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 hights, inspire me to achieve the best within myself, and afford me the opportunity to lead a life filled with unlimited opportunities.

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 [29].

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 [41]. 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:

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

$$q = (w_1, w_2, ..., w_{|V|})$$ (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 9997 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, 45,

31, 44, 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:

$$w_{ij} = tf_{ij} * idf_i \qquad (2.5)$$

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.

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

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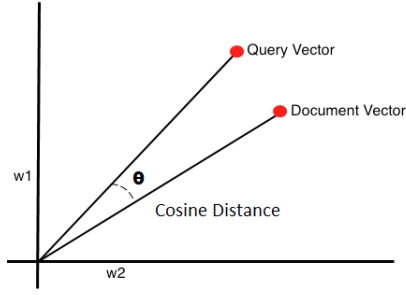
Figure 2.1: The similarity between a query and document in the vector space model is computed as the cosine of the angle between the two vectors.

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

12

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.

$$idf_i = \ln \frac{|D| - df_i + 0.5}{df_i + 0.5} \qquad (2.8)$$

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, $|D|$.

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

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.

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

$$d_1 = \text{``I heart APIs. You heart APIs''}$$
$$d_2 = \text{``I use APIs at work''}$$
$$d_3 = \text{``You work too much''}$$

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

14

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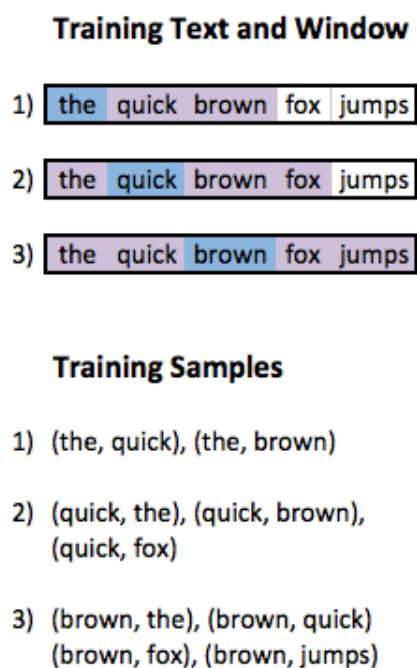
Figure 2.4: Demonstrates how training samples are produced for the Word2Vec neural network from a set of text with a window radius of two.

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,

$$vec(\text{``Germany''}) + vec(\text{``capital''}) \approx vec(\text{``Berlin''}) \tag{2.11}$$

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 modifications to the Okapi BM25 equation to increase the model's accuracy. The next section will discuss other related techniques to analyze unstructured text that do not utilize the Okapi BM25 model.

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 the Okapi BM25 equation, the re-

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.

searchers defines 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 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 hypothesis to test the properties of topic modifying and topic collocating terms:

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 in the form of parts of speech and in the form of bigrams that are composed of words. As opposed to this research paper, this thesis explores how these techniques fair with long queries, as the queries used in this research paper were only 2 terms long. Lastly, the proximity of a term in relation to the front of a document is further explored.

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 2500 text files. Each text file is a passage written by an author. No information was provided on the custom queries. Their results show an increase in precision when observing the first 5 and 10 returned documents by upwards to 20-30% and an increase to the mean average precision by around 15%.

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 2 other hyper parameters. Thus, the total relevance contribution rc is the sum of scores from all non overlapping spans and the new term frequency formula for Okapi BM25 is as shown in Figure 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 [28]. As discussed in [28], 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 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, which 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

corpora.

Research done by Olga Vechtomova, Stephen Robertson, and Susan Jones in their research *Query Expansion with Long Span Collocates* [42] 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 and included the entire corpus (a corpus dependent knowledge model). 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 and 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 depending on the content and size of the query. 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.

A modern approach to query expansion uses language models, which take advantage of the high levels of the contextual dimensionality that a term can appear (language models are explored in detail in Section 3.3). Some researchers such as

---

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

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. Because language models require a lot of training data, their models were trained on the same Trec disks 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 expansions do 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. RDF is a 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. For example, a web page on the movie Deadpool can contain a hyperlink to the movie's director, Tim Miller, which may implicitly indicate that Deadpool was directed by Tim Miller. This information can be expressed explicitly as as a RDF triple: (Deadpool, directed by, Tim Miller), where the movie Deadpool is the subject, directed by is the predicate (encoded as a URI), 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 contents 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 2.7 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 index may be given a positive boost to the term's score because these fields may be highly 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 explicitly defined structure within

documents, which cannot always be assumed.

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 in sum around 95 million documents and 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 exploited 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 (a set of 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 terms that appear in the query that do not show up in a document may still end up close to a document in hyperspace. Therefore, there is a chance that 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."

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 com-bination of words, which belong 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. Queries and documents are relevant to one another if they are composed of a similar combination of topics.

---

foreach *document* $d \in D$ **do**

    foreach *word* $w \in d$ **do**

        foreach *topic* t **do**

            p1 = the proportion of words in d currently assigned to t;

            p2 = the proportion of assignments to t over all documents

             from w;

            p3 = p1 * p2;

            $w \leftarrow (t, p3)$;

        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 a hyper parameter that must be empirically determined. Additionally, LDA does not take into account the proximity and ordering of terms inside a document because it is a bag of words model. 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. Semantic meaning is represented 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.

A language model's predictive behavior lends itself well to solve problems in in other fields beyond document retrieval, such as in speech recognition [43, 41]. This is because a language model will assign higher probabilities to terms that are likely to appear next to each other rather than simply pairing terms based on their acoustics. 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.

