4378
Concat All Layers	0.4610

cabulary list, namely out-of-vocabulary(OoV) words. For OoV words, BERT adapts the WordPiece model in which its subwords represent one OoV word. Since BERT vocabulary has only 30000 fixed words, finding all the related words in the language is impossible. For representing the OoV words, we take an average of subword representations of the word. According to our observations, BERT also performs well in OoV word representations.

After finding the best word embedding extractor from BERT, we continue to follow the remaining step in the query expansion system, the term extraction step.

3.4 Term Extraction

There are two possible places to extract candidate terms for our query expansion system. While the first one chooses the candidate terms from titles, the second is choosing candidate terms from the document body. Both methods have advantages and disadvantages over each other, as shown in Table 3.7. The first factor is computational power. The former offers fewer candidate terms, which lightens the computational power, while the latter offers many candidate terms. The second factor is the ability to be used in data-driven approaches like Machine Learning and Deep Learning applications. Since the latter offers much data, it is more adaptable for different models. The risk of deceiving BERT and choosing the wrong word as a candidate is high in the latter one. As a fourth factor, since the title has dense information for representing the document, words chosen from titles can be more suitable. Regarding all these factors, we adapt using candidates chosen from titles.

Table 3.7: Pros and Cons of Each Method over Another One.

From Titles	Factor	From Bodies
Remarkably less	Computational Power	Hard to handle using one machine
Few data	Tendency to be used in Machine Learning Applications	Suitable
Low	Risk of Choosing Wrong Word	High
Suitable	Dense Information	Sparse information

3.5 Weighting and Ranking of Terms & Selection of Terms

Having decided to use candidate terms from titles, we find top n candidate terms for query expansion. Therefore, we need to decide the way of using the relationship between the query and candidate terms. This relationship was categorized into four methods [7], which we explain in Chapter 2. However, we used only two of them: one-to-one association and one-to-many association. Therefore, when we process the candidate terms, we follow these two strategies. On the one hand, we find the best candidate terms for each term in the query in the one-to-one association. On the other hand, we find the best candidates for the whole query in a one-to-many association.

There are different consequences of using either of these two methods. To investigate the differences, we adapted each method and conducted our experiments with each method. There are different parameters to measure their performance. For example, we need to decide the number of candidate terms to be added to the original query. Regarding these kinds of parameters, we conducted different experiments to observe the effect of each method. According to our findings, these two methods show nearly the same performance when we set the number of candidates the same as the number of query terms. After observing this, we expand the number of candidate terms. We believed that more candidates could better reveal the performance of each method. Since the average size of the queries in the dataset is approximately three, adding only three terms from each method can not show their actual performance. Hence, we set the number of candidate terms twice as the size of the query and then conducted the experiments. With these settings, we realized that one-to-one association gives better performance over one-to-many association. There can be different ideas to explain this performance difference. We share ours in Chapter 5. We also put some of these experiments to show the differences in performance in Table 5.7, so that the effect of each method can be investigated deeply.

3.6 Query Reformulation

In the study of query expansion with Word2Vec [2], for query reformulation step, they assigned the original query weight as one while fixing the candidate term weight as 0.5. This says that while original query terms contribute to the extended query equally between them, the candidate terms can contribute the extended query as half weight of the original query terms. We think this weighting schema can be reasonable because extended query terms should not exceed the weight of the original query term. This policy can be seen as assigning constant weight for terms. However, there are more intelligent ways to assign more meaningful values to terms. We consider this approach, which is choosing dynamic weights for candidates in Chapter 4. We also conduct several experiments to show the difference between each policy in Chapter 5.

3.7 Comparison to Word2Vec model

After all steps, we extracted top-50 candidate terms for each query. In order to share some of the best candidate terms and their relevance scores to the given query, we present an example in Table 3.8. In this example, we share the top 8 candidate terms for the query of "social media and teen suicide". In one-to-many association, we share the top candidate terms which are best associated with the whole query. In one-to-one association, we represent the "based on" column to show the query term that has the closest meaning with the candidate term.

By looking at the table, we can say that the candidate terms are semantically related to the query. Therefore, we can say that the results obtained when adding these candidate terms to the query may find the relevant documents better. By conducting this experiment, we also observe the result of our primary research goal in this thesis, which is observing the performance difference between BERT and Word2Vec. According to our observations, BERT performs slightly better than Word2Vec. We leave all interesting results and comments on the performance difference to Chapter 5, where we conduct various experiments.

Table 3.8: Best candidate terms extracted by BERT in decreasing order of their relevance score to the query.

Query: "social media and teen suicide"				
One-to-Many Association		One-to-One Association		
Candidate	Score	Candidate	Score	Based on
andi	0.34652	prank	0.90266	teen
societal	0.33030	protege	0.89315	teen
teenage	0.32880	societal	0.88624	social
teeny	0.32850	lifestyle	0.84747	social
televisions	0.32525	broadcasting	0.85894	media
socialite	0.32317	culture	0.84452	media
adolescent	0.32196	murder	0.86507	suicide
genders	0.32030	burial	0.84378	suicide

CHAPTER 4

INTEGRATION WITH THE RELEVANCE FEEDBACK

In Chapter 3, we used all the documents in the corpus as our data source to find candidate terms and considered titles of TREC topics as queries. We extracted candidate terms from the titles of the documents. We tried to find the relevant documents by assigning constant weights for the terms in our extended query. These weights are 1.0 for the original query terms and 0.5 for each of the candidate terms. After trying different sizes of candidate terms, we observed that our approach outperforms the work done with Word2Vec model[2]. The results are promising because Word2Vec word embeddings were seen as a cornerstone for creating vector representation of the words. After observing the success of BERT word embeddings, we turn our attention to assigning weights of candidate terms. We need to find a different way of setting weights of candidate terms. In this chapter, we aim to see that a better ranking of candidates improves document ranking.

In intelligent query expansion systems, not all candidates should have the same query weight in the extended query. While some candidates may have a less positive impact on finding relevant documents, some candidates may better indicate relevant documents. To assign weights and ranks to the candidate terms, we need to find a way to weigh the candidate terms according to their relevance. As we mentioned in Chapter 2, several studies use relevance feedback. Therefore, we use two different relevance feedback methods, which are and pseudo-relevance feedback and explicit feedback. The methods used in these two feedback are the same, so we will talk about these methods under a common roof. The specific details of each method and results will be discussed in separate sections.

