IMPROVING ANONYMIZED SEARCH RELEVANCE WITH NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING

THESIS

Niko A Petrocelli, Captain, USAF

AFIT-ENG-MS-22-M-055

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-22-M-055

IMPROVING ANONYMIZED SEARCH RELEVANCE WITH NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Niko A Petrocelli, B.S.E.E., B.S.C.E.

Captain, USAF

March 26, 2021

AFIT-ENG-MS-22-M-055

IMPROVING ANONYMIZED SEARCH RELEVANCE WITH NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING

THESIS

Niko A Petrocelli, B.S.E.E., B.S.C.E.
Captain, USAF

Committee Membership:

Brett J. Borghetti, Ph.D
Chair

George E. Noel, Lt Col, Ph.D
Member

Warren B. Watkinson, Col, Ph.D
Member

AFIT-ENG-MS-22-M-055

Abstract

Users often sacrifice personal data for more relevant search results, presenting a problem to communities that desire both search anonymity and relevant results. To balance these priorities, this research examines the impact of using Siamese networks to extend word embeddings into document embeddings and detect similarities between documents. The predicted similarity can locally re-rank search results provided from various sources. This technique is leveraged to limit the amount of information collected from a user by a search engine. A prototype is produced by applying the methodology in a real-world search environment. The prototype yielded an additional function of finding new documents related to a provided sample document. The prototype is evaluated using real-world search examples. Results indicate that the Siamese network can produce document embeddings superior to current encoders like the Universal Sentence Encoder. Results also show the promising performance of the prototype in improving search relevancy while limiting user data transmission.

Table of Contents

Page

Page

List of Figures

List of Tables

IMPROVING ANONYMIZED SEARCH RELEVANCE WITH NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING

I. Introduction

There is a need for a text-based search system that can provide relevant search results for a constrained-term search query. This system can be used to provide increased privacy, as well as provide more relevant results for user's who already possess a sample of what they are looking for. There are a multitude of communities who would benefit from a system such as this. Companies doing research who want to protect their intellectual property could utilize this tool to limit the amount of information revealed during their online searches and prevent data leakage on what is being researched. Individuals concerned with their online footprint and privacy could leverage the system to ensure their privacy needs are met. Research communities who find an interesting paper could use this system to find more papers like it, without the need to conduct multiple repeated searches using differently formed queries to find documents similar to the desired document. This system will aid in avoiding the aforementioned problem. The intelligence community also has a need for improved search results without providing tracking data to commercial search engines.

Utilizing commercially available search engines to collect data often requires users to sacrifice private data in exchange for improved search results. Major search engines maintain search histories and profiles to provide more relevant results to the end users. This presents a concern to communities that desire both relevant search results tailored to them and anonymity. The expected contribution for this thesis is a machine-learning based system that improves the relevancy of search results without

sacrificing privacy.

Significant research has already been conducted on improving relevancy-ranked search results. Existing work uses natural language processing tools such as topic modeling and semantic networks to improve search engine results This optimization depends on server-side techniques that collect client information to provide personalized results. These search engines build a profile on the user based on aspects such as career, interests, hobbies, search histories, and past links clicked on [1]. This information is used to tailor search results to the user. These processes do not work when information cannot be collected from the client, such as specific search terms (keywords), user traffic (clicking behavior), and website loiter time. This research will focus on developing a process that can provide similar results without transmission of client data.

Developing a machine learning tool that can determine relevancy on a local machine suffers from severe scalability issues. Any solution must take into account that the user may present search queries the model has never encountered and the model must still be able to detect similarities between queries and search returns. The primary goal is to provide more relevant results to a user while maintaining anonymity, but additionally to provide results in a timely manner, and the ability to scale. Relevancy is related to similarity and machine learning solutions exist that are capable of providing scalable similarity determinations between two entities, even if the entities have not been seen before.

1.1 Attributes and Challenges of Desired System

The machine learning tool developed should meet the following requirements:

1. Provide relevant results to search queries it has never encountered before—in training or in use.

2. Produce relevant results for constrained-term search queries by considering a context that is hidden from the underlying search engine.

3. Provide increased privacy to user by limiting the amount of information relayed to a search engine.

4. Eliminate the ability for the underlying search engine to track user behavior, such as clicks and website loiter time.

5. When given a document, it should be able to find more documents like it.

Some of the challenges this system must overcome include:

1. Producing a machine learning model that is able to handle search queries not encountered during training.

2. Using locally stored information to re-rank search engine returns.

3. Being computationally efficient, as local computer resources will be required to perform calculations.

4. Providing relevancy that is equal to or better than major search engines without leaking client information.

5. The system must be independent of the search engine—it should be adaptable to work with any search engine.

1.2 Assumptions, Limitations, and Constraints

A major assumption for this research is that the underlying search engine has the ability to provide good performance. The developed system is designed to work regardless of what search engine is utilized. If the underlying search engine yields low performance, then the search system will also suffer. The better the underlying

search engine, the better the system's performance. This assumption also leads to a limitation—the system requires an underlying search engine to function. Further limitations arise that stem from the underlying search engine, such as potential biases in the original search engine or lack of relevant results to choose.

The computational resources available for model training are limited to a small computer typically used by an individual. As such there is a constraint with how much data can be utilized during model training. The model training is dependent on a training corpus consisting of queries and documents relevant to these queries. This training data only contains documents that are exclusively relevant to a single query. These documents are mutually exclusive, such that no document shall be found by more than one query. Thus, there is no information in the training set on documents which may be relevant to multiple search queries.

1.3 Research Questions

The previous section discusses an overview of the problem domain, some existing research in the field, desired attributes, challenges to overcome, assumptions, limitations, and constraints. This sections provides five research questions to be answered by this research.

1. How can major search engine results be reorganized in a way that is useful to the user while limiting data transmission?

2. How can a technique be scaled to handle new search queries without requiring retraining?

3. What is the performance of the reorganized results when compared to a major search engine?

4. What is the effect of applying a scalable machine learning technique in this problem domain?

1.4 Impact

The research conducted provides a new search tool that is beneficial to various communities of users. There are many organizations, companies, or individuals who would benefit from reduced tracking and data collection. Data privacy is an enormous concern which is often sacrificed for improved convenience such as improved search results, but the research presented identifies a way to obtain relevant search results without sacrificing anonymity. Military intelligence and corporations protecting intellectual property are examples of interested communities.

In addition to the improved privacy the search tool will provide, it aims to improve search results when provided a user specific context. This context can be populated by a multitude of sources. One such application is if a user were to find something of interest when browsing the web. The search tool will make use of this as context and find more information similar to it. Many users would benefit from functionality such as this, one being the research community. When a useful research paper is found, a "find more like this" capability can be used to find more academic papers that are similar to the initial.

1.5 Document Overview

Chapter II presents an overview of background knowledge and existing techniques in the Natural Language Processing (NLP) domain. It also presents machine learning algorithms that can be applied to this research problem. Chapter III outlines the methodology utilized to answer the research questions. Chapter IV presents the results and analysis of all experimentation conducted in accordance with the method-

ology presented. Finally, Chapter V provides the conclusions drawn from the results as well as potential future work to be conducted.

II. Background and Literature Review

This chapter provides an overview of Natural Language Processing (NLP), text pre-processing, feature extraction, and applications of Machine Learning (ML) for NLP. Broad descriptions and important definitions are provided for each concept. First, text pre-processing and its implications are provided. Then various NLP feature extraction techniques are examined. Lastly, a background on ML techniques that can be applied to NLP is presented.

NLP problems involve the extraction of information from text based data. An example of an NLP problem that ML can solve is Sentiment Analysis (SA). SA is a technique for determining the opinions, sentiments, and subjectivity of text using ML [2]. A simple and practical example of SA using ML is the binary classification of movie reviews from the IMDB dataset [3]. The data consists of multiple reviews that are labeled either positive or negative and the goal is to train a model using ML and NLP to predict whether a movie review has a positive or negative sentiment. ML uses features to make these predictions and the first step is to generate features from the movie reviews. Since the movie reviews are text based, this is where NLP is leveraged. The text must be manipulated and transformed, this is called feature generation. A useful technique for generating features is by tokenization—partitioning the sentences into a list of individual words.

2.1 Text Pre-Processing

Before words, sentences, or paragraphs can be turned into features to identify a document, the text needs to be processed. The output of this process is typically a list of tokenized text that has been cleaned up. Each step in this chain as implications that are both positive and negative [4].

1. Make the text lower case.

    (a) Positive impact: if a word is at the start of a sentence it will no longer be treated differently than if it were in the middle of the sentence. Without this step sentences such as "Dogs like food" and "The dogs ran" would treat Dogs differently from dogs.

    (b) Negative impact: it is sometimes impossible to differentiate between a proper noun and a noun. For example Windows, the operating system, becomes the same as windows in a car.

2. Spell out the contractions.

    (a) Positive impact: contractions can be treated the same as their spelled-out counterparts, can't and cannot will no longer be treated as different words.

    (b) Negative impact: none.

3. Remove websites from the text.

    (a) Positive impact: some of the documents contain URLs, these are not useful features to use—eliminating them improves performance.

    (b) Negative impact: none.

4. Remove numbers from the text.

    (a) Positive impact: eliminates non-descriptive document content such as page numbers, or numbers from a list.

    (b) Negative impact: numeric digits can be useful. The technique could remove important dates which could be good descriptors of a document.

5. Remove punctuation from the text.

(a) Positive impact: USA and U.S.A. refer to the same thing.

(b) Negative impact: acronyms that spell words could be transformed into their word counterpart. For example, S.H.I.E.L.D. would become shield, causing confusion between a fictional organization and a medieval tool.

6. Remove non-ascii characters from the text.

   (a) Positive impact: this ensures the text is in a format that can be universally handled.

   (b) Negative impact: some important descriptive characters that are not ascii could exist and be removed.

7. Remove words too short ($\leq$ 3 characters) or too long ($\geq$ 20 characters) from the text.

   (a) Positive impact: short words typically are not descriptive. Long words could be the body of a website URL.

   (b) Negative impact: some of the short or long words removed may be actual descriptive words.

8. Remove a defined list of stop words from the text.

   (a) Positive impact: eliminates words that exist commonly in all languages, reducing unnecessary features.

   (b) Negative impact: there is a low probability that a stop word could be useful.

9. Lemmatize all the words in the text.

   (a) Positive impact: words are reduced their lemma making words like am, are, and is become the same as the word be.

(b) Negative impact: need to find the correct headword from a dictionary for successful lemmatization.

2.2 Feature Extraction for NLP

After the words have been pre-processed, they need to be represented numerically in order for them to be used as features by a ML model. This is typically accomplished in one of two ways: sparse matrix representation or dense vector embeddings. Both techniques transform text into viable numeric features for model training, and have their own benefits and limitations. This section will explore text feature extraction techniques, their benefits, and limitations.

2.2.1 Bag-of-Words

Before text can be used to train a classifier, features need to be extracted. An example technique is to use word counts. The text from the training documents is used to build a vocabulary, a data structure that stores each word encountered during training. Each document will produce a dictionary that maps this vocabulary to the frequency each word appears. The bag-of-words feature extraction technique does have limitations when applied to machine learning, such as high dimensionality and sparsity. High dimensionality refers to the feature count used to train a ML model and the curse of dimensionality is a term used to describe the phenomena that arises when dealing with high-dimensional data [5]. Often high-dimensional data presents detrimental behaviors on ML models, such as decline in performance, over-fitting the data, or other unwanted behaviors . Thus, feature count is a trade-off between having enough to adequately describe the data while minimizing the cost incurred by the curse of dimensionality.

In this context, the primary contributing factor for highly-dimensional data is

sparsity. Sparsity occurs when the features, in this case the dictionary that maps vocabulary to word frequency, contain many zeros. This occurs because the vocabulary consists of words encountered across each training document. Individual documents will only contain a fraction of the words contained in the entire vocabulary, which results in a dictionary that maps much of the vocabulary to zero. Word importance is a critical aspect that is not included in word count feature generation. When a word occurs often in all documents, it likely is an unimportant word. These word counts are used to distinguish between documents, so when all documents possess a high word count for a given word, that word provides no distinguishing information. Term Frequency—Inverse Document Frequency (TF-IDF) extends words counts to incorporate word importance and address this issue.

2.2.2 TF-IDF

A method for feature extraction from text documents is TF-IDF, an extension of simple word counts that takes word importance into account [6]. The basic mathematical principle is [7] :

$$w_{t,d} = \mathrm{tf}_{t,d} \cdot \mathrm{idf}_t \tag{1}$$

$$tf(t,d) = \frac{f_{t,d}}{\sum_{\hat{t}\epsilon d} f_{\hat{t}}} \tag{2}$$

$$\mathrm{idf}(t) = \log_{10}\left(\frac{N}{n_t}\right) \tag{3}$$

In eq. (1), $w_{(t,d)}$ describes the TF-IDF value for word t in document d, $tf_{(t,d)}$ is the percentage of word t out of the total words in document d, eq. (2). Inverse document frequency describes how much information can be gleaned from a specific

word. If a word is unique to a single document, it provides more information than if it were common across all documents. This can be seen from eq. (3), where N is the number of documents and n_t is the number of documents where word t appears. With N remaining the same, as n_t increases the log becomes smaller until finally, it becomes zero when $n_t = N$. In practice, this means that a word that appears in every document would have a weight of zero. In contrast, a word that appears in very few documents would have a much higher weight. This is useful in ensuring meaningful words carry larger weights when used as features.

TF-IDF produces sparse matrices that do not encompass the semantic relationships between words. These semantic relationships are necessary for relevancy determination. Text cannot be naturally used with machine learning algorithms and so feature extraction must take place. TF-IDF can be insufficient for this task for multiple reasons. Firstly, TF-IDF is a bag-of-words approach which treats each word independently and does not consider word orderings or pairings, such as in phrases. The lack of consideration results in a probability calculation solely driven by the frequency of the independent words. With an N-gram approach, conditional probabilities are also considered. The N is the length of the sequence of words being considered as a unit, for example in a bigram model, where $N = 2$, considering the sentence "The boy ran" would calculate $P(boy|the)$ as well as $P(ran|boy)$. It is also possible to capture items other than words in the N-gram window, such as delimiters, sentence starts, or the end of the sentence. The simplistic approach would not consider the preceding word of "boy" when considering the word "ran".

Secondly, TF-IDF, without feature selection, produces a dictionary using every word it has encountered during training. This dictionary becomes a sparse matrix of features when used to transform text. The matrix is sparse because it will populate zeroes whenever it encounters a word that exists in the dictionary, but not in the text

being transformed. The sparse matrix results in high feature dimensionality which can lead to overfitting and poor generalization of the model on unseen data, considered the curse of dimensionality in machine learning [8]. The sparsity of the data results in a waste of memory as all those uninformative zeros still must be stored. While feature-selection can mitigate some of these issues, experimentation from Section 2.4 showed that optimal TF-IDF feature count increased as more queries were considered, because as the model trains for more search queries, the amount of training data must increase, which results in expansion of the dictionary. For example, in a 10-query model there were approximately 25,000 features to choose from, on average feature selection found optimal performance when selecting approximately 8,000 of those features. While this is definitely an improvement in dimensionality, this is still a sparse matrix, wasting space, and highly dimensional suffering from the curse of dimensionality. Dense vector embeddings can be used to produce matrices that are no longer sparse, and exist in a much lower dimension—often in the hundreds.

2.2.3 Word Embeddings

While the bag-of-words approach produces sparse matrices, the word embeddings approach produces dense vectors. A bag-of-word approach is unable to capture the contextual relationships between words in a sentence. These relationships can be taken into consideration using word embeddings. Word embeddings are much more complex than a standard bag-of-words approach, and often take more time to train. This is an important tradeoff to consider when selected between a bag-of-words and a word embedding feature extraction technique.

A major advantage of word embeddings is the ability to manipulate text as though it were numerical. More traditional ML techniques, such as linear regression, often use analysis to determine relationships between an independent and dependent vari-

able [9]. The process for doing this requires numeric features. Numerical features are already in a usable feature, but categorical features must first be converted to numbers. While feature selection and transformation are often utilized to improve ML algorithms, they focus on numeric features [10]. Other techniques that seek to transform text into usable features, such as TF-IDF, suffer from sparsity and high dimensionality. Often the number of features needs to be limited, and a feature selection technique needs to be employed to ensure that important features are not eliminated [11]. The result is a large vector that consists primarily of zeros, and while this is a valid technique for creating features from text, all of contextual information is lost. Some of these contextual relationships provide useful information and enable analysis that would not be possible with a sparse matrix. For example:

1. Gender can be determined from a word such as queen or king.

2. Mathematical operations can be performed on the word embeddings such as king - male + female = queen.

3. Similarities between words can easily be calculated using the cosine similarity.

A good embedder is defined by the ability to produce embeddings which place similar items in tight groups, resulting in distinct groups of embeddings called clusters. Intra-cluster distancing is the distance between elements within a cluster and inter-cluster distances is defined as the distance between clusters [8]. In the case of words, the embedder should place similar words close to each other, resulting in low intra-cluster distancing. Another attribute of a good embedder is to place dissimilar words far away from each other, resulting in high inter-cluster distance. This same concept extends to sentences, paragraphs, and documents.

2.2.4 Universal Sentence Encoder

Google research has provided models for encoding sentences into vector embeddings [12]. These embeddings are specifically designed to target transfer learning into alternate NLP taskings. There are two model variants available that trade off accuracy and computational complexity. The embeddings generated by both variants of the Universal Sentence Encoder (USE) achieve promising results from Word Embeddings Association Tests (WEAT)– a statistical test designed to evaluate word associations from embeddings [13]. Good performance is also reported for transfer learning. Figure 1 shows Deep Averaging Network (DAN) variant of the USE.

Although the USE reports good associations from WEAT, indicating the embeddings produced capture important contextual information and word associations, they are still designed to primarily embed sentences. This presents a major limitation if the desired outcome is to produce embeddings on longer bodies of texts, such as paragraphs, topics, or document embeddings.

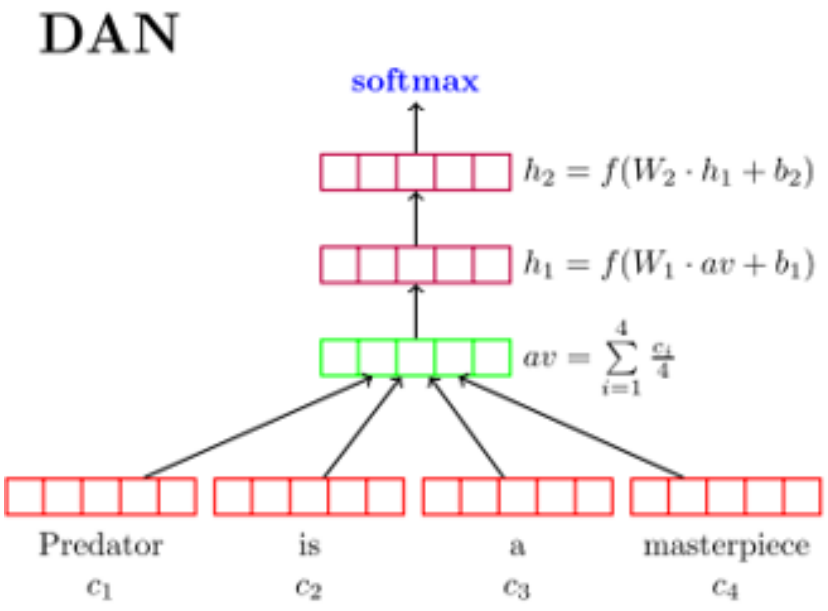


Figure 1: Diagram showing the DAN model of the USE

2.2.5 Word2Vec

Word2Vec has two possible model architectures that turns words into vectors, Continuous Bag of Words (CBOW) and skip-gram [14]. The word2Vec models differ from the USE because they do not embed series of words, sentences, paragraphs, or documents. Word2Vec specifically focuses on transforming a single word into a dense vector. While many pre-trained models exist, it did not fit the application of this project because the domain requires generation of document embeddings. There are techniques available that can combine word embeddings to create paragraph embeddings or even document embeddings. Some of these techniques such as paragraph vector and doc2vec are presented in Section 2.5.2.

2.2.6 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a transformer network model, which maps all output elements to an input element and dynamically adjusts the weightings between them based on their connections [15]. BERT introduces bidirectionally with regards to text, where historically text could only be read from left to right or right to left BERT has the ability to leverage transformers to achieve bidirectionality-the ability to read text from both directions simultaneously.

Sentence BERT is a modification that utilizes triplet loss functions and Siamese networks to derive semantically meaningful sentence embeddings [16]. Previously BERT struggled with tasks that include semantic similarity comparisons, clustering, and information retrieval from semantic searches. Sentence BERT overcomes these shortcomings, improving on the semantic textual similarity benchmark by leveraging the triplet loss function to generate sentence embeddings. This is similar in to the work conducted here, but differs in that the Universal Sentence Encoder is combined

with Siamese networks and the triplet loss function to generate document embeddings as opposed to sentence embeddings. These documents embeddings are used for evaluating semantic similarities between different texts, and determining relevancy of text files in information retrieval systems.

2.3 Techniques for Determining Similarity

Determining similarity between texts can be important in many NLP applications. Similarity could be used to detect plagiarism, or to group documents into topics. In the context of Information Retrieval (IR), text similarity can be leveraged to provide improved relevancy in a question-answer (QA) system. The similarity measurement between the answer and the question could be an indication of how well a search result answers the question.