3.3.1   The Probabilistic Language Model

In order to rank the relevance of a document and a query, a probabilistic language model will compute the probability that a document will generate a query [18]. To perform this calculation, a probability distribution is first built for each document. In a unigram language model, terms are treated as independent events. So the probability that a word appears in a document is given as the conditional probability between the word and the document [21]:

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

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

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

This simple model can incorporate term proximity in order to gain more insight into the semantic meaning of terms. To do so, the probability of a term will be modified to depend on previous terms in its context:

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

Where $P(A, B, C)$ can be calculated as $P(A)P(B|A)P(C|A, B)$ and n is the nth term in a document. However, this over complicates the language model by placing too much dependence on previous terms. 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:

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

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 now depends

on the frequency of phrases that appear before a term and including that term. For example, the probability that a term appears in a document for a bigram model is:

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

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 $P("and"|"happy") = \frac{count("happy","and")}{count("happy")} = \frac{1}{2}$

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:

$$P(q_i) = P(w_1| <s>) \times P(w_2|w_1) \times P(w_3|w_2) \times ... \times P(</s> |w_m) \qquad (3.7)$$

Where $<s>$ 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 and the size of N. Also, it is not likely that a training set will contain every combination of phrases for high values of N. Thus, the probability distribution that is created is discrete and not continuous. Additionally, it is unclear of what 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. Such a distribution does not

close the semantic gap. A natural next step in building a probabilistic language model would be to smooth the generated probability distributions to mimic a continuous distribution. The main purpose of smoothing is to assign a non-zero probability to unseen words. One of the simplest smoothing techniques is called Laplace smoothing [46], 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 [46].

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 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 and not just individual n-grams. 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. More information on neural language models can be found in Section 2.4.2.

Chapter 4

Solution / 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 the inverted index. The second section describes how an extensible version of the Okapi BM25 model was built so that sets of modifications can be easily enabled. And 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

{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.

contains a reference to the original document and a count for how many times 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 now 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 as well as 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. This 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

Key Term $\rightarrow$ [<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.

procedure because a term's part of speech heavily relies on the 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, because the same term can appear in multiple documents, each key from the inverted index points to a list of postings, where each posting represents data from a unique document. Figure 4.2 provides a summary of this thesis's inverted index.

4.2   Creating an Extensible Okapi BM25 Model

This section covers the implementation details for the 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, as well as 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. For example, if a modification rewards two terms for having close proximity to one another, then the modification would need to boost a document's score relative to the perceived importance of the terms being analyzed. Adding a constant value could even negatively harm the system's performance. For example, 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. At this point, the mathematically inclined reader might recognize an important edge case that derives from the unmodified Okapi

42

BM25 model formula. 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 is common 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 the Okapi BM25 model and $Influence$ is a variable determined from a modification that scales a term's score based on the term's positive, Okapi BM25 score.

$$BoostValue = (|OkapiScore| * Influence) - |OkapiScore| \qquad (4.1)$$

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 a user defined hyper parameter or a parameter whose value is determined through training. Then 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 according to activated modifications. For example, one modification attempts to perform query expansion on terms that score the highest inverse document frequency. The idea is that terms that have a high inverse document frequency are rare, and thus have higher significance to the query. This requires that all the inverse document frequency values for each term to be known ahead

of time in order to only expand the terms that rank highly. Additionally, other modifications require knowledge about the previous term that was evaluated, such as the previous term's part of speech. Since this particular modification requires information about the current term's context within the query, it breaks the bag of words definition for Okapi BM25. As a result, the extensible model always traverses the query in order in which the terms are listed. This is unlike pulling words out of a set where the order of words selected does not influence the final score.

4.3   Okapi BM25 Modifications

This section details every modification that was tested in order to improve the Okapi BM25 model. The sections describe relevant algorithms and mathematical equations that are used to derive an *Influence* score for the particular modification. 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 encompasses modifications that analyze the distance between pairs of query terms found within a document. The third theme analyzes the relative position of a single query term with respect to its position within a document. Finally, the fourth theme explores how query expansion can be incorporated into the Okapi BM25 model.

4.3.1   Parts of Speech Modifications

Originally presented in Section 3.1.5, the BM25F probabilistic retrieval model [30] assumes that some terms within a document are more relevant than other

Category Type	Terms Affected	Parameters	Summary
Boost Up	Nouns	I = DTT and I > 1	Nouns have higher *Influence*
Boost Up	Adjectives	I = DTT and I > 1	Adjectives have higher *Influence*
Boost Up	Verbs	I = DTT and I > 1	Verbs have higher *Influence*
Boost Up	Adverbs	I = DTT and I > 1	Adverbs have higher *Influence*
Boost Down	Nouns	I = DTT and I < 1	Nouns have lower *Influence*
Boost Down	Adjectives	I = DTT and I < 1	Adjectives have lower *Influence*
Boost Down	Verbs	I = DTT and I < 1	Verbs have lower *Influence*
Boost Down	Adverbs	I = DTT and I < 1	Adverbs have lower *Influence*

Table 4.1: All isolated part of speech modifications. DTT stands for a Determined Through Training and I is short for *Influence.*

terms according to a term's location within a web document's HTML. For example, terms found in a header tag should be weighted more than terms found in the body because an article's title is usually a concise summary of the body. Then the researchers heuristically adjusted the weights of each term depending on training sets. Although the documents in this study do not contain explicit structure, implicit structure can be still be used to adjust the *Influence* of terms. This thesis hypothesizes that a document retrieval system's accuracy can be improved if the *Influence* of terms are scaled up or down depending on their parts of speech. This hypothesis is fundamental to the research presented in this thesis because many of the following modifications will be tested on individual or combination of terms with specific parts of speech. The level of *Influence* a particular part of speech will have on the term's overall score is determined by slowly incrementing or decrementing the *Influence* value on a training set until a local maximum mean average precision value is reached. Table 4.1 shows all the part of speech modifications that are tested in this thesis.

4.3.2 Term to Term Modifications

Previous research indicates that incorporating term proximity measures can improve a systems accuracy in cases where queries are small (2 to 3 terms) [40]. Conversely, the effects of adding proximity and phrasal measures can be neutral or even negative if the system's ranking function is already strong [27]. This suite of modifications explore hows term proximity affects a document retrieval system's performance using new mathematical formulas on very large queries (10 terms).

The following term to term modifications take inspiration from Bhatia's and Kumar's idea that pairs of terms are either topic modifying or topic collacating [6]. As a reminder, topic modifying terms are pairs of terms where one query term represents a subject and the other query term modifies the subject. And topic collacating terms are pairs of terms that represent a single topic. This thesis refines and combines these definitions to be pairs of terms where the first term is either an adjective or adverb and the second term is either a noun or a verb. The hypothesis is that a modifying term such as an adjective or adverb only contains semantic meaning if found next to its corresponding subject. For example, if the following query was analyzed by the Okapi BM25 model, "Red cars for sale," the model would retrieve documents that may only contain the term "red" without considering the term it is modifying. Therefore, adjective and adverb modifiers should be analyzed with its corresponding subject in order to maximize a model's accuracy.

