RELEVANCE ANALYSIS FOR DOCUMENT RETRIEVAL

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Eric LaBouve

March 2019

COMMITTEE MEMBERSHIP

TITLE:	Relevance Analysis for Document Retrieval
AUTHOR:	Eric LaBouve
DATE SUBMITTED:	March 2019
COMMITTEE CHAIR:	Lubomir Stanchev, Ph.D.
COMMITTEE MEMBER:	Alex Dekhtyar, Ph.D.
COMMITTEE MEMBER:	Franz Kurfess, Ph.D.

Abstract

Relevance Analysis for Document Retrieval

Eric LaBouve

Document retrieval systems recover documents from a dataset and order them according to their perceived relevance to a user's search query. This is a difficult task for machines to accomplish because there exists a *semantic gap* between the meaning of the terms in a user's literal query and a user's true intentions. Even with this ambiguity that arises with a lack of context, users still expect that the set of documents returned by a search engine is both highly relevant to their query and properly ordered.

The focus of this thesis is on ad-hoc document retrieval systems, which explores methods of ordering documents from unstructured, textual corpora according to textual search queries. The main goal of this study is to enhance the Okapi BM25 document retrieval model. In doing so, this research hypothesizes that the structure of text inside documents and queries hold valuable semantic information that can be incorporated into the Okapi BM25 model to increase its performance. Modifications that account for a term's part of speech, the proximity between a pair of related terms, the proximity of a term with respect to its location in a document, and query expansion are used to augment Okapi BM25 to increase the model's performance. The study resulted in 87 modifications which were all validated using open source corpora. The top scoring modification from the validation phase was then tested under the Lisa corpus and the model performed 10.25% better than Okapi BM25 when evaluated under mean average precision.

Acknowledgements

Thank you to my parents who, with their unconditional love and support, push me to greater heights, inspire me to achieve the best within myself, and afford me the opportunity to lead a life filled with unlimited opportunities. Thank you to my loving big sister who cuts for me a path through life, reminds me of my mistakes so that I may grow into a better man, and poses as a symbol of excellence. Lastly, thank you to my dog Sandy who is such a good girl.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Document retrieval systems are useful tools that allow users to obtain a set of documents from a textual user query. In the field of information retrieval, a collection of documents is called a corpus. When people discuss information retrieval systems, most people think about Google's search engine. At its core, Google's search engine is much like a simple document retrieval system, except the corpus it is analyzing is the entire Internet. Their search engine works well because documents on the Internet are highly structured. For example, web pages are structured using HTML, which contain tags that provide context for the document. In addition to analyzing contextual information derived through HTML tags, Google's search engine is able to return relevant web pages because it prioritizes "popular" web pages. Popularity is scored using their page rank algorithm, which is outside the scope of this introduction. However, due to the technical nature of the page rank algorithm, Google's search engine fails when users are trying to search for documents that are unpopular. This is a problem because documents can be both unpopular and relevant to the user's search query. As a result, some documents are circularly discovered by many individ-

uals and many other relevant and unpopular documents go undiscovered. This thesis focuses on improving document retrieval by developing methods to rank unstructured documents based on their contents without relying on preexisting document structure, corpus context, or popularity scores.

It is important to research alternative ways to rank documents for a handful of reasons. First, systems that primarily rely on training data will not operate well if the domain of the training data is disjoint from the domain of the deployment environment. To generalize these models to many datasets, research has been conducted to try to minimize overfitting on training data [3, 34]. Ideally, the subject of training data should be broad enough to cover all possible domains, however such an idealized set of data is difficult to gather. A system that heavily relies on training data will need to be retrained if the system is deployed in an environment which has a context that differs from the domain of the training data. Retraining or partially retraining a document retrieval system can take a significant period of time and also requires substantial effort from developers to collect and clean training data. On the other hand, an ad-hoc document retrieval model that does not require any training data can ideally be plugged into any corpus and operate effectively without any domain specific knowledge. This would save developers valuable time that can be spent on other priorities. A second reason to research alternative document ranking methods is that the structure of a corpus cannot always be assumed. As alluded to earlier in this chapter, Google's page rank algorithm relies on structured content that exists in HTML web pages, such as hyper links, to grade the popularity of web domains. However, not all corpora contain structured content. Thus, the success of Google's search engine is limited to corpora that contain these key contextual items. In a similar methodology to Google's page rank algorithm, some researchers have experimented with

clustering documents based on bibliographic citations located in a paper's bibliography [38]. Graphing documents based on their citations is a useful way to analyze the differences and similarities between documents, but this method is naturally limited to corpora that only contain documents with extensive bibliographic information. This is another example where the success of the document retrieval model is dependent on the specific structure of its documents. Ideally, researchers would like to develop a model that analyzes documents based just on their content and return accurate results without having to rely on structured data or metadata. Such a model would perform efficiently on all corpora because the system's performance would be agnostic to any preexisting structure. Third, it is often the case in a classroom or an industry setting where building a minimal viable product in a short amount of time is important. In these time constrained environments, there simply isnt enough time to train a sophisticated model, build a bibliographic similarity network, or insert structure into all documents inside a corpus. Instead, it might be better to use an ad-hoc document retrieval system that is guaranteed to operate efficiently and effectively on any corpus to speed up initial development and still maintaining high accuracy.

In addition to understanding the advantages of researching ad-hoc document retrieval methods, it is also important to understand why ranking documents based on their content is a difficult and unsolved problem. First, users are unpredictable. Users may have different expectations for the acceptable degree of relevance and ordering for a list of documents for a particular search query. Second, there commonly exists a mismatch between the meaning of the user's literal search query and the true meaning behind what the user intended to type. This mismatch between the user's query and the user's desire is known as the *semantic gap*. Closing the semantic gap is a fundamental area of research that directly af-

fects how well a computer can differentiate between the user's literal search query and the user's desire. If closed, search engines would be able to return relevant documents with extremely high accuracy. Third, languages are dynamic with respect to time and location. This commonly occurs when a written language is shared across multiple cultures. For example, slang is extremely difficult to interpret because its meaning is dependent on the user's culture. Text that reads, Eric likes chips, can be interpreted differently depending on the user's culture. A user from America might had intended the statement to mean, Eric likes tortilla chips, whereas a user from Australia might had intended the statement to mean, Eric likes French fries because French fries are called chips in Australia. Fourth, as information grows at an exponential rate, search engines are expected to return relevant documents in a timely manner. So, data structures and algorithms must be built to perform efficiently. However, this becomes an issue for modern computers when there are many millions, or billions, of documents that need to be quickly searched. In this case, system designers must decide if it is reasonable to search through every document in the corpus or a subset of documents.

As one might imagine, there are many more variables that can be considered when designing a document retrieval system. In addition to the above complexity, improving a document retrieval system is much like working with a black box. This is true because in real-world circumstances, the true relevance rating for a document, according to a search query, is not fully observable. Also, the many nuances of a language makes it difficult to design relevance criteria that can be applied to all corpora. As a result of these constraints, this thesis proposes hypotheses on corpus content and then designs and runs experiments to validate or to invalidate these hypotheses. If all experimental results pertaining to a particular hypothesis produce a better ordering of predefined relevant documents,

then this will be seen as evidence that the hypothesis is correct.

This thesis will explore methods to improve a well known probabilistic document retrieval model called Okapi BM25 [32]. The overarching hypothesis is that the Okapi BM25 model is limited in quality of results because it is a bag of words approach to document retrieval. For example, the model does not take into account term proximity, query expansion, and term parts of speech. The model also lacks the ability to recognize semantically similar terms. For example, the word "smart is treated completely differently than the word "intelligent. This thesis explores 42 different modifications to address the above issues. When the modifications are systematically combined together, 87 unique variations of the Okapi BM25 model are produced. Each model is then validated against four difficult datasets and graded according to their mean average precision scores. The very best model from the validation set is then tested against a separate, large dataset. The resulting model is able to outperform the original Okapi BM25 model in mean average precision by 10.25% and shows an increase in performance when evaluated over a precision recall curve.

The rest of this thesis is structured as follows. Chapter two provides background on document retrieval models and relevant data structures. Chapter three explores related research on term proximity, semantic analysis, topic modeling, and language modeling. Chapter four provides implementation details for the various modifications to Okapi BM25. Chapter five discusses methods of validation, the relevant datasets, experimental protocols, and the results of the experiments. Lastly, chapter six gives concluding remarks and avenues for future research.

Chapter 2

Background

Chapter 2 begins by briefly describing how Google's search engine uses structured content and how the engine can fail. The section that follows describes unstructured search and why unstructured search is valuable. Then, the mathematical foundation for document retrieval systems will be presented. This theoretical section will describe the vector space model, the probabilistic model, and how documents are stored in information retrieval systems. Afterwards, the limitations of the vector space and probabilistic models are discussed by recognizing a semantic gap that exists between a user query and a set of documents. To help close the semantic gap, two open areas of research are introduced: WordNet and word embeddings.

2.1 Google Search

When the topic of search is brought up in conversation, many people will first think of Google's search engine. But many are unaware of how it works and how it might fail. Google's search engine organizes the web into a massive index

using software programs called spiders [14]. Spiders take advantage of structured content inside HTML elements to make sense of the web pages. Spiders use hyper links to jump between web pages in order to discovered new web pages. The observable web is all the websites that are discovered by the spiders.

When a user submits a search query, Google's search engine queries its enormous index and presents relevant links to the user [13]. The relevance for a web page is highly influenced by the number and weight of hyper links that point to a web page, as described in Google's page rank algorithm [28].

Despite the successes of Google's search engine, there exist situations where Google's search engine fails. Due to the very nature of Google's page rank algorithm, Google gives priority to web documents that are visited frequently and are cited by popular websites. As a result, Google rarely orders documents with a small number of visits first, even if the content of the document is relevant to the query. This is a problem because unpopular documents can still contain credible and useful information. For example, a peer-reviewed paper published in a small conference which contains relevant information may never get returned first according to Google's page rank algorithm. As a result of Google's page rank algorithm, the documents which have the highest page rank are returned, which leads to a circular discovery of the same information.

2.2 Unstructured Search

An alternative way to process queries and documents is to rank documents without considering explicit inter and intra document structure. It is important to explore how search engines can rank documents using unstructured methods because not every corpus will contain structured content. In fact, there are

many document collections that have very little structure. An example of an unstructured corpus is a collection of transcribed phone calls. If a user would like to search for conversations using the query, "couples discussing their family vacation plans," a search engine would have to calculate the relevance of each document based on document and query text. These types of datasets are popular amongst speech recognition researchers and developers [40]. Another example of a document collection with no explicit structure is a set of research paper abstracts. Digital libraries, such as ACM and IEEE, have search engines that allow researchers to search papers based on a paper's abstract. Hence, researching new methods to rank unstructured text has the possibility to enhance these search engines.

2.3 Document Retrieval Theory

A formalized mathematical understanding of document retrieval systems is now presented in order to build the foundation for introducing the Okapi BM25 model and related works.

2.3.1 The Vector Space Model

The technical underpinnings of a document retrieval system can be formalized mathematically [21]. Allow the set of distinct terms in a vocabulary to be denoted by $V = \{w_1, w_2, ...\}$ and the corpus of documents to be denoted by $D = \{d_1, d_2, ...\}$. Both documents and queries can be represented as vectors of length $|V|$ and will contain a subset of unique terms from V. Each index in a vector is the count for a unique term in the document or query and each index

represents the same term for all vectors with the same vocabulary. The equations for document and query vectors are shown here:

$$d_j = (w_{1j}, w_{2j}, ..., w_{|V|j}) \tag{2.1}$$

$$q = (w_1, w_2, ..., w_{|V|}) \tag{2.2}$$

where d_j is a document in the corpus D and w_{ij} is the count for a word in d_j. Likewise, q is a query and w_i is the count for a word in q. When the ordering of words in a representation is ignored, the representation is called a *bag of words*. It is important to notice that document and query vectors are extremely sparse, meaning most of the elements inside a vector will be equal to zero. This is true because queries and documents will not contain a large proportion of the total vocabulary. For example, if the vocabulary contains 10,000 distinct terms and the text of a query is "yellow fluffy puppies," then the vector representation of this query will contain three ones, each located at the corresponding index for "yellow," "fluffy" and "puppy," and 9,997 zeros in every other index location.

The size of the vocabulary can grow extremely large, so it is important to only store terms that have semantic meaning. It is very common for search engines to filter out "stop words," which are terms that hold very little semantic meaning, such as {"a", "of", "is", "the"}. In addition to filtering out stop words, many search engines will reduce terms down to their roots in order to further compress the vocabulary size, a process known as *stemming*. For example, "doggy" and "dogs" share the root "dog," so any derivation of "dog" found in a document or vector will be recorded in the "dog" index. Filtering a corpus for stop words and stemming terms is very common in information retrieval experiments [37, 10, 44,

31, 43, 36].

The vector space model slightly modifies the vector representation of documents and queries and uses a geometric distance function to compare the similarity of vectors. The vector space model recognizes the limitations of only accounting for the frequency of terms inside a document. For example, the term "dog" may appear frequently inside a dog-themed corpus. So, the word "dog" holds less meaning than other words found in the corpus. The vector space model fixes this issue by accounting for how rare terms are with respect to all other terms inside the corpus. This property is known as the inverse document frequency of a term and is represented as the logarithmic quantity below:

$$idf_i = log\frac{|D|}{df_i} \tag{2.3}$$

where idf_i is the inverse document frequency for a term i, $|D|$ is the number of documents in the corpus and df_i is the number of documents in the corpus that contain the term i. Notice that each term in a corpus is assigned an idf value that does not change unless the corpus changes.

The vector space model also takes into account the frequency of a term in a document. This is known as the term frequency and is represented as the quantity below:

$$tf_{ij} = \frac{f_{ij}}{max\{f_{1j}, f_{2j}, ..., f_{|V|j}\}} \tag{2.4}$$

where tf_{ij} is the term frequency for a term i and a document j. f_{ij} is the number of times a term i appears in document j and $max\{f_{1j}, f_{2j}, ..., f_{|V|j}\}$ is the highest term frequency in document j. Notice that the tf value for a term changes with respect to the document.

The the weight for a term inside a document or query vector is the product between the term frequency and the inverse document frequency:

$$w_{ij} = tf_{ij} * idf_i \tag{2.5}$$

This tf-idf equation allows a query vector to be compared against document vectors with respect to a single corpus and a single vocabulary. Since documents and queries are represented as vectors, the similarity between two vectors can be computed by taking the cosine of the angle between the two vectors.

$$cosine(q, d_j) = \frac{< q \bullet d_j >}{||q|| \times ||d_j||} \tag{2.6}$$

The idea of the vector space model is that vectors that point in similar directions will be related to each other. When restricting values to the first quadrant (since documents and queries cannot have a negative number of terms), the cosine of the angle between two vectors gives a value from zero to one. A score that is close to zero indicates that two vectors are relevant because they point in similar directions. A score that is close to 1 indicates that two vectors are irrelevant because they point in different directions. Figure 2.1 shows a graphical representation of the cosine similarity metric.