Embeddings can be considered as a point or as a vector. When an embedding is generated, a sequence of values is created to represent the text. This sequence of values is considered a vector. The embedding could also be treated as a point in n-dimensional space. A default vector begins at the origin and continues to the point with the coordinates in the n-dimensional vector. There also exists a vector that can be computed by subtracting embedding A from embedding B. If there is a vector X from A to B, and a vector Y from C to D, then it is possible vector X encodes the relationship between A and B and vector Y encodes the relationship between C and D. If vectors X and Y are similar then it is possible that the relationship between A and B is the same as the relationship between C and D. For example, if A is "Duchess" and B is "Duke" and C is "Princess" and D is "Prince", then X may encode the embedded direction of Female-to-Male as a vector. One technique for determining similarity is to calculate the Euclidean distance between two embeddings. Equation (4) defines the formula used to calculate Euclidean distance between two embeddings that exists

in n-space where n is the width of the embedding. While this distance can be used to determine how similar words are when applied to word vectors, it can also be used to determine how similar documents are by applying it to document vectors. This same concept can also be applied to a search query and answer. If the query is embedded using the same technique as the results, the distance between the query and the result could be used as a metric for relevancy. The requirement for this to hold true is that the texts that are closely related to each other are placed close in the embedding space. Thus good embeddings are defined by how well this underlying requirement is executed.

$$\begin{aligned} d(p,q) &= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \\ p, q &= \text{two points in Euclidean n-space} \\ q_i, p_i &= \text{Euclidean vectors originating from the initial point} \\ n &= \text{n-space} \end{aligned} \tag{4}$$

Another method to measure similarity between two vectors is using cosine similarity. This metric differs from strictly taking the distance by instead taking the cosine angle between the two vectors [17]. The importance of this is that more information is captured than distance alone as shown in fig. 2. France and Rome vectors may be far apart, but when comparing the relationship of France to Paris and Rome to Italy, these comparisons are similar. Cosine similarity captures this information, while euclidean distance does not. In eq. (5) cosine similarity is calculated given two vectors $\vec{v}$ and $\vec{w}$.

$$\cos \vec{v}, \vec{w} = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}} \tag{5}$$

Cosine similarity is able to capture the angle between the embedding vectors without regard to the magnitude, whereas Euclidean distance captures the distance between the two endpoints of the vectors. In Figure 2 the cosine similarity is displayed between different sets of words/word-pairings. The figure displays an example of how two embedding vectors would appear when reduced to two dimensions and plotted next to each other. The angle between these vectors is what is used to determine similarity. Good embeddings will allow for tight grouping of similar items (words, phrases/pairings, sentences, paragraphs, documents, etc.) resulting in smaller angles for related items and larger angles for less similar items. It can also be used to detect opposites, when the angle between the embeddings is close to 180 degrees, they can still be considered similar as shown Figure 2 that compares city-country pairings to country-city pairings [18].

Further examination of the Figure 2 example on city-country/country-city pairings can be used to further explore the differences between cosine similarity and Euclidean distance. If the embeddings captured city-country pairings, then the cosine similarity for Paris-France and Rome-Italy would be high, because these vectors would be pointing in the same direction. Euclidian distance would provide the distance between the

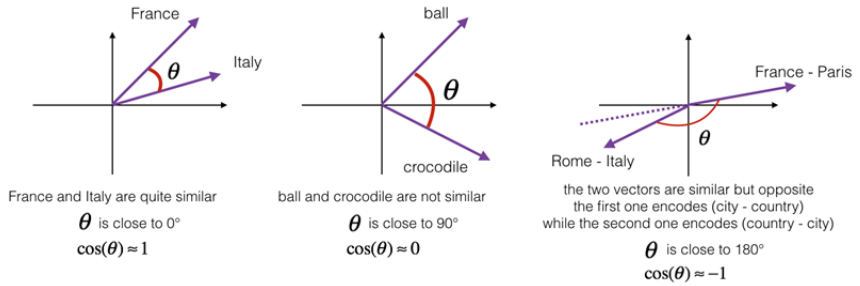


Figure 2: [18] Graphs showing cosine similarity between word vectors

two endpoints of these vectors, providing numerical evidence that Rome-Italy is closer to Venice-Italy than it is to Paris-France.

2.3.1 Techniques for Scoring Search Relevancy

A technique for quantifying the quality of document ranking is a necessary component to score and evaluate the utility of any search engine [19]. Discounted Cumulative Gain (DCG) is a technique that is used to score the performance of a search engine. DCG logarithmically discounts the relevancy rating proportionally to the document's position. Thus if highly relevant documents are ranked low the DCG score is adversely affected. DCG scores can be normalized to compare result lists of varying length. This is achieved by dividing the DCG score by the best possible DCG score. Equation (6) shows the mathematical formula for calculating the DCG of a search engine.

$$\begin{aligned} \mathrm{DCG}_p &= \sum_{i=1}^{p} \frac{\mathrm{rel}_i}{\log_2 (i+1)} \\ \mathrm{DCG}_p &= \text{DCG score at a specific ranking position} p \\ p &= \text{ranking position} \\ \mathrm{rel}_i &= \text{user rated relevancy ranking, if binary} \in \{0,1\} \end{aligned} \tag{6}$$

2.4 Applying Supervised Machine Learning to NLP

Supervised ML is a technique for mapping inputs to outputs. The supervised component refers the data that is provided to the algorithm, which consists of example pairings of inputs to outputs. This data is called training data and is used to develop a function which can be used to predict an output given an input. Supervised ML can be used to solve many problems in the NLP domain, such as sentiment analysis

and text classification.

The input to output pairings that are provided as examples to infer a mapping function is called model training. This data is used as truth data where each input has a known output. This is called a labeled data and can be used to score and improve the function that is inferred to predict outputs. This is achieved by reserving a portion of the labeled data, called a validation set. This validation set will be used to test different models across varying hyperparameters to find the best scoring model. A test set is often reserved from this labeled data as well. This test set will not be used for any model making decisions, instead once the validation set is used to optimize a model, the optimized model is given the test set inputs and is used to predict the outputs. These predictions are compared to the known truth data to generate a score and evaluate the final model.

There are two primary applications of supervised machine learning: regression and classification [8]. The inputs of the model are called features. In a regression problem the output is a number and the features are used to describe the relationship between the dependent variable (output) and one or more independent variables (features). An example of a regression problem is using numerical features (independent variables) such as horsepower, cylinder count, and vehicle age to predict miles per gallon (dependent variable).

In a classification problem, the inputs of the model are features that represent an observation, but the output is a class, or category. An example of this is in NLP sentiment analysis, where a set of words (observation) is mapped to a sentiment (class). For example in a binary classifier movie reviews can have a sentiment of positive or negative. There also exist multi-class classifiers where the movie review could be extremely negative, mildly negative, neutral, mildly positive, or extremely positive. In this example, the movie reviews consists of words which make up the

features used to predict the sentiment. These words need to be transformed into features that can be used for learning. This process is called feature-extraction.

A limitation that arises from supervised ML is the need for those example pairings of inputs to outputs. This need is often referred to as labeled data. The requirement of having labeled data for supervised ML can cause issues if a dataset is not already readily available. Labeling data by hand can be time consuming.

2.4.1 Text Classification Using Multinomial Naïve Bayes

A multinomial Naïve Bayes (NB) model uses the same assumption of all NB models: the probability of every word is independent of the word sense (the intended meaning of the word), its context, and placement in a document [6]. Another critical assumption is that a document's length does not affect the class [20]. This model assumes a multinomially distributed dataset parameterized by $\theta_c = (\theta_{c_{w_1}}, ..., \theta_{c_{w_n}})$ This means for a given class c and every word in the vocabulary, $w_1, \ldots w_n$, the probability of a word appearing in class c is given by $P(w_i|c) = \theta_{c_i}$. Thus $\theta_c = P(w_i|c), ..., P(w_n|c)$ which is a vector produced for every possible class. The multinomial NB classifier estimates the parameter θ_c by using maximum likelihood with a smoothing parameter that can be used to optimize performance [21]. The number of times word i appears in a sample of class c in the training set T is defined by W_{c_i} where, $W_{c_i} = \sum_{w \epsilon T} w_i$, and T is the training set. This can then be used to estimate θ_{c_i} as shown in eq. (7).

$$\hat{\theta}_{c_i} = \frac{w_{c_i} + \alpha}{\sum_{i=1}^{n} w_{c_i} + \alpha + |Vocabulary|} \tag{7}$$

With $\alpha = 1$, Laplace smoothing is occurring. With $\alpha = 0$ there is a possibility for zero probabilities, which can negatively affect model performance. Lastly, with $\alpha < 1$, Lidstone smoothing is happening. Lapcian smoothing can be considered a bias-variance tradeoff, and can be a tunable hyperparameter for model optimization.

The θ_c with the highest probability is chosen as the classification for the document.

2.4.2 Text Classification Using Logistic Regression

Logistic regression is a common tool for using a regression algorithm as a classifier. It classifies by estimating the probability that an observation belong to a class [22]. If this estimated class probability exceeds 50% then logistic regression will predict the observation belonging to that class, if not then it will be predicted as not belonging to this class. This results in a binary classifier, but logistic regression can be extended to multiple classes. This can be accomplished in different ways, such as ensembles, or predicting the percentage of each class and selecting the class with the highest probability, resulting in a multi-class classifier. The logistic regression model is driven by the logistic function, defined in eq. (8).

$$\sigma(t) = \frac{1}{1 + \exp(-t)} \tag{8}$$

Logistic regression can be used for text classification, but like many other NLP problems, must first have numeric features. Once a text is converted to numeric features, logistic regression can be used to classify text as though it were any other ML problem. The feature extraction component of using logistic regression is important because it does add hyper-parameters that can affect performance outside of the scope of normal logistic regression parameters. For example, if the feature extraction technique was TF-IDF, then parameters such as maximum vocabulary, and n-grams could play a role in how the logisitic regression hyper-parameters affect performance. This increases the number of search parameters for ideal performance exponentially.

2.4.3 Partially Supervised/Supervised Latent Dirichlet Allocation

Unsupervised LDA is primarily used to create features, producing topics that are beneficial to categorization. This is useful on large amounts of unlabeled data, but it may be less helpful when the data already has a label. When the model's primary objective is to predict relevance, unsupervised topic modeling may not be the optimal choice. In Blei's paper, supervised LDA is tested against two problems: movie ratings predicted from reviews and web page popularity predicted from text descriptions [23]. This technique could also be applied to estimate the relevance of a document to a given query.

2.4.4 Artificial Neural Networks

Artificial neural networks (ANN) are a type of ML model that are inspired by the way signals are sent through synapses in the human brain. ANNs are especially useful for feature sets that have complex relationships. ANNs are often used in word embedding models because they are capable of capturing the complex relationship between words. Word embeddings can be used as a layer inside a larger ANN and optimized using the triplet loss function to perform one-shot learning and same/not same classification on documents.

An ANN is typically a fully connected neural network where all layer's neurons connect to the following layer's neurons. Each layer receives data as an input, applies a weight, and passes the sum of the weighted data to an activation function. ANNs will improve the results achieved by adapting the weights being applied to the data at each layer or modifying parameters that are part of the system. This tuning can occur during training by minimizing loss.

2.4.5 Siamese Networks for Image Analysis and One-Shot learning

Initially designed to detect fraudulent signatures, the primary role of a Siamese network is to detect similarity between two items [24]. Using a Siamese network, it is possible to create a model that when presented with two images of an animal from the same species it should report that they are the same and otherwise, that they are not. When a new species of animal is encountered that was not seen during training this model could still be used to determine if two images were of the same species, using a reference image and a new image. For example a Siamese network which did not contain images of a giant squid during training can then be shown two unique images of a giant squid and it should report them as the same. If it were shown a picture of a lion and a squid, it should report them as not same. The limitation of this technique is the substantial amount of training data required for the model to learn how to implement this same/not same functionality.

Another example using a Siamese network is for facial recognition, but this presents a new problem, how to train a network on every human's face. One-Shot learning tackles this problem by enabling a Siamese network to classify images it has never seen before by detecting similarity between a known and unknown sample [25]. This results in training the network to learn how to detect similarities between features, rather than similarities between new images and a database of known classes. In facial recognition, a large database of faces is used to train the Siamese network to learn how to detect the same person, regardless of age, or even adaptive features, such as a beard. The network can then be given an image of a new face, not encountered during training, and from a large set of images select images most likely belonging to the same person. An example of this could be providing an image of high value target such as Osama Bin Laden. Even though this network had never encountered images of Osama Bin Laden during training, it could be provided a large set of images from

security cameras and detect which cameras Osama Bin Laden showed up on without a human having to sift through and analyze thousands of photographs. One such function for accomplishing this task is the triplet loss function.

2.4.6 Triplet Loss Function

The triplet loss function is a a technique for numerically comparing a reference input (anchor) to a matched input (positive) and a non-matched input (negative) [26]. Each of these inputs are unique observations. The anchor and the positive must be observations that are similar, while the anchor and the negative must be observations that are markedly different. Equation (9) shows the formula for the function–which uses Euclidean distance to compare the distance between the anchor and the positive, and the anchor and the negative, within a defined margin of α. The anchor is first selected from the training batch at random. From the anchor, a positive and negative are generated. These inputs are then embedded, and the distance is calculated. As this loss is minimized, the embeddings will place anchors closer to positives and further from negatives. This results in embeddings that are capable of placing similar inputs nearer each other in the embedding space, resulting in distinct clusters of like elements.

$$
\begin{aligned}
L(a,p,n) = \max(||f(a) - f(p)||^2 - ||f(a) - f(n)||^2 + \alpha, 0) \\
a = \text{anchor input} \\
p = \text{positive–input similar to anchor} \\
n = \text{negative–input dissimilar to anchor} \\
f = \text{embedding function} \\
\alpha = \text{margin between positive/negative pairs}
\end{aligned}
\tag{9}
$$

2.5 Unsupervised Machine Learning

Unsupervised ML differs from its supervised counterpart by handling data that does not have truth labels. This technique can be useful in handling large quantities of data that has not been curated. This type of machine learning can be combined with supervised techniques to form ensembles, or to label data for future use in supervised learning. It is possible to improve the labeling performance of unsupervised learning by human interaction. A human can review the labels produced and provide feedback to the unsupervised algorithm so that it can improve its labeling capability, this is a form of partially supervised learning.

2.5.1 Topic Modeling/Latent Dirichlet Allocation

Topic modeling is a tool used for classifying topics, especially if they lack labels. One such unsupervised technique that can accomplish this is Latent Dirichlet Allocation (LDA). LDA follows a three-level hierarchical Bayesian model and is a probabilistic generative model [27]. LDA will have a parameter to set the number of topics to emerge, and then will be fed unlabeled data. The algorithm will then produce "topics" defined by the words that occur most frequently in that topic. For an example of an LDA model using topic clustering see Appendix C. In this example LDA was trained on documents related to the Manhattan Project and three topics emerged: topic 1 contains terms that most closely match the Manhattan Project and topics 2 and 3 are topics with words that do not appear often in the Manhattan Project documents. An important observation is that LDA does not name the topics, but rather defines them by their most relevant terms. Therefore, it may still require a human intervention to classify which topic equates to which query, although there may be a way to automate this by extracting key words from a query and comparing them to the topic. The major limitation of this technique is that it requires retraining

every time a new query needs to be added. Creating a way to merge LDA models may overcome this limitation by creating a new model for the new query and exporting its probability matrix. Then exporting the probability matrix of the old model. The two probability matrices would be merged and normalized and then used as a starting point to train a new LDA model with an increased number of topics. In theory, this may overcome the issue of long model training time when a new topic is added.

2.5.2 Paragraph Vector and Doc2Vec

Extending on Word2Vec, is paragraph vector and Doc2Vec. Both techniques retain ordering and semantics of the words [28]. They are unsupervised algorithms that learn fixed-length feature representations. The key component that these techniques expand upon is that they can generate these dense vector feature representations on variable-length pieces of text, such as sentences, paragraphs, and even documents.

2.5.3 Topic2Vec

Expanding further upon Doc2Vec is Topic2Vec. Topic vectors will take a series of document embeddings and cluster them [29]. Documents that appear close together in this cluster are considered of the same topic. These clusters are then used to create an embedding to represent the entire topic. In theory, a query could be embedded and a set of documents, relevant and irrelevant, could be used to generate topic vectors. The topic vector closest to the query embedding, would be returned. This topic vector represents a cluster of documents, that are relevant to the original query. A potential flaw with this technique is difficulty with very specific queries. For example, the specific queries "How many football teams are in the NFL" and "What are common football penalties" will both be close to the same topic vector that represents "football". However, this could be a preliminary technique to gather the generic doc-

uments for the topic vector of "football" to be further sorted by relevancy for specific queries that fall under the topic vector.

Topic2Vec is also another competitor to LDA topic modeling. LDA only works well when the user has a good idea of how many topics they should expect to emerge. Tuning LDA parameters to achieve optimal performance can be time consuming and require a lot of human interaction. Topic2Vec has the potential to solve some of these problems, specifically by allowing topics to emerge naturally, without requiring the number of topics to be specified before training the model. LDA also requires further training whenever a new topic needs to be added. Topic2Vec has the potential to add new topics without needing to be retrained.

2.6 Related Works

Siamese networks have been used to generate word embeddings that enable a ML model to predict a document's topic [30]. This technique varies from unsupervised topic modeling (Section 2.5.1) because it incorporates document labels into the learning process. It is possible for the ML model to discover label-specific topics. This approach to topic modeling can be powerful, but the underlying requirement is that labeled training data is available. If it is not already available, the labeling of the data can cost an extensive amount of time. While predicting a document's topic can be useful for document classification, it does not answer the question of how relevant a document is to a specific search query.

Siamese networks have also been leveraged to detect the similarity between questions [31]. This is a useful technique for determining if two search queries may be identical even if worded differently. For example, the question "Is bronchitis contagious?" and "Can you catch bronchitis from somebody who has it?" can be considered the same question. Siamese networks can be used to detect the similarity between

these two queries and label them as the same question. While this can be useful for a Frequently Asked Questions (FAQ) section of a website or condensing forum topics for already answered questions, it does not solve the problem of determining the similarity between a question and an answer (how relevant or correct is the answer), or between answers for the same question (do the answers agree).

Lastly, a Siamese network has been used to detect the effectiveness of an argument, defined as persuasiveness (how convincing an argument is) [32]. This can be applied to domains other than the field of debate such as information retrieval. The correctness of an answer can be evaluated instead of the persuasiveness of the argument. These three related works justify exploration into the application of a Siamese network in the problem domain of this research.

2.7 Summary

Turning text into features is an integral part of solving NLP tasks. There are multitude of ways to accomplish it, each with their own benefits and limitations. Text rarely comes in a format conducive to creating features, and as such many preprocessing techniques have emerged to generate better performing features. Once features are obtained, there are a multitude of machine learning techniques that can be leveraged to determine document similarity. Document similarity can be used as a metric for calculating relevancy, and eventually rank ordering a list of search returns. ANNs are very powerful tools capable of handling the complex relationships that exist between words. ANNs combined with word embeddings have the potential to solve the NLP problem of detecting document relevancy. Lastly, Siamese neural networks show particularly promising results when applied to problems similar to the one presented in this research.

III. Methodology

3.1 Preamble

This research contains multiple goals. The primary goal is to create a search tool capable of providing relevant search results without compromising data security and privacy. The secondary goal is to create a solution that is scalable, measured by its ability to handle search queries not encountered during training. The performance of the resulting solution is compared to the original search engine, with a goal of equivalent or better performance. The primary technique for accomplishing these goals is by adapting facial recognition algorithms, which detect similarity using Siamese networks, the triplet loss function, and one-shot learning to detecting similarity between bodies of text.

The method for achieving these research goals starts by conducting an analysis of some existing techniques in the supervised and unsupervised classification domain. Feature extraction techniques such as TF-IDF are examined in the problem domain of this research. Then text pre-processing techniques are presented to explain how text is transformed into something useful to model training, and eventually relevancy prediction. After pre-processing, dense vector embeddings are generated, the possible choices are outlined and one path is selected. Next, a Siamese network using triplet loss function is created and trained to improve these embeddings when working with longer documents. The new embeddings are exploited to detect similarities between documents and predict relevancy to a search query. Finally, the developed model is integrated into a research prototype that is applied and evaluated in the problem domain.

3.2 Training Data

The data set that is used to train each of the machine learning models tested comes from the TREC 2020 data set. It consists of an extensive list of search queries. Each search query possesses 100 documents classified as relevant. While each corresponding document is definitively relevant to its search query, they are not all equally relevant. As such the data is ranked by relevancy. This ordering is not taken into consideration during model training. The goal of the data is to train a model to classify the relevancy of a document with respect to a search query. More detail can be found on the training data in Appendix B.

3.3 Text Pre-Processing

The training data in its raw form comes from the TREC 2020 dataset (Appendix B) and must be manipulated into a more conducive format for generating embeddings. Section 2.1 discusses the generic process that text undergoes prior to being sent to the model. The raw dataset comes in a three-file structure: a text file which ties a query identification number (QID) to a ranked list of 100 document identification numbers (DocID), a file that connects each QID to the query text, and a file that contains the document body of each DocID. A visual representation of the file types can be seen in Appendix B.

Due to the extensive file size of the training data, special techniques had to be applied to handle it. In python the files were read line by line, preventing a memory overflow due to file size. Each QID that is going to be used for training is turned into a list of relevant DocIDs. Then the line by line reader scans for the bodies of text belonging to the QIDs on the training list. If the line contains a QID not on the list, it gets skipped, otherwise it gets scanned into memory and stored in a data structure that ties the document text body to the QID it belongs with. Once each QID has all

of its corresponding documents read in, a file is saved that contains all relevant bodies of text pertaining it. This generates a separate file for each QID on the training list, with one corpus that contains 100 documents that each relate to their QID. For future uses, these files can be loaded as the process can take some time. If more QIDs are added to the training list, the ones that already have a generated file can just be loaded. Each step of this process can be seen in Appendix B. The execution logic is shown in Figure 3, each "box" in the figure is saved upon execution to be loaded for any future uses. Appendix B shows the data each step of the way as it progresses through the stages outlined in Section 2.1. The final body of text used in training is a cleaned up list of words, called tokens.