4.1 Adapting Dynamic Weights

The use of relevance feedback brings two main advantages. The first advantage is to score candidate terms better. To achieve this, we adopt some of the Query Expansion (QE) techniques, tf-idf and term co-occurrence. Since we have relevance feedback, we can employ these methods so that we can weight candidate terms dynamically rather than assigning constant weight.

4.1.1 Integration of Tf-Idf property

The use of tf-idf score can bring great improvements. Tf-idf scores of candidate terms can help us finding the terms that represent relevant documents better so that we can increase the weight of a high tf-idf scored candidate term in the expanded query. The tf-idf formula that we use in our experiments as follows:

$$\text{Tf-Idf(c)} = \frac{tf_c}{df_c} \tag{4.1}$$

where tf_c and df_c shows the term frequency and document of frequency of candidate c, respectively.

4.1.2 Integration of Term Co-occurrence property

In applying terms co-occurrence, we exploit the relationship between query terms and candidates from relevance feedback. We increase the candidate's weight if it occurs a lot with any query term in the relevant documents. Term co-occurrence property helps us show the documents that have both the candidate term and query term. If we spot that there are so many documents, we can understand that the candidate term is more important for the query. The formula that we employ for calculating term co-occurrence score of each candidate is as follows:

$$\text{Term Co-occurrence(c)} = \frac{dn_{c \wedge t}}{dn_t} \tag{4.2}$$

where $dn_{c \wedge t}$ is the number of relevant documents that contain both candidate c and any query term t from the query, and dn_t is the number of relevant documents that contain any query term t from the query.

4.1.3 Combination of Tf-idf and Term co-occurrence

We used each of these two scores which are tf-idf and term co-occurrence in our experiments. We also combined them to get a score for each candidate term by averaging the scores from the two methods. Given a candidate term c and any term t from the query, we formulate the dynamic term weight (DTW) as follows:

$$DTW(c) = \frac{1}{2}\left(\frac{tf_c}{df_c} + \frac{dn_{c \wedge t}}{dn_t}\right) \tag{4.3}$$

In the query reformulation step, since we use the calculated score for each candidate term, we do not want candidates' scores to exceed the original queries' scores. We assign the scores of original query terms as 1. Therefore, we need to adjust the scores of candidate terms not to exceed 1. We want to range their score from 0.0 to 1.0. Hence, when DTW exceeds 1.0 due to the high tf-idf score, we reduce DTW to 1.0 in order not to exceed the weights of terms in the original query. Since the term co-occurrence score is the probability score, we do not need to adjust that.

After choosing the top 50 candidates using BERT embeddings, we assign the DTW score to each candidate as a term weight in the extended query. Using these weights, we conducted several experiments. We did various experiments using only tf-idf scores, only term co-occurrence scores, and DTW scores. We observe that using both tf-idf and term co-occurrence increased the ranking of documents substantially when used together and even individually. Therefore, we can say that dynamic weights increase performance significantly compared to using fixed weight for all candidate terms. The effect of assigning dynamic weights rather than predefined constant term weight is discussed in detail in Section 5. We also share each of the significant results in Table 5.7. To give an example for showing the candidate terms, we represented the candidate terms for a sample query in Table 4.1 and Table 4.2.

4.2 Integrating the Pseudo-Relevance Feedback

In situations, when there is no explicit feedback given with the dataset, pseudo-relevance feedback can be a good choice to create the "pseudo" relevant documents. In pseudo-relevance feedback, the relevant documents are formed by retrieving the first-n results returned by the initial query from the search engine. Assuming these documents as relevant, candidate terms are extracted from these documents.

To conduct this experiment, we first direct the queries to the retrieval system. After taking the first 50 documents for each query, we extract candidate terms. In calculating the DTW score shown in Equation 4.3 for each candidate term, we use these documents as our relevance documents. Then, choosing the different sizes of candidate terms, we experiment with different scenarios. Regarding the several successful studies that we present in Chapter 2, we expect to see some significant improvements compared to our base method in Chapter 3. According to our results, the improvements have noticeable effects on the results. The scores taken from this methodology and the related comments are shared in detail in Chapter 5.

4.3 Integrating the Explicit Feedback

When the explicit feedback (relevance judgment) is available with the dataset, using the documents in this explicit feedback as a data source looks like a better idea. Since the documents in this feedback are judged by humans, they should have a closer meaning to the query. Therefore, it should theoretically give better results compared to using pseudo-relevance feedback.

To do this experiment, we first take all relevant documents for each query. To compare the relevant document size with pseudo-relevant document size, we share the statistics of relevant documents in Figure 4.1. Then, we extract the candidate terms from these documents for each query. After applying the data pre-processing step, only filtered candidate terms remain. Having been calculated DTW scores for each candidate term using relevant documents, we choose the best candidate terms. Adapting various candidate size in term selection step, we conduct several experiments. At this point,