2.3.2 The Probabilistic Model

A well known information retrieval model is Okapi BM25 [32]. This model is rooted in statistics and assumes a bag of words interpretation for documents and queries. From statistics, the model assumes that an occurrence of a query term in a document is an independent event. These events happen in a specified

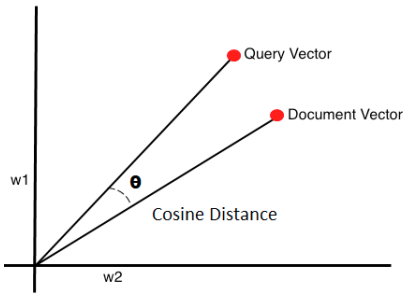

Figure 2.1: The similarity between a query and document in the vector space model is computed as the cosine of the angle between the two vectors.

interval, the start and end of a document. The probability that an event occurs is proportional to how rare a term is in a document collection (inverse document frequency). These criteria satisfy a Poisson distribution [15]. However, a Poisson distribution requires that the rate of occurrences for an event (terms in a document) is known ahead of time. This is problematic because there is no way of knowing the exact mean number of occurrences for a term in a given document ahead of time. Therefore, Okapi BM25 was built as an approximation of the Poisson distribution. The model's equation bellow has been broken down into multiple components to help with the model's overall explanation:

$$okapi(d_j, q) = \sum_{t_i \in q, d_j} idf_i \times tf_{ij} \times qtf_i \tag{2.7}$$

Equation 2.7 gives a high level overview of the model's ranking function. The model takes two inputs, a document d_j and a query q and loops through each term t_i that appears in both the document and the query. The score for a term is the product of three parts: the inverse document frequency (Equation 2.8), the term frequency (Equation 2.9), and the query term frequency (Equation 2.10). The overall score for a pair of vectors is the sum of all the values. It is important to notice that the output value of the equation is unbounded, unlike the cosine

metric used in the vector space model. Higher scores indicate that two vectors are similar to each other and lower scores indicate that two vectors are dissimilar to each other.

$$idf_i = \ln \frac{|D| - df_i + 0.5}{df_i + 0.5} \tag{2.8}$$

The first component of the model is the inverse document frequency. The idf for term t_i (Equation 2.8) is a logarithmic function that gives a higher reward to terms that occur infrequently in the document collection. Similar to Equation 2.3 from the vector space model, idf_i is a function of the number of documents in the corpus, $|D|$.

$$tf_{ij} = \frac{(k_1 + 1)f_{ij}}{k_1(1 - b + b\frac{dl_j}{avdl}) + f_{ij}} \tag{2.9}$$

The second component of the model is the term frequency. The tf for a term t_i in a document d_j (Equation 2.9) is a linear function that gives a higher reward to terms that occur frequently in small documents. Term frequency punishes the document d_j if the length of a document dl_j is longer than the average document length $avdl$ in the corpus. It is important to distinguish between longer and shorter documents because longer documents have more opportunities to contain query terms. b is a hyper parameter that adjusts how much a document is punished for its length and k_1 is another hyper parameter that adjusts the weight of the term frequency with respect to the entire model.

$$qtf_i = \frac{(k_2 + 1)f_i}{k_2 + f_i} \tag{2.10}$$

d_1 = "I heart APIs. You heart APIs"
d_2 = "I use APIs at work"
d_3 = "You work too much"

Figure 2.2: An example corpus containing three short documents.

I: $[< d_1, 1 >, < d_2, 1 >]$	heart: $[< d_1, 2 >]$	APIs: $[< d_1, 2 >, < d_2, 1 >]$
You: $[< d_1, 1 >, < d_2, 1 >]$	use: $[< d_2, 1 >]$	at: $[< d_2, 1 >]$
work: $[< d_2, 1 >, < d_3, 1 >]$	too: $[< d_3, 1 >]$	much: $[< d_3, 1 >]$

Figure 2.3: An example inverted index derived from the corpus in Figure 2.2.

The third component of the model is the query term frequency. The *qtf* for a term t_i (Equation 2.10) is a linear function that gives higher rewards for terms that appear multiple times in a query. f_i is the frequency of a term t_i in a query q. k_2 is a hyper parameter that adjust the influence of the query term frequency with respect to the entire model.

2.3.3 The Inverted Index

An inverted index is an efficient data structure for digesting the information found within corpora by mapping each term in the vocabulary to a list of postings. A posting consists of a list of documents that contain the specified term and other relevant information, such as the term frequency. For example, examine the small corpus consisting of the three documents in Figure 2.2. The entries of the inverted index that would be produced from the three documents is shown in Figure 2.3. For each resulting entry in the inverted index, the key is the term on the left, the value is the list on the right, and a posting is represented as the values inside a pair of angle brackets. Within each posting, the first element is a reference to the document that contains the key term and the second element is the term frequency of the key term within the document. The inverted index

is a helpful data structure in information retrieval systems because it abstracts away all the needed information found within a corpus. After the contents of a corpus is transformed into entries in an inverted index, the corpus can usually be discarded from memory. Inverted indexes also dramatically increase the lookup time to obtain a set of documents that contain a particular word. Since the vector space model and the probabilistic model both rely on looking up documents that contain query terms, it makes sense why document retrieval systems utilize inverted indexes.

2.4 The Semantic Gap

One of the largest issues with the vector space and probabilistic retrieval models is their inability to cope with terms that are semantically similar and do not stem to the same root. For example, "smart" and "intelligent" are treated differently even though they have high semantic similarity. Often, users are more interested in the concepts that their queries represent rather than the exact phrasing of their queries. A document retrieval model that evaluates the query "smart animals" should be able to assign a high score to documents that contain the phrase "intelligent animals." This limitation is known as the *semantic gap*. The research conducted in this thesis utilizes the following two tools to help close the semantic gap during experiments.

2.4.1 WordNet

To help close the semantic gap, researchers from Princeton University have attempted to organize the semantic relationships between English terms into cog-

nitive synonyms in a project called WordNet [25]. A set of terms are considered to be cognitive synonyms, or *synsets*, of one another if the meanings of the terms are so similar to one another that they cannot be differentiated. From these synsets, conceptual, semantic, and lexical relations are built. The WordNet database can be used to extract valuable semantic information from terms, such as term definitions, various parts of speech, synonyms, hypernyms, hyponyms, and links to other related terms.

Words in the English language can have multiple definitions and uses so it is important to recognize these differences. A term inside WordNet subscribes to a set of synsets, where each synset has a different meaning. For example, the word "dog" belongs to eight different synsets. This is because dog can take on different meanings depending on the context of its usage. Some of the synsets that dog subscribes to are: a domestic dog of the Canis family, a smooth-textured sausage of minced beef or pork usually smoked, an informal way to refer to someone, etc. So determining the meaning of a word in a sentence is not as straightforward as looking up the definition of a term inside WordNet. There needs to be a way to determine the correct synset based on the term's context. An area of research that aims to determine the correct usage of a term inside its context is called *word sense disambiguation* and a classic algorithm for determining the correct synset for a word is the Lesk Algorithm [20]. Given an ambiguous word and the context in which the word occurs, Lesk returns a synset with the highest number of overlapping words between: the various definitions from each synsets of each word in the context sentence and the different definitions from each synset of the ambiguous word. Once the appropriate synset is selected, semantically similar words can be chosen by selecting terms that subscribe to this synset. An in depth

example of the Lesk algorithm ran on WordNet can be found in the footnote[1].

Although WordNet shows that there exists a relationship between terms, such as "car" and "automobile," WordNet does not provide the strength of the relationship between terms. Research has demonstrated that a graph can be extracted from the WordNet database to represent the sematic similarity between English terms [39]. In this graph, each node is represented by a term or phrase and each edge holds a weight that represents the probability that a user is interested in another term or phrase when given the current node. In this graph representation of WordNet, the sum of probabilities from all the out edges is equal to one. The similarity between two terms is computed by performing a breadth-first traversal of the graph from each node in parallel to discover a path between the two nodes and then computing the product of the edges along this path [39]. In addition to computing the similarity between two known terms, semantically similar terms can be discovered for a given term by performing many random walks to discover neighboring nodes. A random walk is performed by randomly selecting an out edge, according to the probability distribution from the set of out edges, and traversing to the node pointed to by this edge. This process repeats itself for a given number of intervals defined as a hyper parameter. The nodes that are traversed most often represent the most semantically similar nodes.

Unfortunately, WordNet is difficult to maintain. As language evolves, the project demands continued effort in order to keep up with new additions and modifications to the English language. As a result, WordNet does not have thorough entries for slang terms or figures of speech. For example, the phrase "down to Earth" is semantically similar to the words "practical" or "humble," and this is not provided by WordNet. The best WordNet can do is analyze the

[1] http://www.nltk.org/howto/wsd.html

above phrase through its parts, so "down to Earth" is interpreted as something that is literally close to the Earth's surface.

2.4.2 Word Embeddings

Another method that can be used to help close the semantic gap is word embeddings. The core concept behind word embeddings is the Distributional Hypothesis [33]. The hypothesis states that words that appear in the same context share semantic meaning. In this model, each word in a vocabulary is represented as a vector and semantically similar words will have similar vectors. Just as how WordNet can be used to discover semantically similar terms, as can word embeddings. Semantically similar terms can be discovered by looking up a word's vector representation and then using a distance function, such as cosine similarity, to compute the distance between each word in the vocabulary.

Researchers have been experimenting with word embeddings since the early 2000s. One of the first papers to describe and implement a neural language model to produce word vectors was in [5]. The paper proposes a feed-forward neural network with a linear projection layer and a non-linear hidden layer to learn word vector representations and a neural probabilistic language model. This paper's model took three weeks to train across 40 CPUs and produced perplexity scores that were 10% to 20% better than (at the time) state of the art smoothed trigram probabilistic language models.

Since the early 2000s, generating word embeddings have become much more efficient and accurate. In 2013, Google researchers published a technique called Word2Vec to learn high quality word vectors from huge data sets with billions of words and with millions of distinct vocabulary words [22]. Word2Vec is a

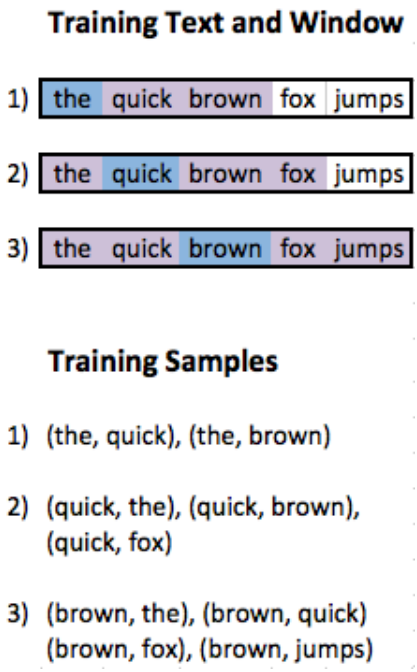

Figure 2.4: Demonstrates how training samples are produced for the Word2Vec neural network from a set of text with a window radius of two.

semantically driven method for representing terms that encode many linguistic regularities and patterns. The paper shows that word vectors could be built in less than a day using a shallow neural network and perform better than other industry standard language models. In a follow up paper by Google researchers [23], the researchers report that the word vectors produced using the Word2Vec method can represent syntactic analogies such as "quick":"quickly" and also semantic analogies, such as country to capital city relationships. This relationship can surprisingly be extracted using simple vector addition. For example,

$$vec(\text{“}Germany\text{”}) + vec(\text{“}capital\text{”}) \approx vec(\text{“}Berlin\text{”}) \tag{2.11}$$

Word vectors allow for precise analogical reasoning and representation of the distributional context in which a word appears. To generate a word embedding

using Google's Word2Vec model, a shallow neural network is used to build feature vectors for each word in the vocabulary. These vectors encode the occurrence of prior and subsequent terms. Visually, prior and subsequent terms are encapsulated by a window, where the size of the window is a hyper parameter. Shown in Figure 2.4, pairs of data points are drawn from the window of words in order to create the training samples for the neural network. For each training sample, the input to the neural network is the center window word and the expected output is a non center window word. Finally, the word vectors are extracted from the weights connecting the hidden layer and the output layer [16].

Chapter 3

Related Works

Chapter 3 explores related research on unstructured search. The first section will discuss previously established modifications to Okapi BM25 to increase the model's accuracy. The next section will discuss other related techniques to analyze unstructured text that do not utilize Okapi BM25.

3.1 Okapi BM25 Modifications

3.1.1 Genetic Programming

Ronan Cummins' and Colm O'Riordan's *Learning in a Pairwise Term-Term Proximity Framework for Information Retrieval* [10] uses a genetic algorithm to modify the Okapi BM25 model. This paper builds off their previous publication, where they propose a genetic algorithm to improve the vector space model [9]. The fitness function for both papers rewards their system when an equation modification results in an increase in the model's mean average precision.

In order to derive a modified version of Okapi BM25, the researchers defines

Proximity Measure	Description
min_dist(a, b, D)	The minimum distance between terms a and b.
diff_avg_pos(a, b, D)	Computes the difference between the average positions of terms a and b.
avg_dist(a, b, D)	The average distance between terms a and b for all positions combinations in D.
avg_min_dist(a, b, D)	The average of the shortest distance between each occurrence of the least frequently occurring term and any occurrence of the other term.
match_dist(a, b, D)	The smallest distance achievable when each occurrence of a term is uniquely matched to another occurrence of a term.
max_dist(a, b, D)	The maximum distance between any two occurrences of terms a and b.
sum(tf(a), tf(b))	The sum of the term frequencies between terms a and b.
prod(tf(a), tf(b))	The product of the term frequencies between terms a and b.
fullcover(Q, D)	The length of the document that covers all occurrences of query terms.
mincover(Q, D)	The length of the document that covers all query terms at least once.
dl(D)	The length of the document.
qt(Q, D)	The number of unique terms that match both the document and query.

Table 3.1: The 12 proximity measurements used as input into the genetic algorithm.

12 different term proximity measures. These proximity measure were fed as input into their genetic algorithm. The algorithm then tried different combinations and weights for each proximity measure and output an equation that maximizes the mean average precision on a set of 69,500 documents and 55 queries. The best three functions produced using the genetic algorithm all resulted in equations that used the minimum distance proximity measure and the average distance proximity measure. This suggests that the minimum distance and the average distance between query terms in a document is correlated to the relevance of a query and document. The proximity measures used in this experiment are defined

in Table 3.1.1.

3.1.2 Semantic Analysis

Bhatia's and Kumar's *Contextual paradigm for ad hoc retrieval of user-centric web data* [6] categorize the semantic relationship between pairs of terms in multi-term queries. The two categories that were defined in the paper were labeled "topic modifying" and "topic collocating." Topic modifying is where one query term represents a subject and the other query term modifies the subject. For example, "Indian Currency" is a topic modifying query. On the other hand, topic collocating is where multiple query terms represent a single topic. For example, "data mining" is a topic collocating query. The researchers propose three different hypotheses to test the properties of topic modifying and topic collocating queries:

1. Important terms always appear in the forefront of a document.
2. Related terms appear in close proximity to one another.
3. Important terms appear repeatedly in a document.

To test these hypotheses, an equation for each of the above hypotheses was derived. Then, 20 topic modifying and 10 topic collocating queries were typed into Google's search engine to obtain the top 20 documents for each query. Each of the documents were then labeled as relevant or irrelevant by a human. The aforementioned equations were then used to reorder the returned documents and the precision was recorded at 5, 10, and 15 documents.

The results show that the reordering of topic modifying queries saw the largest increase in precision when ranking documents based on the smallest distance be-

tween query terms. This makes sense because a modifying term is most useful when found near its corresponding subject. Topic collocating queries saw the largest increase in precision when reordering documents based on their term frequencies and based on the smallest distance between query terms. These results make sense because the appearance of individual topic collocating terms do not have much semantic meaning if they are not found close together.

This thesis adopts the topic modifying and topic collocating principals laid out by Bhatia and Kumar. Unfortunately, the researchers do not provide a mechanism for identifying topic modifying and topic collocating terms. As a result, this thesis assumes that topic modifying and topic collocating terms can be identified by evaluating the parts of speech of adjacent terms. In contrast to this research paper, this thesis explores how these techniques fair with long queries, as the queries used in this research paper were only two terms long. Lastly, this thesis takes inspiration from Bhatia's and Kumar's first hypothesis, that important terms may appear at the front of a document.