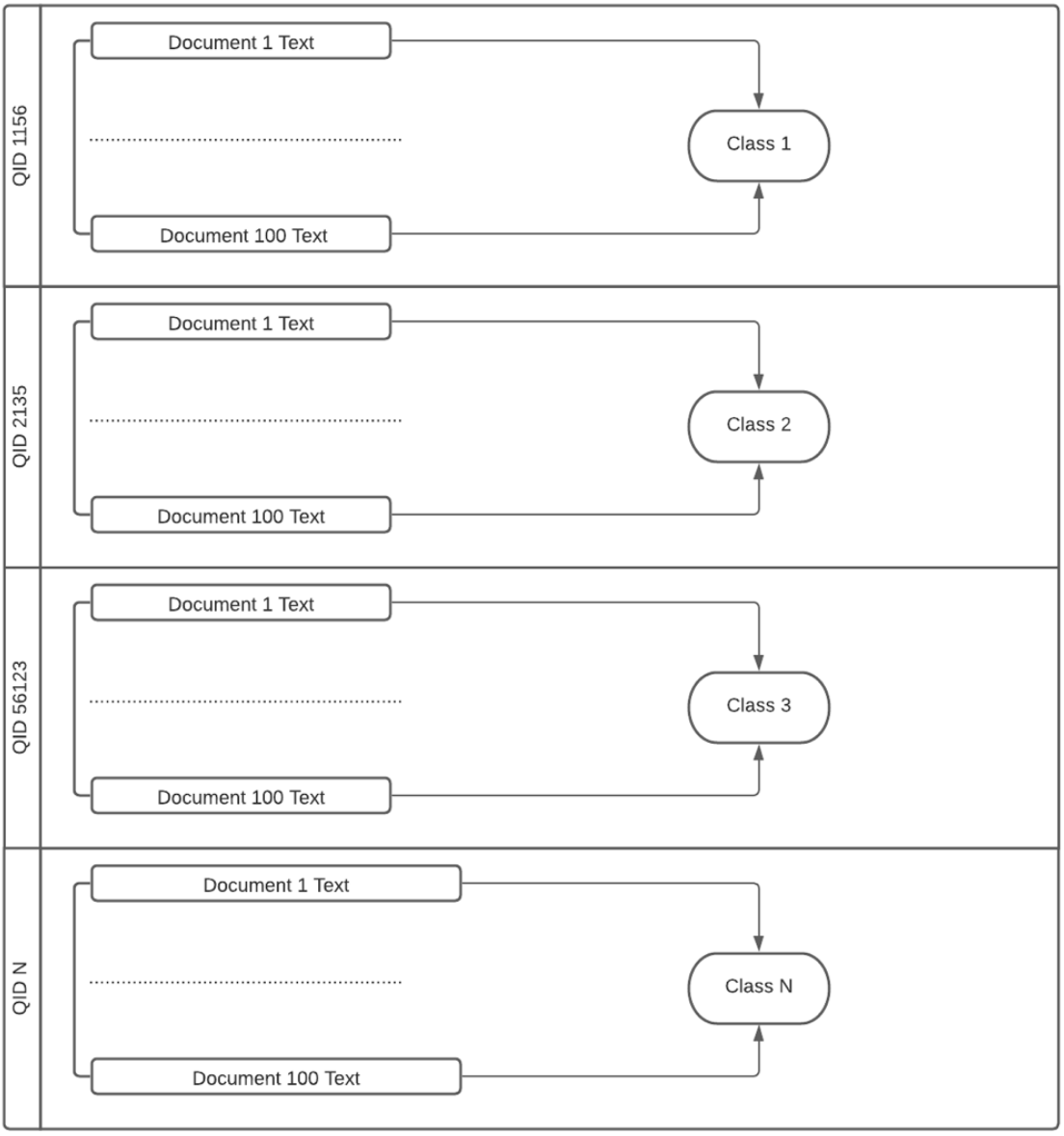


Figure 3: Flow chart of how raw data is organized for training a model. Note N is the total number of QIDs in the training list. The QID numbers in each box are independent of each other and were randomly chosen for the sake of providing an example. They have no significance outside of identifying a query as unique and mapping it to a class.

3.4 Analysis of Existing Techniques

There are many existing techniques that can be leveraged to accomplish the aforementioned research goals. Each have distinct benefits and limitations. Combination of

existing techniques are possible to overcome individual limitations. Some techniques are outright dismissed due to poor adaption to this domain. The techniques presented do not represent every algorithm tested. Logistic Regression (LR) and Multinomial Naïve Bayes (NB) are the two supervised classification techniques presented in this section, but more techniques were also explored, as well as ensembles and multi-stage classifiers. The findings can be found in Appendix A, however, they suffer from the same limitations as LR and NB. The unsupervised technique presented is Latent Dirichlet Allocation (LDA). LDA is also examined in both a partially supervised way via human interaction and combined with supervised techniques such as LR and NB. Lastly, the feature extraction technique used in the classification experiments, term frequency-inverse document frequency (TF-IDF), is analyzed.

3.4.1 Logistic Regression and Multinomial Naïve Bayes for Multiclass Classification

Logistic regression and Multinomial Naïve Bayes are used as a multi-class classifier with the goal of classifying documents, "observations", to queries, "classes". The data consists of 10 search queries each with 100 relevant and irrelevant documents split into three balanced sets: training data (80%), validation data (10%), and test data (10%). The training data is used exclusively for the classifier to learn, while the validation set is used to examine performance as hyperparameters are changed. This is accomplished using halving-grid search with a micro averaged f1 score as a performance metric. The test set is reserved for final score reporting once all parameters are optimized and the model is finalized. The features for all three sets of data are generated using TF-IDF.

Seventeen models are developed, each trained on a different number of search queries, from 3-20. The data for these models is randomly selected from 350 possible search queries. This process is repeated 3 times for both LR and NB, resulting in

102 point estimates of model performance. The model performance results is plotted in Figure 4 and a regression line is fitted to display the trend. The data presented in this figure supports that LR has superior performance to NB and shows that performance in general declines as more search queries are considered. During model training, the validation set is used to optimize the following hyper-parameters, for LR: C value, L1 value, maximum iterations, penalty, solver, and TF-IDF max feature count. The hyper-parameters explored for NB only included alpha and TF-IDF max feature count. The TF-IDF feature count had over 25,000 words to choose from on a 10-query model, the optimal number of words chosen for LR was 4,821 and for NB was 7,128 resulting in an extensive reduction in feature dimensionality for both model techniques. Figure 5 shows the final number of features used after feature selection across the different models tested. Sample confusion matrices produced when the models were executed on the reserved test data can be seen for a 5-query (Figure 6), 10-query (Figure 7), and a 20-query (Figure 8) model. The zero class in the confusion matrices represents a document that does not belong to any of the trained search queries. These basic techniques were also extended to other supervised multi-class classifiers, as well as combination techniques such as ensembles and multi-stage classifiers, sample results can be seen in Appendix A.

Analysis of the experimental observations and data presented in Figure 4–Figure 8 yielded the following observations and limitations:

1. The model must be retrained and hyper-parameters need to be re-optimized to accommodate new queries.

2. From analysis of model performance vs query count (Figure 4): the performance drops significantly as the model grows to accommodate more queries.

3. From analysis of confusion matrices (Figure 6–Figure 8): the zero class, the class

used to identify irrelevant documents, suffers the highest performance impact as scale grows.

4. From analysis of features selected vs query count (Figure 5): the feature dimensionality increases significantly as the model trains for more queries. Even with feature selection techniques, the optimal features trended up as more queries were considered.

The requirement for retraining and re-optimizing holds true regardless of scale. Experimentation showed that even when considering two models trained on the same amount of queries, they always had differing hyper-parameters, performance, and, in general, produced entirely different models.

These experiments exemplify how logistic regression, Naïve Bayes, and classification algorithms in general do not lend themselves to the problem this research seeks to solve. The requirement to retrain and re-optimize a model each time a new search

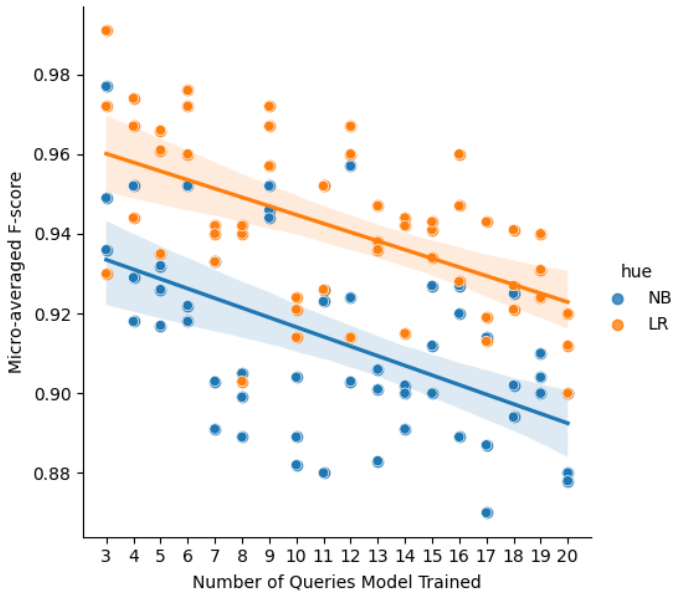


Figure 4: F-score performance of NB and LR as the number of queries the model is trained on increases. The solid lines represent regression lines that show performance to trend down as query count increases.

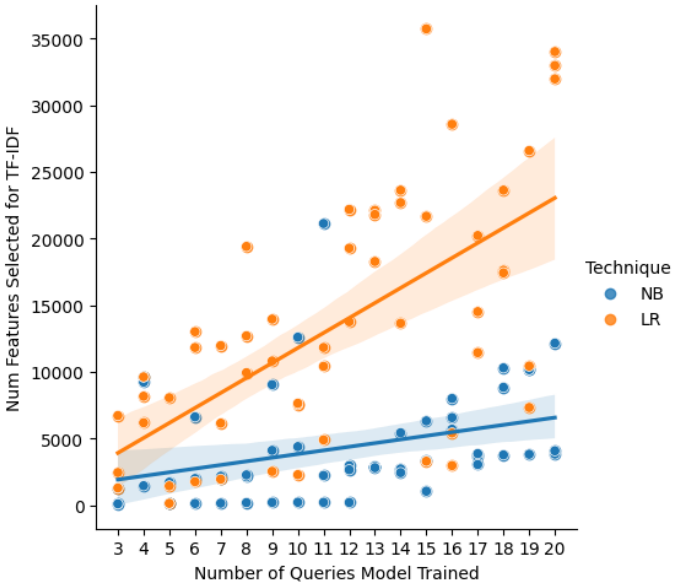


Figure 5: The number of features used after feature selection across trained NB and LR models.

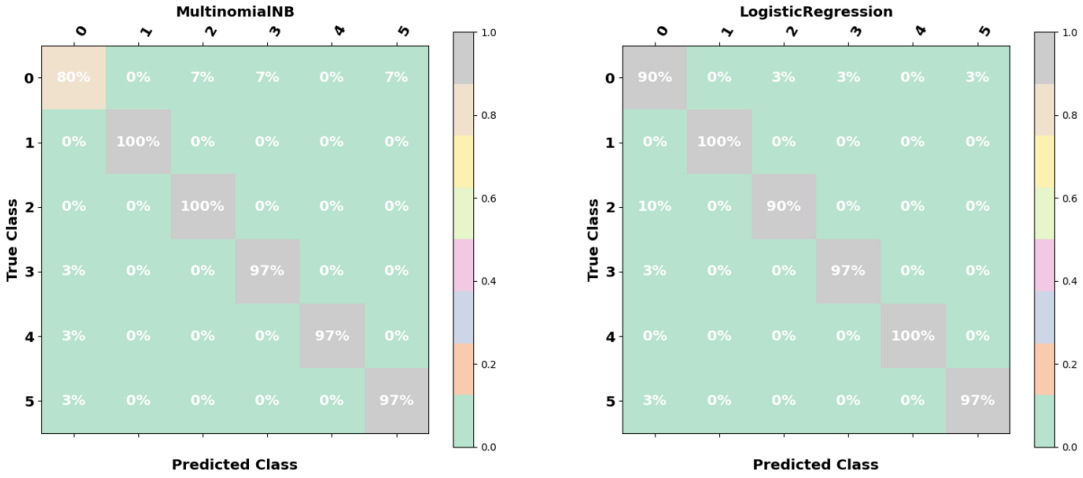


Figure 6: Final model performance on reserved test data for a 5-query model

query needs to be added renders these techniques infeasible. The performance drop as the model is scaled to accommodate more queries is also concerning. A final consideration is the amount of data required, dozens of relevant/irrelevant examples, to train each model iteration, when a new query needs to be added. It is not possible to train the model on every possible search query a user may enter because there is

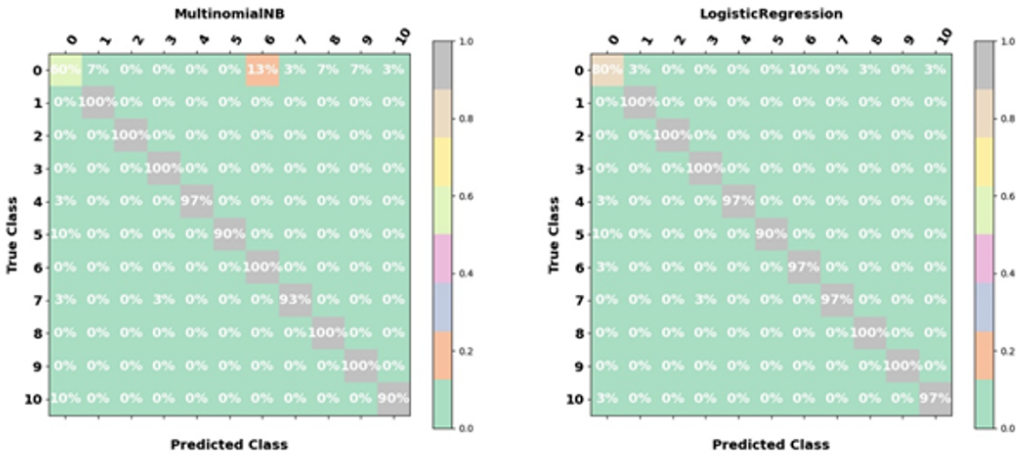


Figure 7: Final model performance on reserved test data for a 10-query model

an infinite amount of queries a user may input. As such, the experiments show that a supervised classifier is insufficient for the task this research seeks to accomplish.

3.4.2 Latent Dirichlet Allocation and Topic Modeling for Relevancy Prediction

An experiment similar to that found in Section 3.4.1 is conducted with unsupervised machine learning. The data for this experiment is for three search queries, each with 100 relevant documents. Again the data is split into three sets: train (60%), validation (30%), and test (10%). The validation set is used to tune the hyperparameters of LDA and find the optimal model. LDA is used to generate four topics. Three of four topics had words closely related to the trained search queries. Unlike supervised models, an advantage LDA provided is that it did not require irrelevant documents to train. Instead, LDA is used to predict the probability a document belongs to one of the topics it produced. If this probability is lower than a threshold, then it means it is a poor match for any of the topics and thus is considered irrelevant. An example of this is a probability vector where it predicts 30%—Topic 1,

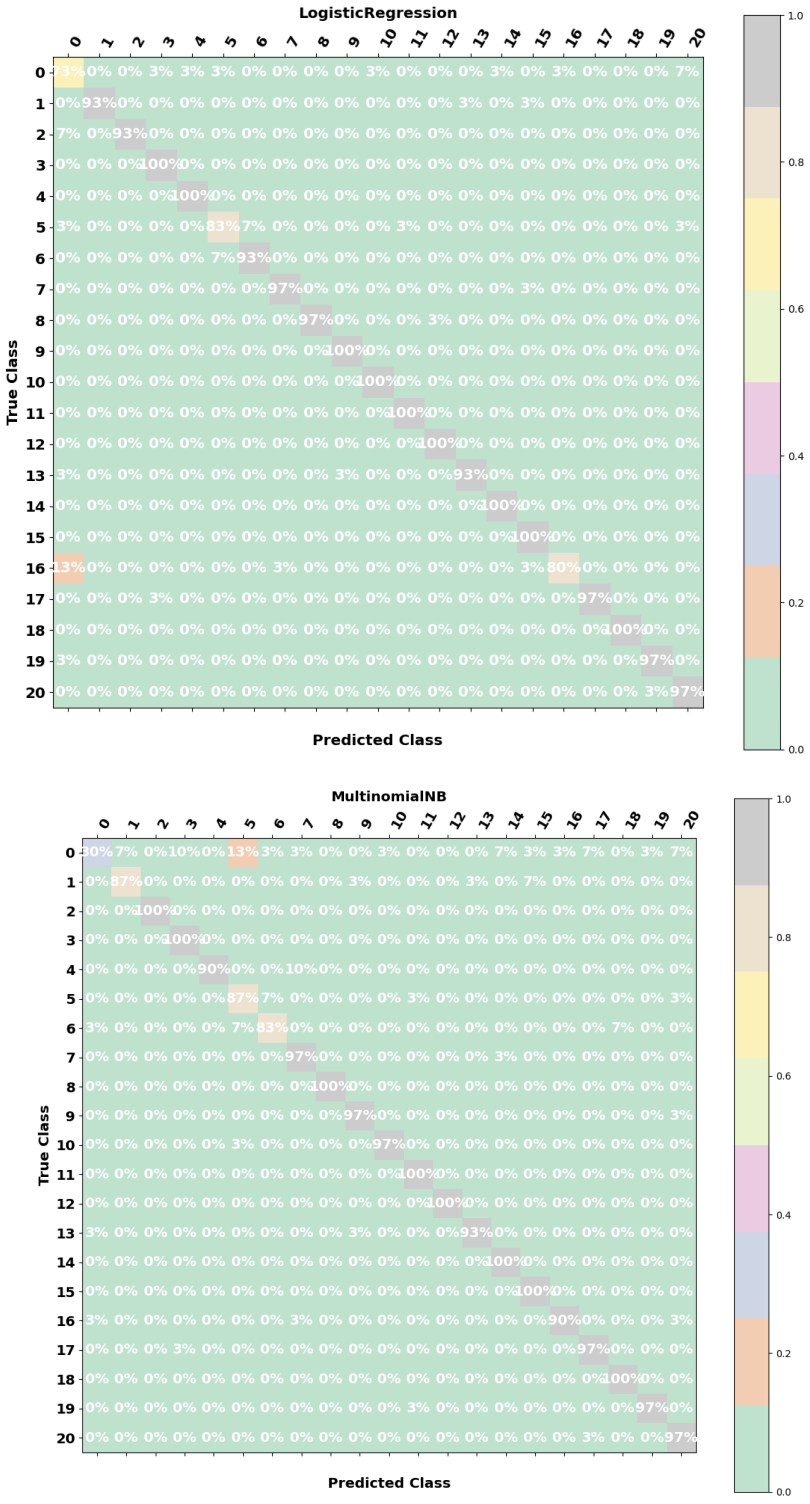


Figure 8: Final model performance on reserved test data for a 20-query model

30%—Topic 2, 40%—Topic 3, this shows a poor match to any of the topics and therefore it would be classified not relevant. This confidence threshold becomes a tuneable hyper-parameter, and its effects on the validation data can be seen in Figure 9. A visualization of how LDA works is shown in Appendix C.

A limitation that arises from using LDA is that the number of topics LDA produces is a hyper-parameter, and the optimal number does not necessarily equal the number

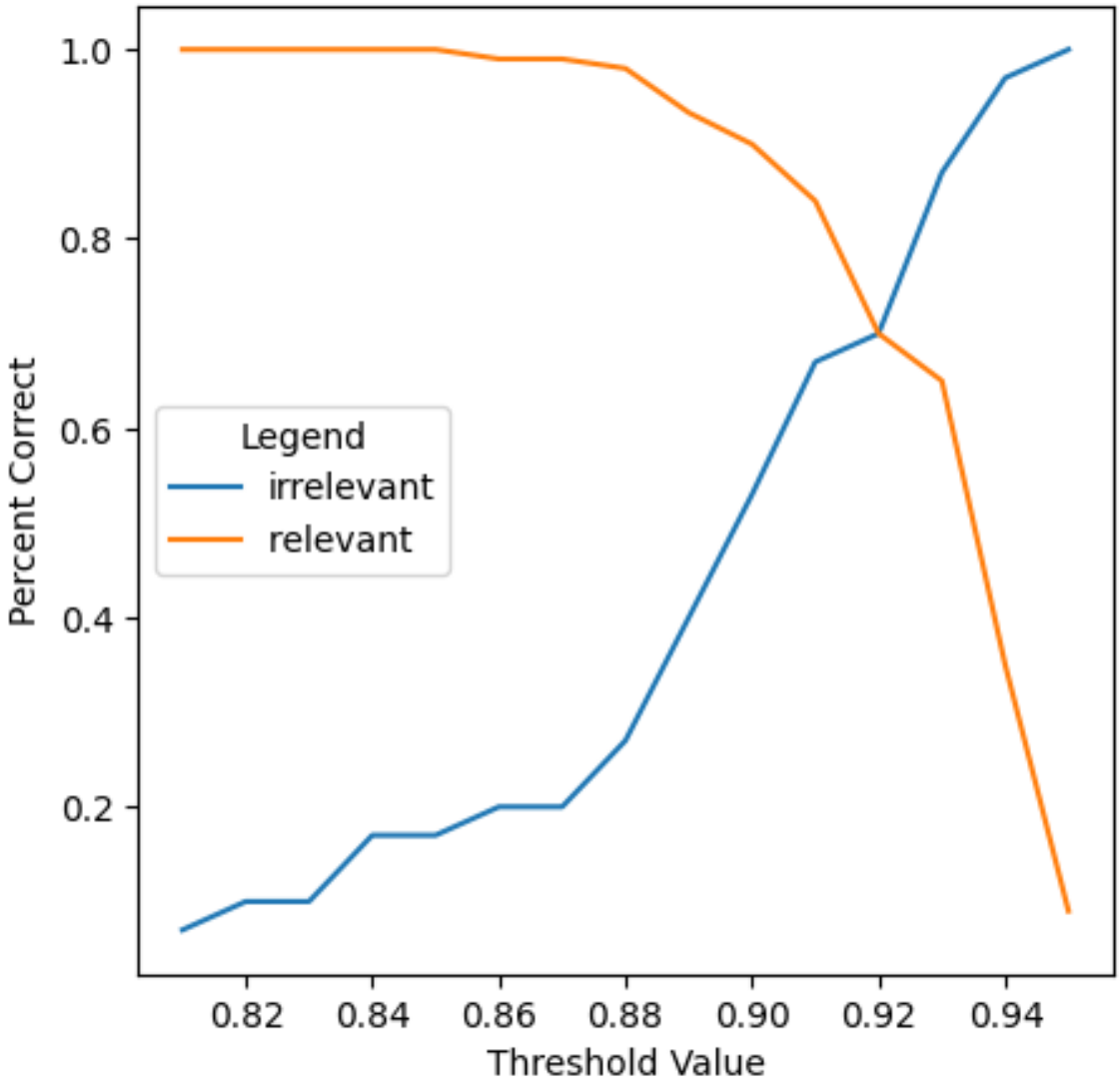


Figure 9: Threshold value for LDA probability that a document is relevant vs percent of correct predictions. The blue line represents irrelevent predictions, while the orange represents relevant predictions.

of search queries. The topic numbers that are generated by LDA do not coincide with the class numbers assigned to each query and so a lookup table must be created. Figure 10 shows the topics that emerged for the queries [‘what helps with burns’], [‘weft knit definition’], [‘size of check in baggage for international’]. Matching the words from the LDA topic descriptors to the search queries yields the lookup table in Table 1. The limitation is the manual labor involved in using LDA to predict relevancy. In this experiment, only three queries were examined, but the issue lies in scalability. Creating a lookup table for a large number of topics becomes much more complex and time-consuming.