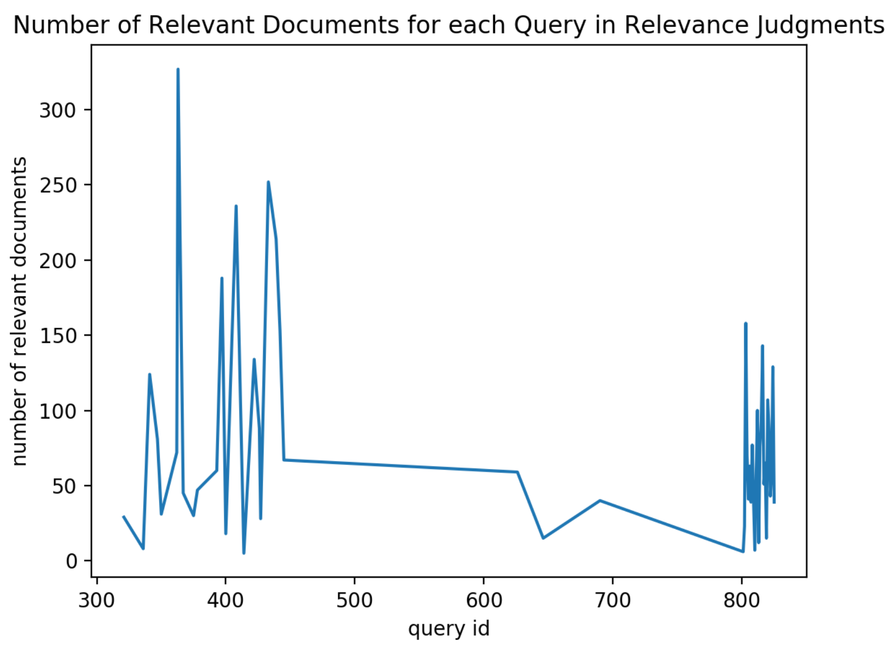


Figure 4.1: Number of Relevant Documents For Each Query In Relevance Judgment

we can say that this methodology gives the best results compared to our base work and pseudo-relevance feedback approach. The results observed in this methodology, including the best score that we found among our experiments, are represented in Chapter 5.

4.4 Query Expansion vs Document Expansion

The second advantage of using relevance feedback is reducing the number of documents in which we search. When we have fewer documents, we can investigate another option for choosing the candidate pool. For the candidate pool, we only considered titles of the documents in Chapter 3. We have wondered whether choosing candidates' terms directly from documents can give better results than choosing terms from titles of documents only. This comparison is also known as query expansion versus document expansion in the literature. This research issue has already been examined considerably in different studies. According to these studies, it can be said that document expansion is not promising compared to query expansion [5, 36]. We also want to experiment with the performance difference of this comparison. As we

Table 4.1: Best candidate terms extracted from titles using BERT in decreasing order of their relevance score to the query.

Query: "social media and teen suicide"				
One-to-Many Association		One-to-One Association		
Candidate	Score	Candidate	Score	Based on
bullying	1.00	facebook	0.45790	social
killed	0.58547	life	0.45736	social
student	0.50014	online	0.42712	media
family	0.44761	video	0.35006	media
life	0.4438	boy	0.64671	teen
death	0.43561	girl	0.56099	teen
online	0.42650	child	0.53594	teen
people	0.41817	bullying	1.00	suicide
police	0.38391	killed	0.59047	suicide
schools	0.37970	death	0.43811	suicide

did in extracting candidate terms from titles, this time, we extracted candidate terms from documents. For making comparisons between them in our experiments and being able to show the results, we added a Title/Body column in Table 5.7. We share the results in detail in Chapter 5. Our findings approve the existing work showing that expansion from titles outperforms expansion from document bodies. The comments and details of each method are shared in Chapter 5.

In this chapter, we represented adapting relevance judgments for conducting several experiments. The first time we need this feedback was the time we assign constant weights to the candidate terms. We believed that not all the candidate terms should have the same weight in the expanded query and the same importance for finding relevant documents. Therefore, we employed two well-known properties of QE systems, which are tf-idf and term co-occurrence. We observed significant improvements in the results. Hence, we can say that we can apply for the work in this thesis anywhere when there are some relevant documents. We share our intention to use this system in another research topic, which is personalization in Chapter 6. As a second advantage

Table 4.2: Best candidate terms extracted from document bodies using BERT, decreasingly ordered by their relevance score to the query.

Query: "social media and teen suicide"				
One-to-Many Association		One-to-One Association		
Candidate	Score	Candidate	Score	Based on
sexual	0.27685	people	0.42692	social
tragic	0.26276	community	0.29026	social
polices	0.26	internet	0.45785	media
teenage	0.25102	video	0.35006	media
internets	0.195	boy	0.64671	teen
youth	0.14996	girl	0.56099	teen
womens	0.105	child	0.53594	teen
terrorist	0.10480	killed	0.59047	suicide
crime	0.095	death	0.43811	suicide
criminal	0.09	tragic	0.26568	suicide

of using relevance judgments, we reduced the number of documents that we look at. In the first part in Chapter 3, we could do our experiments using only the titles of the documents since there is a considerable amount of documents and for each document, thousands of words as candidate terms in the document body. However, in this chapter, we have fewer documents, which are only relevant documents to the query. Therefore, we point out an essential issue in NLP, which is the performance difference between query expansion and document expansion. We show that document expansion underperforms. There could be several reasons for this, and we discuss this and other details in the next chapter.

CHAPTER 5

EXPERIMENTS AND EVALUATIONS

In this chapter, we show the experimental results of all the methods mentioned in the previous chapters. We present comparisons and performance differences between some basic methods. In the end, we represent our final table, which summarizes the main points of each experiment. Before diving into the details of the comparisons, we share our computational environment in which we did all the experiments. In our opinion, sharing our environment is vital since it encourages the researchers to reproduce the results of some specific method that they focus on. In recent years, the reproducibility has especially been important more than ever. We also plan to integrate Docker [1] to our experiment results so that any researcher can reproduce the experiments or improve parts of our work in seconds. We next present the evaluation metrics that we use to observe the performance differences between different methods. We choose the most popular metrics so that any academician who works on TREC 2018 Common Core Track can easily compare their results with ours.

We present all experiments that we conducted to measure the performance of different methods that we represent in the previous chapters. Using the metrics that we stated in the previous section, we compare some of the notable methods. In the first experiment, we measure the performance of word embeddings from BERT over Word2Vec word embeddings, which is our primary research goal. The performance of BERT has already shown with different studies that we indicated in Chapter 2. However, to the best of our knowledge, the success of BERT on word embeddings and its performance comparison with the Word2Vec model on a real dataset was not examined before. Therefore, we believe that the result of this comparison is quite impressive. After that, we focus on different query expansion techniques. We give comparisons

[1] https://www.docker.com

on some controversial topics like using one-to-one association or one-to-many association in extracting terms. Results can vary in other models since this comparison is made using the BERT model. Next, we point out the contribution of the relevance feedback to our query expansion system. Following that, we share the performance comparison between the candidates from titles and the candidates from the document bodies. As the last point, we share our best result by combining all positive points from previous experiments.