3.1.3 Spans

A well researched area in term proximity is the idea of "spans." A span is a segment of text from a document that incorporates all query terms, or a subset of the query terms. Spans have been shown to increase document retrieval accuracy for a number of cases. In Muhammad Rafique's and Mehdi Hassan's *Utilizing Distinct Terms for Proximity and Phrases in the Document for Better Information Retrieval* [31], the two researchers derive an equation that gives a greater reward if a large number of query terms appear close together in a document. The researchers' implementation is unique because the system only

accounts for the first occurrence of each query term. This modification allows the researchers to compress the system's inverted index by only storing the location of the first occurrence of a term inside each posting. To validate their system, the researchers designed 25 custom queries and ran them the C50 dataset. The C50 dataset contains two repositories, one for testing and one for training, each containing 2,500 text files. Each text file is a passage written by a well known author and no information was provided on the custom queries. Their results show an increase in precision when observing the first five and ten returned documents by upwards to 20-30% and an increase to the mean average precision by around 15%, which is evidence that spans are a useful machanism for increasing the accuracy of document retrieval systems.

Researchers Ruihua Song, Ji-Rong Wen, and Wei-Ying Ma used spans to derive contextual information from queries in their paper, *Viewing Term Proximity from a Different Perspective* [37]. The paper introduces a technique to replace term frequency with an algorithm that incorporate term proximity using spans. Their algorithm splits a documents into non overlapping segments that contain one or more query terms according to the rules of the algorithm. Intuitively, the relevance contribution is proportional to the density of each non-overlapping span. Mathematically, the score for an individual span is a function of the number of unique query terms divided by the width of the span, multiplied by two other hyper parameters. Thus, the total relevance contribution, rc, is the sum of scores from all the non-overlapping spans. The new term frequency formula for Okapi BM25 is as shown in Equation 3.1. Experiments were conducted on TREC (Text Retrieval Conference) disks 9, 10, and 11. Their results show an increase in precision when observing the first 5 and 10 returned documents by

around 0.3% for disk 9, 10.4% for disk 10, and 4.4% for disk 11.

$$tf_{ij} = \frac{(k_1 + 1)rc}{k_1(1 - b + b\frac{dl_j}{avdl}) + rc} \tag{3.1}$$

3.1.4 Query Expansion

Query expansion is the process of reformulating a query to improve retrieval performance in information retrieval operations. The main motivation of query expansion is to include additional terms to express the original query in a more detailed way in order to increase the number of relevant documents identified [27]. As discussed in [27], there are three major areas of query expansion: query expansion using corpus dependent knowledge models, query expansion using relevance feedback, and query expansion using language models. Query expansion using corpus dependent knowledge models group similar words together in order to find suitable expansion terms. Query expansion using relevance feedback expands the query by extracting terms from either the first few returned documents or from known relevant document. Extracting terms from the first few returned documents is not very effective for ad-hoc feedback systems because the first few returned documents are not guaranteed to be relevant to the query and will result in query drift [1]. Last, query expansion using language models selects new terms according to the highest probability that the new term will appear in the context of the original query.

There are a number of researchers who have experimented with query expansion. Edward Fox's research *Lexical Relations: Enhancing Effectiveness of Information Retrieval Systems* [12] explores a corpus dependent query expansion method. His algorithm builds unique lexical relations for document collections

and shows how query expansion can affect the recall level of an information retrieval system. His main contribution is that the recall level of information retrieval systems can be improved if the knowledge model being used to expand the query is built from a corpus that shares the same lexical relations as the test corpus.

Research done by Olga Vechtomova, Stephen Robertson, and Susan Jones in their paper *Query Expansion with Long Span Collocates* [41] expounded on the idea of collocates[1]. The researchers aimed to identify all terms that significantly co-occur with query terms within a specified window size. Their algorithm builds a list of possible expansion terms and weights these terms by their significance of association using statistical methods, such as Z-score. The researchers built two different knowledge models for query expansion. The first model was constructed from a global point of view, which included the entire corpus (a corpus dependent knowledge model) and the second model was built from a local point of view which contained a subset of the corpus that contained known relevant documents (a corpus dependent knowledge model with relevance feedback). To test their models, each query was ran using Okapi BM25 to gather average precision and recall scores. Then the query was expanded using one of the two models and was again ran using Okapi BM25 to gather average precision and recall scores. Their results show that Okapi BM25 consistently performed worse when the query was expanded using the global model. When using the local model, the average change in precision both improved and degraded for different sized queries. The researchers reason that their models fail to consistently and correctly expand queries because query terms have a very high level of dimensionality that can be derived from their contexts of occurrence.

[1] Words which co-occur near each other with more than random probability are known as collocates

A modern approach to query expansion uses language models. Language models take advantage of the high levels of the contextual dimensionality that a term can exhibit within a document (language models are explored in detail in Section 3.3). Some researchers, such as Saar Kuzi, Anna Shtok, and Oren Kurland, in their paper *Query Expansion Using Word Embeddings* [17] have successfully been able to expand queries and increase mean average precision using language models trained on the same corpus that their document retrieval system is analyzing. Since training language models from scratch require a lot of training data, their models were trained on large Trec disks, which were also the same datasets that their document retrieval system was analyzing. This type of local training is consistent with the preceding corpus dependent models as it appears that query expansion is best conducted when a lexicon is built from the same domain as the target corpus.

Despite some of the successes of the above researchers, other researchers doubt the potential of query expansion for ad-hoc retrieval. Anton Bakhtin, Yury Ustinovskiy, and Pavel Serdyukov in their work *Predicting the Impact of Expansion Terms Using Semantic and User Interaction Features* [2] suggest that query expansion does not improve query performance. The researchers examine query expansion via corpus dependent knowledge models and claim that query expansions will more than likely hurt the system's recall due to vocabulary mismatch, or hurt the system's precision due to topic drift. The researchers sampled 35,000 unique queries from a Yandex search engine query log and computed the difference between the F-score, precision, and recall for documents being retrieved from a query without any expansions and from a query with expansions. The results of their experiments show that in around 84% of cases, query expansion does not change the query's overall performance, which may imply that query

expansions are not a very efficient mechanism for improving ad-hoc document retrieval systems.

3.1.5 BM25F

Some researchers have proposed an extension of the Okapi BM25 model that assumes that some parts of a document are more relevant to the query than other parts of a document. This idea was first developed for web search and takes advantage of explicit structure inside HTML tags and RDF triples. RDF is a *Semantic Web* technology which stands for Resource Description Framework that was developed to give semantic meaning to elements on web pages [24]. Metadata in the RDF model is expressed as triples: subject, predicate, and object, which are encoded as URI's. For example, a web page on the movie Deadpool can contain a hyperlink to the movie's director, Tim Miller, which indicates that Deadpool was directed by Tim Miller. This information can be expressed explicitly as a RDF triple: (Deadpool, directed by, Tim Miller), where the movie Deadpool is the subject, directed by is the predicate, and Tim Miller is the object. RDF triples allow humans to encode documents with semantic meaning which computers can then use to better understand the content of documents.

Researchers first took advantage of HTML tags and RDF triples to create the BM25F probabilistic retrieval model [30]. BM25F is similar to Okapi BM25 with the addition that for each term, a weight variable is used to either scale up or scale down the relevance of the term depending on the context in which the term is discovered. The weight variable is heuristically set. For example, terms discovered in a HTML title element or a RDF triple subject may be given a positive boost to the term's score if these fields appear to be relevant to the

query. Experiments have shown that using BM25F can improve the Okapi BM25 model when evaluated over precision and mean average precision. The downside to the model is that BM25F requires that documents be structured with HTML and/or RDF triples.

With the growing popularity and success of BM25F, some researchers set out to apply similar techniques to unstructured text. For example, Roi Blanco and Paolo Boldi in their paper *Extending BM25 with Multiple Query Operators* [7] built a model to generalize BM25F to unstructured text. The basic idea of their approach is that a document is split into "virtual regions." Much like Okapi BM25F, each region represents a different level of relevance to the query and will be weighted proportionally to its statistical significance. These virtual regions are generated similar to the span examples in Section 3.1.3, where a high density of query terms in a subsection of a document may indicate greater importance than a less dense area of a document. In a two pass procedure, Blanco's and Boldi's algorithm first partitions a document and assigns weight values to each partition. Then, the document is scored in the same way as BM25F. Their algorithm was tested against five large document collections containing around 95 million documents and they report a consistent increase in mean average precision over both Okapi BM25 and BM25F.

3.2 Topic Models

Topic models attempt to understand the semantic structure of text by assuming that there is some hidden structure in a document that can be discovered and exploit it in order to cluster similar documents.

3.2.1 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is an early topic model that uses singular-value decomposition in order to expose semantic relationships between documents in a corpus [35]. Singular-value decomposition is a mathematical procedure that attempts to reduce the rank of a matrix by approximating the matrix's row (or column) vectors by a smaller set of linearly independent vectors (vectors are linearly independent of each other if two vectors are not scalar multiples of each other or a linear combination of other vectors in the set) [26]. In LSI, the matrix that is to be approximated is the corpus, where each row is a document and each column is a term in the vocabulary.

In an early paper that presents the LSI algorithm [35], the researchers proposed that a document-term matrix can be approximated by a set of 100 linearly independent features that can be linearly combined to approximate each document in the corpus. Each document in the corpus can then be plotted in hyperspace according to the document's linear approximation formula. A document's position in space serves as a document's identity and neighboring documents in this space should have similar topics. Similarly, a query can be analyzed as a weighted combination of terms and be plotted in the same hyperspace. From here, LSI borrows from the vector space model by computing the angle between the query point and the surrounding document points to compute similarity scores.

LSI takes into account the semantic similarities between words by nature of its design because a query with terms that do not appear in a document may still end up close to a document in hyperspace. As a result, a query can return documents with terms that are semantically similar to query terms. However, LSI falls short because it does not take into account words that have multiple

definitions. For example, the term "bark" can be used in two different contexts, "the dog barks" and "tree bark." Therefore, a document discussing how a dog barks and another document discussing tree bark may appear close to each other in hyperspace. Since so many words have multiple use cases, it makes sense to extend LSI to better accommodate words that relate to multiple topics, or a set of high level ideas such as sports, music, education, etc.

3.2.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is similar to LSI, but it is more robust to word ambiguity [11]. LDA is more powerful than LSI because each word in the vocabulary can be expressed by a set of topics with a corresponding set of weights. For example, the term "bark" can be expressed by a mixture of topics such as "dog" and "tree." The word "bark" can also relate to "dog" and "tree" in different proportions, such as 60% "dog" and 40% "tree." These proportions will depend on the subject domain of the corpus. In LDA, a document is a combination of words, where each word belongs to a combination of topics. So a document can also be expressed as a combination of topics and weights, much like a recipe. For example, a document may be 35% sports, 45% music, and 20% education. These topic proportions can then be used to compare queries and documents. For LDA, queries and documents are relevant to one another if they are composed of a similar combination of topics.

```
foreach document d ∈ D do
    foreach word w ∈ d do
        foreach topic t do
            p1 = Proportion of words in d currently assigned to t;
            p2 = Proportion of assignments to t over all documents from w;
            p3 = p1 * p2;
            w ← (t, p3);
        end
    end
end
```

Algorithm 1: High level algorithm for how LDA assigns topics and weights to words in a corpus.

LDA must be trained using a corpus in order to determine the best combination of topics and weights for a particular term. Initially, words are assigned randomly to topics. Then, for each word in every document and every topic, a word is reassigned to a topic according to the probability that a topic generates that word. A rough overview of the pseudocode can be found in Algorithm 1. The disadvantage to LDA is that the number of topics is chosen as a hyper parameter that must be empirically determined. Another disadvantage is that LDA does not take into account the proximity and ordering of terms inside a document because it is a bag of words model. Dispite these disadvantages, LDA is still an effective tool for clustering documents. The next section will explore language models, which emphasize term proximities in order to extract a term's semantic meaning.

3.3 Language Models

A language model can be used to derive semantic meaning from terms using term proximity. Language models represent the semantic meaning of terms as a probability distribution that represents the likelihood that a term can be found near other terms. This term probability distribution can then be used to predict the likelihood that a term may appear near a set of different terms. As a result, language models lend themselves well to solve problems in fields beyond document retrieval, such as speech recognition [42, 40]. A common problem in speech to text programs is that a sound recording can be represented in many different ways, most of which make no grammatical sense. For example, the phrases "I saw a van" and "eyes awe of an" are acoustically similar to one another, but "I saw a van" would be a more likely transcription of the audio file because there is a higher probability that these terms would appear in each other's contexts. Language models solve this problem by calculating the probability that a term will appear in a particular context rather than simply returning terms based on their acoustics. This behavior is exploited by document retrieval engines to both derive query expansion terms that make sense according to the term's context and to directly score the similarity between documents and queries.

3.3.1 The Probabilistic Language Model

A probabilistic language model can be used to score the relevance between a document and a query by computing the probability that a document will generate a query [18]. To perform this calculation, a probability distribution is first built for each document. In unigram language models, terms are treated as independent events. Hence, the probability that a word appears in a document

is given as the conditional probability between the word and the document [21], as shown in Equation 3.2,

$$P(w_{ij}|d_j) = \frac{f_{ij}}{|d_j|} \tag{3.2}$$

where f_{ij} is the frequency of term i in document j and $|d_j|$ is the total number of words in document j. Therefore, the probability that a document generates the terms in a query, $q = (w_1, w_2, ..., w_m)$, of length m is the product of all the probabilities that each query term appears in the document [18], as shown in Equation 3.3.

$$P(q_i|d_j) = \prod_{i=1}^{m} P(w_{ij}|d_j) \tag{3.3}$$

A unigram language model can be expanded to account for a term's context in order to gain more insight into a term's semantic meaning. To do so, the probability of a term will be modified to depend on previous terms in its context, as given by Equation 3.4,

$$P(w_n) = P(w_n|w_1, w_2, ..., w_{n-1}) \tag{3.4}$$

where $P(A, B, C)$ is expanded to $P(A)P(B|A)P(C|A, B)$ and n is the n^{th} term in a document. As one might suggest, accounting for all the preceding terms over complicates the language model. In order to simplify training and to reduce the combination of terms that can appear in another term's context, the probability of a term can be simplified to only depend on a few preceding terms. The number of preceding terms will be indicated by the variable k. This is known as the *Markov Assumption* [8]. Using the Markov Assumption, the probability of a term in a document reduces to Equation 3.5.

$$P(w_n) = P(w_n|w_{n-k}..., w_{n-1}) \tag{3.5}$$

When $k = 1$, or $P(w_n) = P(w_n|w_{n-1})$, the system is called a bigram language model. When $k = 2$, or $P(w_n) = P(w_n|w_{n-2}, w_{n-1})$ the system is called a trigram language model. The probability that a term appears in a document for small values of k now depends on the frequency of phrases that appear before a term and including that term. The probability that a term appears in a document for a bigram model is shown in Equation 3.6,

$$P(w_i|w_{i-1})_j = \frac{count(w_{i-1}, w_i)_j}{count(w_{i-1})_j} \tag{3.6}$$

where $count(w_{i-1}, w_i)_j$ is the number of times the 2-tuple (w_{i-1}, w_i) appears in document j and $count(w_{i-1})_j$ is the number of times the word w_{i-1} appears in document j. For example, if a document contains the text "My dog makes him happy and her happy" then $P(\text{“}and\text{”}|\text{“}happy\text{”}) = \frac{count(\text{“}happy\ and\text{”})}{count(\text{“}happy\text{”})} = \frac{1}{2}$

Language models compute the similarity between a query and a document by determining the probability that a document's probability distribution generates a query. For example, the probability that a query $q_i = (w_1, w_2, ..., w_m)$ is generated from a document in a bigram language model is the product of each conditional probability for each pair of terms, as shown in Equation 3.7.

$$P(q_i) = P(w_1|s) \times P(w_2|w_1) \times P(w_3|w_2) \times ... \times P(s'|w_m) \tag{3.7}$$

In Equation 3.7, s and s' are special symbols reserved for the start and end of queries and document. When k is generalized to any number, the model is called an N-gram language model.