Three different solutions are built to satisfy the above criteria, where each solution contains two unique modifications: one for adjective-noun pairs and another for adverb-verb pairs. The first solution excludes the score of all adjective

or adverb query term unless the term that immediately follows is a noun or verb. In order to satisfy this solution, the Okapi BM25 model is modified. As the Okapi BM25 model loops through all the terms in a query, the model saves the score calculated for any tagged adjective or adverb. The modified model does not add this term's score to the document's overall score until the next pass. When the model evaluates the next term in the query, the model checks whether or not the current query term is a noun or verb. If this current term is a subject and the previous is the corresponding modifier and the modifier and the subject appear next to each other in the same sentence within the document, the sum of both score are added to the document's overall score. If one of the conditions is not met, then just the subject's score is added to the document's overall score and the adjective or adverb's score is thrown away. For particularly short queries, this solution may have a negative affect on the model's accuracy because modifier query terms may contain semantic value, even if the term does not appear near its subject. For this reason, a second solution is developed. The second solution determines whether or not there exists bigrams constructed from adjacent query terms that also appear inside the document. If such a bigram is found, the document's overall score is boosted proportionally to the subject's score. Unlike the first solution, the second solution does not remove the score generated from the modifier term. The second solution requires that the *Influence* value be a hyper parameter that is determined through training since there is no mathematical equation that computes how much influence a bigram should have on a document's score.

Now consider the case where a document contains a modifier and a subject that are not directly adjacent to one another. For example, the query "Cheap cars for sale," contains one adjective that influences the subject, namely "cheap."

A document with the following phrase, "Our company offers cheap, red cars at an affordable price" contains two adjectives that both influence the subject, namely "cheap" and "red." However this time, the modifier and the subject are not directly adjacent to one another in the document. Therefore, a distance function needs to be derived to score modifiers and subjects that appear near each other and still grammatically influence one another. Equation 4.2 is used to determine the *Influence* for two query terms that appear near each other within a document.

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

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 one because if two terms are found right next to each other, the difference between their absolution locations is equal to one. For example, in the example document described above, "red" is at position four and "cars" is at position five, so the difference between these two values is equal to one. m is a negative slope value that represents how quickly the distance function should punish two terms as the two terms appear farther and farther away from one another within the same sentence. Finally, $(b - m)$ is the y intercept for the function. The y intercept is adjusted for that fact that when x is equal to one, the value of the function is b. Equation 4.2 takes the shape shown in Figure 4.3 when m is set to -0.25 and b is set to 2.

The distance function is used in three different modifications. The first modification produces distance scores for all adjacent query terms without considering their parts of speech. The second and third modifications produce distance scores for all adjective-noun pairs and adverb-verb pairs. A complete list of the modifi-
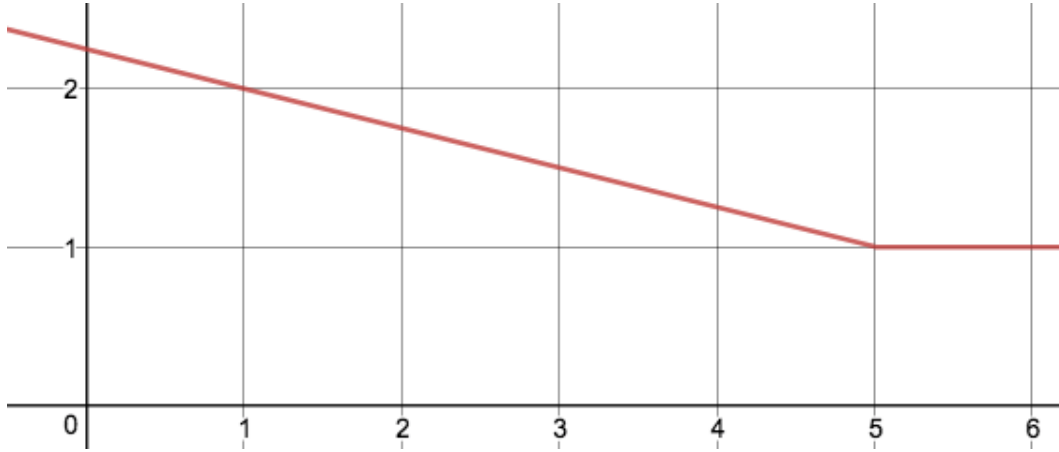
Figure 4.3: The graph of Equation 4.2 when *m* is set to -0.25 and *b* is set to 2. This graph is distance function used to determine an *Influence* score for modifier and subject terms that are found near each other within a document.

cations developed for this theme is summarized in Table 4.2.

4.3.3 Term to Document Modifications

Understanding human psychology is fundamental to the following modifications. Imagine a user has typed a query into a document retrieval system, such as Google. The user is then presented with a set of documents ordered in terms of relevance. What is the user going to do next? Most likely, the user will select the top document to read. But how much of this document will the user read? Most likely, the user will skim the first few lines of the document and then determine if the overall document is relevant to their query. Even if the document contains relevant information at the middle or at the end of the text, the user will not reach this part of the document and will proceed to the next document. The hypothesis is that users expect their query to be highly relevant to the beginning parts of a document rather than the ending parts of a document.

Category Type	Terms Affected	Parameters	Summary
Modifiers and Subjects	Adjectives	N/A	Ignore adjectives not found next to nouns
Modifiers and Subjects	Adverbs	N/A	Ignore adverbs not found next to verbs
Bigrams	All	I = DTT	Reward a document for containing any bigrams from the query
Bigrams	Adjectives and Nouns	I = DTT	Reward a document for containing adjacent adjective, noun pairs
Bigrams	Adverbs and Verbs	I = DTT	Reward a document for containing adjacent adverb, verb pairs
Close Pairs	All	I = max(m*x + (b - m), 1)	Reward a document for containing terms that are close together
Close Pairs	Adjectives and Nouns	I = max(m*x + (b - m), 1)	Reward a document for containing adjectives and nouns that are close together
Close Pairs	Adverbs and Verbs	I = max(m*x + (b - m), 1)	Reward a document for containing adverbs and verbs that are close together

Table 4.2: All term to term modifications tested in the research. DTT stands for Determined Through Training and I is short for *Influence*.