5.1 Computational Resource

We conduct our research on a computer that we can easily make use of parallelization. The main properties of the server provided by our department are:

- 128GB of memory size
- 64 cores
- AMD Opteron (tm) Processor as CPU model

Some of the experiments are conducted in a short time. However, when we are dealing with all the corpus in some experiments, we need to wait longer to see the results. To solve that, we used Python processes and Ray library [2]. Despite these, some experiments still take some considerable time.

5.2 Evaluation Metrics

Evaluation metrics in information retrieval systems are used to evaluate how well the method performs on a given dataset. While there are two different types of evaluation categories, we use offline evaluation metrics. There are a lot of different metrics in offline evaluation systems. To name a few, we can say Precision, Recall, F-score, average precision, precision at K, mean average precision(MAP), and normalized discounted cumulative gain(NDCG). To evaluate our methods with the most popular

[2] https://github.com/ray-project/ray

metrics, we tried to use widely employed ones. We also want to evaluate our methods from a different perspective. While some metric system uses precision, some other focus on recall. In order to observe all these criteria, we use MAP, P at K, and NDCG.

5.2.1 MAP

Precision and recall metrics are generally used on the whole-list of documents. While the former focuses on relevant items from retrieved items, the latter pays attention to retrieved relevant items from relevant items.

$$\text{Precision} = \frac{\text{\#(relevant items retrieved)}}{\text{\#(retrieved items)}} \tag{5.1}$$

$$\text{Recall} = \frac{\text{\#(relevant items retrieved)}}{\text{\#(relevant items)}} \tag{5.2}$$

By employing recall and precision rates, we can draw a precision-recall curve. Using the area under the curve, we can employ both precision and recall to calculate a combined metric, average precision. Given precision P and n as the number of retrieved documents, AP can be formulated as:

$$AP = \frac{1}{r_k} * \sum_{i=1}^{n} P(i) * R(i) \tag{5.3}$$

where r_k represents the number of relevant documents and recall R is a pairwise function, which is 1 when the returned document is relevant, otherwise 0.

Similarly, MAP is calculated as an average value of average precision value for each query. Given the Q as the query set, MAP can be formulated as follows:

$$MAP = \frac{1}{|Q|} \sum_{i=1}^{|Q|} AP(i) \tag{5.4}$$

MAP is seen as one of the most potent indicators for dataset evaluation. It is widely used in information retrieval systems.

5.2.2 P@k

For today's information retrieval systems, using MAP loses its attention and importance. In today's search engines, the user only cares about the first page, which includes just ten documents, not all the relevant documents. Most of the time, the user does not turn to the second page. Therefore, evaluation metrics are focused on the fixed-level precision values, like precision at k. The most popular one is P at 10. We integrate this metric to our evaluation system.

5.2.3 NDCG

Discounted Cumulative Gain uses the relevance of the documents. The premise is that the documents which have a higher relevance score should be listed at the top of the page. If not, there should be some penalty to the score. It is defined as follows:

$$DCG_l = \sum_{i=1}^{l} \frac{rel_i}{log_2(i+1)} \tag{5.5}$$

where *l* shows rank location of the document.

The purpose of NDCG is that more relevant documents appearing lower in the search result should be penalized. This is achieved by dividing the relevance score by the logarithmic value of the position of one individual result. We also adopt this standard metric in our evaluation system.

5.3 Anserini

Lucene[3] has been the dominant approach for indexing the documents for years. However, due to the lack of support for the evaluation of standard test collections, the researchers started creating different platforms. To close this gap, Anserini[4] is created. Following that, to bridge the gap between the projects written in Python and Anserini, Pyserini is created. For indexing the documents and retrieving them, Pyserini[5] offers

[3] https://lucene.apache.org
[4] https://github.com/castorini/anserini
[5] https://github.com/castorini/pyserini/

Table 5.1: Comparison between BERT and Word2Vec in query expansion system

					TREC 2018 Common Core Track		
Methodology	Title/Body	One-to-One/One-to-Many	Term Selection	Term Re-weighting	MAP	P@10	NDCG
JARIR-sg-re(Best)	Title	One-to-Many	Query Size	0.5	0.2040	0.3700	**0.4861**
BERT	Title	One-to-One	Query Size	0.5	0.2049	**0.3720**	0.4696
BERT	Title	One-to-Many	Query Size	0.5	**0.2101**	0.3700	0.4793
Standard BM25	-	-	-	-	0.2495	0.4580	0.5375

an excellent interface to academicians. Its proper documentation and simple interface ease the job of researchers. In addition to that, the steps for indexing various accessible datasets are already explained in their documentation.

5.4 Experimental Results

5.4.1 BERT vs Word2Vec

We first investigate whether BERT outperforms Word2Vec in extracting accurate word embeddings. Using the same experimental environment with the recent work [2], we observe that the query expansion system with BERT performs better than Word2Vec in MAP and P@10 metrics. We obtain better performance with two different settings shown in Table 5.1. While one-to-one association gives better results in P@k, the one-to-many association produces better results in MAP. Regarding the success of BERT in MAP and P@k, we can say that word embeddings extracted from BERT can be a better choice to represent the words. Therefore, they can be more accurate in finding the best candidate terms than Word2Vec word embeddings. It should also be noted that in the recent work [2], they trained their model on TREC Washington Post Corpus before extracting the candidate terms. However, in our case, we only used BERT pretrained base model that was never trained on this dataset before. At this point, we can say that the BERT pretrained model's contextualized embeddings are comparable with no-context embeddings like Word2Vec word embeddings in representing the words. However, we also notice that both models underperform the standard BM25 score.

In all previous prospering works, the model should be trained on the corpus to get meaningful representations of the words. However, in our case, the model that we used was pre-trained on the union of Google BookCorpus and Wikipedia. By using this pre-trained model on a real dataset, observing its success over trained models is worth taking attention. Regarding the success of transfer learning in Computer Vision (CV), this success of pre-trained models like BERT in NLP shows its potential for incoming projects.

