EXPLOITING WORD AND SENTENCE EMBEDDINGS FOR DIVERSIFICATION IN CRAWLING AND RANKING

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

BY

CAN DURAN ÜNALDI

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

SEPTEMBER 2022

Approval of the thesis:

EXPLOITING WORD AND SENTENCE EMBEDDINGS FOR DIVERSIFICATION IN CRAWLING AND RANKING

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

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. İsmail Sengör Altıngövde
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering, METU _____

Prof. Dr. Özgür Ulusoy
Computer Engineering, Bilkent University _____

Assoc. Prof. Dr. İsmail Sengör Altıngövde
Computer Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Can Duran Ünaldı

Signature :

ABSTRACT

EXPLOITING WORD AND SENTENCE EMBEDDINGS FOR DIVERSIFICATION IN CRAWLING AND RANKING

Ünaldı, Can Duran
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. İsmail Sengör Altıngövde

September 2022, 100 pages

The increase in the volume of the Web and Microblogging sites caused copious amounts of duplicate or near duplicate content which emerged the diversification paradigm. On a typical search system, there are three main components, namely, a crawler, an indexer and a query processor. While most diversification approaches aim at the query processing stage of the search system, in this work, we aim to apply the diversification paradigm to both crawling and query processing stages. First, we introduce a diversification-aware focused crawler, which considers all the aspects of a given search query in order to construct a collection that contains equal coverage of them. Second, we focus on the diversification of short texts, such as social media posts, for the query processing stage. For both contributions, we apply well-known diversification approaches in the literature and extend them by exploiting the neural language models that are state-of-the-art for several information retrieval and natural language processing tasks. Our experiments, in which we evaluate both approaches with well-crafted experimental setups, show that the diversification paradigm is successful for both the crawling stage and short texts. Moreover, neural language models perform comparable results for the diversification paradigm.

Keywords: Word Embeddings, Sentence Transformers, Document Embeddings, Diversification, Focused Crawling, Microblogging

ÖZ

TARAMA VE SIRALAMADA ÇEŞİTLENDİRME AMACIYLA KELİME VE CÜMLE VEKTÖRLERİNDEN YARARLANMA

Ünaldı, Can Duran
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. İsmail Sengör Altıngövde

Eylül 2022 , 100 sayfa

Web ve mikroblog sitelerinin hacmindeki artışın yarattığı çok sayıda kopya veya kopyaya yakın içerik çeşitlendirme paradigmasını ortaya çıkardı. Tipik bir arama sisteminde üç ana bileşen bulunmaktadır, bunlar tarayıcı, endeksleyici ve sorgu işleyicisidir. Çoğu çeşitlendirme yöntemi sorgu işleme bileşenini hedeflerken, bu çalışmada çeşitlendirme paradigmasını hem tarayıcı hem de sorgu işleyiciye uygulamak hedeflenmektedir. Öncelikle, sorgunun alt anlamlarını kullanarak alt anlamları dengeli bir dağılımda içeren bir koleksiyon oluşturmayı amaçlayan çeşitlendirmeye duyarlı odaklı tarayıcı geliştirildi. Sonrasında, sorgu işleme aşaması için sosyal medya içerikleri gibi kısa metinleri çeşitlendirmeye odaklanıldı. Her iki katkıda da, literatürde tanınmış çeşitlendirme yaklaşımları uygulandı ve bir çok bilgi getirimi ve doğal dil işleme yönteminde kullanılmış modern sinirsel dil modelleri kullanılarak bu yaklaşımlar genişletildi. Her iki yaklaşım da iyi hazırlanmış deney ortamlarında denendiğinde çeşitlendirme paradigmasının hem tarama aşamasında hem de kısa metinlerde başarılı olduğu görüldü. Ayrıca, çeşitlendirme paradigması için sinirsel dil modellerinin kıyaslanabilir sonuçlar aldığı görüldü.

Anahtar Kelimeler: Kelime Temsilleri, Cümle Dönüşütürücüler, Doküman Temsilleri, Farklılaştırma, Odaklı Tarayıcılık, Mikrobloglar

To my family

ACKNOWLEDGMENTS

First of all, I want to express my gratitude to my supervisor İsmail Sengör Altıngövde since this thesis would not have been possible without his continuous support and guidance. Also, I'm grateful to my thesis committee members Prof. Dr. İsmail Hakkı Toroslu and Prof. Dr. Özgür Ulusoy, for their positive encouragement and valuable comments.

I am deeply indebted to my colleague Yiğit Sever for his continuous support and helpful brainstorming through this thesis. I am also extremely grateful to my lab partners Abdullah Doğan, Sena Terzi, and Bilge Eren, for their valuable contributions to this thesis. Special thanks to Makbule Gülçin Özsoy and Kezban Dilek Önal for their contributions on this study. Moreover, I would like to thank my colleague Merve Asiler for her understanding and Furkan Murat for his support.

And finally, I would like to offer my special thanks to my parents for their support throughout my life and for their understanding and supportive attitude throughout this thesis. Moreover, I could not have undertaken this journey without Göknur Ercan and I would like to express my deepest gratitude for her love and support. Without them, I would not be able to accomplish this study.