LDA Topic Number	Class
0	0
1	2
2	1
3	3
No Fitting Topics	0

Table 1: LDA lookup table for classification

The results for LDA on the validation data can be seen in the top left of Figure 11. This shows excellent performance at predicting relevancy for the trained queries, but poor performance at detecting irrelevent documents. Thus, combination with supervised techniques is explored. Equally weighted ensembles with LR and NB were examined, Figure 11, and a multi-stage classifier of feeding LDA predictions into LR is examined in Figure 12. As Figure 9 shows, it is also possible to prioritize classifying relevant documents. While this improved the performance of LDA by itself, it resulted in a minimal effect to the ensembles, and performance degradation in the multistage classifier.

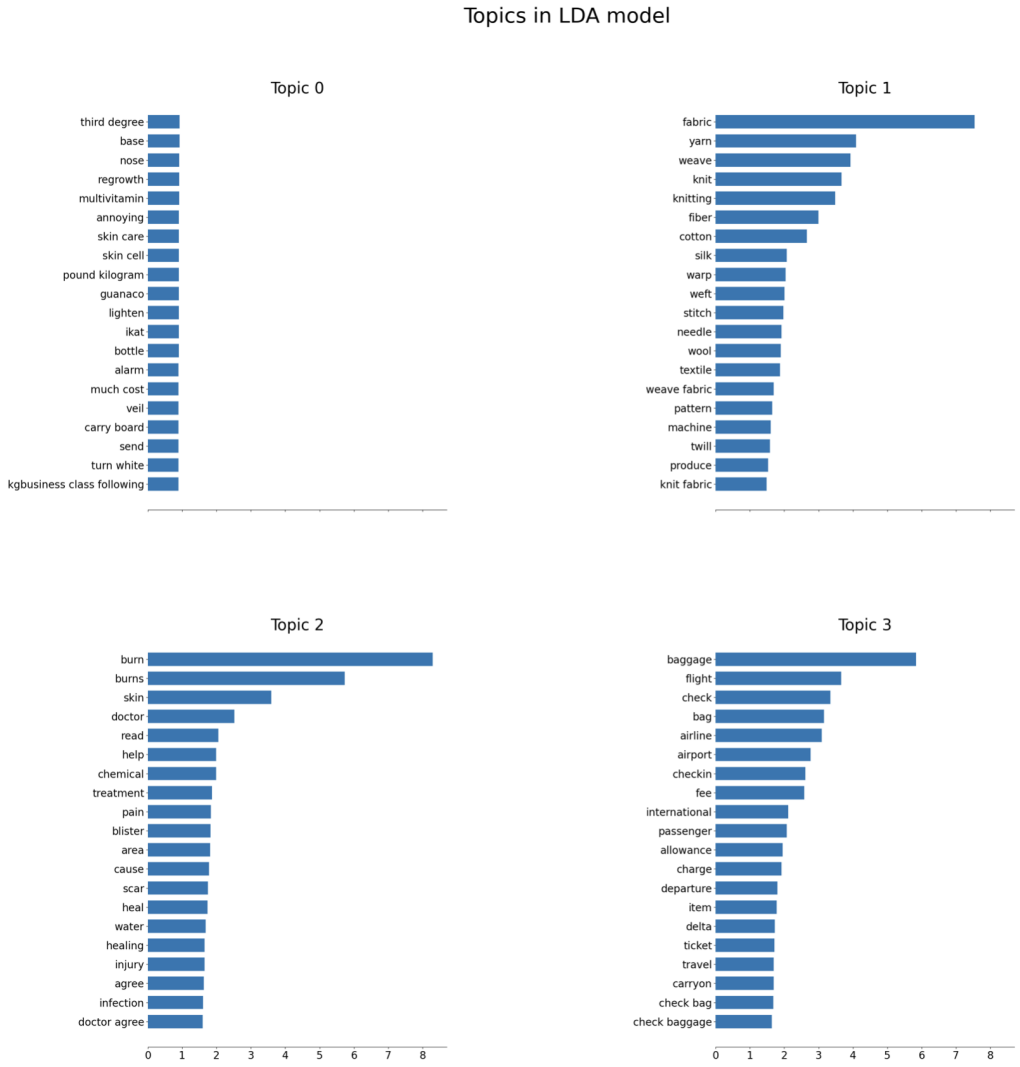


Figure 10: The topics that emerged from running LDA on the training data. Each topic is defined by the set of words shown.

While the ensembles and multistage classifier solve the problem of poor performance at predicting irrelevant documents, new flaws now exist. Firstly, the same limitations that existed for supervised classifiers carry over and compound. LDA and the supervised techniques need to be retrained each time a query is added. LDA alone has the ability to export the probability matrices as a starting point for retraining when new queries need to be added, substantially reducing the retraining time. The

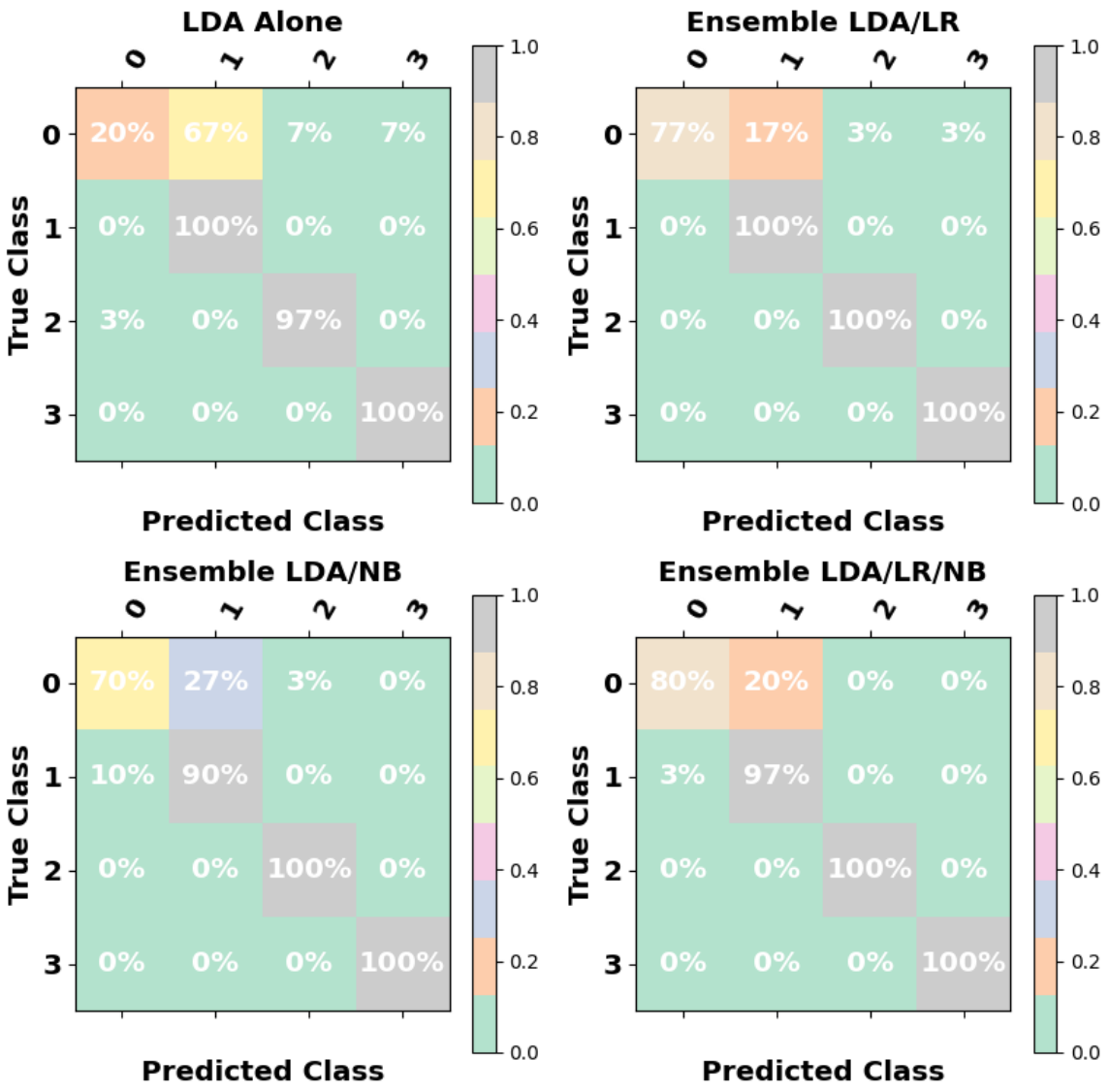


Figure 11: Confusion matrices generated by LDA and Ensembles

addition of a supervised technique now re-applies the limitation of retraining. Secondly, the number of topics that emerge from LDA does not need to equal the number of classes. It is possible that a situation where $P(\text{class}_1) = P(\text{topic}_2) + P(\text{topic}_3)$ could arise. This means that the probability of class 1 could equal the sum of probabilities for two (or more) topics. This occurs when multiple LDA topics relate to a single class. In the experiment conducted, this did not occur, but a similar issue occurs

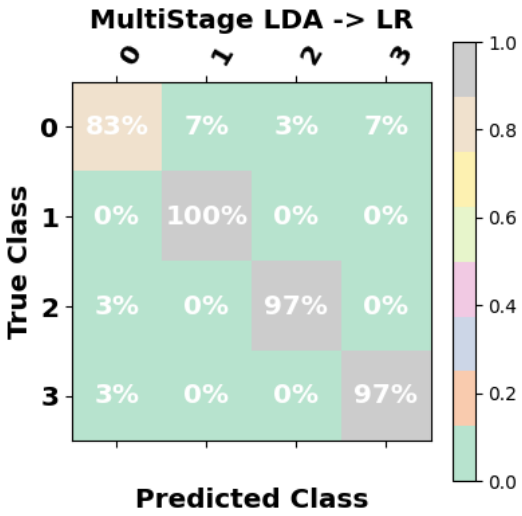


Figure 12: Confusion matrix generated by LDA multistage classifier

where topics do not line up directly with classes (seen in Table 1).

The code that required to handle these techniques creates a list and a dataframe. The rows in this dataframe are documents, and the columns are topics, each cell filled in with a probability. An example of a row is Document 42: 5%, 1%, 90%, 4%. Indicating Document 42 has a 5% match to topic 0, a 1% to topic 1, a 90% to topic 2, and a 4% to topic 3. For each row in this dataframe (for every document), the topic and the probability are examined. A dictionary is created representing each class and initialized to zero. If the topic is 0, then the probability gets added to class 0; if the topic is 1, then the probability gets added to class 2, topic 2 goes to class 1, and topic 3 goes to class 3. This logic follows the lookup table from Table 1, except it, uses probabilities.

This technique can be adapted for when multiple topics point to a single class. For example, the case $P(\text{class}_1) = P(\text{topic}_2) + P(\text{topic}_3)$, the probability for topics 2 and 3 contribute to the probability of class 1. To accomplish this in code, class 1 would have probability added whenever the topic number matched 2 or 3. This is

even more complex to implement than just using a lookup table and further suffers from scalability issues. Repeating this process by hand on a model with many queries is unrealistic. The multitude of problems that arise from trying adapt LDA to the problem this research seeks to solve show that unsupervised and partially supervised topic modeling are not valid approaches.

3.5 Generating Document Embeddings

There were three possible paths for generating embeddings (Section 2.2.3): make new ones from scratch, use existing embeddings, or attempt to improve existing embeddings. The approach taken in this research was to improve existing embeddings. The reasoning behind this choice is that generating new embeddings from scratch would be computationally costly, take a long time to train, and require a large amount of labelled data. Using existing embeddings is a valid option, except that the readily available embedders such as Universal Sentence Encoder (USE), word2vec, paragraph2vec, and BERT thrive at creating shorter embeddings, such as word, sentence, or paragraph embeddings. What was ideal for this research is for a document embedding. While there is an existing document embedder, doc2vec, available, it relies on unsupervised learning and may not be trained appropriately for the use case that was ideal for this research. As such, the path chosen was to improve existing embeddings to create quality embeddings for entire documents.

3.6 Improving Embeddings Through Siamese Network with Triplet Loss Function

Once the data is properly organized and pre-processed, it is time to train the model. The model used is a Siamese neural network trained on the triplet loss function. From the processed data, an anchor will be chosen. An anchor is a document

that will be used to produce a positive and a negative. The positive is a separate unique document of the same class(QID) as the anchor, whereas the negative is a separate unique document of a different class (QID) than the anchor. These documents are then embedded using the Universal Sentence Encoder (USE) and passed into a fully connected deep neural network. The triplet loss is calculated and the weights are adjusted. The process is repeated until the loss is minimized. The network is able to take the embeddings from the USE and add relevance information between documents into the embedded values. This allows for the embeddings to be more suited for entire documents, as the initial embeddings produced by the USE are designed to perform well with shorter inputs, such as words and sentences.

3.6.1 Randomly Select Document

In this stage a document from the list of training data will be selected at random from any possible QID on the training list–this document becomes the anchor. Once chosen, another distinct document belonging to the same QID as the anchor is chosen–this document becomes the positive. Lastly, a third document is chosen belonging to any QID other than that of the anchor–this document becomes the negative. These three documents, collectively referred to as a "triplet" will be used as a single training sample. There are a number of training samples in each batch. Each training step will make use of a different triplet batch, that is the triplet (A_i, P_i, N_i) is used at most once. There are a number of steps per training epoch, and a number of total training epochs. While the triplets will not be re-used for multiple steps within a single epoch, it is possible, though statistically unlikely, that they can be reused in a single step inside a different epoch. So triplets are unique per step, but not necessarily per epoch. For this research batch training was utilized, with a batch size of 64, 100 steps per epoch and 500 epochs.

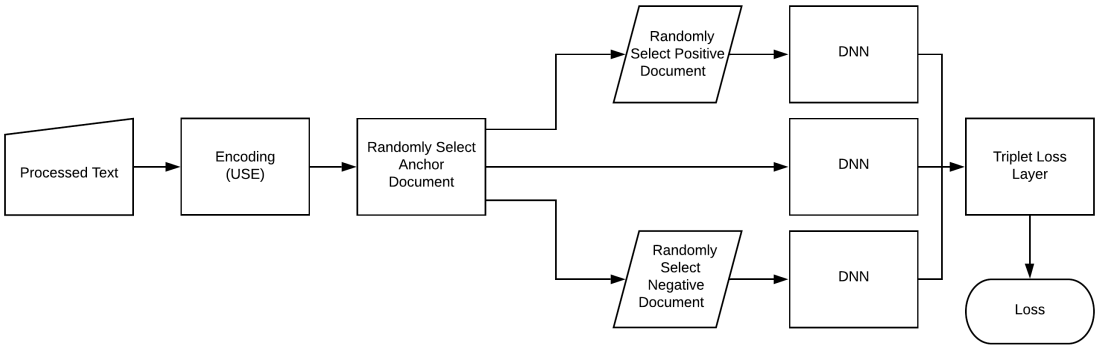


Figure 13: Flow chart of training a Siamese network with triplet loss using the document texts and embeddings.

3.6.2 Input Embedding Layer

In order to generate the input embeddings, first all the documents are pre-processed and then passed into the USE to generate the input embeddings. The output of the USE results in a matrix of size $(N, 512)$ where N represents the number of training documents, and 512 represents the embedding size produced by the USE. Comparing the feature size of this technique to a bag-of-words approach presents a large reduction in feature dimensionality. TF-IDF produced a vocabulary size of over 50,000 words after feature reduction for a 20-query model, resulting in a sparse matrix of size $(N, 50000)$. The USE produces a matrix where N has no bearing on the second matrix dimension. Whereas with TF-IDF, as the value of N increases, so does the second dimension, as shown in Figure 5. From here these input embeddings will go into the DNN, which will further reduce dimensionality of the embeddings, and generate improved document embeddings.

3.6.3 Fully Connected Deep Neural Network

The output of the embedding stage of the network produces three vectors for each triplet component in the training batch. The vectors are 512-wide for each

of the three documents–anchor, positive, negative. Each of these vectors enters an identical (defined as having the same weights and bias) fully connected DNN. The DNN layer of the overarching network is a sub-network consisting of three dense layers, each going through batch normalization as seen in Figure 14. The output width of each of these dense layers can be treated as hyperparameters that can be tuned for highest performance, see Appendix D. The training data consisted of 34,500 documents balanced across 345 search queries. A validation set was created using 500 documents across five search queries. Finally, a test set was reserved, also using 500 documents across five search queries. The validation set was used to make design decisions, such as hyperparameter tuning. The test set was left untouched until final performance reporting. Figure 14 shows the final model architecture used in this research with an output vector of size 320. The different architectures attempted and their corresponding performance is shown in Appendix D.

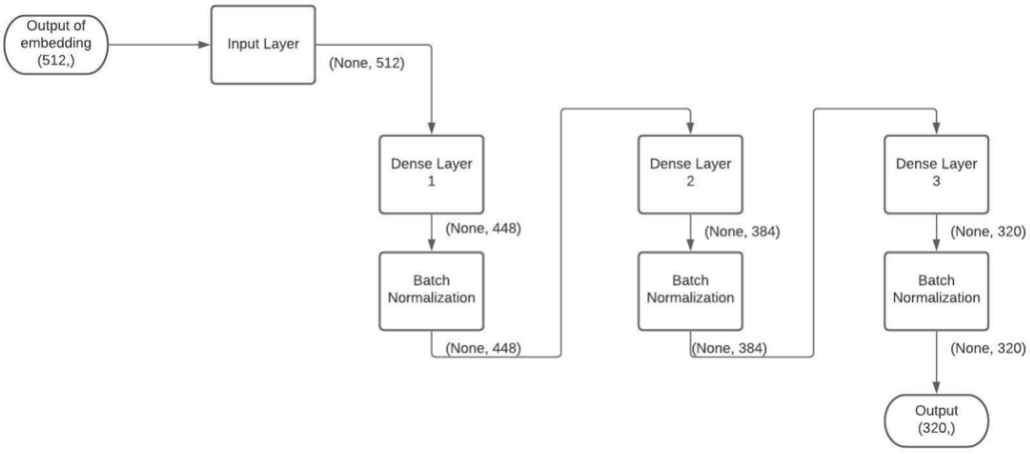


Figure 14: Sub-Network architecture of the Deep Neural Network.

3.6.4 Triplet Loss Layer

The triplet loss layer produces the loss to be minimized during training. It utilizes two distances to compute this loss. The first distance is called p-distance (p) which is defined as the distance between the anchor and the positive (eq. (10a)). The second distance is called the n-distance (n), defined as the distance between the anchor and the negative (eq. (10b)). Each of these distances can be calculated using Euclidean distance (eq. (4)), but other distance formulas could be utilized as well. Next, the n-distance (n) is subtracted from the p-distance (p), and if greater than zero, this is the loss returned, otherwise zero is returned (eq. (10c)). By minimizing the loss, the network is learning how to place anchors closer to the positives and further from the negatives in the document embedding space. A zero loss indicates that the anchor is closer to the positive than it is the negative.

$$p = \text{dist}(\text{anchor}, \text{positive}) \tag{10a}$$

$$n = \text{dist}(\text{anchor}, \text{negative}) \tag{10b}$$

$$\text{loss} = \max(p - n, 0) \tag{10c}$$

$$\text{dist}(a, b) = \text{distance between } a \text{ and } b \tag{10d}$$

3.7 Using Document Embeddings to Determine Document Relevancy

The new embeddings are then used to determine cosine similarity between two documents from the validation set. Since the validation set consists of labeled data, it is possible to derive an accuracy score of the document embeddings. Two documents are considered related if the both belong to the same QID. Thus each document in the validation set is embedded using the trained DNN layer and its cosine similarity to every other document in the set is calculated. If the cosine similarity score between

two documents is high (≥ 0.6) and the two documents belong to the same QID, then this indicates a correct relevancy prediction. Similarly, if the cosine similarity score between two documents is low (< 0.6) and the two documents did not belong to the same QID, then this indicates a correct irrelevancy prediction. This can be used to generate an accuracy score by dividing the total number of correct predictions by the total number of predictions. This process is repeated using the USE, the input to the DNN. These two accuracy scores can be compared to quantify how the Siamese network improved the original embeddings. After all tuning is complete, the process is repeated with the reserved test set to acquire a final accuracy score.

An important consideration when determining relevancy between validation/test QIDs is that no documents relevant to these QIDs were encountered during training. As such, the network learned how to determine the relationship between documents as opposed to simply learning which documents belong to what QID, as a classifier does (Section 3.4.1). This is important for the scalability of the research. The network needs to be able to embed search results it has never seen before and still be able to detect relevancy between them. The goal of the DNN is to place documents of varying length that are related to each other close in embedding space, creating clusters of relevant documents. These clusters should be distinct and separate from clusters belonging to other QIDs.