5.4.2 One-to-One Association vs. One-to-Many Association

The next point we analyze is comparing the performance of candidate extraction methods: one-to-one association and one-to-many association. In one-to-one association, the expanded term, which is highly correlated to at least one query term, is chosen. In one-to-many association, however, the chosen candidate term should be correlated to the whole query. It is a crucial decision to be taken to choose the candidate terms. Therefore, in order to observe their performance in detail, we conduct several experiments by changing the candidate size, as shown in Table 5.2.

Table 5.2: Comparison between one-to-one association and one-to-many association with different candidate size and term re-weighting.

					TREC 2018 Common Core Track		
Methodology	Title/Body	One-to-One/One-to-Many	Term Selection	Term Re-weighting	MAP	P@10	NDCG
BERT	Title	One-to-One	Query Size	1.0 - 0.5	0.2049	0.3720	0.4696
BERT	Title	One-to-Many	Query Size	1.0 - 0.5	0.2101	0.3700	0.4793
BERT	Title	One-to-One	2 * Query Size	1.0 - 0.5	0.1876	0.3420	0.4418
BERT	Title	One-to-Many	2 * Query Size	1.0 - 0.5	0.1776	0.3180	0.4442
BERT	Title	One-to-One	2 * Query Size	0.8 - 0.2	**0.2253**	**0.3960**	**0.4954**
BERT	Title	One-to-Many	2 * Query Size	0.8 - 0.2	0.2169	0.3920	0.4895

When we set the candidate size to the query size, we see that each method's performance is nearly the same. This is expected since the size of query terms is approximately three, we can not precisely observe their performance. Therefore, in order to see more reliable results, we increased the number of candidates. When we add more candidates (in the number of 2 * Query Size) to measure each technique's effect, we have expected that one-to-many association would outperform one-to-one association since only one word is not enough to represent the query. However, we are surprised by the results. According to our findings, one-to-one association performs better. It is also the case when we change the weights of the terms. We can interpret these by saying that BERT already adds context information to its candidates, so each term is context-aware. So far, this was the exact opposite that one-to-many association generally outperformed one-to-one association in previous studies. In that case, although words are trained on the corpus with a window mechanism, words can not learn the context. On the contrary, BERT's bidirectional way of analyzing the words adds context information to the candidate terms. Therefore, for us, this is an exciting result.

At this point, we also want to share one interesting point to show the weakness of one-to-one association. Although one-to-one performs better than one-to-many in most cases, there is also a compelling edge case that one-to-one association underperforms one-to-many association. To exemplify, for the query "Amazon rain forest", "Amazon" has chosen its closest candidate as "Facebook", though "Facebook" is not related to the original query, "Amazon rain forest". However, this is only special to this kind of example. This example occurs in cases where there are private names like company name or person's name. To handle these cases, there may be different mechanisms. However, in general, one-to-one association is more suitable for query expansion techniques in contextualized embeddings. The details of example can be examined from Table 5.3.

Table 5.3: For the query, "Amazon rain forest", the candidate terms chosen by each method.

Original Query: "Amazon rain forest"

Method	Generated Candidate Terms
One-to-One Expansion Method	"Facebook" + "snow" + "wildlife"
One-to-Many Expansion Method	"Climate" + "Brazilian" + "change"

5.4.3 The Effect of Integration of Pseudo-Relevance Feedback

We observe that all previous models that we have constructed so far underperform standard BM25 baseline. There are several reasons for this performance weakness. The first one is the number of candidates. When we use all corpus, the number of candidate terms after all data preprocessing is 30926. Regarding this considerable amount of possible candidate terms, the risk of choosing the wrong candidate term increases. In order to solve this problem, we need to reduce that number to a reasonable amount. The second reason for this performance weakness is that all query terms have the same weight in the expanded query. This makes the query more fragile to be affected by bad candidate terms. Instead, we need dynamic weights for each candidate terms. The candidate terms should be weighted in expanded query within their relevance to the query.

The first idea is to integrate pseudo-relevance feedback (PRF) to our system. Choosing the relevant documents from PRF brings two main advantages. The first one is to narrow down the set of candidate terms. It reduces the number of candidate terms, so the election occurs from "pseudo" relevant documents rather than all documents, which reduces the risk of choosing a non-relevant term. After adapting PRF, for each query, we have 50 documents to search for. Comparing these numbers with the first part of the experiments, the number of candidate terms is remarkably fewer. Besides lessening the number of candidates, adapting PRF brings the second and most important insight. We can employ the properties of the terms that appear in the relevant documents. If we choose the terms and order them with their distinctiveness, we can get better candidate terms. To discriminate against the relevant documents from non-relevant ones, we can use tf-idf and term co-occurrence scores of the candidate terms. These scores are unique to each term; they also show their relevance to the relevant documents. Therefore, by using the DTW score, an average of tf-idf, and term co-occurrence scores as weights in the expanded query, we can achieve dynamic weighting of candidate terms.

We conducted several experiments, as shown in Table 5.4. In the first row, we give the standard BM25 baseline score on this dataset. Trying different candidate size, we get several results. Besides using DTW, we also observe the performance of tf-idf and term co-occurrence(TC) individually. According to our observations, DTW gives better than using tf-idf and TC alone. To conclude, we observe considerable improvements over BERT model that we did in the previous section. Although this approach outperforms the standard BM25 baseline, it is still worse than the standard BM25+RM3 baseline where RM3 states the standard query expansion method.

Therefore, we still continue to search for different ways.

Table 5.4: The effect of integrating the PRF to our system

					TREC 2018 Common Core Track		
Methodology	Title/Body	One-to-One/One-to-Many	Term Selection	Term Re-weighting	MAP	P@10	NDCG
Standard BM25	-	-	-	-	0.2495	0.4580	0.5375
BERT + PRF + DTW	Title	One-to-Many	Query Size	Dynamic	0.2390	0.4500	0.5117
BERT + PRF + DTW	Title	One-to-One	Query Size	Dynamic	0.2460	0.4460	0.5198
BERT + PRF + Tf-Idf	Title	One-to-One	Top40	Dynamic	0.2649	0.4540	0.5376
BERT + PRF + TC	Title	One-to-One	Top40	Dynamic	0.2586	0.4500	0.5279
BERT + PRF + DTW	Title	One-to-One	Top40	Dynamic	0.2686	0.4540	0.5409
BERT + PRF + DTW	Title	One-to-One	Top35	Dynamic	**0.2693**	0.4540	**0.5425**
Standard BM25 + RM3	-	-	-	-	0.3135	0.5120	0.5889

5.4.4 The Effect of Integration of Relevance Judgments

To score better than BM25+RM3, we look for other relevance feedback, relevance judgment (RJ). In our dataset, there is a list of relevant and non-relevant documents for each query. Like using PRF, adapting RJ also lessens the number of candidate terms. The average number of candidate terms for each query is 312 when we choose from titles. On the other hand, that number is 7098 when we choose from document bodies. Comparing these numbers with our base work, the number of candidate terms is significantly fewer.