The N-gram language model has a few disadvantages. The number of different possible phrases grows exponentially with respect to the vocabulary size,

especially when k is large. Also, it is not likely that a training set will contain every combination of phrases for high values of k. Thus, the discrete probability distribution that is created most likely has missing values and it is not clear what exactly to do when a query contains a word that is not contained in a document. This is a problem because missing words will result in zero valued probabilities. To fix this problem, language models are usually smoothed to mimic continuous distributions. The main purpose of smoothing is to assign a non-zero probability to unseen words and phrases. One of the simplest smoothing techniques is called Laplace smoothing [45], where an extra count is added to each term. Another popular way to smooth a language model is by using linear interpolation with a background collection model, or by using word sense information from semantic databases (such as WordNet) [19]. Many more techniques are outlined in [45].

3.3.2 Neural Language Models

The absence of a continuous distribution in the probabilistic language model motivates a language model that utilizes neural networks. By the nature of their design, neural networks lend themselves well to generating continuous distributions because the training process will adjust the weights and biases for groups of neurons. Similar to a statistical language model, a neural language model computes the similarity score between a document and a query as the probability that a query is generated by a document. Neural language models are used to generate the high quality Word2Vec word embeddings as described in Section 2.4.2. Unlike statistical language models, neural language models are trained using both previous and subsequent terms. In the case of Word2Vec, two interesting architectures are derived. The Continuous Bag-of-Words (CBOW) model is able to predict a term according to the terms in the current term's context [22]. This

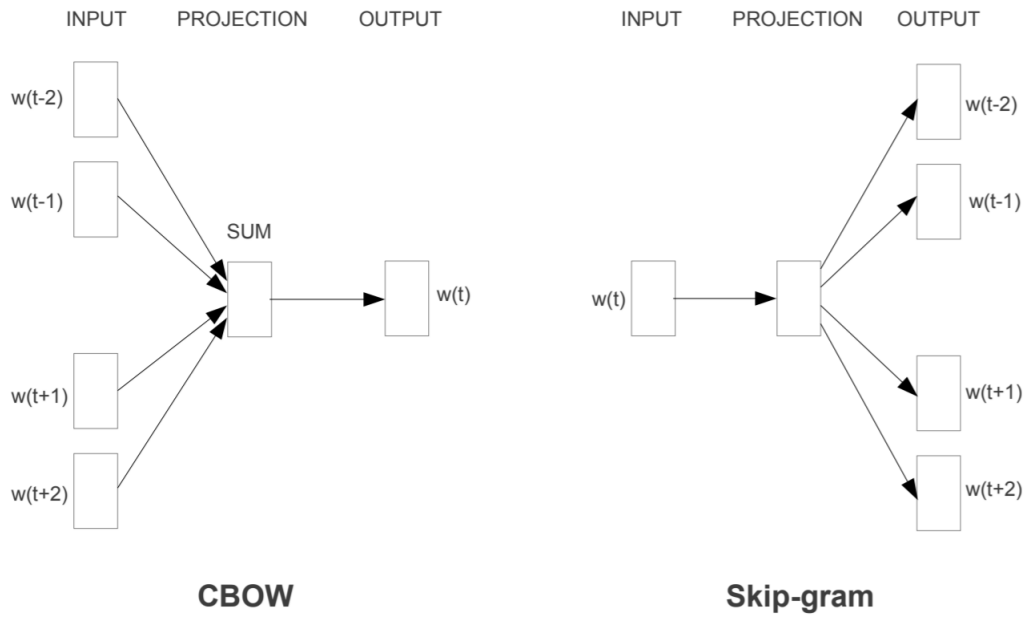

Figure 3.1: The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

behavior is similar to that of the behavior for probabilistic language models. The other Word2Vec architecture is the Continuous Skip-gram model, which tries to classify a word based on another word in the sentence [22]. More precisely, this model is used to predict words within a certain range before and after the current word. The Skip-gram model can hypothetically be used for query expansion because it allows search engines to recognize terms that are likely to appear to each other's contexts. For example, the term "Hong" is commonly found right before the term "Kong." So if a user leaves out "Kong" when searching for the country Hong Kong, the Skip-gram model can be used to infer the user's intension. Figure 3.1 provides a visual summary of the CBOW and Skip-gram models.

Chapter 4

Implementation

This chapter explains the implementation details of relevant system architecture and Okapi BM25 modifications. To make the explanation easier, this chapter has been split into three parts. The first section discusses how information from the datasets is extracted and stored in an inverted index. The second section describes how Okapi BM25 is extended so that sets of modifications can be easily enabled. The third section lists all the modifications that were created and how a term's score is boosted depending on the outcome of these modifications.

4.1 Building the Inverted Index

The goal of this section is to describe how the inverted index for a dataset is built and what information can be found in a term's posting. A lot of data cleaning, data reshaping, and data analysis must be performed before any queries can be executed on a set of documents. As previously presented, a basic inverted index maps terms found inside a corpus to a list of postings, where a posting contains a reference to the original document and a count for how many times

{a, about, above, all, along, also, although, am, an, and, any, are, aren't, as, at, be, because, been, but, by, can, cannot, could, couldn't, did, didn't, do, does, doesn't, e.g., either, etc, etc., even, ever, enough, for, from, further, get, gets, got, had, have, hardly, has, hasn't, having, he, hence, her, here, hereby, herein, hereof, hereon, hereto, herewith, him, his, how, however, i, i.e., if, in, into, it, it's, its, me, more, most, mr, my, near, nor, now, no, not, or, on, of, onto, other, our, out, over, really, said, same, she, should, shouldn't, since, so, some, such, than, that, the, their, them, then, there, thereby, therefore, therefrom, therein, thereof, thereon, thereto, therewith, these, they, this, those, through, thus, to, too, under, until, unto, upon, us, very, was, wasn't, we, were, what, when, where, whereby, wherein, whether, which, while, who, whom, whose, why, with, without, would, you, your, yours, yes}

Figure 4.1: The stop words that were used in all the experiments.

that term appears in that document. In this thesis, the original definition of an inverted index is expanded to incorporate more contextual information about a term inside a document.

The first step in building an inverted index is to gather a set of terms to be used as keys for the data structure. As discussed in Section 2.3.1, evidence from prior research shows that removing stop words can increase a document retrieval system's performance. This thesis pays close attention to minimizing the number of keys in the inverted index so that the search engine can perform experiments quickly. Figure 4.1 shows the set of stop words used in all the experiments. It is worth noting that there is no universal set of stop words used by natural language processing tools.

Additional research presented in Section 2.3.1 shows that the key set for an inverted index can be further reduced without compromising accuracy if inflectional endings from terms are removed. Term inflections can be removed either by stemming or lemmatization. A stemmer crudely removes the ending of a word based on patterns in a particular language, whereas a lemmatizer uses a dictionary

to reduce a term down to it's lemma, or the version of that term that appears in the dictionary. For all experiments, a rule based stemmer called the Porter Stemmer was used due to speed advantages over a lemmatizer. The Porter Stemmer appears to be a standard tool from related information retrieval research from Section 2.3.1. In addition to excluding stop words and using a Porter Stemmer, other minor checks are conducted across all documents to further compress the inverted index. For example, terms that are less than or equal to two characters long are ignored, all terms are converted to lower case, and all punctuation is removed before terms are added to the inverted index.

Now that the procedure for collecting and cleaning keys for the inverted index has been discussed, the contents of a posting can be unraveled. The first two items in a posting are the same from the previous definition of a posting. The first item is a reference to the document containing the term. In the code, document references are represented by a unique identification number. Documents are accessed through a dictionary that maps document id's to document vectors. Second, the posting contains the number of times a key term appears in the document. In addition to these two previously discussed elements, this thesis expands the definition of a posting by adding three new elements. As the program reads through the corpus, the absolute position of the term and the sentence number within the document are recorded. The absolute position for a term represents the number of terms that precede this term in the document. Both of these values are stored in lists whose sizes are equal to the number of times the key term appears in the document. Last, a posting also stores a term's part of speech. Including a term's part of speech is a new modification that is not proposed in any of the related research. Determining the part of speech for a term is not a straightforward procedure because a term's part of speech heavily relies on the

Key Term → [<Document ID, Term Frequency, [Sentence Indices], [Absolute Indices], [Parts of Speech]>, ...]

Figure 4.2: A summary of the inverted index used in this thesis. The arrow represents a mapping function. The angle brackets represent a single posting. The square brackets represent list notation.

context in which the word appears. As a result, a term can take on multiple parts of speech. The English language is especially tricky because there are a countless number of exceptions to standard grammatical rules. To ultimately determine a term's part of speech, a part of speech tagger tool from python's natural language processing tool kit is used. To simplify later contextual analysis, the output of the tool was modified to only include nouns, verbs, adjectives, adverbs, and all other parts of speech are considered to be "other." These parts of speech values are abstracted as enumerations inside the code and the posting contains a list of these enumerations corresponding to each time the term is used in a document. The length of this list is also equal to the total number of occurrences of this term in the document. Finally, since the same term can appear in multiple documents, each key from the inverted index points to a list of postings, where each posting holds data from a different document. Figure 4.2 provides a summary of this thesis's inverted index.

4.2 Extending Okapi BM25

This section covers the implementation details for a generic Okapi BM25 function used in all the modifications as described in Section 4.3. With the construction of a more intimate inverted index, a more sophisticated Okapi BM25 algorithm can be designed. The goal is to create a customized version of the Okapi

BM25 model that allows for extensibility, meaning a developer should not have to create a new Okapi BM25 function for each modification. Ideally, a researcher should be able to quickly toggle different modifications during the instantiation of the model so that experiments can be quickly developed and ran.

Although the exact details of each modification is described in Section 4.3, the overarching idea is that the score generated for a single term is a function of the term's term frequency, inverse document frequency, query term frequency, and a collection of boosts that get applied to that term. The final score for a document is then the sum of all the scores calculated from each matching query term. When designing the system, the modification's influence over a term's individual score is important to consider. In most cases, it is not sufficient to simply add a constant value to the term's score if a term satisfied a particular modification. This is because a term's significance may very depending on context. Adding a constant value could even negatively harm the system's performance. This is because adding constant values will affect short queries more than long queries because the constant's value will represent be a higher proportion of the short query's score while the long query will hardly feel the affect.

Boosting sets of terms based on their perceived importance is an expansion of Bhatia's and Kumar's experiments involving topic modifying and topic collacating terms [6]. Thus, this thesis boosts a term's score proportionally to the term's unmodified Okapi BM25 score. When boosting proportionally to a term's unmodified Okapi BM25 score, an important edge case arises. As defined in Equation 2.8, the function for the inverse document frequency is logarithmic and can become negative when the quantity is less than one. This can occur when a term commonly appears throughout the corpus (df_i approaches the value $|D|$). Although this edge case does not appear very often, boosts are defined to be

strictly positive values and are calculated as a proportion of the absolute value of the term's original score. Equation 4.1 shows the exact boosting function used in many of the experiments, where *OkapiScore* is the original score calculated from Okapi BM25 and *Influence* is a variable determined from a modification that is responsible for scaling a term's score based on the term's positive, Okapi BM25 value.

$$BoostValue = (|OkapiScore| * Influence) - |OkapiScore| \tag{4.1}$$

In order to toggle modifications, the constructor for the model contains a set of optional parameters that turn on and off modifications. The *Influence* of a modification can also be specified in the constructor as an optional parameter during construction for modifications that either require user defined hyper parameters. Next, at runtime, the model checks which modifications are enabled and computes the appropriate boosts for each term.

The extensible model also contains checks to determine values that can be precomputed. For example, one modification attempts to perform query expansion on terms that score the highest inverse document frequency. The idea behind this modification is that terms that have a high inverse document frequencies are rare and may have higher significance to the query. This requires that all the inverse document frequency values for each term be known ahead of time so that to only expand terms that have high inverse document frequency scores. Additionally, other modifications require knowledge about the previous term in the query, such as the previous term's part of speech. As a result, the extensible model always traverses the query in the order in which the terms are appear in the query. This restriction technically breaks the bag of words definition of Okapi BM25 since the order in which query terms are evaluated affect the final score.

4.3 The Okapi BM25 Modifications

This section describes relevant algorithms and equations that are used to derive a modification's *Influence* score. The modifications have been categorized into four themes to help clarify the overall explanation. The first theme analyzes a term's part of speech in isolation. The second theme includes modifications that analyze the distance between pairs of query terms found within a document. The third theme analyzes the position of a single query term with respect to its location within a document. Finally, the fourth theme explores methods for query expansion.

4.3.1 Parts of Speech Modifications

The simplest set of modifications introduced in this research is to scale the *Influence* of an individual term according to its part of speech. *Influence* values for this suite of modifications are set after training the modifications on the Cranfield corpus until a local maximum mean average precision value is reached. Since it is unknown whether or not the *Influence* value should be greater than one or less than one for a particular part of speech, each part of speech is paired with two modifications. The first modification has an *Influence* value that is greater than one and the second modification has an *Influence* value that is less than one. These two sets of modifications are categorized as "Boost Up" and "Boost Down" respectively. Table 4.1 summarizes all of the part of speech modifications.

Category	Affects	Parameters	Summary
Boost Up	Nouns	I > 1	Nouns have higher *Influence*
Boost Up	Adjectives	I > 1	Adjectives have higher *Influence*
Boost Up	Verbs	I > 1	Verbs have higher *Influence*
Boost Up	Adverbs	I > 1	Adverbs have higher *Influence*
Boost Down	Nouns	I < 1	Nouns have lower *Influence*
Boost Down	Adjectives	I < 1	Adjectives have lower *Influence*
Boost Down	Verbs	I < 1	Verbs have lower *Influence*
Boost Down	Adverbs	I < 1	Adverbs have lower *Influence*

Table 4.1: Parts of speech themed modifications. I is short for *Influence*.

4.3.2 Term to Term Modifications

Refining the idea of topic modifying and topic collacating terms from [6], this thesis assumes that pairs of terms are considered related when the first term is either an adjective or adverb and the second term is either a noun or a verb. The idea behind this assumption is that a modifying term, such as an adjective or adverb, contains the most semantic meaning if found next to or near its corresponding subject. For example, in the query "Red cars for sale," the term "red" is semantically insignificant if it is found in a document that does not also contain the word "car."

Three different categories of modifications are built to evaluate pairs of semantically related terms. Shown in Table 4.2 under "Modifiers", the first two modifications exclude the score from adjective and adverb query terms unless the term that immediately follows in the document is the query's subject. These two modifications do not require *Influence* values because they are simply removing the scores from adjectives and adverbs that are not found next to their corresponding subjects.

The second set of modifications can be found under the "Bigrams" category of Table 4.2. This set of modifications considers the possibility that completely removing a term's impact from a document's score might have a negative consequence on a system's overall accuracy. Instead, the "Bigrams" category rewards a document for containing pairs of adjacent query terms. Unlike the "Modifiers" category, the "Bigrams" category does not exclude any term scores. The bigrams are constructed from one of three ways: all adjacent query terms, only adjacent adjective and noun query terms, or only adjacent adverb and verb query terms. Since the significance of a bigram is unknown, the *Influence* values for bigram modifications are computed by training the modifications on the Cranfield dataset until local maximum mean average precision values between one and two are discovered. The boost value is then applied to the subject.