A suite of modifications are developed in order to test the above hypothesis. If users expect relevant information to appear at the forefront of a document, then a term's score should be rewarded based on the term's position within a document. Equation 4.3 shows the distance function that is developed in order to produce *Influence* values that scale a term's score proportionally to the term's original Okapi BM25 value.

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

Where dl_j is the document length of document j measured as the sum of all the terms in the document, idx is the absolute index location of term i measured from the start of the document, and K represents an integer hyper parameter which dictates the upper bound for the function. Equation 4.3 has an inclusive range from one to K, which means that a term is not penalized for occurring at the end of a document, but is rather rewarded for occurring at the start of a document. During experiments, K is set to two with the assumption that terms at the forefront of the document are twice as important as terms that appear at the end of the document.

The modifications that are derived from this equation alter term scores either for all the terms in a query, or according to the term's part of speech. It should be noted that in practice, the relevance scores between all queries and documents for all copora in this thesis are determined by unknown third parties. As a result, these relevance scores may not reflect normal human behavior, that is, the hypothesized human instant to skim the first part of a document in order to determine the document's relevance. It is possible that relevance scores are based on multiple evaluations of query-document pairs that were read through

Category Type	Terms Affected	Parameters	Summary
Is Early	All	$I = (2*dl_j - idx_i)/dl_j$	Rewards terms that appear early in a document
Is Early	Nouns	$I = (2*dl_j - idx_i)/dl_j$	Rewards nouns that appear early in a document
Is Early	Adjectives	$I = (2*dl_j - idx_i)/dl_j$	Rewards adjectives that appear early in a document
Is Early	Verbs	$I = (2*dl_j - idx_i)/dl_j$	Rewards verbs that appear early in a document
Is Early	Adverbs	$I = (2*dl_j - idx_i)/dl_j$	Rewards adverbs that appear early in a document

Table 4.3: All term to document modifications tested in the research. I is short for *Influence*.

to the end. Also, the documents composing each corpus are relatively short (at minimum, around 31 words), so it is possible that these documents are short enough where reading the entire document to determine a document's relevance to a query is not a variable. In either case, the results of the modifications can be found in Chapter 5. A complete list of the modifications developed for this theme is summarized in Table 4.3

4.3.4 Query Expansion Modifications

Related research discussed in Section 3.1.4 shows that query expansion can either help or hinder a document retrieval system's accuracy. In some cases, a system's accuracy is increased when a system uses a language model that is trained on the same corpus that the system is evaluating [17]. Other research suggests that query expansion does not improve a system performance, even if corpus dependent knowledge models are used [2]. The problem with corpus dependent knowledge models is that the document retrieval system that is created is no longer portable to other corpora, so the success of query expansion is totally

dependent on the content of the training corpus. Also, even if query expansion increases a system's accuracy for a particular corpus, retraining the model every time a change is introduced into the corpus is expensive and time consuming, especially when a corpus contains millions of documents. Due to the disadvantages of using corpus dependent knowledge models, this thesis implements non-corpus specific methods to research generic ways to incorporate query expansion into Okapi BM25. Generic query expansion techniques would not rely on the content of corpora (except in cases where a different language is used) and will theoretically be portable to any corpus.

WordNet APIs

The first implementation for query expansion uses the APIs exposed by Word-Net. The implementation is non-trivial because words may have multiple definitions. Word sense disambiguation from the context in which the word appears in the query must be used in order to identify the correct usage of a term. Likewise to how words may have multiple definitions, words in WordNet may subscribe to multiple synsets. The Lesk algorithm, as described in Section 2.4.1, is implemented in order to determine which synset, if any at all, should be extracted from WordNet. Once a synset is obtained from WordNet, the contents of the synset must be cleaned. Elements that contain multiple term phrases from each synset are removed and the remaining elements that contain single terms are stemmed using a Porter Stemmer. For example, given the query "What chemical kinetic system is applicable to hypersonic aerodynamic problems?" the Lesk algorithm analyzes the returned list of synsets and determines that "aerodynamic" can be expanded with the terms ["streamlined", "flowing", "sleek"] and that "problems" can be expanded with the term ["trouble"]. To limit runtime computation in ex-

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

Figure 4.4: Json data structure format for accessing query expansion terms at runtime.

periments, query expansion terms are precomputed for each corpus and stored in a json object that is loaded at runtime. The structure of this json object can be found in Figure 4.4. For example, the above example query is located in the "cran" corpus and has an ID of 1, so the expansion terms for "aerodynamic" are obtained by indexing the json file as: jsonObj["cran"]["1"]["aerodynamic"] .

The Okapi BM25 model is modified to support query expansion using this precomputed json by allowing researchers to toggle WordNet API query expansion through the model's constructor. Unfortunately, WordNet does not provide the strength of similarity between a term and a synset that it subscribes to, so the *Influence* values for this suite of modifications are user defined hyper parameters. This thesis assumes that the original query terms are the most relevant to the user's true intention, so all expanded terms have an *Influence* value of less than one, specifically 0.9, because these terms have a lower probability of being relevant to the original query. Once an *Influence* value is chosen, a query term's score is computed as the sum of the term's original Okapi BM25 score and a boost score which is calculated as the product of the Okapi BM25 algorithm applied to the expansion(s) term and the *Influence* value. In most cases, the boost applied to a query term is equal to zero because it is relatively rare for a expansion term to appear within a document.

Additionally, much like the previous modification themes, the query expansion modifications are divided by parts of speech. Unlike previous modifications, two other modifications are developed. These two modifications are designed to only expand terms that have a relatively high or low inverse document frequency. The

Category Type	Terms Affected	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: All WordNet API modifications tested in the research. I is short for *Influence*.

goal is to determine whether or not expanding rarer or common words lead to a more positive affect on the system's performance. In order to implement these two modifications, the inverse document frequency for all terms in a query are computed and sorted at runtime prior to the main Okapi BM25 algorithm and then only expanding terms that appear in this list. The number of terms to expand from the inverse document frequency list is a user defined variable that can be specified as a parameter to the model's constructor. Table 4.4 provides a summary of the WordNet API modifications.