We tested the effect of using RJ and dynamic weighting using tf-idf and term co-occurrence (TC) scores. All experiments by changing the number of candidates, which can be examined in Table 5.5. When we make a term weighting dynamic using tf-idf rather than some constant value, the system's ranking improved dramatically. This performance rise is something we expected since not all terms have the same contribution to the query. Compared to tf-idf, term co-occurrence (TC) shows a smaller improvement. In TC, we observe significant increases in MAP. After observing these two scores, we also give the performance of DTW. It is important to note that a tf-idf based and DTW approach starts to outperform BM25+RM3, which is the standard score in the literature.

Table 5.5: The effect of integrating the RJ to our system

					TREC 2018 Common Core Track		
Methodology	Title/Body	One-to-One/One-to-Many	Term Selection	Term Re-weighting	MAP	P@10	NDCG
Standard BM25 + RM3	-	-	-	-	0.3135	0.5120	0.5889
BERT + RJ	Body	One-to-One	Query Size	0.5	0.2198	0.4060	0.4954
BERT + RJ	Title	One-to-One	Query Size	0.5	0.2563	0.4580	0.5383
BERT + RJ	Title	One-to-Many	Query Size	0.5	0.2521	0.4300	0.5375
BERT + RJ + Tf-Idf	Title	One-to-One	Query Size	Dynamic	**0.3185**	**0.5560**	**0.5978**
BERT + RJ + TC	Title	One-to-One	Query Size	Dynamic	0.3081	0.5060	0.5875
BERT + RJ + DTW	Title	One-to-One	Query Size	Dynamic	**0.3174**	**0.5340**	**0.5893**

5.4.5 Candidates from Title vs. Candidates from Body

Next, we do some more experiments to see the performance comparison between candidate terms from the titles or candidate terms from document bodies. Since we adapt RJ, the number of documents decreased to a reasonable number. Previously, we only extracted the candidate terms from the title of the documents. Now, we can extract candidate terms from document bodies. This comparison can also be associated with the comparison between query expansion versus document expansion. In document expansion, we extend the query from the terms of document bodies. As we examined in Chapter 2, there is an interesting study of document expansion, in which they try to guess the queries that the document can answer to [25]. In addition to that, there are several studies which compare the query expansion and document expansion. We also want to investigate this research topic in our study.

We extracted candidate terms from relevant documents for each query. The statistics of the number of relevant candidate terms chosen from titles and document bodies are represented respectively in Figure 5.1 and Figure 5.2. As it can be seen, the average candidate size is approximately 300 in title candidates, while it is nearly 7500 in document body candidates. Therefore, the possibility of choosing wrong terms increases in document body candidates.

We conducted different experiments to observe the results by changing the settings of various methods like query size and term weighting strategy. We observe that the candidates from the titles performed better than the candidates from the document bodies in all experiments. The explanation for this performance difference could be that titles represent its documents as a summary. In other words, titles offer more