The third set of modifications are located under the "Close Pairs" category in Table 4.2 and are designed to boost modifiers and subjects that may not appear directly adjacent to each other. For example, in the query "Red and blue cars for sale," the term "red" does not appear right next to the term "car." However, the term "red" still modifies the term "car." When there exists a separation between the modifier and the subject, Equation 4.2 is used to determine the appropriate *Influence* value between the two query terms. As with the "Bigrams" category, the boost is applied to the subject.

$$Influence = max(m * x + (b - m), 1) \tag{4.2}$$

In Equation 4.2, x is an integer value that represents the minimum distance between a pair of query terms found within a document. The minimum value of x is equal to one because if two terms are found right next to each other,

the difference between their absolute locations is equal to one. m is a negative value that represents the rate at which the reward for two terms should diminish. Equation 4.2 does not apply to modifiers and subjects that span across multiple sentences to avoid situations where related query terms may appear near each other but in unrelated contexts. For example, if the modification is searching for instances where the terms "red" and "car" appear close to one another, the sentence "She has red hair. Her car is blue" would not be considered by the modification. Finally, $(b - m)$ is the y intercept for the function. The y intercept is adjusted for the fact that when x is equal to one, the value of the function is equal to b. For the experiments ran in this thesis, m is set to -0.25 and $(b - m)$ is set to two.

4.3.3 Term to Document Modifications

When scoring documents, it is not only important to gather documents that relate to the query somewhere within the document. It is also important to gather documents that relate to the query at the front of the document. The proposition is that if users expect relevant information to appear at the start of documents, then a term's score should be positively rewarded for occurring earlier in a document. Table 4.3 contains modifications that reflect this proposition. Each modification in Table 4.3 uses Equation 4.3 to reward terms based on a term's first occurrence in a document.

$$Influence = \frac{K * dl_j - idx_i}{dl_j} \tag{4.3}$$

Equation 4.3 is a linear function where the variable dl_j is the length of document j, measured as the sum of all the terms in the document. idx_i is the absolute

Category	Affects	Parameters	Summary
Modifiers	Adjectives	N/A	Ignore adjectives not found next to nouns
Modifiers	Adverbs	N/A	Ignore adverbs not found next to verbs
Bigrams	All	$I > 1$	Reward a document for containing any adjacent bigrams from the query
Bigrams	Adjectives and Nouns	$I > 1$	Reward a document for containing adjacent adjective, noun bigrams
Bigrams	Adverbs and Verbs	$I > 1$	Reward a document for containing adjacent adverb, verb bigrams
Close Pairs	All	$I = max(-0.25x + 2, 1)$	Reward a document for containing terms that are close together
Close Pairs	Adjectives and Nouns	$I = max(-0.25x + 2, 1)$	Reward a document for containing adjectives and nouns that are close together
Close Pairs	Adverbs and Verbs	$I = max(-0.25x + 2, 1)$	Reward a document for containing adverbs and verbs that are close together

Table 4.2: Term to term themed modifications. I is short for *Influence*.

index location of term i, where the first term in the document has an idx_i value of zero. K is an integer hyper parameter which dictates the upper bound for the function. During experiments, K is heuristically set to two because terms at the front of a document might be twice as important as terms that appear at the end of a document. Notice that the function never penalizes for a term's position. At the very worst case, a term's score is unmodified if it is located at the very end of a document.

Category	Affects	Parameters	Summary
Is Early	All	I = $\frac{2 \times dl_j - idx_i}{dl_j}$	Rewards terms that appear early in a document
Is Early	Nouns	I = $\frac{2 \times dl_j - idx_i}{dl_j}$	Rewards nouns that appear early in a document
Is Early	Adjectives	I = $\frac{2 \times dl_j - idx_i}{dl_j}$	Rewards adjectives that appear early in a document
Is Early	Verbs	I = $\frac{2 \times dl_j - idx_i}{dl_j}$	Rewards verbs that appear early in a document
Is Early	Adverbs	I = $\frac{2 \times dl_j - idx_i}{dl_j}$	Rewards adverbs that appear early in a document

Table 4.3: Term to document themed modifications. I is short for *Influence*.

4.3.4 Query Expansion Modifications

This research explores global query expansion methods so that the resulting model can be portable to any corpus. Three methods for query expansion are implemented, the first uses the WordNet API, the second uses a graph generated through WordNet, and the third uses word embeddings generated from Word2Vec.

The first method for query expansion uses the APIs exposed by WordNet. WordNet is a project by Princeton University that organizes the semantic rela-

$$\{CorpusName : \{QueryID : \{QueryTerm : [ExpansionTerms]\}\}\}$$

Figure 4.3: JSON data structure format for accessing query expansion terms at runtime.

tionships between English terms into cognitive synonyms [25]. Using the APIs is non-trivial because words may have multiple definitions and parts of speech. In order to look up the correct word in WordNet, the Lesk algorithm [20] is implemented to perform word sense disambiguation. For example, given the query from the Cranfield dataset with ID of one, "What chemical kinetic system is applicable to hypersonic aerodynamic problems," the Lesk algorithm determines that "aerodynamic" can be expanded to the terms ["streamlined", "flowing", "sleek"] and that "problems" can be expanded to the term ["trouble"]. To help limit runtime computation in experiments, query expansion terms are precomputed for each corpus and stored in a large JSON object that is loaded at runtime. The structure of this JSON object is described in Figure 4.3. To get the expansion terms for "aerodynamic" in the previous query, the JSON object is indexed as follows: jsonObject["Cranfield"][1]["aerodynamic"].

With a logical guess of the proper definition and part of speech of a term, cognitive synonyms can then be extracted from WordNet. Unfortunately, WordNet does not provide the strength of similarity between a term and its cognitive synonyms, so we set the *Influence* value for all WordNet API expansion terms to 0.9. The assumption is that expansion terms have a slightly lower probability of being relevant than the original query term, but still maintain a high *Influence* value because they are cognitive synonyms. With the *Influence* value set, a query term's score is computed as the sum of the term's original Okapi BM25 score, plus a collection of boost values for each expansion term. In order to generate the boost values for each expansion term, Equation 4.1 is modified

so that the *OkapiScore* variable is the score generated by Okapi BM25 applied to each expansion term. Table 4.4 provides a summary of the WordNet API modifications. All three query expansion categories will contain modifications that not only target all available query terms, but will also target nouns, verbs, adjectives, adverbs, low inverse document frequency terms, and high inverse document frequency terms. Both low and high inverse document frequency terms are expanded in separate categories in order to determine if a performance increase will consistently occur for one of the two categories.

Category	Affects	Parameters	Summary
WordNet API	All	I = 0.9	Expand all query terms
WordNet API	Nouns	I = 0.9	Expand query terms that are nouns
WordNet API	Verb	I = 0.9	Expand query terms that are verbs
WordNet API	Adjectives	I = 0.9	Expand query terms that are adjectives
WordNet API	Adverbs	I = 0.9	Expand query terms that are adverbs
WordNet API	Low IDF	I = 0.9	Expand query terms that score a low inverse document frequency value
WordNet API	High IDF	I = 0.9	Expand query terms that score a high inverse document frequency value

Table 4.4: WordNet API modifications for the query expansion theme. IDF is short for Inverse Document Frequency. I is short for *Influence*.

Although WordNet does not quantify the similarity between terms, recent research shows that similarity scores can be calculated if the WordNet corpus is arranged in a probability graph [39]. The directed graph stores data about the strength of the relationship between words, expressed as decimal numbers, and is created using probability theory that corresponds to a simplified version of a

Bayesian network [29]. A node in the graph is created for every *word form*[1] and every *sense* in the WordNet corpus. The weight of an edge is the approximation of the probability that a user is interested in the concept that is described by the destination node of the edge given that they are interested in the concept that is described by the source node.

Semantically similar terms can then be discovered from the probability graph by computing random walks from the node that represents the unexpanded query term. A random walk is performed by first selecting all the unmarked outgoing edges from the unexpanded term's node. The weights are then normalized so that their sum is equal to one. From here, the weights are ordered in a discrete probability distribution and a random number is generated to determine which edge is to be selected. For example, suppose a node has three outgoing edges whose weights have already been normalized to one: $\{e_1 = 0.5, e_2 = 0.3, e_3 = 0.2\}$ then the set of edges will be sorted on a domain from zero to one such that $e_1 \in [0, 0.5]$, $e_2 \in (0.5, 0.8]$, and $e_3 \in (0.8, 1]$. If a random number, say $r = 0.65$, is generated, then edge e_2 will be selected because $0.5 < r < 0.8$. The selected edge is then used to traverse to the new node and the outgoing edge pointing from the new node to the original node is marked so that the algorithm does not revisit the source node. The current node is then marked as visited and a frequency dictionary that maps node names to the number of times this node has been visited is updated. This process is repeated once more to reach a depth of two.

The nodes that are traversed the most often after running the random walk algorithm 1,000 times on the unexpanded term's node are the semantically similar terms. The algorithm is ran 1,000 times in order to generate enough sample data

[1] WordNet uses the term *word form* to refer to both the words and the phrases in the corpus

to construct a proper probability distribution. At a high level, the similarity score between the unexpanded term and an expansion term is calculated as the proportion of times the expansion term's node was visited after 1,000 random walks. At a lower level, the frequency dictionary is used in the following way to calculate the similarity score between the original term and each key in the map. First, all multi word senses are filtered out. Then the values in the frequency dictionary are summed together (this sum will be less than $iterations \times depth$ because some senses have been removed). Next, this quantity is then used to normalize the map's values so that the sum is equal to one. Then the keys to the frequency dictionary are stemmed using a Porter Stemmer and a descending list of (term, similarity score) pairs is returned. For example, if the algorithm runs 1,000 times for a depth of two and multi-term senses were visited 500 times, then a single-term sense that is visited 90 times will have a similarity score of $\frac{90}{(1000*2)-500} = 0.06$, or in other terms, this word sense is 6% similar to the original term. Since nodes in the probability graph can have a high branching factor, a minimum acceptable similarity score between the query term and the expanded terms is heuristically set in order to avoid query drift.

The similarity score calculated in the above algorithm represents the probability that a user is interested in a particular term, given the original query term. This score is then used as the expansion term's *Influence* value when calculating the set of boosts to be applied to the original query term. Table 4.5 summarizes the WordNet Graph query expansion modifications. Notice that the modifications target various parts of speech and high and low inverse document frequency terms.

The last query expansion category that is implemented uses English word vectors generated using the Word2Vec algorithm [22, 23] described in Section

Category	Affects	Parameters	Summary
WordNet Graph	All	I =RWSS	Expand all query terms
WordNet Graph	Nouns	I = RWSS	Expand query terms that are nouns
WordNet Graph	Verb	I = RWSS	Expand query terms that are verbs
WordNet Graph	Adjectives	I = RWSS	Expand query terms that are adjectives
WordNet Graph	Adverbs	I = RWSS	Expand query terms that are adverbs
WordNet Graph	Low IDF	I = RWSS	Expand query terms that score a low inverse document frequency value
WordNet Graph	High IDF	I = RWSS	Expand query terms that score a high inverse document frequency value

Table 4.5: WordNet probability graph modifications for the query expansion theme. RWSS is short for Random Walk Similarity Score. I is short for *Influence*.

2.4.2. The vectors are 300 dimensions large and were generated from the Google News corpus[2], which is a large database of text containing three billion running words and about three million unique words. The main advantage to using word vectors for query expansion is that the similarity between any two word vectors can be computed using the cosine similarity function, as described in Equation 2.6. The resulting cosine similarity score is then used as the *Influence* value.

In order to find the top similar word vectors for a given term, the cosine similarity is computed between the query term and all other word vectors. Since the binary file containing all the word vectors is too large to hold in memory for the machine running all the experiments (3.64 gigabytes), the python gensim[3] library is used to only load the top one million word vectors. Although this

[2]GoogleNews corpus: https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[3]The gensim library is an open source vector space modeling and topic modeling toolkit.

dramatically reduces the number of word vectors used in experiments, calculating the similarity between a single word and a million other word vectors is a time consuming procedure. As a result, query expansion terms are computed ahead of time and the highest scoring terms and their cosine similarity scores are stored in a JSON object, which is loaded during runtime. To prevent query drift, a lower bound similarity score of 0.5 is heuristically set so that expansion terms that have a lower similarity score than 0.5 are ignored. Unlike the JSON structure presented in Figure 4.3, the structure of the Word2Vec JSON is simply a mapping from a query term to a list of related term-score pairs. The simplified structure of this JSON object is due to Word2Vec's main disadvantage. Unlike the expansion terms derived from the WordNet API category, Word2Vec expansion terms for a particular query term are the same no matter the context in which the term appears in the query because word vectors are agnostic to queries. A summary of the Word2Vec modifications can be found in Table 4.6. Table 4.6 has a similar structure to the WordNet API and WordNet Graph tables with respect to the terms affected.

Category	Affects	Parameters	Summary
Word2Vec	All	I = CSS	Expand all query terms
Word2Vec	Nouns	I = CSS	Expand query terms that are nouns
Word2Vec	Verb	I = CSS	Expand query terms that are verbs
Word2Vec	Adjectives	I = CSS	Expand query terms that are adjectives
Word2Vec	Adverbs	I = CSS	Expand query terms that are adverbs
Word2Vec	Low IDF	I = CSS	Expand query terms that score a low inverse document frequency value
Word2Vec	High IDF	I = CSS	Expand query terms that score a high inverse document frequency value

Table 4.6: Word2Vec modifications for the query expansion theme. CSS is short for Cosine Similarity Score. I is short for *Influence*.

Chapter 5

Experimental Setup

The main goal of this chapter is to explore the measures, benchmarks, hypotheses, and experimental procedures. The measures section lays the mathematical foundation for judging the accuracy of all the modifications and document retrieval systems. Afterwards, the publicly available benchmarks that were used to validate and test the modifications will then be presented. Next the hypotheses that this research investigates will be state. Lastly, the experimental procedure for applying and combining modifications will be described in detail.

5.1 Measures

The validation metrics that are used for measuring modification improvements on individual queries are precision and recall, which are explained here. For the following equations, let s be the number of relevant documents in a document collection D for query q, i be the number of documents that the system returns, and s_i be the number of relevant documents the system returns after returning i number of documents. Thus, precision is the fraction of how many relevant

documents were returned over how many documents the systems returned [21], as shown in Equation 5.1. Similarly, recall is the fraction of how many relevant documents the system returned over how many relevant documents are available to be returned [21], as shown in Equation 5.2.

$$p(i) = \frac{s_i}{i} \tag{5.1}$$

$$r(i) = \frac{s_i}{s} \tag{5.2}$$

The validation metrics that are used for measuring modification improvements on a set of queries are mean average precision and weighted average recall. Mean average precision (MAP) is a measurement of document ordering. The mean average precision for a single query is the sum of precisions for each recall level $l \in L$ at which a relevant document is returned. In other words, the precision is calculated once every time a known relevant document is returned. This sum is then divided by the number of relevant documents R_q for the query q. To obtain the average MAP score over a set of queries, all MAP scores for each query are summed together and then the total is divided by the total number of queries, $|Q|$. The average MAP score over a set of queries and a corpus is shown in Equation 5.3.