WordNet Probability Graph

Although WordNet does not measure the similarity between terms and their synsets, recent research shows that similarity scores can be heuristically derived if the WordNet database is arranged in a probability graph and then the random walk algorithm is applied to the node corresponding with a term [39]. In order to execute the random walk algorithm, the probability graph must be loaded from five large json files: adjList.json provides all the adjacency nodes of each node, edgeList.json provides edges between two nodes and the weight of each edge,

idToLabel.json and labelToId.json provide mappings to and from an ID and a node label, and wfVertexDB.json provides a list of all the nodes that are word forms (i.e. word expressions) and the rest of the nodes are assumed to be single term senses. A high level overview of the random walk algorithm can be found in Section 2.4.1 and the exact implementation of the algorithm using the above files is as follows.

Initially when expansion words are requested for a query term, that word's corresponding node is located in the graph. Then, all the unmarked outgoing edges from this node are collected as well as the edge's weights. 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, if a node has three outgoing edges and weights: $\{e_1 = 0.5, e_2 = 0.3, e_3 = 0.2\}$ 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 nodes. 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.

Although this algorithm is scalable to any depth, the implementation runs this algorithm for a depth of two. The random walk algorithm with a depth of two is ran 1000 times in order to generate an accurate probability distribution of which nodes are visited, which is recorded in the frequency dictionary. The frequency dictionary is now used 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 will be less than $iterations \times depth$ because some senses have been removed). Finally, this quantity is then used to normalize the map's key-value pairs such that the sum is equal to one. Then the terms are stemmed using a Porter Stemmer and a descending list of (term, similarity score) pairs is returned. For example, if the algorithm runs 1000 times for a depth of 2 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.

The similarity score calculated from the above algorithm represents the probability that a user is interested in a particular term, given the original query term. The advantage of using the WordNet probability graph is that the similarity score can be used as a term's *Influence* value. One hyper parameter is then required to set a threshold for the minimum acceptable similarity score between the query term and the expanded terms. The Okapi BM25 modifications introduced for the WordNet probability graph affect all parts of speech as well as terms which score a low or high inverse document frequency. Table 4.5 summarizes the modifications.

Word2Vec

The last query expansion technique that is implemented uses English word vectors generated using the Word2Vec algorithm, which is described in Section 2.4.2. The vectors are made up of 300 dimensions and are generated from the Google News corpus[1], which is a large database of text containing three billion running words and about three million unique words. The first few attempts in loaded these vectors crashed the laptop running the experiments because the

---

[1]GoogleNews corpus: https://github.com/mmihaltz/word2vec-GoogleNews-vectors

Category Type	Terms Affected	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: All WordNet probability graph modifications tested in the research. RWSS is short for Random Walk Similarity Score. I is short for *Influence*.

memory required to load the binary is too large for the machine, (the binary is about 3.64 gigabytes). As a result, the python gensim library, which is a open source vector space modeling and topic modeling toolkit, is used to only load the top one million word vectors. Just as how query expansion terms were precomputed in the WordNet API modifications, query expansion terms were also precomputed for the Word2Vec modifications in order to speed up experiments.

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. In order to find the most similar word vectors for a given term, the cosine similarity is computed between the query term and all other word vectors. The highest scoring terms, as well as their scores, are then saved in a json object, which is loaded during runtime. Unlike the json structure presented in Figure 4.4, 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. The disadvantage to using Word2Vec to generate query expansion terms is that the returned list of related terms are the same no matter the context in which the term appears in the query. A word vector is a an amalgam of all term's usages within the training set, as opposed to a specific use case. As a result, word sense disambiguation algorithms cannot be used to extract relevant terms that are specific to a query term's context. This explains why the structure of the Word2Vec json is simply a mapping between a term and its expansion terms and does not require indexing into a particular corpus and query.

The suite modifications for Word2Vec query expansion is the same as Sections 4.3.4 and 4.3.4, namely each modification targets a specific part of speech and low or high inverse document frequency scores. The *Influence* value is set to the cosine similarity score between a query term and its more relevant expansion terms. There is only one user defined hyper parameter, which is to assign a lower bound to similarity scores. This value is set to 0.5 for all experiments. In other words, terms that are only at least half as similar to the query term are used for query expansion. A summary of modifications can be found in Table 4.6.

Category Type	Terms Affected	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: All Word2Vec modifications tested in the research. CSS is short for Cosine Similarity Score. I is short for *Influence*.

Chapter 5

Validation / Results

The main goal of this chapter is to explore the measures, benchmarks, hypotheses, experimental procedures, the experimental results. The measures section will lay the mathematical foundation for judging the accuracy of all the modifications and document retrieval systems. 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. Then the experimental procedure for applying and combining modifications will be described in detail. Lastly, the results of the validation will be presented and the top scoring modification will be tested.

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.

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

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

The validation metrics that are used for measuring modification improvements on a set of queries are weighted average recall and mean average precision. 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. Let the total number of relevant documents over all queries be $|R|$. Equation 5.3 is the weighted average recall formula, where the sum of all the recall contributions adds to one, as shown in Equation 5.4.

$$r_{avg}(i) = \sum_{q \in Q} r(i) \times \frac{s_i}{|R|} \tag{5.3}$$

$$\sum_{q \in Q} \frac{s_i}{|R|} = 1. \tag{5.4}$$

Mean average precision 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, divided by the number of relevant documents R_q for the query q. The mean average precision for a set of queries is the average mean average precision score for all queries, as shown in Equation 5.5.

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{|R_q|} \sum_{l \in L} p(l) \tag{5.5}$$

The performance of document retrieval systems can 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 $r_l = 0\%, 10\%, 20\%, ..., 100\%$. Since each recall level may not be obtained, interpolation is performed by computing the precision at at each recall level using Equation 5.6.

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

If the metrics listed in this section increase in value after a modification, then this is evidence that the modification has improved the accuracy of Okapi BM25.