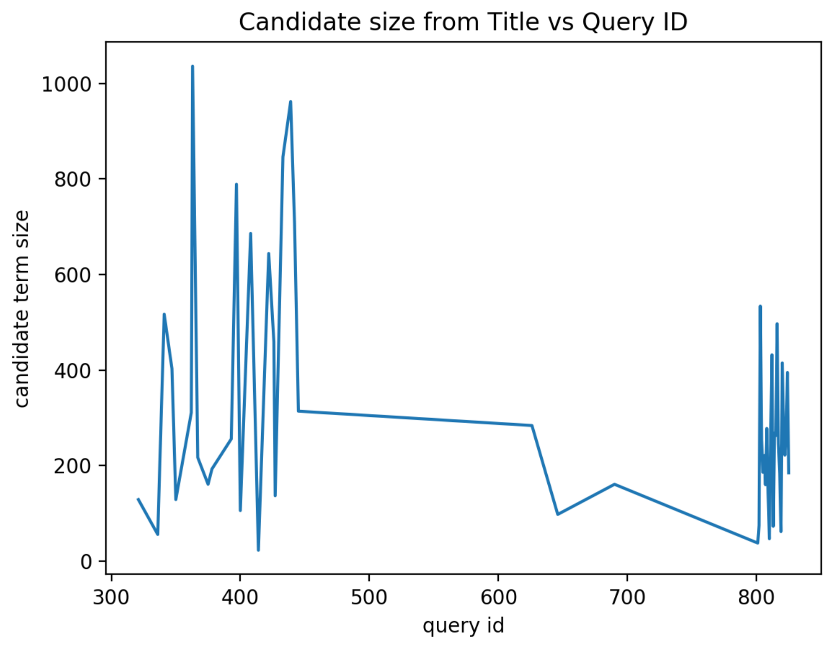


Figure 5.1: Candidate size chosen from Titles vs Query ID

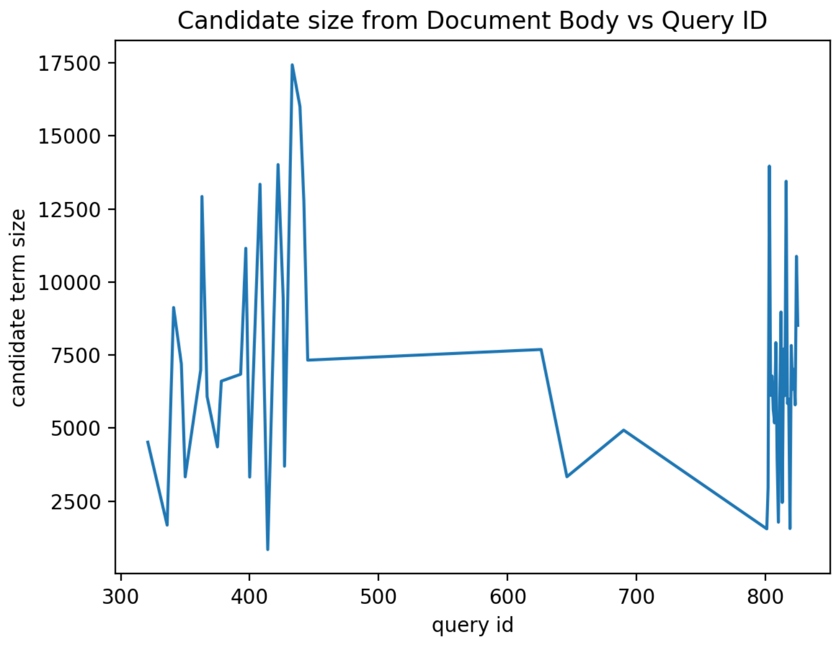


Figure 5.2: Candidate size chosen from Document Bodies vs Query ID

meaningful candidates since they are the essence of the document. The experiments for this comparison can be seen in Table 5.6.

Table 5.6: Comparison between the candidate terms chosen from titles versus candidate terms chosen from document bodies.

					TREC 2018 Common Core Track		
Methodology	Title/Body	One-to-One/One-to-Many	Term Selection	Term Re-weighting	MAP	P@10	NDCG
BERT + RJ	Body	One-to-One	Query Size	0.5	0.2198	0.4060	0.4954
BERT + RJ	Title	One-to-One	Query Size	0.5	0.2563	0.4580	0.5383
BERT + RJ + DTW	Body	One-to-One	2 * Query Size	Dynamic	0.2905	0.5060	0.5675
BERT + RJ + DTW	Title	One-to-One	2 * Query Size	Dynamic	0.3335	0.5420	0.6046

In our opinion, document expansion methodology is more suitable for data-driven approaches like deep learning models due to its colossal size. On the other hand, query expansion methodology is more suitable for algorithmic approaches like graph traversal. For example, locating candidate terms and query terms in the graph nodes and traversing the graph, we can find meaningful terms from the graph. Therefore, this comparison needs more research to observe their performance difference. We discuss the potential improvements to this thesis in Chapter 6.

5.4.6 Best Configuration

As a final remark, we want to point out the best combination of methods. From previous experiments, we concatenate some of the important observations and form the final table. The highest performance (BERT+RJ+DTW) is achieved using titles in choosing data sources, one-to-one association, and dynamic re-weighting. We conducted several experiments in the term selection step. With a varying size of candidate terms, we get the highest result when the candidate terms are chosen from the top 20. The complete results can be seen in Table 5.7. To the best of our knowledge, the best score listed in the table is the highest score that we have observed in this dataset.

Table 5.7: Results of different methods combined in one table

					TREC 2018 Common Core Track		
Methodology	Title/Body	One-to-One/One-to-Many	Term Selection	Term Re-weighting	MAP	P@10	NDCG
JARIR-sg-re(Best)	Title	One-to-Many	Query Size	0.5	0.2040	0.3700	**0.4861**
BERT	Title	One-to-One	Query Size	0.5	0.2049	**0.3720**	0.4696
BERT	Title	One-to-Many	Query Size	0.5	**0.2101**	0.3700	0.4793
BERT	Title	One-to-One	2 * Query Size	0.5	0.1876	0.3420	0.4418
BERT	Title	One-to-Many	2 * Query Size	0.5	0.1776	0.3180	0.4442
Standard BM25	-	-	-	-	0.2495	0.4580	0.5375
BERT + PRF + DTW	Title	One-to-Many	Query Size	Dynamic	0.2390	0.4500	0.5117
BERT + PRF + DTW	Title	One-to-One	Top35	Dynamic	0.2693	0.4540	0.5425
Standard BM25 + RM3	-	-	-	-	0.3135	0.5120	0.5889
BERT + RJ	Body	One-to-One	Query Size	0.5	0.2198	0.4060	0.4954
BERT + RJ	Title	One-to-One	Query Size	0.5	0.2563	0.4580	0.5383
BERT + RJ	Title	One-to-Many	Query Size	0.5	0.2521	0.4300	0.5375
BERT + RJ + Tf-Idf	Title	One-to-One	Query Size	Dynamic	0.3185	0.5560	0.5978
BERT + RJ + TC	Title	One-to-One	Query Size	Dynamic	0.3081	0.5060	0.5875
BERT + RJ + DTW	Title	One-to-One	Query Size	Dynamic	0.3174	0.5340	0.5893
BERT + RJ + DTW	Body	One-to-One	2 * Query Size	Dynamic	0.2905	0.5060	0.5675
BERT + RJ + DTW	Title	One-to-One	2 * Query Size	Dynamic	0.3335	0.5420	0.6046
BERT + RJ + DTW	Title	One-to-One	Top 20	Dynamic	**0.4028**	**0.6540**	**0.6639**

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we demonstrated the use of BERT in query expansion to find similar terms to the query. To extract word embeddings from BERT, we applied different methods and compared their performance on a manually created dataset. According to the results, we chose the best word embedding extractor from BERT and used First Layer's embeddings in representing the word in vector space.

First, we focused on our primary research goal and compared BERT performance on representing words with the Word2Vec model, one of the well-known models for representing terms in vector space. To compare our study with a recent work[2], we used the same dataset, TREC 2018 Common Core Track, and the same test environment. We observed their performance with three different evaluation metrics, MAP and P at k, and NDCG. By looking at the results of various experiments, we conclude that BERT contextualized embeddings are better used in query expansion than Word2Vec word embeddings.