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{|R_q|} \sum_{l \in L} p(l) \tag{5.3}$$

Weighted average recall can be viewed as the average number of relevant documents the system returns for a query $q \in Q$ within a given number of returned documents i. Each query is weighted proportionally to its total number

of relevant documents in the document collection D. Let the total number of relevant documents over all queries be $|R|$. Equation 5.4 is the weighted average recall formula, where the sum of all the recall contributions adds to one, as shown in Equation 5.5.

$$r_{avg}(i) = \sum_{q \in Q} r(i) \times \frac{s}{|R|} \tag{5.4}$$

$$\sum_{q \in Q} \frac{s}{|R|} = 1. \tag{5.5}$$

The performance of document retrieval systems can also be compared graphically using a precision-recall curve [21]. The graph's x-axis is the average recall and the graph's y-axis is the average precision. The curve is commonly plotted against 11 recall levels $r_l = 0\%, 10\%, 20\%, ..., 100\%$, where a percentage indicates that X% of relevant documents have been returned. Since each recall level may not be exactly obtained, interpolation is performed by computing the precision at at each recall level by computing the maximum precision value between the current recall level and all the following recall levels (Equation 5.6).

$$p(r_l) = max_{r_l \leq r \leq r_{100\%}} p(r) \tag{5.6}$$

Comparing two document retrieval systems using the metrics listed in this section is straightforward. The system that scores the highest metric values and who's precision-recall curve has the highest y-values for each recall level is the more accurate system.

5.2 Benchmarks

During validation, Okapi BM25 modifications were ran against four publicly available information retrieval benchmarks[1]: Cranfield, Adi, Medline, and Time. Each benchmark contains a set of documents, a set of queries, and a list of relevance scores for all query-document pairs. During testing, the best modification is ran against the largest publicly available information retrieval benchmark in the group, Lisa. which stands for Library and Information Science Abstracts. It is important to emphasize that no document retrieval systems were ran against Lisa until all the validation was completed. Table 5.1 holds document meta data for each benchmark in the validation and test sets. Cranfield and Adi benchmarks contain research paper abstracts on aerospace and information management and Medline and Time benchmarks contain articles written by a medical journal and the magazine named Time. An important feature to notice is that the average number of terms inside each document is low, less than 100 words, with the exception of the Time benchmark. Since there are very few words inside each document, there are fewer opportunities for there to be an exact matching between query terms and document terms. This means that the benchmarks are very challenging. In total, there are just under 3,000 documents in the validation set. Lisa, the test benchmark, contains 5,872 documents with an average number of 86.3 terms per document. As with Cranfield and Medline, there are a small number of terms per document, which indicates that Lisa is a challenging benchmark.

Table 5.2 provides document parts of speech meta data for each benchmark. The data was gathered using the python nltk library, which sometimes makes

[1]Available at http://ir.dcs.gla.ac.uk/resources/test_collections

Benchmarks	Theme	Documents	Average Number of Document Terms
Cranfield	Abstracts	1,400	86.6
Adi	Abstracts	82	30.9
Medline	Articles	1,033	90.3
Time	Articles	423	346.3
Lisa	Abstracts	5,872	86.3

Table 5.1: Document metadata for each benchmark.

mistakes. So these numbers serve as an approximation and not the ground truth. From these approximations, it appears that benchmarks are nearly 50% composed of nouns, 20% composed of verbs, 20% composed of adjectives, 5% composed of adverbs, and 5% composed of other miscellaneous parts of speech.

Benchmarks	Nouns	Verbs	Adjectives	Adverbs	Other
Cranfield	46.8	16.85	17.64	2.59	2.69
Adi	17.96	6.05	5.26	0.66	1
Medline	48.02	14.83	20.64	3	3.79
Time	169.19	72.8	64.6	18.23	21.45
Lisa	30.36	9.38	10.63	1.00	1.33

Table 5.2: Document parts of speech metadata for each benchmark.

Table 5.3 contains query meta data for each benchmark. An important feature to notice about the set of queries in the validation set is that the average number of related documents per query is relatively low, with the exception of the Medline benchmark. Since there are only a few related documents per query, small changes in the mean average precision are expected because there are fewer opportunities to return a related document. Another feature to notice is that the average length of each query is large, around ten terms after stop word removal. As a comparison, the average length of a query submitted to a web search engine is around two to three terms [4]. Cranfield is used as the training set for a handful

of modifications that require trained parameters because Cranfield contains the most queries. In total, there are 370 queries in the validation set. In the testing set, Lisa contains an average of 10.8 related documents per query and each query contains around 34.7 terms after stop word removal.

Benchmarks	Queries	Average Number of Related Documents	Average Number of Query Terms
Cranfield	222	5.6	10.6
Adi	35	4.9	9.1
Medline	30	23.2	12.9
Time	83	3.9	9.5
Lisa	35	10.8	34.7

Table 5.3: Query metadata for each benchmark.

Table 5.4 provides query parts of speech meta data for each benchmark. Just as how Table 5.2 is an approximation of the parts of speech identified using the nltk library, as is this table. From the approximations in Table 5.4, it appears that queries are nearly 60% composed of nouns, 15% composed of verbs, 20% composed of adjectives, 2.5% composed of adverbs, and 2.5% composed of other miscellaneous parts of speech.

Benchmarks	Nouns	Verbs	Adjectives	Adverbs	Other
Cranfield	5.78	1.66	2.75	0.27	0.2
Adi	5.86	1.4	1.29	0.26	0.26
Medline	9	1.17	2.2	0.4	0.13
Time	5.71	1.31	1.88	0.11	0.45
Lisa	20.31	5.40	7.54	0.86	0.89

Table 5.4: Query parts of speech metadata for each benchmark.

5.3 Hypotheses

This thesis hypothesizes that Okapi BM25's performance can be improved if it is modified to take advantage of semantic information located in documents and queries. This hypothesis is explored by designing four themes, where each theme is further divided into a suite of modifications. If many of the modifications corresponding to a specific theme perform well on the validation benchmarks, then the theme is considered to contain valuable semantic information. If the final model incorporates one or more modifications that span across different themes and outperforms the original Okapi BM25 during testing, then this will be considered as evidence in support of the hypothesis. The assumptions for each theme is as follows:

1. Okapi BM25 can be improved if terms are weighted differently depending on their part of speech.

2. Okapi BM25 can be improved if terms are weighted differently according to their proximity to other terms.

3. Okapi BM25 can be improved if terms are weighted differently according to their initial location in a document.

4. Okapi BM25 can be improved if query terms are expanded.

5.4 Experimental Procedure

At a high level, the experiment is split up into one training round, three validation rounds, and one testing round. During the training round, hyper parameter values for relevant modifications are discovered. Modifications that

require training for hyper parameters include the parts of speech themed modifications and the term to term themed modifications for the Bigrams category. Hyper parameters are chosen by slowly incrementing or decrementing an initial value with a small delta value of 0.1. The resulting value will either be between 0.1 and 0.9 or 1.1 and 3.0, which depends on the specific modification. During training, the modifications are ran against the Cranfield corpus and the hyper parameter value that results in the greatest mean average precision is used for the rest of the experiments. Hyper parameters are not recalculated if multiple modifications are combined. For example, if a modification is set to boost nouns and bigrams, the hyper parameters used in both of these modifications are the values discovered during their isolated training periods.

Once all modification hyper parameters are discovered, the first validation round starts. During the first validation round, all 41 individual modifications are ran independently on the four validation corpora: Cranfield, Adi, Medline, and Time. The modifications that result in a lower mean average precision score than the original Okapi BM25 model on at least two datasets will be eliminated. Those modifications that result in a higher mean average precision scores than the original Okapi BM25 model will move on to validation rounds two and three.

The second validation round combines the successful, intra-themed modifications from the first validation round. In other words, validation round two combines modifications within the same theme. An example modification that can be tested in this round could be a query expansion modification that uses WordNet API, WordNet Graph, and Word2Vec. Once all intra-themed modifications are created, they are scored using mean average precision in the same way as validation round one. This is done by running each modifications against Cranfield, Adi, Medline, and Time, and then eliminating modifications that do

not improve the Okapi BM25 model in at least half of the benchmarks.

The third validation round combines successful, inter-themed modifications from rounds one and two. In other words, round three combines modifications across separate themes. An example modification might use query expansion on WordNet API, WordNet Graph, and Word2Vec, boost terms based on their proximity within a document, boost terms based on their part of speech, and boost query bigrams that appear within a document. These modifications are then validated against Cranfield, Adi, Medline, and Time using the same procedure as validation rounds one and two.

Once all three validation rounds are completed, a single modification across all three rounds must be selected for testing. The best performing modification is taken to be the largest sum of mean average precision scores across Cranfield, Adi, Medline, and Time subtracted by the sum of mean average precision scores accumulated by the unmodified Okapi BM25 model. The best performing modification will then go through a testing round where it will be ran against the Lisa benchmark. The score from this benchmark will then be presented as the result of the study and the performances of the two models will be plotted and compared using precision-recall curves

Chapter 6

Results

This chapter details the results after conducting the experimental procedure as described in Section 5.4. The results are divided into four sections. Sections 6.1, 6.2, and 6.3 list and analyze the results from validation rounds one, two, and three. Then Section 6.4 selects the best performing modification and tests it against the Lisa benchmark. For all the following tables, if a cell is unfilled then it takes the value above it and ΔMAP% stands for the percentage change in mean average precision between the modification and Okapi BM25.

6.1 Validation Round One

Validation round one contains 41 modifications where 14 outperformed Okapi BM25 on Cranfield, 14 outperformed on Adi, 12 outperformed on Medline, and 12 outperformed on Time. Of these 41 modifications, 13 of them performed better than Okapi BM25 on at least half of the validation benchmarks, 8 performed better on at least three validation benchmarks, and 1 performed better on all of the validation benchmarks.

Theme	Category	Terms Affected	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
Expansions	WordNet Graph	All	-0.12%	0.41%	-0.39%	-0.23%
		Noun	-0.07%	0.47%	-0.01%	-0.08%
		Verb	0.00%	0.00%	0.07%	-0.16%
		Adj	-0.01%	-0.02%	-0.23%	-0.16%
		Adv	0.00%	0.00%	0.02%	0.00%
		Bottom IDF	0.02%	0.49%	-0.03%	0.00%
		Top IDF	0.12%	-0.11%	-0.11%	-0.16%
	WordNet APIs	All	0.00%	0.02%	0.19%	-0.50%
		Noun	0.05%	0.00%	0.19%	-0.50%
		Verb	0.00%	0.02%	0.00%	-0.01%
		Adj	0.00%	0.00%	0.00%	-0.01%
		Adv	0.00%	0.00%	0.00%	-0.01%
		Bottom IDF	-0.01%	-0.12%	0.39%	-0.45%
		Top IDF	0.05%	0.00%	-0.03%	-0.06%
	Word2Vec	All	-0.50%	-1.34%	-1.55%	0.16%
		Noun	0.17%	-1.49%	-2.21%	0.23%
		Verb	-0.09%	0.00%	0.38%	-0.08%
		Adj	-0.59%	-0.63%	0.10%	0.00%
		Adv	0.00%	0.00%	0.00%	0.00%
		Bottom IDF	0.09%	-1.11%	-0.40%	0.31%
		Top IDF	-0.59%	-1.15%	-0.83%	-0.23%

Table 6.1: The change in mean average precision for modifications that involve query expansions.

Table 6.1 shows the results for modifications that involve query expansions. From the table, it is easy to see that most percentage differences in mean average precision are negligible. In fact, no query expansion modification improved the mean average precision score on at least two benchmarks, except for the Word2Vec-noun modification. However, the benefits of the Word2Vec-noun modification are outweighed by the negative impact it had on the Adi and Medline benchmarks. What is interesting about these results is that the only time a modification actually improved over Okapi BM25 in at least one benchmark and across WordNet Graph, WordNet APIs, and Word2Vec was when the terms affected include either nouns or bottom inverse document frequency terms (excluding the "All" modification since this modification covers nouns and bottom IDF). The WordNet Graph modification that targets low inverse document frequency terms has a net positive affect on the system's accuracy because it improves the system's score for the Adi benchmark and hardly changes the results for the other benchmarks. However, since most of these modifications have a negligible effect on most of the datasets, it is worth investigating ways of combining these modifications in round 2 in order to produce more pronounced results.

Upon further analysis, query expansion has minor effects on a model's mean average precision because there were only a few expansion terms discovered for each query. To support this claim, the average number of expansion terms were calculated for each category on each benchmark. For simplicity in later calculations, assume that the average number of query terms across all four validation benchmarks is equal to ten (the actual average is 10.525). Individual averages for each benchmark can be found in Table 5.3.

For WordNet API modifications, it turns out that there are about two expansion terms being added for each query, as shown in Table 6.2. Adding two terms

Benchmark	Terms Found for Expansion	Total Expansion Terms	Average Expansion Terms Per Query
Cranfield	1277	2416	1.89
Adi	160	298	1.86
Medline	164	302	1.84
Time	367	725	1.97

Table 6.2: WordNet API query expansion data.

Benchmark	Terms Found for Expansion	Total Expansion Terms	Average Expansion Terms Per Query
Cranfield	555	835	1.50
Adi	179	142	0.79
Medline	70	28	0.4
Time	195	45	0.23

Table 6.3: WordNet Graph query expansion data. Expansion terms were calculated with a minimum similarity score of 0.02.

to each query equates to an approximate 20% expansion of terms for each query. For WordNet Graph modifications, there are far fewer terms being expanded, as seen in Table 6.3. Across all four validation datasets, only about 0.7 expansion terms are added to each query, which only equates to an approximate 7% expansion of terms for each query. In Table 6.3, the total number of expansion terms are actually lower than the number of terms found for expansion because many of the expansion terms did not meet the minimum similarity threshold. Lastly, Word2Vec modifications discovers the most number of expansion terms per query, as shown in Table 6.4. On average, Word2Vec adds around four terms per query, which equates to an approximate 40% expansion of terms for each query. Although these percentages may sound high when compared to the average length of each query, the number of added expanded terms is evidently not high enough to significantly affect the scores of each modification because we observe small changes in mean average precision. In reality, there is a very low probability that an expanded query term will actually appear within a document

Benchmark	Terms Found for Expansion	Total Expansion Terms	Average Expansion Terms Per Query
Cranfield	2,644	10,345	3.91
Adi	337	1,503	4.45
Medline	365	1362	3.73
Time	825	3,678	4.45

Table 6.4: Word2Vec query expansion data. Expansion terms were calculated with a minimum similarity score of 0.5.

because the documents do not contain many words. Therefor, the majority of expansion terms have no affect on a document's score. So, validation round two will increase the probability of discovering an expansion term within a document by enabling multiple query expansion modifications.

The next theme, labeled Term-Document, contains the modifications that reward terms that appear closer to the start of a document, as shown in Table 6.5. Each Term-Document modification successfully increases the mean average precision for at least three of the benchmarks. The modifications always increased the accuracy against the Cranfield and the Medline benchmarks and increased the accuracy against the Adi and the Time benchmarks in three out of five cases. The modification that targeted only adjectives consistently increased the model's accuracy across all benchmarks and performed surprisingly well on the Time benchmark, despite the fact that the documents in the Time dataset are composed of the same proportion of adjectives as the other validation corpora (around 20%). On average, modifications that targeted parts of speech that compose a larger majority of a corpus gave rise to larger swings in the accuracy of the system, and vice versa. For example, adverbs make up a small minority of the terms inside a document, so the observed changes for the modification that targeted adverbs are small. On the other hand, the modification that targets all parts of speech resulted in the largest swings in accuracy for each benchmark

(with the exception of the adjective modification on the Time benchmark).

Theme	Category	Terms Affected	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
Term - Document	Is Early	All	3.18%	1.80%	3.21%	-4.20%
		Noun	2.73%	0.70%	2.86%	-3.81%
		Adj	1.48%	0.16%	0.73%	6.14%
		Verb	0.65%	-0.60%	0.15%	1.17%
		Adv	0.71%	-0.16%	0.01%	0.31%

Table 6.5: The change in mean average precision for modifications that involve the position of a term within a document.

Since the mean average precision increased in the vast majority of cases, there appears to be a positive correlation between the relevance of a document and how early the query terms appear in that document. Additionally, because all modifications positively increased the mean average precision on at least half of the datasets, validation round two will explore combinations of Term-Document modifications in order to potentially produce an even better system.