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 an exhaustive list of relevance scores for all query-document pairs. And during testing, the best

---

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

modification is ran against the largest publicly available information retrieval benchmark in the group, Lisa.

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 Time magazine. An important feature to notice is that the average number of terms inside each document is low, with the exception of the Time benchmark. In total there are just under 3,000 documents in the validation set. Lisa, which stands for Library and Information Science Abstracts, will be used as the test benchmark because it contains the largest number of of documents at 5,872 and the average number of terms in each document is 86.3, which is similar in size to that of Cranfield and Medline.

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.

Table 5.2 provides document parts of speech meta data for each benchmark. The data was gathered using the nltk python library so these numbers serve as an approximation and not the ground truth. From these approximations, it appears that documents 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.

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

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.

of related documents per query, except for the Medline dataset, is relatively low. 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 10 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 the small set of modifications that require parameters that are discovered through training because it contains the highest number of 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 on average contains 34.7 terms, which is higher than that of the validation benchmarks.

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 these approximations, 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

As alluded to in Section 4.3, this thesis explores four overarching hypotheses, where each hypothesis corresponds to a single modification theme. If a suite of modifications in the same theme perform well on the test set, then this will be considered evidence to support the theme's corresponding hypothesis. And if the final model incorporates one or more of the modification themes, then this will also be considered evidence to support the theme's corresponding hypothesis. The list of hypotheses are listed here:

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 within the same document.

3. Okapi BM25 can be improved if terms are weighted differently according to their position inside a document.

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

5.4　Experimental Procedure

This section outlines a procedure for training, validating, eliminating, combining, and testing modifications. The procedure 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 hyper parameter training include all the parts of speech modifications and term to term modifications for bigrams. Hyper parameters are chosen by slowly incrementing or decrementing the value with a small delta 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. Then the modifications are ran with each possible hyper parameter value against the Cranfield corpus. The hyper parameter that results in the greatest mean average precision is used for the rest of the experiments whenever that modification is toggled. 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 obtained, the first round of validation begins. During the first validation round, all 41 individual modifications are ran independently on the four validation corpora: Cranfield, Adi, Medline, and Time. The isolated modifications that result in a lower mean average precision than the original Okapi BM25 model on at least two datasets will be eliminated from future rounds and isolated modifications that result in a higher mean average precision than the original Okapi BM25 model will be used in validation rounds two and three.

The second validation round combines the successful, intra-themed modifications from round one. In other words, 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. As described in Section 4.2, activating multiple modifications is easy since all that is required is to enable the modification in the model's constructor. This new set of modifications is then tested for accuracy in the same way as round one by running the modifications against Cranfield, Adi, Medline, and Time, and eliminating modifications that do not improve the Okapi BM25 model in at least half of the datasets.

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 a document's score when query bigrams appear within a document. These modifications are then validated against Cranfield, Adi, Medline, and Time.

Once all three validation rounds are completed, a single modification across all three rounds must be selected to be tested. The best performing modification is the largest sum of mean average precision scores across Cranfield, Adi, Medline, and Time minus the sum of mean average precision scores for the unmodified Okapi BM25 model. This 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.

5.5 Results

This section details the results after conducting the experimental procedure as described in Section 5.4. The results are divided into four sections. Sections 5.5.1, 5.5.2, and 5.5.3 list and analyze the results from validation rounds one, two, and three. Then Section 5.5.4 compares the results across all three validation rounds, selects the best performing modification, and tests this modification against the Lisa benchmark.

5.5.1 Round 1 Validation

Round 1 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. For all the following tables, if a cell is unfilled then it takes the value above it.

Table 5.5 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 datasets. What is interesting about these results is that the only time an expansion modification actually improved over Okapi BM25 in at least one benchmark, across WordNet Graph, WordNet APIs, and Word2Vec, is when the terms af-

Theme	Category	Terms Affected	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.5: **The change in mean average precision for modifications that involve substitutions.**

fected include either nouns or bottom inverse document frequency terms (this observation is excluding the "All" category since this category 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 better 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 substitution category on each benchmark. For simplicity, assume that the average number of query terms across all four validation benchmarks is equal to ten. The actual values can be found in Table 5.3.

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 5.6: 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 5.7: WordNet Graph query expansion data. Expansion terms were calculated with a minimum similarity score of 0.02.

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

For WordNet API expansions, it turns out that there are about two expansion terms being added for each query, as shown in Table 5.6. Adding two terms to each query equates to an approximate 20% expansion of terms for each query. For WordNet Graph expansion, there are far fewer terms being expanded, as seen in Table 5.7. 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 5.7, 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 query expansion discovers the most number of expansion terms per query, as shown in Table 5.8. On average, Word2Vec adds around four terms per query, which equates to an approximate 40% increase in 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. In reality, there is actually a very low probability that an expanded query term will actually appear within a document. Therefor, the majority of expansion terms have no affect on a document's score. So, in order to confirm or deny Hypothesis 4, round 2 will increase the probability of discovering an expansion term within a document by enabling multiple expansion modifications.

The next theme contains the modifications that reward terms that appear closer to the start of a document, as shown in Table 5.9. 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 Adi and Time in three out of five cases. The modification that targeted only adjectives consistently increased the model's accuracy across all datasets and performed surprisingly well on the Time dataset, despite the fact that the documents in the Time dataset are composed of the same proportion of adjectives as the other validation corpora, which is around 20%. On average, modifications that targeted parts of speech that compose a larger majority of a document 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 dataset).

Theme	Category	Terms Affected	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.9: 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. Therefor, this is interpreted

Theme	Category	Terms Affected	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.10: The change in mean average precision for modifications that involve a single term's part of speech.

as evidence in favor of Hypothesis 3. Additionally, because all modifications positively increased the mean average precision on at least half of the datasets, round 2 will explore combinations of term-document modifications in order to potentially produce an even better system.

The part of speech themed modifications are 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 5.10 is that the modifications that target the same term, but are located in opposite categories, are somewhat inversely related. For example, the two modifications that affect only adjectives are good examples of this relationship. Notice when the Boost Down adjective modification increases the mean average precision score for a particular benchmark, the Boost Up adjective modification decreases the system's accuracy. The same trend is followed by the adverb modifications and the noun modifications also follow this trend with the exception of the results from the Medline dataset.