Moreover, we adapted relevance feedback information to rank the terms more accurately. In Chapter 3, we were only choosing the candidate terms and assigning them the same constant weight in the expanded query. However, this approach is not so accurate because not all candidate terms should have the same relevancy to the query. Adapting relevance feedback to query expansion provided us with two main advantages. The first one was reducing the size of the data source. Since we have a fewer amount of documents to inspect, we could also use document bodies from which we can select the candidate terms. We studied using both titles and document bodies as a resource for query expansion. We observed that expanding from title is highly preferable over expanding from document body. The second advantage of using relevance feedback is to adapt tf-idf and term co-occurrence scores to query expansion techniques. Using tf-idf and term co-occurrence scores bring the advantage of using dynamic weights instead of constant values. We showed their contributions using different evaluation metrics in Table 5.7. As the last point, we also represent our best configuration, which is achieved by using the top 20 candidates. To the best of our knowledge, this is the best score done in the dataset.

As future work, we want to use the findings in this thesis to develop new ideas for using BERT in personalized search. Like news document ranking in this study, we can adapt user query logs to offer a personalized search environment to each user separately. Since each clicked URL on the web is also a document, we do not need to change our architecture. To offer a personalized search, we can create user profiles by using titles and bodies of user-clicked URLs. In place of relevance feedback information in this thesis, the session information and query logs can be used because we can refer that the user is interested in that page if he clicks an URL after seeing all the results. We plan to conduct this experiment on the AOL dataset [1], which offers millions of queries and click URLs.

In addition to a personalized search environment, we also want to revise our method in extracting the word embeddings. Enlarging a manual dataset and using deep learning models may give better indicators to show the best technique to extract word embeddings from BERT. We also want to integrate Docker to our project since reproducibility is essential more than ever in today's research. Apart from these, we can train our BERT model on the dataset to get more meaningful word representations. Finally, in our opinion, there are various ways of researching this topic. We plan to do each of them in our future projects.

[1] https://jeffhuang.com/search_query_logs.html

REFERENCES

[1] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*, 2019.

[2] Billel Aklouche, Ibrahim Bounhas, and Yahya Slimani. Query expansion based on nlp and word embeddings. In *TREC*, 2018.

[3] Hiteshwar Kumar Azad and Akshay Deepak. Query expansion techniques for information retrieval: A survey. *Information Processing & Management*, 56(5):1698–1735, 2019.

[4] Jagdev Bhogal, Andrew MacFarlane, and Peter Smith. A review of ontology based query expansion. *Information processing & management*, 43(4):866–886, 2007.

[5] Bodo Billerbeck and Justin Zobel. Document expansion versus query expansion for ad-hoc retrieval. In *Proceedings of the 10th Australasian Document Computing Symposium*, pages 34–41. Citeseer, 2005.

[6] Claudio Carpineto, Renato De Mori, Giovanni Romano, and Brigitte Bigi. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems (TOIS)*, 19(1):1–27, 2001.

[7] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *Acm Computing Surveys (CSUR)*, 44(1):1–50, 2012.

[8] Kerry G Coffman and Andrew M Odlyzko. Growth of the internet. In *Optical fiber telecommunications IV-B*, pages 17–56. Elsevier, 2002.

[9] Kevyn Collins-Thompson and Jamie Callan. Query expansion using random walk models. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 704–711, 2005.

[10] Ovidiu Dan and Brian D Davison. Measuring and predicting search engine users' satisfaction. *ACM Computing Surveys (CSUR)*, 49(1):1–35, 2016.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891*, 2016.

[13] Annegret M Gross. Search engine behavior and satisfaction of arab students from a user perspective. *Int. J. Comput. Linguist. Res.*, 5(3):85–98, 2014.

[14] Bernard J Jansen and Michael D McNeese. Evaluating the effectiveness of and patterns of interactions with automated searching assistance. *Journal of the American Society for Information Science and Technology*, 56(14):1480–1503, 2005.

[15] Susan Jones, Mike Gatford, Steve Robertson, Micheline Hancock-Beaulieu, Judith Secker, and Steve Walker. Interactive thesaurus navigation: intelligence rules ok? *Journal of the American Society for Information Science*, 46(1):52–59, 1995.

[16] Diane Kelly and Jaime Teevan. Implicit feedback for inferring user preference: a bibliography. In *Acm Sigir Forum*, volume 37, pages 18–28. ACM New York, NY, USA, 2003.

[17] Robert Krovetz. Viewing morphology as an inference process. *Artificial intelligence*, 118(1-2):277–294, 2000.

[18] Saar Kuzi, Anna Shtok, and Oren Kurland. Query expansion using word embeddings. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 1929–1932, 2016.

[19] Yinghao Li, Wing Pong Robert Luk, Kei Shiu Edward Ho, and Fu Lai Korris Chung. Improving weak ad-hoc queries using wikipedia asexternal corpus. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 797–798, 2007.

[20] Melvin Earl Maron and John Larry Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):216–244, 1960.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[23] 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.

[24] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[25] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.

[26] Ramith Padaki, Zhuyun Dai, and Jamie Callan. Rethinking query expansion for bert reranking. In *Proceedings of the 42nd European Conference on Information Retrieval (ECIR 2020).*

[27] Jiaul H Paik, Dipasree Pal, and Swapan K Parui. Incremental blind feedback: An effective approach to automatic query expansion. *ACM Transactions on Asian Language Information Processing (TALIP)*, 13(3):1–22, 2014.

[28] Dipasree Pal, Mandar Mitra, and Kalyankumar Datta. Improving query expansion using wordnet. *Journal of the Association for Information Science and Technology*, 65(12):2469–2478, 2014.

[29] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[30] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

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

[32] Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608*, 2016.

[33] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American society for information science*, 41(4):288–297, 1990.

[34] Amanda Spink, Dietmar Wolfram, Major BJ Jansen, and Tefko Saracevic. Searching the web: The public and their queries. *Journal of the American society for information science and technology*, 52(3):226–234, 2001.

[35] Ellen M Voorhees. Query expansion using lexical-semantic relations. In *SIGIR'94*, pages 61–69. Springer, 1994.

[36] Xing Wei and W Bruce Croft. Modeling term associations for ad-hoc retrieval performance within language modeling framework. In *European Conference on Information Retrieval*, pages 52–63. Springer, 2007.

[37] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*, 2019.

[38] 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*, pages 505–514, 2017.

[39] Haoyu Zhang, Yeyun Gong, Yu Yan, Nan Duan, Jianjun Xu, Ji Wang, Ming Gong, and Ming Zhou. Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243*, 2019.