The following theme, labeled Part of Speech, is split into two categories. The first category, Boost Up, contains *Influence* scores that are greater than one and the second category, Boost Down, contains *Influence* scores that are less than one. A trend that can be observed in Table 6.6 is that the scores of the modifications that target the same part of speech, but are located in opposite categories, are somewhat inversely related. For example, the Boost Up and Boost Down modifications that affect only adjectives are good examples of this relationship. Notice in Table 6.6 that when the Boost Up adjective modification decreases the mean average precision score for a particular benchmark, the Boost Down adjective modification increases the system's accuracy. The same trend is observed by the adverb modifications and the noun modifications, with the exception of the results from the Medline benchmark.

Theme	Category	Terms Affected	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
Part of Speech	Boost Up	Noun	2.57%	0.25%	-0.70%	1.48%
		Adj	1.20%	-2.26%	-0.47%	-1.79%
		Verb	-0.82%	1.05%	0.47%	-1.71%
		Adv	-0.02%	-1.13%	-0.88%	0.54%
	Boost Down	Noun	-1.10%	-3.46%	-0.33%	-2.56%
		Adj	-0.12%	1.13%	0.41%	0.23%
		Verb	0.57%	1.77%	-1.29%	-0.93%
		Adv	0.35%	0.74%	0.12%	-0.70%

Table 6.6: The change in mean average precision for modifications that involve a single term's part of speech.

Some of the modifications from Table 6.6 performed better when the *Influence* of a particular part of speech was decreased, such as the Boost Down adjective and adverb modifications. Conversely, other modifications performed better when the *Influence* of a particular part of speech was increased, such as the Boost Up noun modifications. Validation round two will explore different part of speech ratios to determine if there exists a combination that produces better results.

The final theme, labeled Term-Term, places a strong emphasis on the distances between terms and the results can be found in Table 6.7. There are three categories: Modifier and Subject, Bigrams, and Close Pairs. In all three categories, the change in mean average precision is small when the modification specifically targets adverbs because adverbs consist of a small proportion of each benchmark. For modifications that target more common parts of speeches, the changes in performance are larger, but in many cases, still relatively small. This is most likely because the probability of two terms with specific parts of speech appearing chronologically near each other in a relatively small document is a rare event. As a result, there may not be many opportunities for a Term-Term modification to change the system's behavior.

Theme	Category	Terms Affected	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
Term - Term	Modifier and Subject	Adj	-15.22%	-8.91%	-6.73%	7.30%
		Adv	-0.11%	1.66%	0.09%	-1.09%
	Bigrams	All	0.93%	-0.70%	0.21%	-2.10%
		Adj and Noun	0.43%	0.26%	-0.01%	-0.85%
		Adv and Verb	0.21%	0.00%	0.00%	0.00%
	Close Pairs	All	-2.79%	-2.37%	-0.10%	-6.76%
		Adj and Noun	0.08%	1.33%	-0.47%	1.01%
		Adv and Verb	0.08%	0.01%	0.09%	1.01%

Table 6.7: The change in mean average precision for modifications that involve the proximity between a pair of terms.

The first category in Table 6.7, Modifier and Subject, does not include the score from the modifying term unless the corresponding subject appears immediately afterwards. As evident in Table 6.7, removing a modifier altogether resulted in very negative scores, with the exception of the Time benchmark that actually saw an increase in accuracy for the adjective modification. Rather than removing an adjective or adverb for not appearing next to its corresponding subject, the Bigram category rewards a document for containing pairs of adjacent terms constructed from the query. Although the bigram modifications performed better than the Modifier and Subject modifications, none of the three Bigram modifications improved the mean average precision score on more than two benchmarks. The last category, Close Pairs, linearly rewards a document for containing a higher density of adjacent query terms. As evident from the modification that targets all parts of speech, rewarding a document for containing a high density of all adjacent query terms actually lowers the accuracy of the system across all benchmarks. This may be because the modification is placing too much empha-

sis on pairs of term which do not represent the main subject of the query. For example, in this query from the Time dataset, "India fears of another communist China invasion," the emphasis of the query should be placed between the terms "India" and "China" because these terms are the two subjects of the query. But the modification may disproportionately place emphasis on the terms "fears" and "communist," which can result in unrelated articles. On the other hand, focusing on specific terms, such as adjectives and nouns, appear to have a positive affect on the system's accuracy. From all the Term-Term results, it is unclear if simply measuring the closeness between terms will result in a more accurate system. In half the cases, the system's performance drops and in the other half of cases, the system's performance slightly increases. These results contrast the results from related research in Sections 3.1.1 and 3.1.3, where researchers observed a consistent increase in their system's performance when proximity measures were introduced into Okapi BM25.

6.2 Validation Round Two

Validation round two contains 20 modifications that were created by combining successful modifications within the same theme from validation round one. Of these 20 modifications, 12 outperformed Okapi BM25 on Cranfield, 13 outperformed on Adi, 7 outperformed on Medline, and 8 outperformed on Time. Of these 20 modifications, 14 of them performed better than Okapi BM25 on at least half of the validation benchmarks and 7 performed better on at least three validation benchmarks. For the following tables, the Category column has been combined with the Terms Affected column to specify which category affects which terms.

The first theme in validation round two combines query expansion modifications, as shown in Table 6.8. Although the query expansion modifications from validation round one did not satisfy the criteria for moving on to validation round two, the analysis from validation round one concluded that there were not enough terms affected by any one expansion category to significantly affect the system's accuracy. So in order to increase the total number of expanded terms, each modification in validation round two takes advantage of all categories: WordNet Graph, the WordNet API's, and Word2Vec. It is clear from looking at Table 6.8 that using query expansion from all the globally built models results in a decrease in the system's performance. The decreases in the system's performance is evidence against the potential advantages of using global query expansion methods. Validation round three will not include any multi-query expansion modifications. However, validation round three will include the bottom inverse document frequency modification from the WordNet Graph category from round one because this modification slightly improved the system's precision on the Adi benchmark and only resulted in minor changes to the other benchmarks.

The second theme in validation round two combines modifications that boost a term's score proportionally to how close that term appears at the front of a document, as shown in Table 6.9. All of the modifications improve the system's performance on exactly three benchmarks and decrease the system's performance on the Time benchmark, with the exception of modification with ID 4. A good performing modification is ID 4 because it has the most consistent increase in the system's mean average precision and the lowest decrease in precision for the benchmark that it negatively impacted. Another modification that performed well is ID 3 because it had the least negative impact on the Time benchmark and improvements in the other benchmarks. Although, combining Term - Document

ID	Theme	Terms Affected	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
1	Expansions	WNG:All, WNA:All, W2V:All	-0.67%	-1.20%	-1.80%	-0.42%
2		WNG:BIDF, WNA:BIDF, W2V:BIDF	0.12%	-1.09%	-0.38%	-0.17%
3		WNG:Noun, WNA:Noun, W2V:Noun	0.19%	-1.20%	-2.04%	-0.24%
4		WNG:Noun, WNA:Noun, W2V:Verb	-0.11%	0.41%	0.27%	-0.56%

Table 6.8: The change in mean average precision for modifications that involve query expansions across multiple categories. WNG is short for WordNet Graph. WNA is short for WordNet API. W2V is short for Word2Vec. BIDF is short for Bottom Inverse Document Frequency.

modifications did not expose a ratio that outperformed the very best Term - Document modifications from validation round one, many modifications resulted in improvements on the validation benchmarks. This is evidence in favor for measuring how early a term appears in a document as a way to increase a system's performance. The two best modifications from this round and the best modification from validation round one will proceed to validation round three.

The third theme in validation round two combines modifications that either

ID	Theme	Terms Affected	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
1	Term - Document	Noun, Adj, Verb, Adv,	0.53%	2.51%	3.57%	-5.45%
2		Noun, Verb	1.65%	0.17%	3.22%	-5.11%
3		Noun, Adj	0.48%	3.31%	3.11%	-4.18%
4		Verb, Adv	1.22%	-0.75%	0.11%	1.93%

Table 6.9: The change in mean average precision for modifications that involve the position of a term within a document across multiple parts of speech.

boosts the weight of a term up or down based solely on the term's part of speech. The best performing modification that takes into account all parts of speech: nouns, verbs, adjectives, and adverbs, is modification with ID 3 because it increased the system's performance on the most benchmarks. This particular modification positively boosts the weights of nouns and adjectives while negatively boosting the weights of verbs and adverbs. The best performing modifications that took into account two parts of speech were modifications with ID's 5, 6, and 7 because each modification increased the system's performance on at least three benchmarks. Since there were so many Part of Speech modifications that improved Okapi BM25's performance across validation rounds one and two, only one modification is chosen to be used in validation round three to decrease the number of possible modification combinations. Modification with ID 7 is a good candidate because it resulted in relatively high increases to precision for three of the benchmarks and only a relatively small decrease in precision for the other benchmark. Since there were a vast number of successful Part of Speech modifications that increased Okapi BM25's performance, this is taken as evidence that boosting according a term's part of speech is an effective way to increase the system's performance.

The last and final theme of modifications for round 2 determines if there is a combination of Term - Term modifications that will increase a system's accuracy. As evident from Table 6.11, combining the Close Proximity and Bigrams categories resulted in an overall negative change to the system's mean average precision. Neither of the two modifications improved more than one validation benchmark so these modifications will not appear in validation round three. So, the best performing modification from the Term-Term theme across validation rounds one and two is the Close Pairs modification which boosted according to

ID	Theme	Terms Affected	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
1	Parts of Speech	Noun:↑, Adj:↑, Verb:↑, Adv:↓	2.91%	-0.27%	0.11%	-1.28%
2		Noun:↑, Adj:↓, Verb:↑, Adv:↓	-1.62%	-0.14%	-0.41%	1.37%
3		Noun:↑, Adj:↑, Verb:↓, Adv:↓	2.92%	3.09%	-2.31%	0.89%
4		Noun:↑, Adj:↓, Verb:↓, Adv:↓	-1.07%	2.65%	-2.87%	3.35%
5		Noun:↑, Verb:↑	1.93%	-0.37%	0.05%	0.40%
6		Noun:↑, Verb:↓	1.93%	1.54%	-2.64%	2.30%
7		Noun:↑, Adj:↑	2.95%	0.39%	-0.53%	2.07%
8		Noun:↑, Adj:↓	-0.74%	0.18%	-1.15%	3.45%
9		Verb:↑, Adv:↓	-0.62%	1.90%	0.54%	-2.29%
10		Verb:↓, Adv:↓	0.73%	2.90%	-1.04%	-1.39%

Table 6.10: The change in mean average precision for modifications that involve a single term's part of speech across multiple categories. ↑ symbolizes Boost Up. ↓ symbolizes Boost Down.

ID	Theme	Terms Affected	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
1	Term - Term	CP:Adj-Noun B:All	-0.04%	1.32%	-0.43%	-1.29%
2		CP:Adj-Noun B:Adj-Noun	-0.04%	1.32%	-0.43%	-1.29%

Table 6.11: The change in mean average precision for modifications that involve proximity between pairs of terms across multiple categories. CP is short for Close Proximity. B is short for Bigram.

the close proximity of adjectives and nouns.

6.3 Validation Round Three

Validation round three contains 25 modifications that combine aspect between different categories. From the list of modifications, 24 outperformed Okapi BM25 on Cranfield, 22 outperformed on Adi, 7 outperformed on Medline, and 11 outperformed on Time. In terms of overall performance, 23 outperformed Okapi BM25 on at least half of the validation benchmarks and 16 outperformed on at least three of the validation benchmarks.

Since there are many modifications from validation rounds one and two that improved the accuracy of Okapi BM25 on at least half of the validation benchmarks, a subset of modifications were selected to decrease the total number of combinations. At least one modification from each theme is included in validation round three in order to experiment with models that take advantage of a diverse set of properties.

From the query expansion theme, the modification that expands the bottom inverse document frequency terms from the WordNet Graph was chosen because it did not negatively impact any validation benchmark and displayed slight im-

provements on the Adi benchmark. Recall from Sections 6.1 and 6.2, all other expansion modifications from validation rounds one and two negatively impacted multiple datasets and only showed marginal improvements on the datasets that were improved. From the Term-Document theme, three different modifications are selected. The first is chosen from round one boosts adjectives that appear towards the front of the document. This modification is an obvious candidate for validation round three because it increases the model's accuracy on all four validation benchmarks. Two more modifications were selected from validation round two: the modification that boosts nouns and adjectives (ID 3) and the modification that boosts verbs and adverbs (ID 4). The noun - adjective modification was chosen because it displayed the highest, percentage increase in overall performance across each validation benchmark. The verb-adverb modification was chosen because it had the smallest negative impact on the benchmark that decreased in its mean average precision. From the Term-Term theme, the modification from the Close Pairs category that measures the density of adjacent adjective and noun query terms from validation round one was chosen. This modification was chosen because it displayed the highest percentage increase in the model's performance across Term - Term modifications that increased the model's performance on at least half of the validation benchmarks. Most of the other modifications within this theme performed very poorly, especially when these modifications were combined in validation round two. Finally from the Parts of Speech theme, the modification with ID 7 was chosen.

Table 6.12 displays the results from round three. In order to limit the size of the graph, abbreviations are used for each modification. The query expansion modification is aliased as QE, which stands for "Query Expansion." The Term - Document modification from round one is aliased as TDA, which stands for "Term

ID	Modifications	ΔMAP% Cranfield	ΔMAP% Adi	ΔMAP% Medline	ΔMAP% Time
1	QE, TD	0.71%	0.45%	0.00%	0.32%
2	QE, TDNA	0.47%	3.96%	3.12%	-4.19%
3	QE, TDVA	1.19%	0.82%	0.09%	1.93%
4	QE, TT	-0.02%	1.05%	-0.43%	-1.30%
5	QE, POS	2.85%	0.32%	-0.51%	2.10%
6	TD, TT	0.34%	1.17%	-0.40%	-0.96%
7	TDNA, TT	0.96%	4.24%	3.08%	-2.74%
8	TDVA, TT	0.36%	0.58%	-0.23%	-1.48%
9	TDVA, POS	3.61%	-0.63%	-0.26%	1.86%
10	TD, POS	3.09%	0.36%	-0.30%	2.22%
11	TDNA, POS	2.60%	4.08%	1.47%	-0.41%
12	TT, POS	1.55%	1.04%	-0.40%	0.11%
13	QE, TD, TT	0.36%	1.02%	-0.42%	0.00%
14	QE, TDNA, TT	0.60%	4.74%	3.05%	-2.82%
15	QE, TDVA, TT	0.34%	1.39%	-0.25%	-1.47%
16	QE, TT, POS	1.56%	1.03%	-0.38%	0.14%
17	QE, TD, POS	3.00%	0.29%	-0.29%	2.25%
18	QE, TDNA, POS	2.60%	4.69%	1.37%	-0.41%
19	QE, TDVA, POS	3.48%	-0.46%	-0.25%	1.88%
20	TD, TT, POS	1.78%	1.01%	-0.35%	0.26%
21	TDNA, TT, POS	1.97%	4.27%	1.51%	-0.51%
22	TDVA, TT, POS	1.73%	0.02%	-0.31%	-1.29%
23	QE, TD, TT, POS	1.81%	1.00%	-0.33%	0.29%
24	QE, TDNA, TT, POS	1.97%	4.63%	1.49%	-0.51%
25	QE, TDVA, TT, POS	1.72%	0.26%	-0.30%	-1.27%

Table 6.12: The change in mean average precision for modifications across multiple themes.