3.8 Applying the Trained DNN into a Prototype

Now that the model is capable of placing documents of varying length in an embedding space it can be applied to the original problem. A search query is sent through a search engine and the results are stored. Next, a document that is known to be relevant to the search query is provided as a context, this information is stored locally and never seen by the search engine. This context and all the search query

results are then embedded by the DNN. The cosine similarity is calculated between each search return and the provided context. The results are then re-ranked by this similarity score. There are two use cases of this application tested:

1. The user has a constrained-term search query and a relevant context file. This simulates a "find more like it" functionality, where searching the internet, a user comes across something of interest. This is used to re-rank results to find more documents similar to that which was previously found.

2. The user has knowledge of a subject but wishes to obscure their search while still achieving relevant results. This allows the user to transmit a less specific search, and re-rank the results offline using more specific keywords.

Each of these use cases can be quantified by leveraging normalized Discounted Cumulative Gain (Section 2.3.1), but the technique behind the quantification varies between use cases.

For use case 1, a search is performed using the protoype with a constrained-term search query and a relevant context file. The top 198 results are downloaded and labeled with their original rank (no context consideration) and their new rank (with context consideration). The labels are then hidden and the results are randomly shuffled. A user then goes through all 198 results and scores their relevancy on a scale of 1-5. Once all results are scored, the labels are revealed and their ranking combined with the blind user score are used to calculate a DCG score for both the prototype with context and the underlying search engine with no context. The best possible DCG score is obtained by manually ordering the user's blind scores in ascending order and calculating the DCG. Both the prototype and the search engine's DCG get divided by the best possible DCG score to produce a normalized score that is between zero and one. If the prototype yields are higher normalized DCG score than the underlying search engine then it is considered to have met objectives and

improved search relevancy. This process is conducted on 20 searches, 10 specific to military related searches and 10 inspired by recent most common searches [33].

The technique for scoring use case 2 varies slightly from use case 1. This time three scores must be considered: the search engine without hidden context, the prototype, and the search engine with hidden intent. Use case one does not consider a score for the search engine with hidden context, because in that scenario the context is an entire text document. When an entire document is typed into a search engine, the search query becomes too long and no results are provided. Use case 2 differs by using a few added keywords to hide true intentions. An example military search is shown in Table 2. The top 24 results from the search "missile" are downloaded and labeled with their rank and source (search without context). The top 24 results from the search "Intercontinental Ballistic Missile" are downloaded and labeled with their rank and source (search with context). Lastly, the top 24 results from the prototype using the search "missile" and the hidden keywords are downloaded and labeled with their rank and source (prototype). The labels are hidden and the results are scrambled. The DCG calculations are the same as it was for use case 1, except this time 3 normalized scores are obtained. The objective for use case 2 is met if the prototype score exceeds the search without context and matches the search with hidden intent. This process is repeated for 24 searches, 12 specific to military related searches and 12 inspired by recent most common searches [33]. A complete list of the use case 2 searches conducted can be found in Appendix E.

Constrained-term Search	Missile
Added Hidden Keywords	ICBM, Nuclear, Long Range, Global Strike
Hidden Intent	Intercontinental Ballistic Missile

Table 2: Example use case 2 search

3.9 Summary

Supervised multi-class classification techniques such as Logistic Regression and Multinomial Naïve Bayes were examined in this problem space and determined insufficient due to lack of scalability. Unsupervised techniques such as Latent Dirichlet Allocation and topic modeling were examined and also deemed to require too much human interaction to be a viable solution. Bag-of-Words feature extraction techniques such as TF-IDF were considered and found to be inferior to dense vector embeddings.

Text pre-processing techniques were explored in order to transform document text into quality model training features. Different embedding techniques were considered, and improving existing embeddings is attempted. A Siamese network is utilized with the triplet loss function to create document embeddings from the USE. The improved embeddings are examined to determine their effectiveness at generating embeddings for varying document lengths. The DNN is tested in the problem domain by creating document embeddings and using them to rank search results based on relevancy of a context file. Two use cases are identified for utilizing the DNN in a prototype. Normalized discounted cumulative gain is used to evaluate the performance and determine if objectives were met.

IV. Results and Analysis

4.1 Preamble

This chapter focuses on two different sets of results. The first set of results presents the comparison of the Siamese network embeddings to that of the Universal Sentence Encoder (USE) when used to cluster documents of different lengths. The first set of results seeks to answer research question 4 (Section 1.3), exploring the performance effects of applying a Siamese network to the USE. The second set of results answers research questions 1-3 (Section 1.3) by analyzing the normalized Discounted Cumulative Gain (DCG) (Section 2.3.1) of multiple searches.

4.2 Siamese Network Embedding Performance

Scoring the accuracy was performed as described in Section 3.7, utilizing the top 20, 50, and 75 rankings of the reserved test set on both the Siamese Deep Neural Network (DNN) trained on the triplet loss function and the USE. The results are shown in Figure 15. These results show that the overall scores for the DNN and the USE decline as more rankings are considered. The data indicates that USE declines more rapidly than the DNN when more rankings are considered. It also shows a relationship between performance and document length, specifically with the USE. Stable performance is observed across varying document length for the DNN, while the USE experiences significant performance drops as longer documents are considered.

Further exploration of performance stability is conducted by examining the document embeddings produced by different length inputs. As the input length increases, there are more words being used to generate the fixed-width embedding. Thus, the more words, the less each individual word contributes towards the overall embedding. Therefore, it is expected that the embeddings produced for identical documents pre-

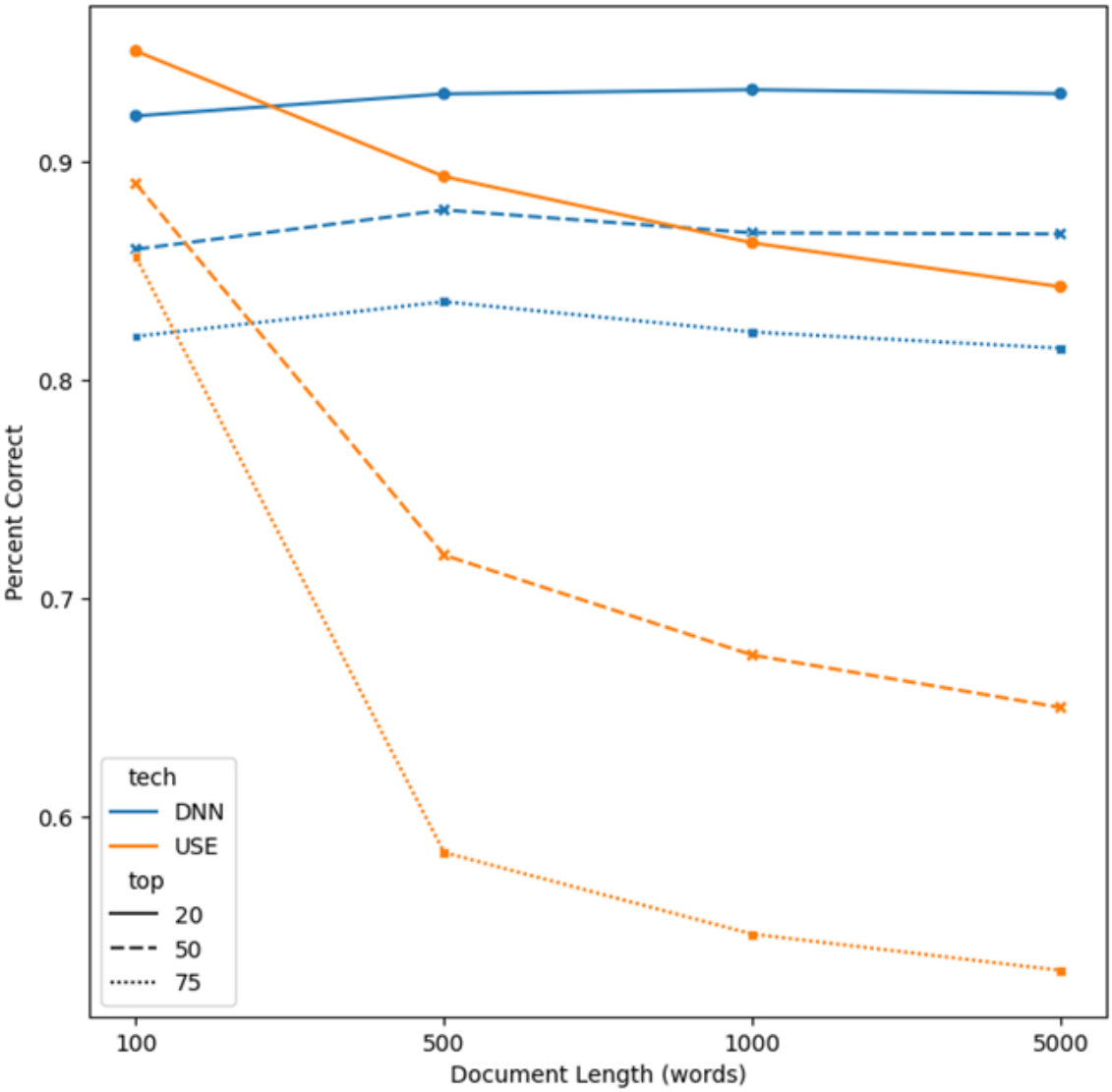


Figure 15: Accuracy scores of Siamese Network and USE across varying document lengths. The top 20, 50, and 75 rankings are considered during scoring.

processed to different lengths will vary. This variance will play a part in the stability of the performance as input length is changed, and can be quantified using cosine similarity.

The cosine similarity between embeddings generated for identical documents, using an input length of 100, 500, and 1000 words are examined. A cosine similarity score closer to one indicates an embedding that has not significantly diverged as input

length increased, whereas a cosine similarity score closer to zero indicates an embedding that has vastly diverged as input length was increased. Two comparisons are presented: 100 vs 500 word input lengths in Figure 16 and 100 vs 1000 word input lengths in Figure 17. The interesting finding presented by these histograms is that the Siamese network produces embeddings with a definitively higher cosine similarity score between identical documents with varying input length than the USE does. This metric further indicates added performance stability when the Siamese network is leveraged over the USE alone, supporting the data shown in Figure 15.

Visualization of how these embeddings work is possible through t-distributed stochastic neighbor embedding (TSNE). Figure 18 visualizes the placement of document clusters using the embeddings created by both USE and the DNN. This visual representation supports the findings that the DNN is more capable at generating

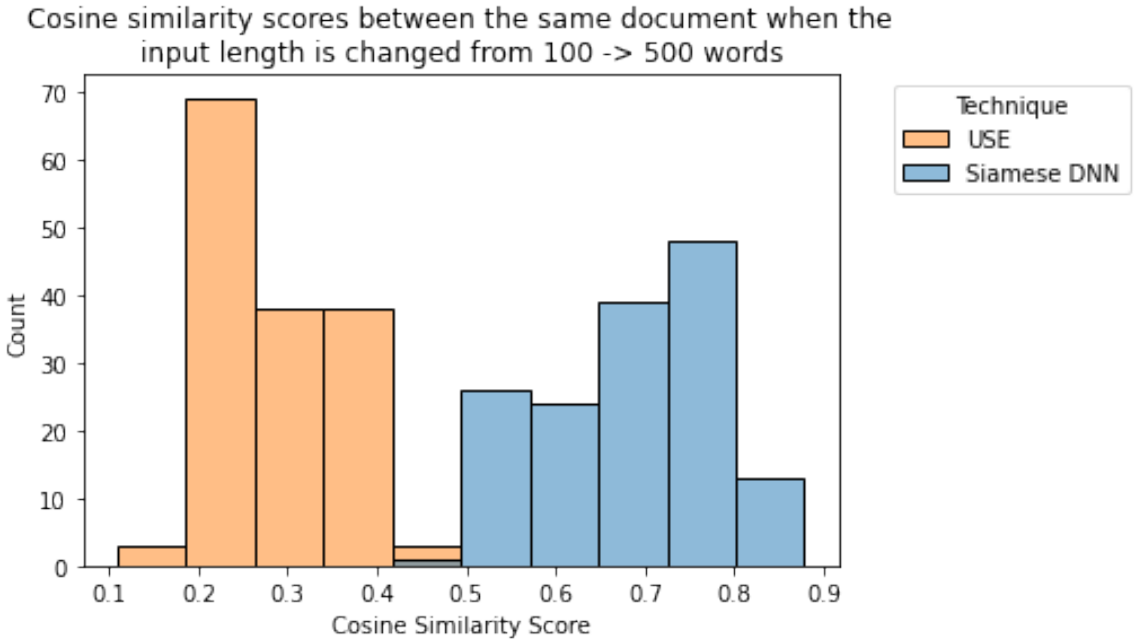


Figure 16: The cosine similarity score between the embeddings produced by USE and Siamese DNN for the same document, but with different input lengths is compared. The input lengths were 100 words and 500 words. The Siamese DNN is shown to produce higher cosine similarity scores between the identical documents as input length is increased.

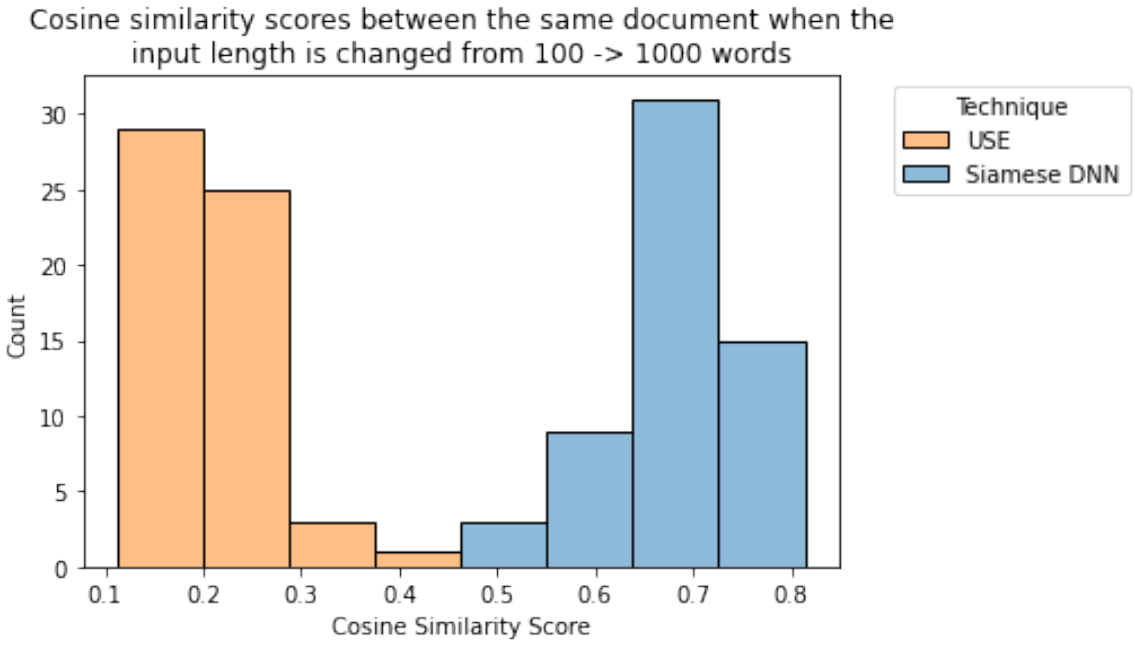
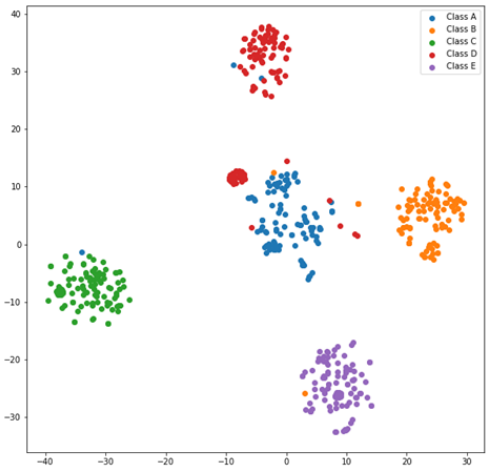


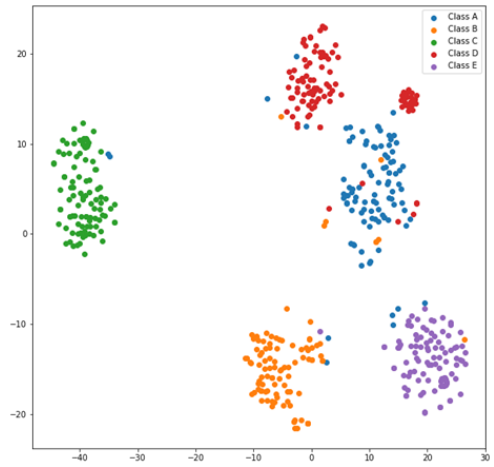
Figure 17: The cosine similarity score between the embeddings produced by USE and Siamese DNN for the same document, but with different input lengths is compared. The input lengths were 100 words and 1000 words. The Siamese DNN is shown to produce higher cosine similarity scores between the identical documents as input length is increased.

quality document embeddings. It shows that as document length increases from 100 words, Figure 18a and Figure 18b, to 1000 words, Figure 18c and Figure 18d, the USE clusters begin to space out, while the DNN embeddings manage to stay more tightly packed. [8] Intra-cluster distance is defined as the distance between elements within a cluster and inter-cluster distance is defined as the distance between clusters. In the context of this experiment, a good intra-cluster distance means that similar documents are close to each other and a good inter-cluster documents means that dissimilar documents are spaced out. An example this can be seen by considering the green points in Figure 18a, these points represent documents belonging to class C. Their intra-cluster distance is defined by the distance between green points and their inter-cluster distance is defined as their distance between green points and purple, blue, orange, or red points. Good embeddings are defined as those which

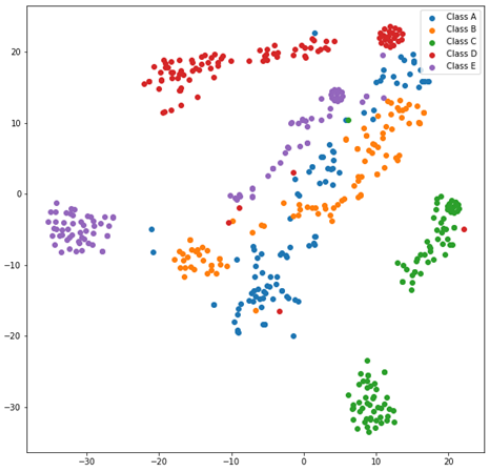
place green points close to each other and far away from purple, blue, orange, or red points. Figure 18e and Figure 18f show this trend to continue—USE embeddings intra-cluster distances increasing and inter-cluster distances decreasing with length 5000, and the DNN maintaining tight intra-cluster distances and distinct inter-cluster distances irrespective of document length. This means that the DNN is producing superior embeddings, specifically on longer document lengths, because the inter-cluster distance and intra-cluster distance indicate the ability to place relevant documents close to each other and irrelevant documents further away from each other. These results confirm those seen in Figure 15 and support the claim that the DNN generates embeddings more suited to detecting document relevancy via cosine similarity.



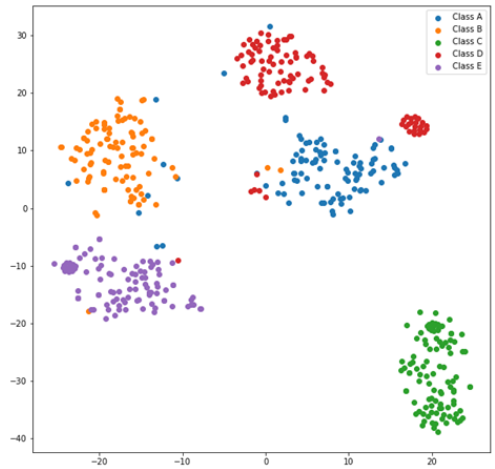
(a) USE—100 word documents



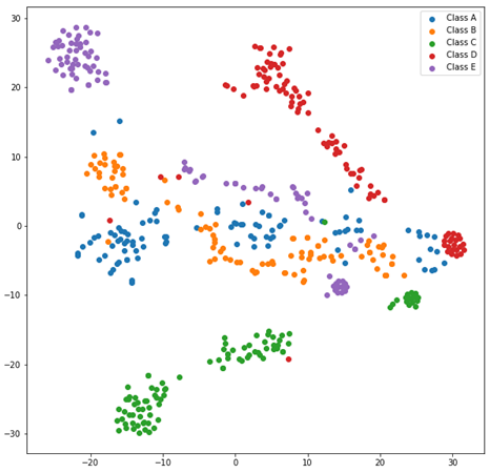
(b) DNN—100 word documents



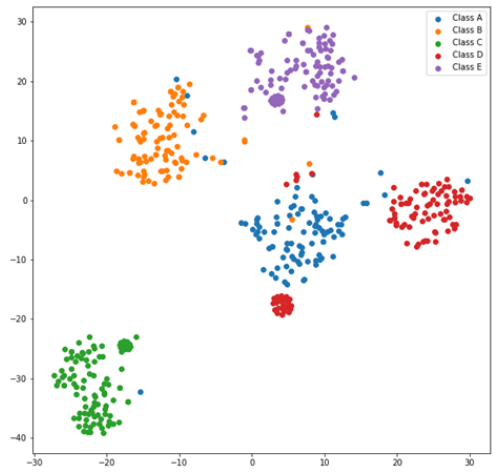
(c) USE—1000 word documents



(d) DNN—1000 word documents



(e) USE—5000 word documents



(f) DNN—5000 word documents

Figure 18: USE vs DNN TSNE to display the intra-cluster and inter-cluster distances on varying document lengths.

4.3 Performance and Utility Analysis of Search Tool

Now that the network has the ability to generate quality document embeddings, it is implanted in a prototype and examined. The prototype utilizes a commercial search engine, specifically Google US Servers, but it has the ability to be adapted to work with any search engine, and even databases or hierarchical file systems. It takes three parameters: the search query, which will be transmitted to the search engine, the context, which will be hidden from the search engine, and the depth, the number of search pages to examine. The context and each search return across the depth is embedded using the trained DNN, and then the cosine similarity is calculated. The higher the cosine similarity score is, the more relevant the result is likely to be, thus the results are re-ranked in ascending order using this metric.