Some of the modifications from Table 5.10 performed better when the *Influence* of a particular part of speech was decreased, such as the adjective and ad-

verb modifications. Conversely, other modifications performed better when the *Influence* of a particular part of speech was increased, such as the noun modifications. Maybe there is a combination of *Influence* scores across multiple parts of speech that will consistently increase the accuracy of the system. Round 2 will explore this possibility to determine if there exists a efficient ratio between each part of speech's *Influence* levels. If a successful ratio is discovered, then this will be taken as evidence in support of Hypothesis 1.

The final theme places a strong emphasis on the distances between terms and the results can be found in Table 5.11. 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 all documents in each benchmark. For modifications that target parts of speeches that are more common in the datasets, the changes in the system's accuracy 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 is a rare event. So there may not be many opportunities for a term-term modification to change the system's behavior.

The first category, Modifier and Subject, does not include the score from the affected terms unless the corresponding subject appears right afterwards the modifier in a document. As evident in Table 5.11, removing a modifier altogether resulted in very negative scores, with the exception of the Time benchmark that actually saw an increase in accuracy with the adjective modification. Rather than removing an adjective or adverb for not appearing next to its corresponding subject, the Bigram category rewarded a document if it contained pairs of adjacent terms observed from the query in a document. Although the bigram modifica-

Theme	Category	Terms Affected	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.11: The change in mean average precision for modifications that involve the proximity between a pair of terms.

tions 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.

As opposed to the Bigrams category, the Close Pairs category 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 lowers the accuracy of the system across all benchmarks. This may be because the modification is placing extra emphasis on pairs of term which negatively drift the meaning 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 are the two subjects of the query. But the modification disproportionally places emphasis on "fears" and "communist," which can result in unrelated articles. On the other hand, rewarding a document

based on the density of more specific terms such as adjectives and nouns appear to have a positive affect on the system's accuracy. To relate back to Hypothesis 2, it is unclear that simply measuring the proximity of terms will result in a more accurate system. In half the cases, the system's accuracy drops or 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.

5.5.2   Round 2 Validation

Round 2 contains 20 modifications that were created by combining successful modifications within the same theme from round 1. 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 replaced by an ID column because all the modifications combine each category.

The first theme combines query expansion modifications, as shown in Table 5.12. Although the query expansion modifications from round one did not satisfy the criteria for moving to round 2, the analysis of round 1 concluded that that there were not enough terms affected by any one expansion criteria to significantly affect the system's accuracy. So in order to increase the total number of expanded terms, each modification in validation round 2 takes advantage of all categories: WordNet graph, the WordNet API's, and Word2Vec. It is clear from looking at Table 5.12 that using query expansion from all the globally built

ID	Theme	Terms Affected	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.12: **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.**

models results in a decrease in the system's performance. The decrease in the system's precision over each dataset and each modification is evidence against Hypothesis 4. Validation round 3 will not include any multi model query expansion modifications. However, validation round 3 will include the bottom inverse document frequency modification from round 1 since this modification slightly improved the system's precision on the Adi benchmark and did not affect the other benchmarks.

The second theme 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 5.13. All of the modifications improve the system's accuracy on exactly three benchmarks and decrease the system's accuracy on the Time benchmark, except for modification with ID 4. The best modification is number 4 because it has the most consistent increase in the system's mean average precision and

ID	Theme	Terms Affected	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.13: The change in mean average precision for modifications that involve the position of a term within a document across multiple parts of speech.

the lowest decrease in accuracy for the benchmark that it negatively impacted. The second best modification is number 3 since it resulted in the largest increase in the system's performance for the benchmarks that were positively affected. Overall, combining Term - Document modifications did not expose a ratio that outperformed the very best Term - Document modifications from validation round 1. The best two modifications from this round and the best modification from validation round 1 will proceed to validation round 3.

The third theme combines modifications that either boosts the weight of a term up or down based on the term's part of speech. The best performing modification that takes into account all nouns, verbs, adjectives, and adverbs is modification with ID 3. This particular modification positively boosts the weight of nouns and adjectives while negatively boosts the weights of verbs and adverbs. Of the modifications that only decreased the system's accuracy in one benchmark, modification with ID 7 has the best overall ranking. This is because it has the smallest impact on the benchmark it performed worse on and the most positive impact on the benchmarks for which the system improved. In fact, modification with ID 7 performs more accurately than any of the Part of Speech modifications from round 1 when judging based on the same criteria. This is evidence that a ratio can be assigned to terms with specific parts of speech to increase the sys-

tem's accuracy. Therefor, this is interpreted as evidence in favor of Hypothesis 1.

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 5.15, 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 at least two validation benchmarks so these modifications will not appear in validation round 3. The best performing modification across the Term - Term theme from validation round 1 and round 2 is the Close Pairs modification that affected adjectives and nouns.

5.5.3 Round 3 Validation

Round 3 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 datasets, a subset of modifications were selected from validation rounds one and two to combine across multiple themes. Although some themes outperformed another themes, at least one modification from each theme is included in the following list in order to experiment with models that take advantage of a diverse set of properties.

ID	Theme	Terms Affected	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.14: The change in mean average precision for modifications that involve a single term's part of speech across multiple categories. ↑ is short for up. ↓ is short for down.

ID	Theme	Terms Affected	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.15: **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.**

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 improvements on the Adi benchmark. Recall from Sections 5.5.1 and 5.5.2, 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 and assigns higher *Influence* scores to 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 assigns higher *Influence* scores to nouns and adjectives and the modification that assigns higher *Influence* scores to verbs and adverbs. The noun - adjective modification was chosen because it displayed the highest, net 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 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

net 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 that assigned higher *Influence* values to nouns and adjectives was chosen. This modification had the smallest impact on the benchmark it performed worse on and the most positive impact on the benchmarks for which the system improved.

Table 5.16 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 - 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."