- Document for Adjectives" and the modifications from round two are aliased as TDNA and TDVA, which stand for "Term - Document for Nouns and Adjectives" and "Term - Document for Verbs and Adverbs" respectively. The Term - Term modification is aliased as TT, which stands for "Term - Term." Lastly, the Parts of Speech modification is aliased as POS, which stands for "Parts of Speech."

6.4 Selecting and Testing a Modification

Validation rounds one, two, and three exposed 87 different modifications to Okapi BM25. The procedure to select a single modification to test is now detailed. In the validation rounds, the percentage change in a modification's precision from the unmodified Okapi BM25 model is used as a general heuristic to guide the creation of new modifications. However, in order to determine which modification improved Okapi BM25 the most, the numerical mean average precision scores are used instead of the percentage values. The modification which resulted in the greatest sum of the differences between the modification's mean average precision and the unmodified Okapi BM25's mean average precision across each validation benchmark is the highest scoring modification. Equation 6.1 is the precise definition for how a score is calculated for each modification. Allow B to be the set of validation benchmarks {Cranfield, Adi, Medline, Time}, *mod* represent a modification, and $MAP(m, b)$ be the mean average precision value of model m applied to benchmark b.

$$Score(mod) = \sum_{b \in B} MAP(mod, b) - \sum_{b \in B} MAP(Okapi\ BM25, b) \tag{6.1}$$

The implication of using this procedure, as opposed to comparing the raw percentages, is that some benchmarks will be weighted more than others. The goal is to minimize the affect of a random ordering of documents that may had lead to a high percentage impact for benchmarks that result in low mean average precision scores. For example, the Time benchmark generally resulted in very low precision values for both the unmodified and modified version of Okapi BM25. So the percent change for the Time benchmark should be weighted less than other benchmarks. On the other hand, the Medline benchmark scored high mean

average precision values for both the unmodified and modified versions of Okapi BM25. Higher percentage changes from the Medline benchmark will therefor have a larger impact on the modification's score because the percentage change represents a more significant change in the underlying precision.

Using Equation 6.1, a score for each of the 87 modifications is computed[1]. The modification that resulted in the highest score is ID 14 from validation round 3. This modification combines three different themes of modifications. From the Query Expansion theme, the modification uses the WordNet Graph to expand the bottom inverse document frequency query terms. From the Term - Document theme, the modification rewards nouns and adjectives for appearing closer to the start of a document. Lastly from the Term - Term theme, the modification rewards adjacent adjectives and nouns found in the query for having a close proximity to each other in a document.

As a result of obtaining the highest score, this modification is tested against the Lisa benchmark. Recall that Lisa is the largest benchmark used in this thesis, containing 5,872 documents and 35 queries. Since the best modification includes query expansion from the WordNet Graph, it is worth mentioning how this benchmark compares to the query expansion information provided in Tables 6.2, 6.3, and 6.4. In Lisa, the ratio for the total number of expansion terms for each query to the number of terms that can be expanded is about 2.17 terms per query, which is slightly higher than the WordNet Graph ratios in Table 6.3. Around two expansion terms per query is similar to the WordNet API ratios from Table 6.2 and half as large as the ratios calculated using the Word2Vec model in Table 6.4.

The final results from testing the top scoring modification is what follows.

[1]The results of the calculations will be presented upon request.

When the unmodified version of Okapi BM25 is ran against Lisa, the resulting mean average precision value is 0.357 and when the top modification is ran against Lisa, the resulting mean average precision value is 0.393. The difference between these results represents a 10.25% improvement in mean average precision. A 10.25% improvement is larger than expected so sources of bias were checked for in the Lisa benchmark. A potential avenue for bias may exist in each document because the start of each document begins with a title. After the titles were removed, the unmodified version of Okapi BM25 scored a mean average precision value of 0.304 and the modification scored a mean average precision value of 0.326. This difference between these new results represents a 7.31% improvement in mean average precision. Since the modification outperformed the unmodified Okapi BM25 model on the unmodified and modified versions of Lisa, the test suggests that the modification has improved Okapi BM25's performance.

The performance of the top modification to the unmodified Okapi BM25 model can also be compared using weighted average recall, where the recall scores are weighted proportionally to the number of relevant documents in each query, as shown in Equation 5.4. When ran against Lisa, Okapi BM25 returned recall scores of 0.145, 0.237, and 0.332 on the first 5, 10, and 20 documents returned for each query. Compared to the top modification, the top modification returned recall scores of 0.155, 0.224, and 0.343 on the first 5, 10, and 20 documents returned for each query. From the first 5 document returned, the top modification obtained a recall that is 7.27% better than Okapi BM25. Then the recall score for the modification dipped bellow Okapi BM25 once 10 documents were returned by around -5.56%. However in the long term, after 20 documents are returned, the modification returns a recall score that is 3.17% better than Okapi BM25. The results from the weighted average recall scores indicate that the modification

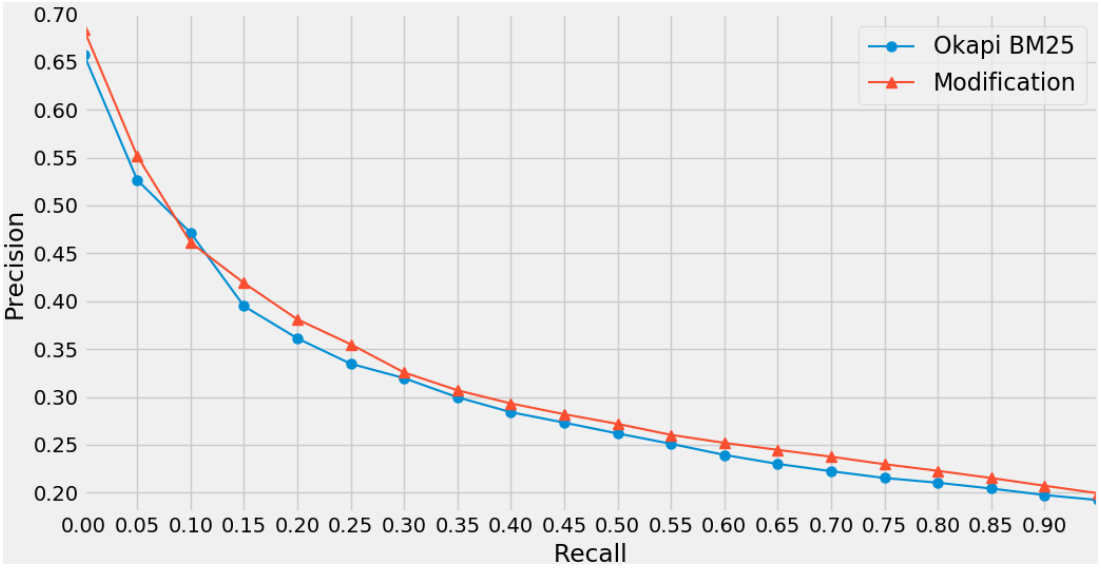

Figure 6.1: Compares the precision and recall of Okapi BM25 and the top modification at recall bucket sizes 0.05.

was able to return more relevant documents sooner than Okapi BM25.

The performance of both Okapi BM25 and the top modification can be displayed on a precision-recall curve to gain more granular insight into the performance of each system with respect to both precision and recall. In Figure 6.1, the blue line with circle markers is the Okapi BM25 curve and the red line with triangle markers is the top modification's curve. In the graph, the recall levels are split into buckets of 0.05, which is similar to saying that each tick on the x-axis represents a point where an additional 5% of the relevant documents are returned for each query. From the graph, it is clear that the modified system scores a higher mean average precision value than Okapi BM25 at all recall levels, except the small recall range between 0.08 and 0.12. Since the final modification combines multiple themes and improves Okapi BM25's mean average precision and recall over the test set in a large majority of cases, this is taken as strong evidence in favor of this thesis' hypothesis.

Chapter 7

Conclusion and Future Work

By modifying Okapi BM25 to take advantage of semantic information located in documents and queries, the model performed upwards to 10.25% better in mean average precision and also scored higher recall scores for most recall levels during testing. The new model has three modifications. First, the model uses query expansion on query terms with low inverse document frequency scores. Second, the model measures how close noun and adjective query terms appear from the start of a document. Third, the model measures the closeness between adjacent adjectives and nouns query terms in a document.

There are plenty of avenues for future work. We do not claim that the hyper parameters used in this study are optimal, so there is a lot of room for improvement. In this study, parameters were either heuristically set or discovered through training on the Cranfield benchmark. However much larger benchmarks, or combinations of benchmarks, can be used to discover more accurate hyper parameters.

Apart from adjusting hyper parameters, research can be done to take more

advantage of language models. A language model can be used to split a query into sections based on the probability that one term follows the next. These sections can then be combined into N-grams to identify multi-term ideas such as city names like San Jose, San Francisco, San Luis Obispo, etc. This is in contrast to creating bigrams according to parts of speech. The discovered N-grams can then be queried for in a corpus. Furthermore, various Skip-gram language models can be used to make more educated guesses for query expansion. For example, a Skip-gram language model may predict that the most likely words to come after "San" would be "Jose" and "Francisco."

This thesis used Wordnet APIs, Wordnet Graph, and Word2Vec in order to obtain appropriate query expansion terms. However, we noted that these models are not efficient at identifying slang terms. One website that specializes in slang is Urban Dictionary[1]. In order to obtain slang terms for query expansion, Urban dictionary can be scraped and built into a probability graph, much like the WordNet Graph The random walk algorithm can then be used in order to discover related, neighboring nodes. In addition to using Urban dictionary to identify slang terms, RDF triples can be used to identify conjunction of ideas. DBPedia[2] contains RDF datasets created from Wikipedia for this exact purpose.

Although this research made significant progress on breaking a query down into its parts of speech, more natural language processing can be conducted. For example, a long query can be broken down into phrases by identifying conjunction words. These phrases can then be ran independently and the results can then be combined.

In addition to trying to improve Okapi BM25, another interesting avenue for

[1]https://www.urbandictionary.com/
[2]https://wiki.dbpedia.org/

future research would be to repeat this experiment for another document retrieval model, such as for cosine similarity. Ranking with cosine similarity would be more resistant to variable query sizes because values can be normalizes before plotting a document in hyperspace.

Bibliography

[1] B. Al-Shboul and S. H. Myaeng. Analyzing topic drift in query expansion for information retrieval from a large-scale patent database. In *2014 International Conference on Big Data and Smart Computing (BIGCOMP)*, pages 177–182, Jan 2014.

[2] A. Bakhtin, Y. Ustinovskiy, and P. Serdyukov. Predicting the impact of expansion terms using semantic and user interaction features. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 1825–1828, New York, NY, USA, 2013. ACM.

[3] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

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

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003.

[6] M. P. S. Bhatia and A. Kumar. Contextual paradigm for ad hoc retrieval of user-centric web data. *IET Software*, 3(4):264–275, August 2009.

[7] R. Blanco and P. Boldi. Extending bm25 with multiple query operators. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 921–930, New York, NY, USA, 2012. ACM.

[8] M. Collins. Language modeling.

[9] R. Cummins and C. O'Riordan. Evolving local and global weighting schemes in information retrieval. *Inf. Retr.*, 9(3):311–330, June 2006.

[10] R. Cummins and C. O'Riordan. Learning in a pairwise term-term proximity framework for information retrieval. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 251–258, New York, NY, USA, 2009. ACM.

[11] M. I. J. David M. Blei, Andrew Y. Ng. Latent dirichlet allocation. Technical report, January 2003.

[12] E. A. Fox. Lexical relations: Enhancing effectiveness of information retrieval systems. *SIGIR Forum*, 15(3):5–36, Dec. 1980.

[13] Google. How search algorithms work.

[14] Google. How search organizes information.

[15] D. Griffiths. *Head first statistics*. OReilly, 2009.

[16] K. Jain, A. Jain, T. Srivastava, and NSS. Intuitive understanding of word embeddings: Count vectors to word2vec, Jun 2017.

[17] S. Kuzi, A. Shtok, and O. Kurland. Query expansion using word embeddings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1929–1932, New York, NY, USA, 2016. ACM.

[18] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 111–119, New York, NY, USA, 2001. ACM.

[19] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. *SIGIR Forum*, 51(2):251–259, Aug. 2017.

[20] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM.

[21] B. Liu. *Web data mining exploring hyperlinks, contents, and usage data*. Springer, 2011.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, ed-

itors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[24] E. Miller. An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology*, 25(1):15–19, 11 1998.

[25] G. A. Miller. Wordnet: A lexical database for english, 1995.

[26] P. Mills. Singular value decomposition (svd) tutorial: Applications, examples, exercises, Oct 2017.

[27] J. Ooi, X. Ma, H. Qin, and S. C. Liew. A survey of query expansion, query suggestion and query refinement techniques. In *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)*, pages 112–117, Aug 2015.

[28] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[29] J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proc. of Cognitive Science Society (CSS-7)*, 1985.

[30] J. R. Pérez-Agüera, J. Arroyo, J. Greenberg, J. P. Iglesias, and V. Fresno. Using bm25f for semantic search. In *Proceedings of the 3rd International Semantic Search Workshop*, SEMSEARCH '10, pages 2:1–2:8, New York, NY, USA, 2010. ACM.

[31] M. I. Rafique and M. Hassan. Utilizing distinct terms for proximity and phrases in the document for better information retrieval. In *2014 Interna-*

tional Conference on Emerging Technologies (ICET), pages 100–105, Dec 2014.

[32] S. Robertson and S. Walker. Okapi/keenbow at trec8. In *The Eighth Text REtrieval Conference (TREC8)*, page 151162. Gaithersburg, MD: NIST, January 2000.

[33] M. Sahlgren. The distributional hypothesis. pages 20 (1) 33–53, 2008.

[34] H. Sanders and J. Saxe. Garbage in, garbage out: How purportedly great ml models can be screwed up by bad data. Technical report, July 2017.

[35] G. W. F. T. K. L. R. H. Scott Deerwester, Susan T. Dumais. Indexing by latent semantic analysis. Technical report, September 1990. Previous number = SIDL-WP-1999-0120.

[36] R. T. Selvi and E. G. D. P. Raj. An approach to improve precision and recall for ad-hoc information retrieval using sbir algorithm. In *2014 World Congress on Computing and Communication Technologies*, pages 137–141, Feb 2014.

[37] R. Song, J.-R. Wen, and W.-Y. Ma. Viewing term proximity from a different perspective. Technical report, May 2005.

[38] L. Soulier, L. Ben Jabeur, L. Tamine, and W. Bahsoun. Bibrank: A language-based model for co-ranking entities in bibliographic networks. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '12, pages 61–70, New York, NY, USA, 2012. ACM.

[39] L. Stanchev. Creating a similarity graph from wordnet. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, WIMS '14, pages 36:1–36:11, New York, NY, USA, 2014. ACM.

[40] D. Vazhenina, I. Kipyatkova, K. Markov, and A. Karpov. State-of-the-art speech recognition technologies for russian language. In *Proceedings of the 2012 Joint International Conference on Human-Centered Computer Environments*, HCCE '12, pages 59–63, New York, NY, USA, 2012. ACM.

[41] O. Vechtomova, S. Robertson, and S. Jones. Query expansion with long-span collocates. *Inf. Retr.*, 6(2):251–273, Apr. 2003.

[42] H.-m. Wang and B. Chen. Content-based language models for spoken document retrieval. In *Proceedings of the Fifth International Workshop on on Information Retrieval with Asian Languages*, IRAL '00, pages 149–155, New York, NY, USA, 2000. ACM.

[43] Z. Ye, J. X. Huang, and J. Miao. A hybrid model for ad-hoc information retrieval. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1025–1026, New York, NY, USA, 2012. ACM.

[44] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 334–342, New York, NY, USA, 2001. ACM.

[45] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. *SIGIR Forum*, 51(2):268–276, Aug. 2017.