In Section 3.8, two use cases were identified. The first use case sought to answer research questions 1-3 (Section 1.3) by implementing the "find more like it" functionality. Figure 19 shows an example of use case 1 where the constrained-term search query is "Viper" and the context is a Wikipedia file discussing the F-16 fighter jet. Qualitatively, it can be seen that the prototype was able to provide relevant results. Some relevant results that can be observed are in position 1, 3, 5, and 6 in the newly re-ranked results and were originally ranked in 103, 177, 84, and 101 respectively. A user would have had to sift through hundreds of search returns to find these results without using the prototype.

The effectiveness can be quantified following the methodology outlined in Section 3.8. In total, 20 searches are conducted, 10 with respect to military and 10 inspired by common internet searches [33]. The scores are collected and displayed in Figure 20. In this graphic, the normalized DCG score is compared between the prototype's search with a context file and the search without the context file (GWO). It quantitatively shows the prototype outperforming the search engine without context.

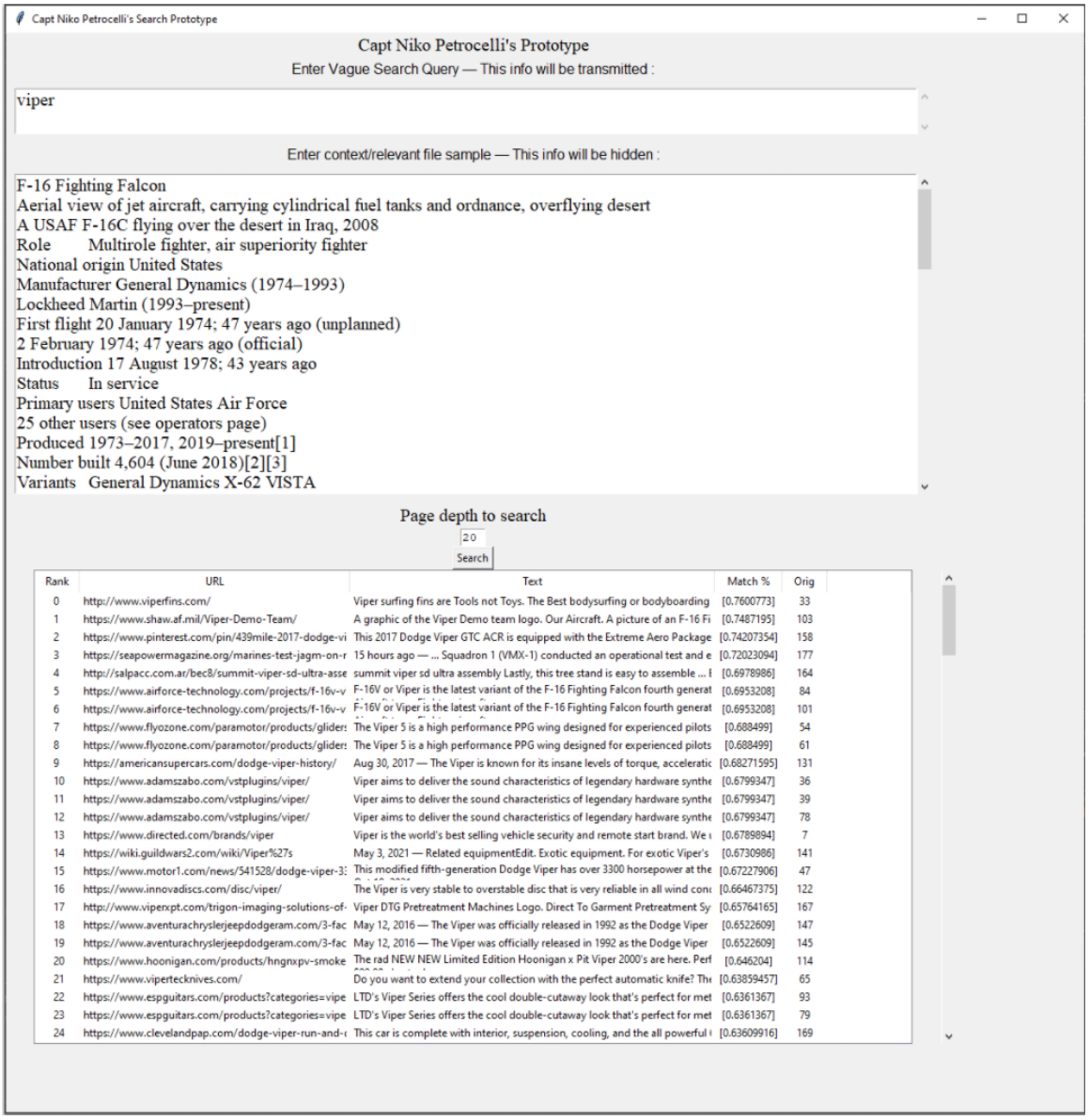


Figure 19: Prototype Use Case 1 Search Example

The second use case identified in Section 3.8 seeks to answer research question 1 (Section 1.3) by implementing the "hidden intent" functionality. Figure 21 shows an example of use case 2 where the constrained-term search query is "intelligence" and the hidden keywords derive from a hidden intent of "artificial intelligence". Similar to the behavior shown with use case 1 in Figure 19, relevant results previously ranked low are re-ranked much higher. The top 3 results seen in Figure 21 are particularly

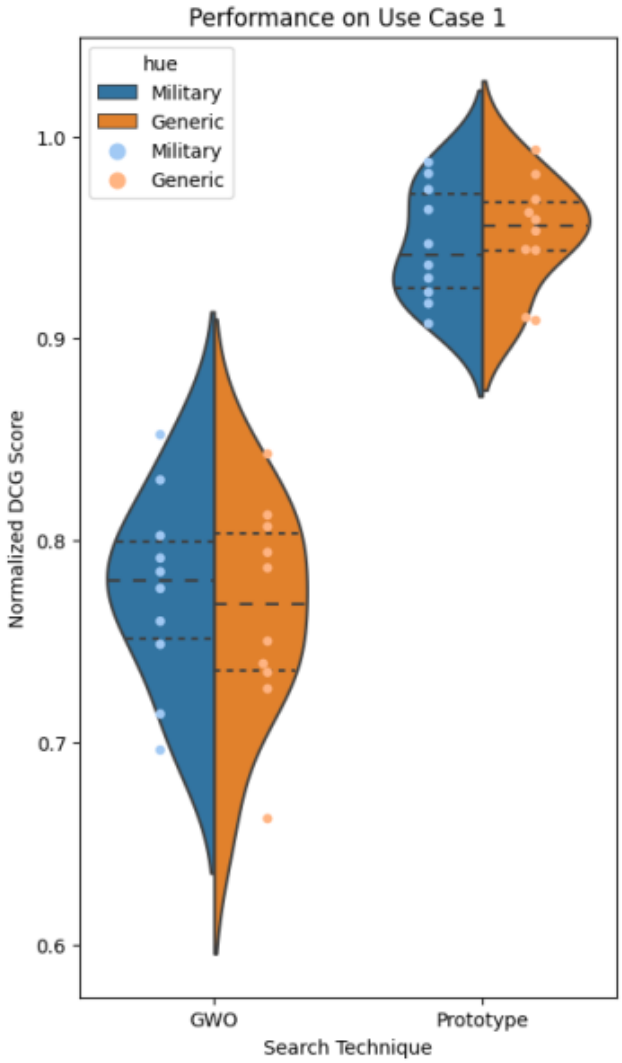


Figure 20: Performance of the "find more like it" functionality of use case 1. This figure compares the normalized DCG score of the search engine without context (GWO) to the prototype. It shows the distribution of the scores across military and generic search queries. The prototype's distribution shows there is a high probability that the prototype will outperform the search engine without context on a given search query.

relevant to the hidden intent and were originally ranked 79, 62, and 147 respectively. Thus, as in use case 1, it appears use case 2 qualitatively improves results over a constrained-term search without the hidden intent.

Use case 2 can be quantitatively scored by using DCG. Figure 22 shows the normalized DCG scores obtained utilizing the methodology as described in Section 3.8.

The figure displays the normalized DCG scores of 24 searches, 12 with respect to military related queries and 12 inspired by [33] common internet searches. It compares the performance of the search engine with the constrained-term search query (GWO) and the performance of the search engine with the full search query (GW) to the prototype with the constrained-term search query with hidden keywords.

The primary objective for the prototype is to exceed the performance of GWO. The secondary objective is to match the performance of GW. Analyzing the DCG scores from Figure 22 quantitatively shows that the prototype achieves the main objective by exceeding the performance of GWO—defined as having a higher overall DCG score. Even the lowest DCG scores shown for the prototype exceed the highest DCG scores for GWO. The secondary objective is not fully achieved. Figure 22 shows that the prototype is not always able to match GW performance. The higher variability observed for the prototype makes it insufficient to conclude that the prototype performs as well as GW.

The prototype offers a substantial privacy improvement over GW, so while it may not always match the performance of GW, it may still be a better option for a user. The performance reported for GW is generated by a search that relays the entirety of the hidden intent, and so the search provider knows exactly what the user is searching for. The prototype only relays the constrained-term search query, and provides a locally stored list of returns. GW and GWO search would both allow the search engine to track the user's traffic, to see which links they click on and how long they loiter. The prototype blocks this activity as well, masking not only the true intent of the search, but the user's traffic and their loiter time. The complete list of searches that were used to generate this plot can be seen in Appendix E.

There were interesting findings that arose from evaluating the performance of the prototpye in both use cases. Firstly, both Figure 20 and Figure 22 show wide

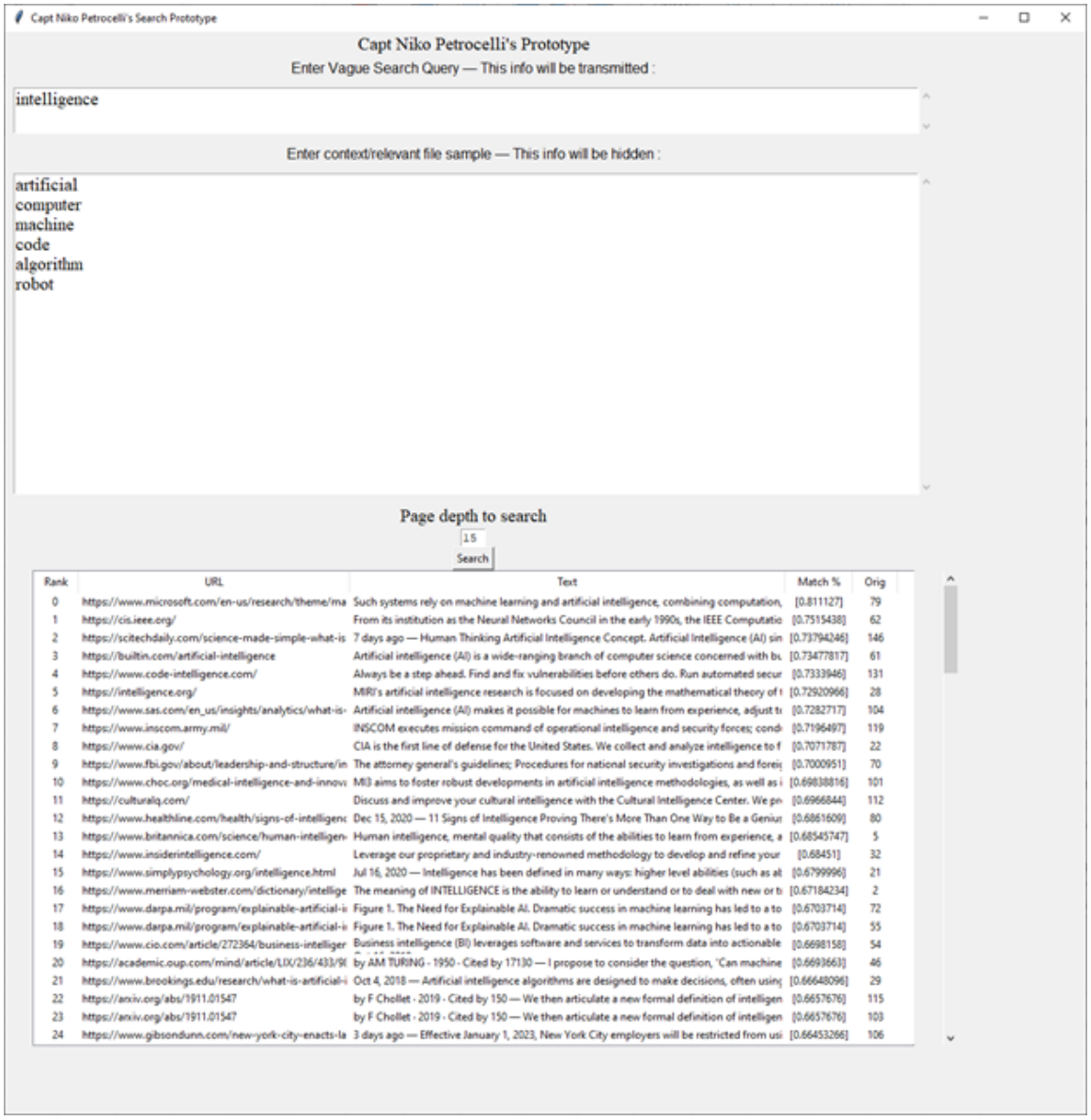


Figure 21: Prototype Use Case 2 Search Example

variability in performance for GWO. Analysis of the searches that generate the data, presented in Appendix E, yield some insight on what may be contributing to this. Often the constrained-term query has multiple use cases, for example the search "Fighter Jet". If that query is presented to a search engine when the intent was F-15 it performs worse than if that same search were entered with an intent of F-35. With GWO, context is not considered and the constrained-term search is highly dependent

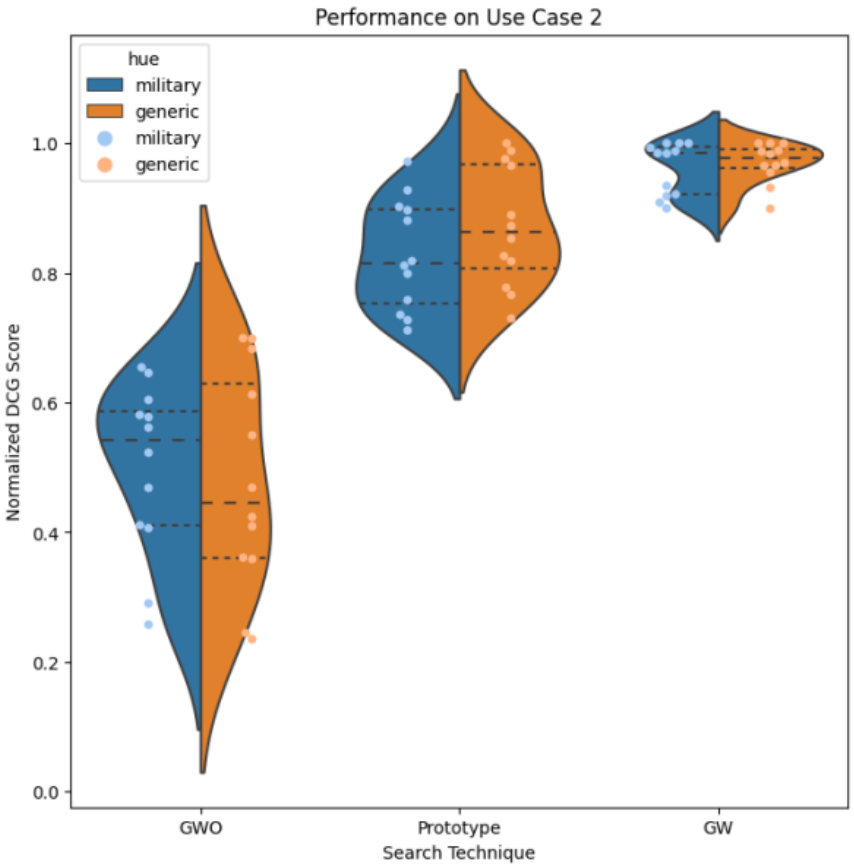


Figure 22: Performance of the "hidden intent" functionality of use case 2. This figure compares the performance distribution of the search engine without context (GWO), the prototype, and the search engine with context (GW) across a series of military and generic search queries. The prototype is shown to provide performance better than GWO, and slightly less performance than GW.

on hot searches at the moment the query is entered. A hot search is a popular search query that has gained popularity from a temporarily higher frequency of user requests for a specific search as compared to alternative searches. If F-35 is more popular or trending on the search engine at the time this experiment is conducted, it makes sense that GWO score for this search with a hidden intent of F-35 is relatively high while that same search with a hidden intent of F-15 is low. There is low set overlap between identical searches with differing intent: if GWO performs reasonably well for one context, it must perform worse for the other. There is some set overlap because

the possibility exists that a search result could equally discuss F-15 and F-35, making that result relevant to both intents. This is more than likely the primary driver to the variability of the GWO scores shown in both of these figures.

Another interesting finding arose from further consideration of a specific search found in Appendix E. The constrained-term search query of "Vaccine" with the hidden intent of COVID has excellent GWO performance. The cause of this is the popularity of COVID and COVID vaccinations during the time at which these experiments are conducted. In order to find any other context of the constrained-term search "Vaccine" a user would have to search an extensive amount of results because the search tool depends on the underlying search engine for results. If the underlying search engine only provides results of a singular context, such as COVID, then the prototype is unable to re-rank relevant results to a different context, such as Polio, because it has no relevant results to choose from. This uncovers a limitation of the tool, as there is a computational limit on how many pages can be searched. While it is possible that given hundreds or thousands of pages to consider the prototype could find relevancy to alternate contexts, it is not always feasible to scrape that many pages. This same limitation can also be considered for use case 1, if insufficient page depth is examined, it is possible there are no relevant results from the underlying search engine for the prototype to choose.

4.4 Summary

The DNN embeddings were compared to USE embeddings to quantify the difference in performance, specifically when used for document embeddding in Section 4.2. The DNN produced document embeddings superior to those produced by the USE because the DNN had stable performance across a variety of document lengths, whereas the USE performance declined with longer documents. Next, Section 4.3 conducts

qualitative and quantitative evaluations of the usefulness and the performance of using the DNN document embeddings in a search tool. The search tool is shown to provide promising performance in multiple use cases, and has the added benefit of increased user privacy. The results presented in this chapter provide sufficient insight to answer the research questions initially posed in Chapter I.

V. Conclusions

This chapter seeks to answer each research question from Section 1.3 based on the data presented in Chapter IV. Section 5.5 covers generalized conclusions that arose during the research. Lastly, Section 5.6 presents potential future work to be conducted in this domain, and completes the chapter.

5.1 How can major search engine results be reorganized in a way that is useful to the user while limiting data transmission?

Section 4.2 shows that a Siamese Deep Neural Network (DNN) is capable of generating quality document embeddings independent of document length. The DNN is able to determine similarities between two documents, even if the documents are of a topic not encountered during training. This allows for a user to provide a context file and the DNN can determine similarity between this context and other files. The ability of the DNN to produce quality document embeddings irrespective of document length is important, because the context file could be any length, as could the documents with which it is being compared. This technique is applied to a search prototype in Section 4.3. The context file used is stored locally and the results are provided by an underlying search engine. This allows the prototypes to re-rank the major search engine's returns without transmitting the information provided as context. Leveraging these techniques allows major search engine results to be reorganized, in a useful way, while also preventing unnecessary data transmission.

5.2 How can a technique be scaled to handle new search queries without requiring retraining?

An important research goal is to provide a solution that is capable of handling newly encountered search queries without retraining a model. Supervised machine learning using Naïve Bayes/logistic regression (Section 3.4.1) and unsupervised learning using latent Dirichlet allocation (Section 3.4.2) displayed classification solutions that cannot be scaled. The application of a Siamese network to produce document embeddings proved to be a promising solution that is capable of scaling. This is because the Siamese network works on similarity detection instead of classification. This can be adapted from existing machine learning solution such as facial recognition software.

Facial recognition software is able to detect if two images are of the same person, even if it has never encountered that person. One of the techniques used to do this is one-shot learning using a Siamese Deep Neural Network and a triplet loss function. This technique can be modified to work with Natural Language Processing (NLP). This research focuses on developing a way to implement these algorithms in a new domain. Instead of detecting if two photos are of the same person, it detects the relevancy between two documents it has never encountered. These techniques allow the machine learning prototype developed in Section 4.3 to scale to new search queries without the need of retraining. The data the model was trained on was from search queries and documents provided by the TREC 2020 data set. Whereas the search queries presented to the tool during testing were hand crafted based on military interests and the most common internet searches, none of which were used during training.

5.3 What is the performance of the reorganized results when compared to a major search engine?

Section 4.3 presents the performance of two use cases identified in Section 3.8. Use case one simulates the case where a user has a context file and wants to find more results like it. The performance for this use case is shown to provide better overall results than if the user just input a search query without providing this context.

For use case two, two comparisons need to be made. In this use case, it is simulated that the user is obscuring their search, by generating a constrained-term search query. The user then provides a context that specifies what their intent is, while hiding it from the underlying search engine. The performance of the prototype is compared to the constrained-term search query without context and the search where the hidden intent is not obscured and is fully transmitted to the search engine. The performance for use case two is shown to be a definitive improvement over the case where the search engine is just provided the constrained-term search. It does not always match performance of the search without obscurity. This becomes a tradespace for the user between privacy and performance. The prototype is shown to perform well while providing added privacy.

The prototype presented in Section 4.3 has benefits over major search engines. It can be adapted to be used with any search engine, whether it is online or offline. It can even work with file structure or database searches. The prototype also prevents the ability for search engines to follow user traffic by preventing the ability to detect where the user clicks. It also prevents tracking of loiter time, which tells the search engines how long a user spent on a specific site.