5.5.4   Selecting a Modification and Testing

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 pre-

ID	Modifications	ΔMAP% Cran	ΔMAP% Adi	ΔMAP% Med	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 5.16: The change in mean average precision for modifications that involve proximity between pairs of terms across multiple themes.

cision scores are used instead of the percentage values. Now, the procedure for determining the best modification is simply 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. Equation 5.7 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.

$$Score(mod) = \sum_{b \in B} MAP(mod, b) - \sum_{b \in B} MAP(Okapi\ BM25, b) \qquad (5.7)$$

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 lead to a high percentage impact because a high percentage impact may correspond to a low numerical change to the benchmark's underlying score. 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 5.7, a score for each of the 87 modifications is computed[2]. The modification that resulted in the highest score is number 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 5.6, 5.7, and 5.8. 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 5.7. Around two expansion terms per query is similar to the WordNet API ratios from Table 5.6 and half as large as the ratios calculated using the Word2Vec model in Table 5.8.

The final result of testing the top scoring modification is what follows. 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 resultingmean average precision value is 0.393. The difference between these results represents a 10.25% improvement in mean average precision. A 10.25%

---

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

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 the abstract's 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 in overall mean average precision.

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 5.3. 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 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 perfor-
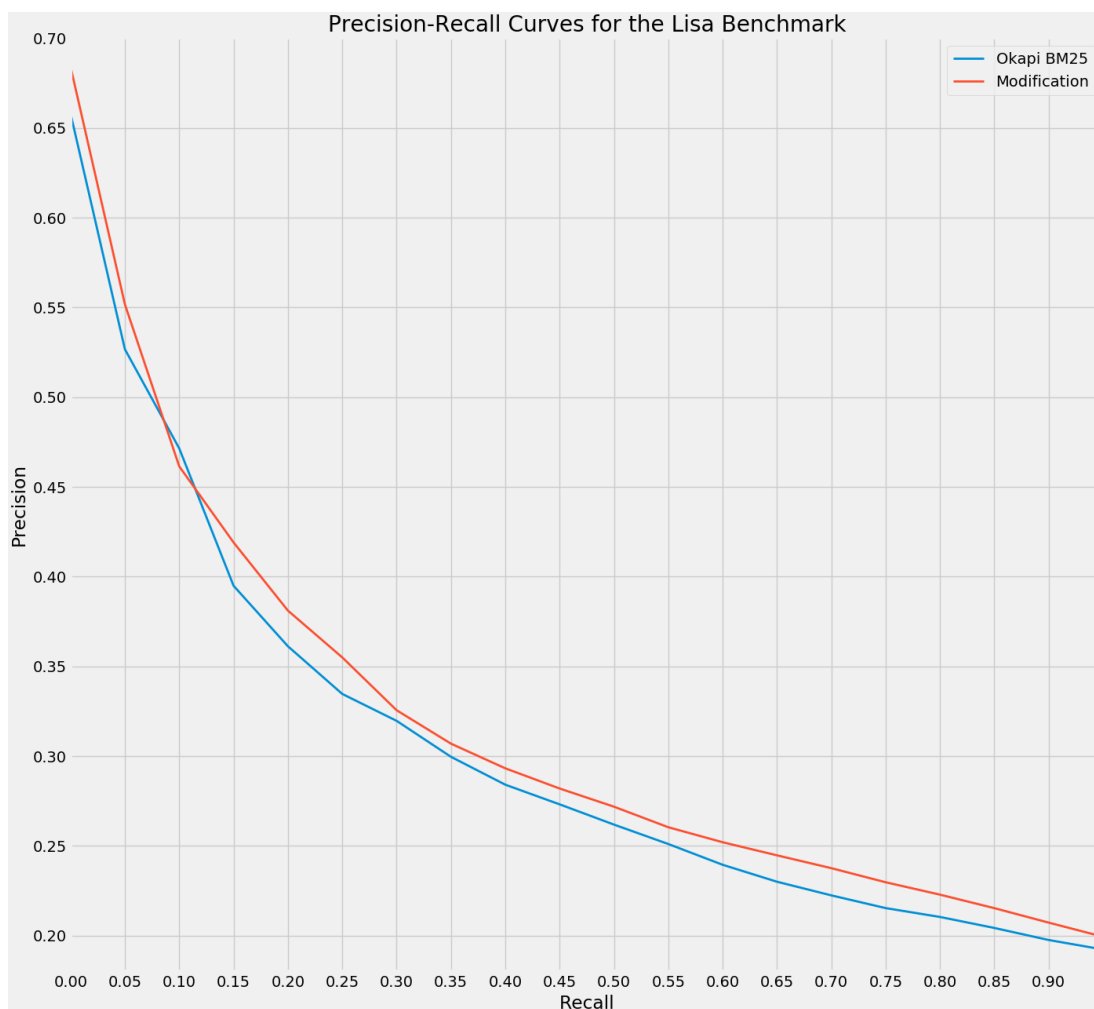
Figure 5.1: Compares the precision and recall of Okapi BM25 and the top modification at recall bucket sizes 0.05.

mance of each system with respect to both precision and recall. In Figure 5.1, the blue line is the Okapi BM25 curve and the red line is the top modification's curve. In the graph, the recall levels are split into buckets of 0.05 each, which is similar to saying that each tick on the x-axis represent 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 where Okapi BM25 outperformed the modification.

Chapter 6

Conclusion and Future Work

Talk about how not all relevant documents have an exact matching between query terms and document terms. Show a few examples. Explain how this can be fixed with query substitution using WordNet.

Use a language model to split a query into n-grams based on the probability that one term follows the next. Then run these n-grams in the bag of words Okapi BM25 model.

Break long queries up by their conjunctions. Then run each query independently and combine the results.

Scrape Urban dictionary for slang terms to incorporate into WordNet datastructures for better query expansion

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] M. Mitra, C. Buckley, A. Singhal, and C. Cardie. An analysis of statistical and syntactic phrases. In *Computer-Assisted Information Searching on Internet*, RIAO '97, pages 200–214, Paris, France, France, 1997. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

[28] 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.

[29] 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.

[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 International 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] K. M. Svore, P. H. Kanani, and N. Khan. How good is a span of terms?: Exploiting proximity to improve web retrieval. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 154–161, New York, NY, USA, 2010. ACM.

[41] 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.

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

[43] 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.

[44] 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.

[45] 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.

[46] 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.