5.4 What is the effect of applying a scalable machine learning technique in this problem domain?

Section 4.2 presents the results required to answer this question. Figure 15 shows the accuracy score of the Siamese DNN embeddings versus the accuracy score of an existing embedding technique, the Universal Sentence Encoder (USE). It shows that the Siamese DNN is able to not only provide superior accuracy on documents of longer length, but that it is able to achieve stable performance across multiple document lengths. The USE shows substantially accuracy drops as document length is increased. Additionally, the DNN shows superior performance when more rankings are considered. The performance as the top 50, and 75 ranked documents are considered is definitively higher than the USE, specifically as longer documents are considered.

Figure 18 shows the inter-cluster and intra-cluster distances of both the USE and the Siamese DNN. Five classes of documents are considered and the cluster distances are compared side-by-side across varying document lengths. The Siamese DNN shows superior intra-cluster and inter-cluster distance on documents of longer lengths. Thus the data supports that applying a Siamese DNN with triplet loss function to the existing embedding technique of USE provides superior embeddings for documents, specifically when used to embed documents of varying lengths. While USE may perform better when the documents are short, the Siamese DNN displayed stable performance across document length. Therefore, in the case where document length is unknown, may vary between documents, or is known to be lengthy ($\geq$ 100 words), the Siamese DNN provides superior embeddings.

5.5 General Conclusions

The goal of this research was to develop a model that could be integrated into a search tool. This search tool would be used to provide improved search results while providing added user privacy. A variety of techniques were explored to accomplish this, and the best technique discovered was by adapting facial recognition algorithms to work with text. While difficult to quantify usefulness, initial analysis of the search tool by users in the intelligence community show the tool provided an increase in efficiency over their existing techniques. This research resulted in a new search tool that acts as a proof of concept showing that further research in this area, and improvement of this tool would be a valuable contribution.

5.6 Future Work

This research sought to improve search relevancy while providing improved privacy for a user. While a proof of concept was created in the form of a prototype, there is still much room for improvement. There are a multitude of areas within this research that can benefit from further exploration including: model training, anchor selection, performance comparisons, human studies, model application, and computation/performance tradeoffs. This section will explore some of these areas and how they can be expanded on.

One potential area for improvement lies in model training. At the time of this research, the document anchor, positive, and negative, collectively referred to as a triplet, are selected randomly. There exist techniques that could be leveraged to ensure the best possible triplets are chosen. The difficulty of this is glimpsed from Equation (11d), where the number of possibilities is a combinatoric in the number of 3-tuples of each class, and grows even further if more training data is added. One technique for choosing better triplets is using batch hard triplet loss [34].

Another area of future work would be to explore the possibility of using queries as anchors in the triplet loss instead of documents. In Table 10, model 10 explored this concept. When this was conducted the number of possible anchors dropped from 34500 to 345. This is because there were 100 documents per query and the model was trained on 345 queries. If a query is used as an anchor there could only be 345 possible anchors. This is a limitation imposed by the amount of training data used, specifically due to time constraints. If more data were processed and made available for model training, then more queries would be available as possible anchors. It would be meaningful to compare the performance of a Siamese network trained with document anchors vs a Siamese network trained with query anchors on a larger scale.

$$\text{number of possible anchors} = \binom{34500}{1} = 34500 \tag{11a}$$

$$\text{number of possible positives} = \binom{99}{1} = 99 \tag{11b}$$

$$\text{number of possible negatives} = \binom{34400}{1} = 34400 \tag{11c}$$

$$\text{number of possible triplets} = 34500 \cdot 99 \cdot 34400 = 117493200000 \tag{11d}$$

An experiment that compares Doc2vec (Section 2.5.2) to the embeddings created from the Siamese DNN would be meaningful future work. The experiment would use the same methodology for ranking relevancy as seen in Chapter III, except it would do so with two prototypes, one using the Siamese DNN embeddings and the other using Doc2vec embeddings. Results similar to those seen in Section 4.3 would be produced where DCG scores are used to evaluate the performance of each prototype. The prototypes could then be directly compared to each other, to determine which document embeddings yield higher performance. Another comparison that can be conducted is by using alternate input embeddings (such as doc2vec, or BERT) instead

of USE to train the Siamese DNN and then comparing performance of the different Siamese networks produced.

A user study could further quantify the usefulness of the search tool. People in various communities and backgrounds could be given a blind test to rank perceived utility of the developed search tool versus a plain search. This blind test would hide where the search results are coming from and the user would rate two sets of results based on which they perceive as superior, one being the results provided by the prototype and the other being the set of results from the search engine. This data can be aggregated from all the people who participated in the test and the user score's can be averaged and used to directly compare perceived utility of the search prototype to that of a plain search engine.

There exists future work to be conducted in the application of the model. The search tool currently only works with online search engines, but has the potential to be adapted to work with database searches, or hierarchical computer file structures. The performance in this new domain can be quantified to further explore utility.

Another search tool improvement lies in how the cosine similarity is calculated. There is tradeoff between search time and relevancy directly related to the amount of text processed during calculations. Processing more text acquired from the search engine results may yield improved re-ranking, but will cost more computationally—resulting in longer search times. Adding a feature that allows a user to tune between optimizing search time, optimizing search relevancy, or somewhere in between could be useful. Lastly, experimentation could be conducted on integrating the tool into a portfolio based solution, where context is provided by a locally stored user profile. Tracking user clicks, loiter time, and feedback from users may lead to improved ranking and the evolution of a user's profile.

Lastly for future work, a feature could be added that expands on the "find more

like it" functionality presented in use case one of the prototype (Section 3.8). A user would click a button when they find a search result considered high quality. Next, the important information from that high quality result is automatically extracted and used as a context. This context can be utilized in subsequent searches to continue to provide more results that appeal to the user. Similarly, if the user does not like a link, the extracted context could be used to preclude results similar to the disliked result in future searches. This preclusion can be implemented using the same methodology presented in Section 3.8 except instead of considering results with high cosine similarity scores it would seek to find results with low cosine similarity scores. This low score would represent documents that are far away from the context. A conjunction of this could be implemented where the prototype searches for results that are close to one context and far away from another.

Appendix A. Additional Results from Supervised Classification Techniques

Figure 23 shows the logic behind how each model tested was optimized for the following supervised techniques: Random Forest Classifier (RFC), Multinomial Naïve Bayes (NB), Support Vector Classifier (SVC), and Logistic Regression (LR). Table 3 shows the parameters searched across each model type. Table 4 presents the training and validation scores of the different supervised classifiers, as well as the reduced feature count from the TF-IDF optimization. Figure 24 displays the confusion matrices generated on the validation set for each supervised technique. In Figure 25 the validation set confusion matrix for the multistage classifier, LR being fed forward into NB. Lastly, Figure 26 presents the confusion matrix results on the validation set for multiple ensemble types.

Model	Parameters
RFC	TF-IDF max features, number of estimators, RFC max features, max depth
NB	TF-IDF max features, alpha (smoothing parameter)
SVC	TF-IDF max features, kernel type, max iterations
LR	TF-IDF max features, solve type, penalty type, max iteration

Table 3: Supervised model hyperparameters examined for each model type

	RFC	NB	SVC	LR
f-score (train)	0.924	0.879	0.927	0.938
f-score (validation)	0.921	0.903	0.933	0.939
Num Features	2515	7128	4821	4821

Table 4: Micro averaged f-scores and feature counts for supervised classification models

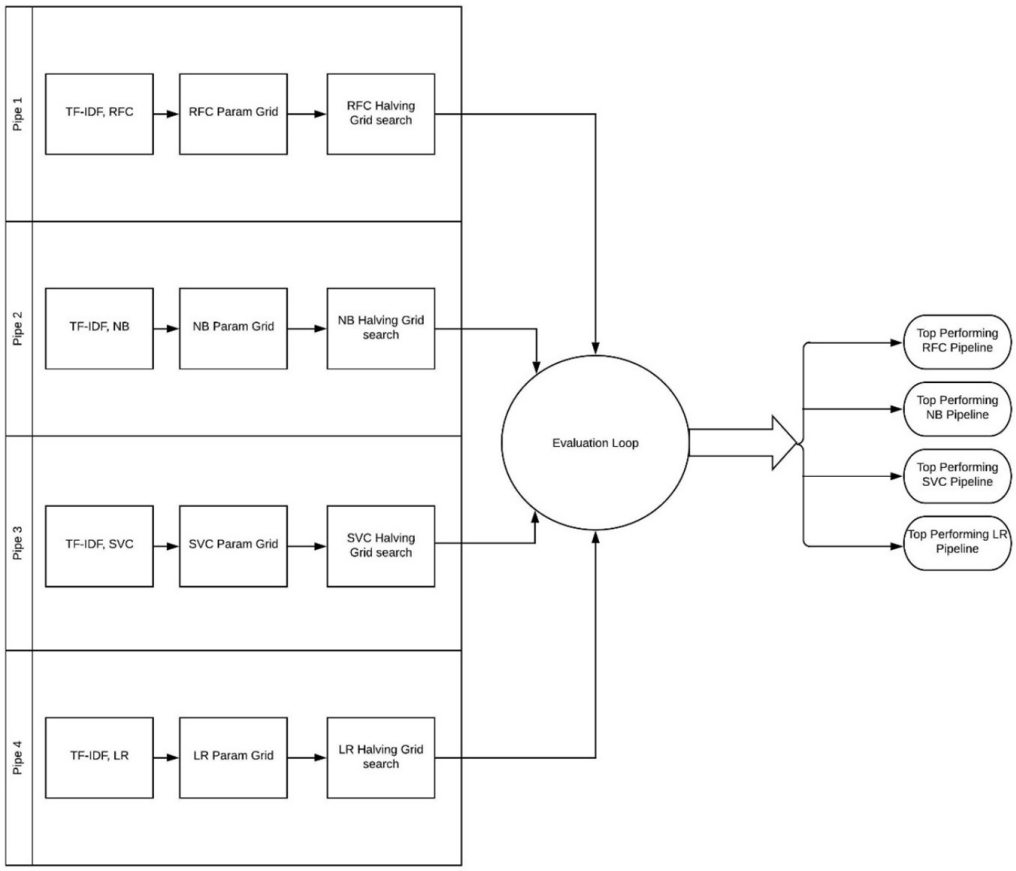


Figure 23: Pipeline logic for finding the optimal supervised classification model parameters

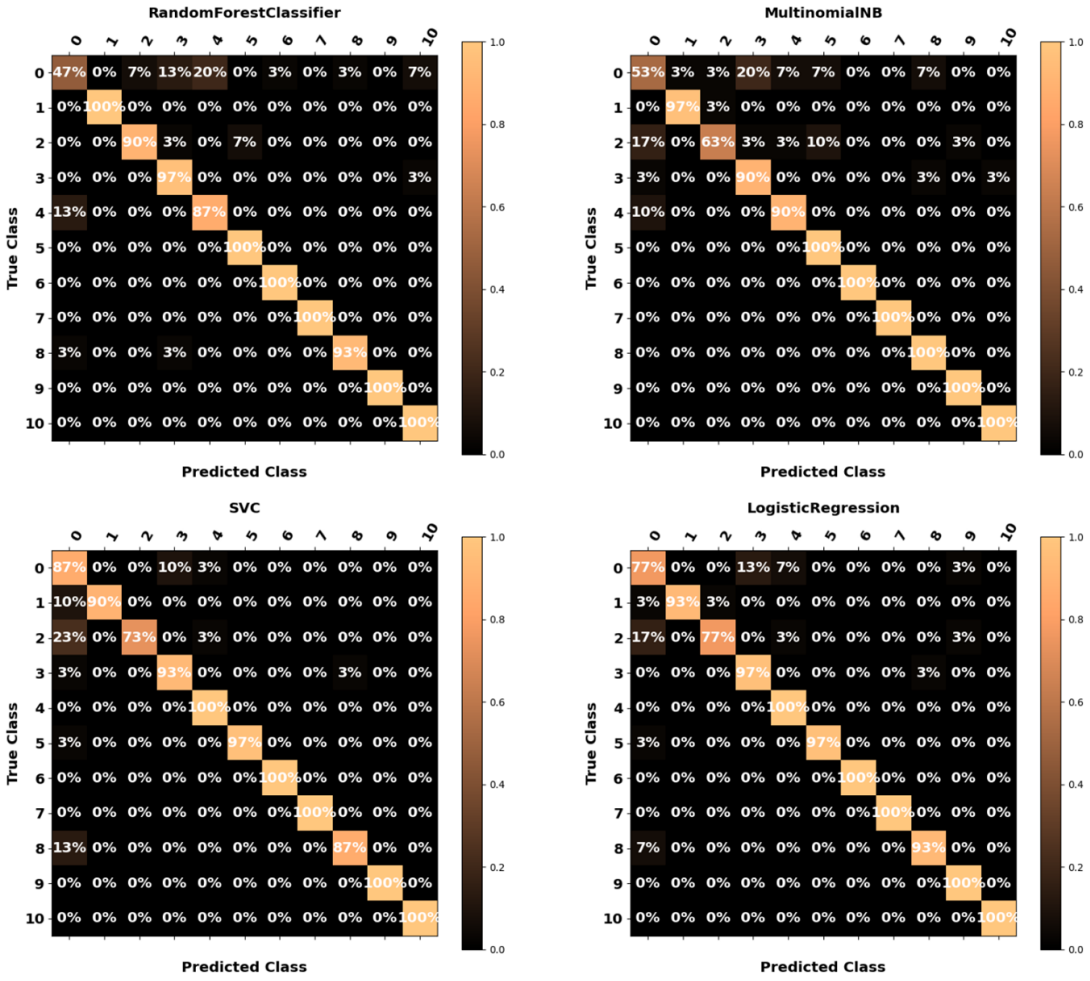


Figure 24: Confusion matrices for the supervised techniques on the validation set

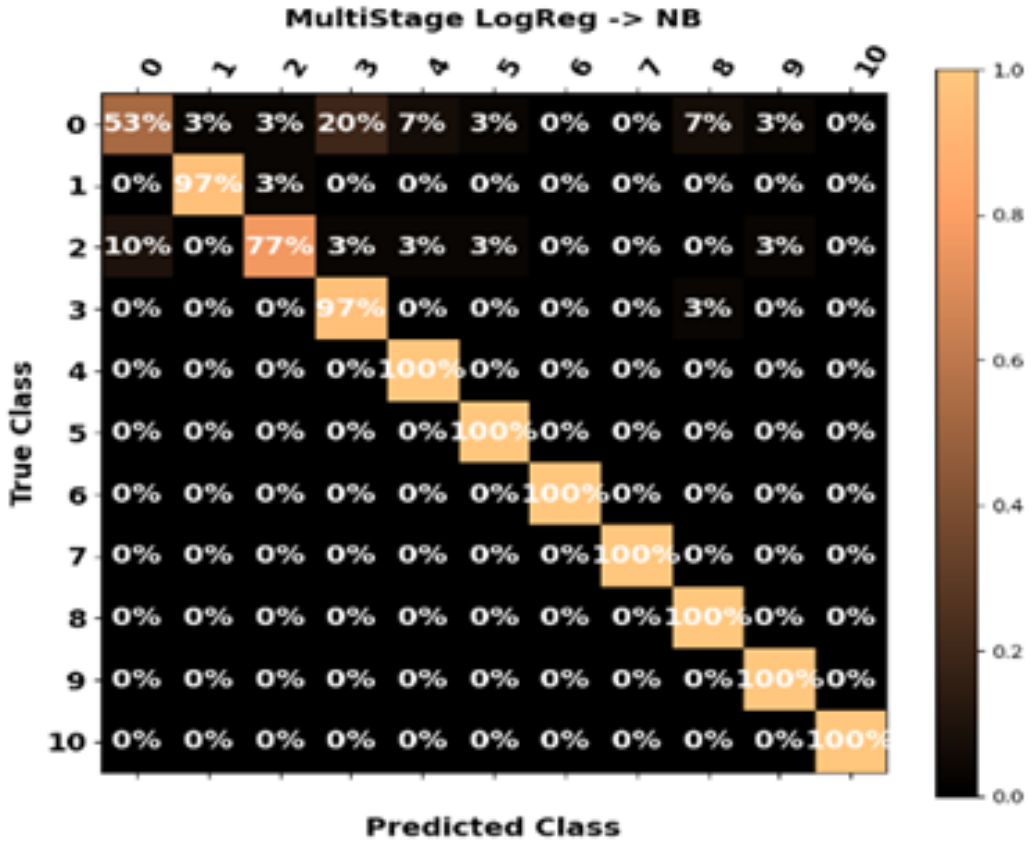


Figure 25: Confusion matrices for the multistage classifier on the validation set

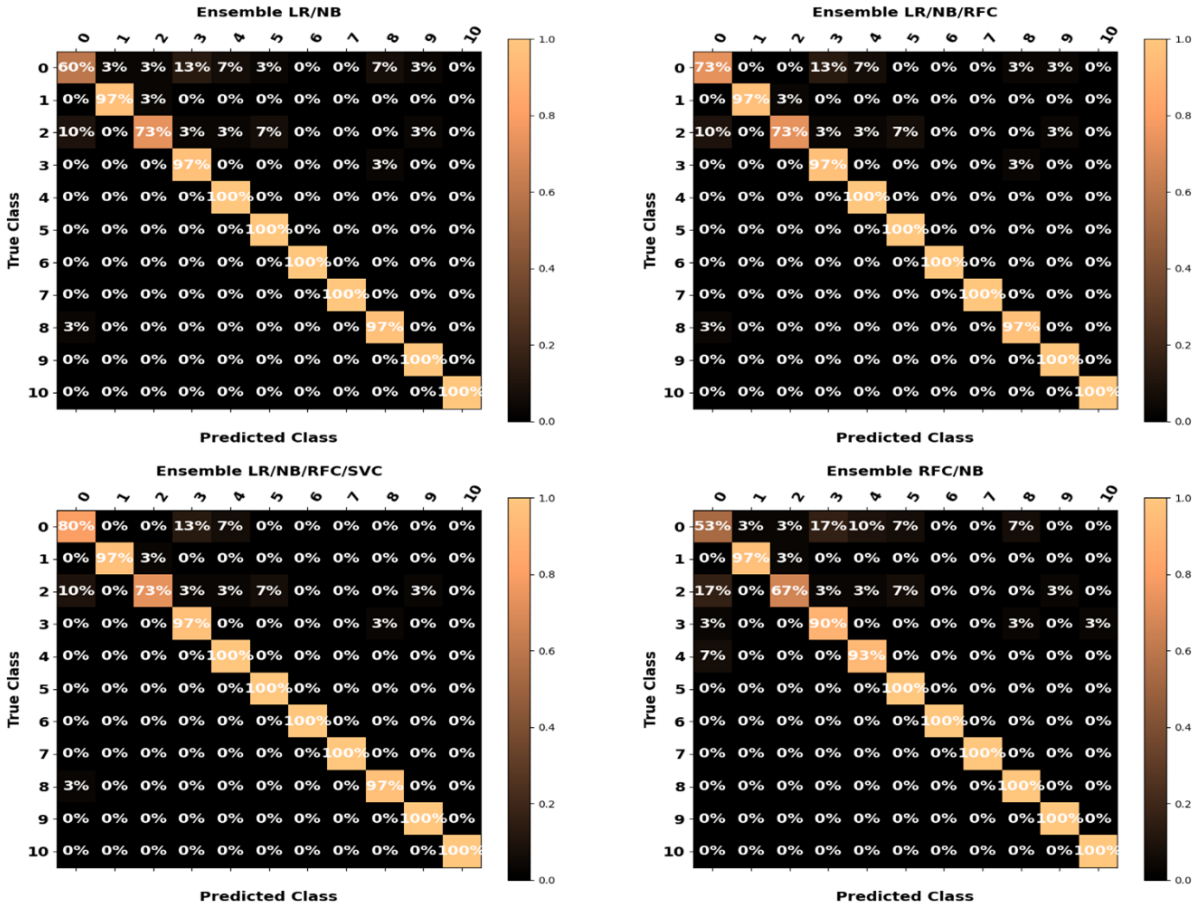


Figure 26: Confusion matrices for the supervised ensemble techniques on the validation set

Appendix B. Raw Training Data Handling

The raw data that was used to train the neural network consists of three files. The first is shown in Table 5 which ties each QID to a DocID. The next file type ties each QID to the search queries text, shown in Table 6. The final file type ties each DocID to the document text, an example can be seen in Table 7. After this raw data is processed, it will form the data structure seen in Figure 27. There are three columns, DocID refers to the document's identification number, values contains the entire document text—not yet cleaned, and the y column contains a unique class identifier that has a one-to-one mapping with the QID list. Next, each document goes through the process discussed in Section 2.1. The end result can be seen in Table 8.

1185869 Q0 D59221 1 -4.80433 IndriQueryLikelihood
1185869 Q0 D59220 2 -4.92127 IndriQueryLikelihood
1185869 Q0 D2192591 3 -5.05215 IndriQueryLikelihood
.....
1162849 Q0 D669890 98 -6.24166 IndriQueryLikelihood
1162849 Q0 D782042 99 -6.24259 IndriQueryLikelihood
1162849 Q0 D1537983 100 -6.24275 IndriQueryLikelihood

Table 5: QID tied to DocID

1185869)what was the immediate impact of the success of the manhattan project?
1185868 justice is designed to repair the harm to victim, the community and the offender caused by the offender criminal act. question 19 options:
1183785 elegxo meaning
645590 what does physical medicine do

Table 6: QID tied to query text

D3513209 https://answers.yahoo.com/question/index?qid=20100208181857AABxq2z What is the typical result when a diploid cell undergoes meiosis? (8.14) A) two diploid cells B) two haploid? Science & Mathematics Biology What is the typical result when a diploid cell undergoes meiosis? (8.14) A) two diploid cells B) two haploid? What is the typical result when a diploid cell undergoes meiosis? (8.14) A) two diploid cells B) two haploid cells C) four diploid cells D) four haploid cells E) two haploid cells and two diploid cells Follow 1 answer Answers Best Answer: Meiosis is 2 cell divisions. The first reduces the chromosomes number in half. The second is equal (mitosis) division. So for humans: A diploid cell has 46 chromosomes. Before starting meiosis, the chromosomes replicate (becoming double). So 46 double chromosomes separate into 2 cells of 23 double chromosomes each. Since each cell contains 23 chromosomes, they are haploid. But the chromosomes are double, so each cell divides, producing 4 cells, each with 23 single chromosomes. The answer is D 4 haploid cells. Roland · 8 years ago1 0 Comment Maybe you would like to learn more about one of these? Consolidate Your Student Loans Explore Digital Home Security Want to build your own website? Look For an Accident Attorney ”

Table 7: DocID tied to document text

DocID	Values	y
D216392	D216392\thttp://study.com/articles/What_Does_a_Sports_Medicine_Doctor_Do.html\tWhat Does a Sports Medicine Doctor Do?\t"Glossary of Career Education Programs / M...	4.00000
D2306673	D2306673\thttps://tatring.com/getting-tattooed/Tattoo-Infection\tInfected Tattoo: Signs and Treatments\tTat Ring » Getting Tattooed » Problems Infected Tattoo: Signs and...	9.00000
D621795	D621795\thttp://www.definitions.net/definition/weft+knitting\tDefinitions &Translations\t"Vocabulary What does weft knitting mean? Definitions for weft knitting Here are ...	6.00000
D1814826	D1814826\thttp://www.landwatch.com/Colorado_land_for_sale/Clark\t.\tCLARK, ROUTT COUNTY, NORTHWEST, COLORADO LAND FOR SALE: 1 - 15 of 94 listings Save Save ...	5.00000
D812398	D812398\thttps://authoritytattoo.com/tattoo-and-tanning-in-the-sun/\tTattoos In The Sun â What NOT To Do When Tanning\t2Tattoos In The Sun – What NOT To Do When...	9.00000
D566077	D566077\thttp://drdnaturopath.com/what-can-lower-blood-pressure-within-minutes/\tWhat can lower blood pressure within minutes?\tWhat can lower blood pressure wit...	8.00000
D705162	D705162\thttp://www.barbercosmo.ca.gov/forms_pubs/publications/faqs.shtml\t.\t"Frequently Asked Questions Click on the heading below to view the question listed, or ...	9.00000
D2307854	D2307854\thttp://cpr-test.org/how-long-should-you-spend-checking-for-breathing/\tHow long should you spend checking for breathing?\tHow long should you spend c...	8.00000
D812402	D812402\thttp://forum.slowtwitch.com/forum/Slowtwitch_Forums_C1/Triathlon_Forum_F1/Inked_Triathletes_-_How_long_do_you_wait_to_swim_after_getting_a_new_tatto...	9.00000
D1301035	D1301035\thttp://daily.barbellshrugged.com/breathing-meditation-techniquewod/\tBoost Performance with Box Breathing and Meditation â TechniqueWOD\tBoost Perfor...	8.00000
D3274575	D3274575\thttp://www.themedicalquestions.com/Med/what-does-a-do-after-a-doctors-name-stand-for.html\tWhat does a DO after a doctors name stand for\t"Home > ...	4.00000
D2697197	D2697197\thttps://quizlet.com/28714900/breathing-emergenciescpr-flash-cards/\tBreathing Emergencies/CPR\t61 terms kj02327Breathing Emergencies/CPRLearn Flashcar...	8.00000
D2075669	D2075669\thttp://www.answers.com/Q/FAQ/2226-2\tHair\t"Wiki Answers ® Categories Health Beauty Hair Unanswered | Answered Hair Parent Category: Beauty This is a ca...	10.00000
D3495831	D3495831\thttps://www.joelandsonfabrics.com/uk/learn-about-fabrics/\tLearn About Fabrics\t"Home Learn About Fabrics Learn About Fabrics Know the ins and outs of a s...	6.00000
D2626919	D2626919\thttps://greenwichmeantime.com/time-zone/usa/north-dakota/time/\tNorth Dakota Time\tNorth Dakota Time What time is it in North Dakota Time Zone / Curr...	2.00000
D102858	D102858\thttp://www.landwatch.com/Michigan_land_for_sale/Irons\tIRONS, LAKE COUNTY, WEST CENTRAL, MICHIGAN LAND FOR SALE: 1 - 15 of 81 listings\t"IRONS, LAK...	5.00000
D1988144	D1988144\thttps://www.healthtap.com/topics/how-many-calories-do-i-burn-doing-sit-ups\tTop 30 Doctor insights on: Calories Burned While Doing Sit Ups\t"Top 30 Doct...	3.00000
D2151228	D2151228\thttps://answers.yahoo.com/question/index?qid=20090304110024AA8J0tM\tCan I take a duffel bag on an airplane? 10 points to most informative.?\t"Travel Air Tr...	7.00000
D401428	D401428\thttps://www.berkeleyparentsnetwork.org/advice/pregnancy/hairloss\tHair Loss & Pregnancy\tLosing clumps of hair - had a baby 4 months ago Massive hairloss ...	10.00000
D2022008	D2022008\thttps://www.sokanu.com/careers/sports-medicine-physician/\tWhat does a Sports Medicine Physician do?\tWhat is a Sports Medicine Physician? A Sports Medi...	4.00000
D415846	D415846\thttp://salarygenius.com/ca/1/salary/criminologist-salary\tCriminologist Salary in California\tCriminologist Salary in California In order to provide the most accura...	5.00000
D1284458	D1284458\thttp://www.rowingmachineking.com/calories-burned-on-rowing-machine/\tCalories Burned on Rowing Machine\tCalories Burned on Rowing Machine Many p...	3.00000
D3299798	D3299798\thttps://birdsna.org/Species-Account/bna/species/697b/articles/introduction\tHawaiian Coot\t© Carl Giometti Macaulay Library ML28488021Common Gallinule ...	2.00000
D1890380	D1890380\thttp://www.azlyrics.com/lyrics/total/whataboutus.html\tTotal Lyrics\t"""What About Us"" lyrics Total Lyrics""What About Us"" (feat. Missy Elliott) (from ""Soul F...	4.00000
D621735	D621735\thttp://pennsylvania.hometownlocator.com/maps/feature-map,ftc,2,fid,1174114,n,elk%20lake.cfm\tElk Lake in Wayne County PA\tElk Lake - Cultural Feature (Res...	5.00000
D621743	D621743\thttp://www.healthforworld.com/does-brushing-your-hair-make-it-grow-faster/\tDoes Brushing your Hair Make it Grow Faster?\tDoes Brushing your Hair Make it ...	10.00000
D2405293	D2405293\thttps://www.niddk.nih.gov/health-information/health-communication-programs/nkdep/learn/living/medicines/Pages/medicines.aspx\tManaging Chronic Kid...	4.00000
D3347392	D3347392\thttp://www.stylecraze.com/articles/home-remedies-to-control-hair-fall/\t20 Effective Home Remedies And Tips To Control Hair Fall\tHome » Hair Care » Hair Fa...	10.00000
D3094014	D3094014\thttp://tattooartistmagazineblog.com/2013/03/04/emily-mccombs-xojane-how-not-to-be-a-dick-at-the-tattoo-shop-tamblog-tattoo-artist-magazine-blog-tam...	9.00000
D2747461	D2747461\thttps://www.seatguru.com/airlines/Spirit_Airlines/baggage.php\tSpirit Baggage\tOverview Planes & Seat Maps Check-in Baggage Infants Minors Pets What is Sp...	7.00000
D695888	D695888\thttp://www.answers.com/Q/How_long_do_you_keep_a_tattoo_covered_after_laser_removal_treatment\tHow long do you keep a tattoo covered after laser remova...	9.00000
D2326661	D2326661\thttp://www.petwave.com/Emergency/Dog-First-Aid/How-to-Perform-Canine-CPR.aspx\tHow to Perform CPR on a Dog\tHome | Emergency | First Aid How to ...	8.00000
D1748811	D1748811\thttps://wizzair.com/en-gb/information-and-services/travel-information/baggage\t\t"Baggage FAQHAND LUGGAGE55 x 40 x 23cm Changed size MAX 10 KGFRE...	7.00000
D992959	D992959\thttp://www.airlinesbaggagefees.com/american-airlines-baggage-fees/\tAmerican Airlines Baggage Fees\tAmerican Airlines Baggage Fees In our American Airline...	7.00000
D3141816	D3141816\thttps://www.healthtap.com/topics/heating-pad-rash\tTop 30 Doctor insights on: Heating Pad Rash\tTop 30 Doctor insights on: Heating Pad Rash Share Follow ...	3.00000
D768846	D768846\thttp://military.wikia.com/wiki/Panzer_Lehr_Division\tPanzer Lehr Division\t"Panzer Lehr Division Tactical symbol of Panzer Lehr Country Nazi Germany Branch Ar...	2.00000
D2607977	D2607977\thttps://uk.answers.yahoo.com/question/index?qid=20071214035756AAjGwoS\tFemale lab in heat?\tPets Dogs Female lab in heat?we have an 8 month old choco...	0.00000
D2263881	D2263881\thttp://info.drillinginfo.com/offshore-rigs-primer-offshore-drilling/\tOil and Gas Offshore Rigs: a Primer on Offshore Drilling\tOil and Gas Offshore Rigs: a Primer ...	0.00000
D274838	D274838\thttp://www.americantowns.com/nd/carrington/info\tCarrington, North Dakota Census Data & Community Profile\t"Carrington, North Dakota Census Data & Co...	2.00000
D1505319	D1505319\thttp://www.answers.com/Q/How_many_seconds_ahead_should_you_look_when_driving\tWhen driving in the city how far should you look ahead?\t"Clownz Rsc...	8.00000
D1507082	D1507082\thttp://www.answers.com/Q/How_long_should_you_wait_to_exercise_after_getting_a_tattoo\tHow long should you wait to exercise after getting a tattoo?\t"Ans...	9.00000
D3079360	D3079360\thttp://bellaartetattoo.com/faq/\tFAQ\tFAQWhat forms of payment do you accept? We accept all major credit cards, and of course cash, for payments. There is n...	9.00000

Figure 27: Raw data after processing

['unite', 'state', 'conduct', 'first', 'nuclear', 'test', 'series', 'world', 'war', 'july', 'appropriately', 'name', 'operation', 'crossroads', 'mankind', 'indeed', 'crossroadsone', 'road', 'leading', 'international', 'control', 'atomic', 'energy', 'frightening', 'nuclear', 'arms', 'race', 'operation', 'crossroads', 'bright', 'idea', 'lewis', 'strauss', 'aide', 'secretary', 'navy', 'james', 'forrestal', 'later', 'chairman', 'atomic', 'energy', 'commission', 'perhaps', 'weapon', 'past', 'atomic', 'bomb', 'threaten', 'instant', 'absolescence', 'navy', 'american', 'navy', 'immediate', 'reaction', 'hiroshima', 'find', 'impact', 'new', 'weapon', 'future', 'naval', 'force', 'bitterly', 'resent', 'air', 'force', 'monopoly', 'means', 'delivery', 'weapon', 'desperately', 'want', 'role', 'future', 'advocate', 'air', 'power', 'however', 'challenge', 'view', 'ship', 'could', 'survive', 'atomic', 'bombing', 'strauss', 'therefore', 'recommend', 'american', 'navy', 'test', 'ability', 'ship', 'withstand', 'force', 'generate', 'atomic', 'bomb'... "

Table 8: Example of a processed and tokenized document

Appendix C. Latent Dirichlet Allocation Visualization

Figure 28 visualizes the topic modeling which occurs when LDA creates a topic for the query "What was the significance of the Manhattan Project". The top 30 most relevent words of the topic cluster are shown on the right side of the graphic. These words can be used to classify new documents as relevant to this topic.

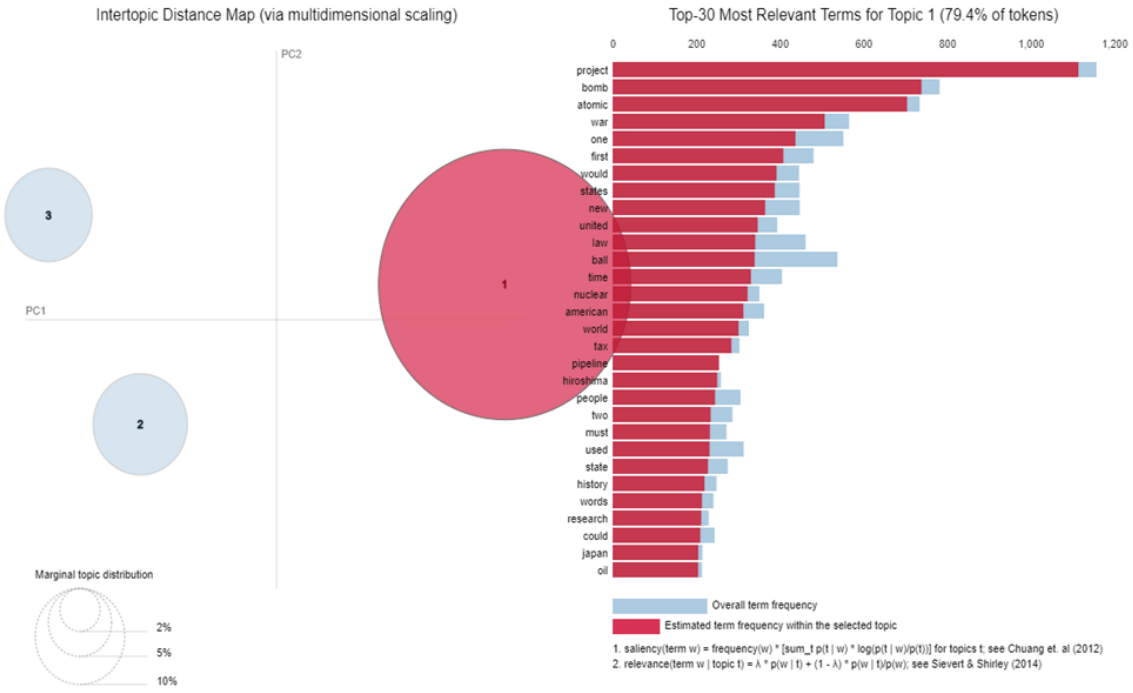


Figure 28: LDA topic that emerged from a specific search query

Appendix D. Siamese Network Model Generation Results

The figures in this appendix compare the Siamese networks performance to the Universal Sentence Encoder across varying paramaters and document lengths. The goal is to find the top performing model and keep it. Due to long training time for the network, parameters are first swept with reduced training, Table 9, the input width is always 512 and cannot be changed as that is the size of Universal Sentence Enconder's (USE) output. The performance results across varying document lengths is presented in Table 10. The top two models are highlighted yellow and these models will be provided increased training and examined closer, Table 11. Finally, the best model is used on the reserved test data, the scores are displayed in Table 12.

Model	Doc Size (words)	L1 Width	L2 Width	L3 Width	Other Alterations
1	100	256	128	none	normalization final layer
2	100	256	64	16	
3	100	256	128	64	normalization final layer
4	500	512	256	none	normalization final layer
5	500	448	384	320	
6	1000	448	384	320	
7	5000	448	384	320	
8	10000	448	384	320	
9	5000	448	384	320	softmax final layer
10	5000	448	384	320	query text used as anchors
Training Conditions: Batch Size: 64, Steps per Epoch:100, Num Training Epochs: 100					

Table 9: Preliminary Siamese Network Setups

	Document Length (words)					
	100		1000		5000	
Name	Model	USE	Model	USE	Model	USE
1	0.8438	0.9508	0.63179	0.8629	0.6214	0.8428
2	0.8593	0.9508	0.8438	0.8629	0.813	0.8428
3	0.8705	0.9508	0.683	0.8629	0.655	0.8428
4	0.9041	0.9508	0.7918	0.8629	0.7784	0.8428
5	0.9171	0.9508	0.8839	0.8629	0.8696	0.8428
6	0.9066	0.9508	0.9115	0.8629	0.8928	0.8428
7	0.9205	0.9508	0.9079	0.8629	0.8896	0.8428
8	0.9097	0.9508	0.8903	0.8629	0.8839	0.8428
9	0.8816	0.9508	0.8265	0.8629	0.8106	0.8428
10	0.9284	0.9508	0.9197	0.8629	0.9098	0.8428

Table 10: Preliminary Siamese Network Results

	Document Length (words)					
	100		1000		5000	
Name	Model	USE	Model	USE	Model	USE
6	0.9105	0.9508	0.9117	0.8629	0.9056	0.8428
7	0.9183	0.9508	0.9216	0.8629	0.9205	0.8428
10	0.9196	0.9508	0.9179	0.8269	0.9043	0.8428
Num Training Epochs: 100 $\rightarrow$ 500						

Table 11: Secondary Siamese Network Results

	Document Length (words)					
	100		1000		5000	
Name	Model	USE	Model	USE	Model	USE
7	0.9211	0.9508	0.9331	0.8629	0.9313	0.8428

Table 12: Final Siamese Network Results on Test Data

Appendix E. Prototype Use Case 2 Searches

This appendix presents all the searches conducted to generate Figure 22.

Vague Search Query	Hidden Intent	Category
War	World War 2	Military
War	Russia	Military
Military Tank	American	Military
Military Tank	Russian	Military
Missile	Anti-Aircraft	Military
Missile	ICBM	Military
Missile	Defense	Military
Fighter Jet	F-16	Military
Fighter Jet	F-35	Military
Fighter Jet	F-15	Military
Gun	Violence/Law/Policy	Military
Gun	Purchase/Store	Military
Game	Videogame	Common Searches
Game	Board/Tabletop	Common Searches
Computer	Science/School	Common Searches
Computer	Store/Purchase	Common Searches
Restaurants	Fine Dining	Common Searches
Restaurants	Delivery	Common Searches
Vaccine	COVID	Common Searches
Sick	Cold	Common Searches
Sick	Company	Common Searches
Printer	Ink	Common Searches
Printer	3d	Common Searches
Intelligence	Artificial/AI/ML	Common Searches

Table 13: Prototype Use Case 2 Searches

Bibliography

1. M. Speretta and S. Gauch. Personalized search based on user search histories. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 622–628, 2005.

2. Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, 2014.

3. Yash Alpeshbhai Patel. Sentiment analysis on imdb movie review, Oct 2021.

4. Niko A Petrocelli. CSCE 699: Natural Language Processing Project Report, 2021.

5. Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*, pages 758–770. Springer, 2005.

6. Niko A Petrocelli. CSCE 623: Machine Learning Project Report, 2021.

7. Juan Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. *Proceedings of the first instructional conference on machine learning*, 242(1):29–48, 2003.

8. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013.

9. Dastan Maulud and Adnan M. Abdulazeez. A Review on Linear Regression Comprehensive in Machine Learning. *Journal of Applied Science and Technology Trends*, 1(4):140–147, 2020.

10. H. Liu and H. Motoda. Feature transformation and subset selection. *IEEE Intelligent Systems and their Applications*, 13(2):26–28, 1998.

11. Armin Askari, Alexandre D'Aspremont, and Laurent El Ghaoui. Naive Feature Selection: Sparsity in Naive Bayes. 108, 2019.

12. Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. *EMNLP 2018 - Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Proceedings*, pages 169–174, 2018.

13. Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

14. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pages 1–12, 2013.

15. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 10 2018.

16. Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. 8 2019.

17. Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic Cosine Similarity. *Semantic Scholar*, 2(4):4–5, 2012.

18. Berhane, Fisseha. Operations on word vectors. `https://datascience-enthusiast.com/DL/Operations_on_word_vectors.html`, 2021. Accessed: 2022-02-23.

19. Georges Dupret. Discounted cumulative gain and user decision models. In Roberto Grossi, Fabrizio Sebastiani, and Fabrizio Silvestri, editors, *String Processing and Information Retrieval*, pages 2–13, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

20. Andrew Mccallum and Kamal Nigam. A comparison of event models for naive bayes text classification. *Work Learn Text Categ*, 752, 05 2001.

21. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):108–122, 2011.

22. Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, Sebastopol, CA, 2017.

23. David M. Blei and Jon D. McAuliffe. Supervised topic models. *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*, pages 1–22, 2009.

24. Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille, 2015.

25. Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

26. Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737, 2017.

27. David M. Blei, Andrew Y. Ng, and Michael T. Jordan. Latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 3:993–1022, 2002.

28. Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *31st International Conference on Machine Learning, ICML 2014*, volume 4, pages 2931–2939, 2014.

29. Dimo Angelov. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*, 2020.

30. Minghui Huang, Yanghui Rao, Yuwei Liu, Haoran Xie, and Fu Lee Wang. Siamese network-based supervised topic modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4652–4662, 2018.

31. Arpita Das, Harish Yenala, Manoj Chinnakotla, and Manish Shrivastava. Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 378–387, Berlin, Germany, August 2016. Association for Computational Linguistics.

32. Martin Gleize, Eyal Shnarch, Leshem Choshen, Lena Dankin, Guy Moshkowich, Ranit Aharonov, and Noam Slonim. Are you convinced? choosing the more convincing evidence with a siamese network, 2019.

33. Voitsekhovskaia, Valeria. Top trending google searches. `https://www.semrush.com/blog/most-searched-keywords-google/`, 2021. Accessed: 2022-02-23.

34. Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24-03-2022	Master's Thesis	Sept 2020 - Mar 2022

4. TITLE AND SUBTITLE

Improving Anonymized Search Relevance with Natural Language Processing and Machine Learning

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Petrocelli, Niko A. Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-22-M-055

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Users often sacrifice personal data for more relevant search results, presenting a problem to communities that desire both search anonymity and relevant results. To balance these priorities, this research examines the impact of using Siamese networks to extend word embeddings into document embeddings and detect similarities between documents. The predicted similarity can locally re-rank search results provided from various sources. This technique is leveraged to limit the amount of information collected from a user by a search engine. A prototype is produced by applying the methodology in a real-world search environment. The prototype yielded an additional function of finding new documents related to a provided sample document. The prototype is evaluated using real-world search examples. Results indicate that the Siamese network can produce document embeddings superior to current encoders like the Universal Sentence Encoder. Results also show the promising performance of the prototype in improving search relevancy while limiting user data transmission.

15. SUBJECT TERMS

Natural Language Processing, Search Relevance, Machine Learning, Siamese Networks

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Brett J. Borghetti, AFIT/ENG
U	U	U	UU	104	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636; brett.borghetti@afit.edu