This work is partially funded by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant no. 117E861.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

ABBREVIATIONS	
NLP	Natural Language Processing
WE	Word Embeddings
ASCII	American Standard Code for Information Interchange
TF	Term Frequency
IDF	Inverse Document Frequency
TFIDF	Term Frequency - Inverse Document Frequency
BERT	Bidirectional Encoder Representations from Transformers
SBERT	Sentence-BERT
MMR	Maximal Marginal Relevance
WMD	Word Mover's Distance
NLTK	Natural Language Toolkit
JWNL	Java WordNet Library
RT	Retweet

CHAPTER 1

INTRODUCTION

A typical search (or text-retrieval) system includes 3 main components, namely, a general-purpose or focused crawler, to obtain the underlying collection from the Web, an indexer and a query processor (i.e. ranker). The recently emerging paradigm of diversification is essentially associated with the last component, ranking, and several approaches are proposed to post-process a candidate ranking to obtain more diversified top results where, broadly, diversification may imply results different than each other and/or covering different aspects of the query at hand. In this thesis, we aim to achieve diversification in both focused crawling and ranking stages; and while doing so, we exploit neural language models that set the state-of-the-art for several information retrieval (IR) and natural language processing (NLP) tasks.

1.1 Diversification-Aware Focused Crawler

The initial step of a typical search system is a crawler, which can be a general-purpose crawler or a focused crawler. Unlike the general-purpose crawler, the focused crawler aims to gather pages that is relevant to the given target topic. Although the main goal is to gather only relevant pages, since the crawler can not know the content of the page before fetching it, all of the pages might not be relevant to the topic. The crawler uses the information of the given seed URLs and continuously fetches the URLs that are located in the fetched pages. While doing so, a focused crawler exploits the textual content of the fetched page to predict the likelihood of the URL to be relevant to the target topic. In this thesis, following the practice in Chakrabarti et al. [4] and Pant et al. [5], we implement a baseline focused crawler which assess the relevance of a

URL based on its anchor text and nearby text, as well as the content of the entire page including the URL.

For a given search query, there can be many aspects of that query each can lead to different pages. Assuming there is no aspect of the search query might lead to different pages that is only have one meaning of the query. However, if we add the knowledge of the aspects to the crawler we may gather more diverse results that may satisfy all meanings of the aspects and the query. In this thesis, we introduce the notion of diversification-aware focused crawling, which aims to incrementally construct a collection with equal coverage of alternative aspects of a target crawling topic. To this end, we adopt an approach from Ozdemiray & Altingovde [6] that has been proposed for search result diversification. The proposed approach is extended to compute similarity between target topic and Web pages using word embeddings [7] [8] [9] [10] and sentence transformers [11].

1.2 Diversifying Rankings for Tweet Search

A crucial component of a typical search system is the ranker. After fetching and indexing the search results, a search system should present ordered results to the user. The main goal of the ranking stage is to sort the results using the relevance score between the query and results. An emerging paradigm considered during ranking is diversification, which aims to eliminate similar search results and favor results that include novel information and/or cover alternative interpretations of the query. In this study, we focus on short text (namely, Twitter posts) where the problem becomes more challenging, as argued in [12].

Microblogging sites are popular and their popularity increases over time. On average, more than 500 million tweets per day and more than 200 billion tweets per year are tweeted. Although the growth on the tweets are declining, the increase in the number of tweets are still increasing. In this study, we aim to improve the results for tweet search, which is a more challenging problem since a tweet is limited to the 280 characters. Understanding the meaning of a sentence becomes harder if the sentence becomes shorter as explained in [12]. Although the Twitter has 280 character limit

now, back in 2013 which our dataset is created at, the Twitter was only has 140 characters limit which limits our ability to extract the meaning from the sentence.

To this end, following the practice in [12], we use three well-known diversification approaches from the literature (namely, MMR [13], Sy [14] [15] and xQuaD [3] [16]) with the well-known scoring methods such as BM25 3.1.2, tf-idf 3.1.1 and distance metrics such as Cosine 3.3.1 and Jaccard 3.3.2. We extend these diversification approaches to employ word embeddings and sentence transformers representations to investigate the impact of the models on the diversification problem on ranking stage.

1.3 Contribution

To the best of our knowledge, it is the first time that word embeddings and sentence transformers are applied to a diversification aiming search system. Our contributions in this work are follows:

- Create a focused crawler that is using sentence transformers and word embeddings.
- Introduce the notion of diversified focused crawler which exploits the aspects of target topic in addition to page content and link text.
- Create a ranking system that applies diversification paradigm on a microblogging site, Twitter, by using sentence transformers and word embeddings.

Finally, we extensively evaluate the proposed approaches in a well-crafted experimental setup. For the focused crawling, our experiments involved real-time crawling of tens of thousands of Web pages for the target topic Covid 19. For the ranking part, we employed a benchmark dataset, Tweets2013 [14]. Our experiments reveal that the proposed approaches are promising to improve diversification for both focused crawling and ranking components of a search system.

1.4 The Outline of the Thesis

The rest of this thesis is organized as follows. Chapter 2 reviews the related work and Chapter 3 provides the preliminary information about the methods, metrics and algorithms that are used in this study. In Chapter 4, we propose our approach of diversification aware focused crawling and provide experimental results. Chapter 5 presents how embeddings are employed during diversification of rankings for tweet search, and provides a detailed experimental evaluation of their impact on diversification performance. Finally, we conclude and point to future work directions in Chapter 6.

CHAPTER 2

RELATED WORK

In this chapter, studies that are inspired or used as a basis and studies that are specified in the context of this work including baseline methods, general diversification problem approaches, twitter studies and focused crawler related studies are explained.

2.1 Focused Crawling

The rapid increase in the number of pages on Web, caused too much information to extract and decide whether the information is useful or not. By this mean, there are lots of focused crawler implemented for some specific or general contents.

Although it was not a complete focused crawler, the idea behind the ARACHNID [17] using agents to get best neighbors just like a crawler uses fetcher threads to crawl best links. The resource of a crawler is the computational power and ARACHNID simulates this resource by giving energy to its agents. Moreover, the score between the query and neighbors are calculated using cosine similarity in order to select the best neighbor to jump. Not only working on local system, but also ARACHNID tested on actual on-line Web environment.

The work of Chakrabarti et al. [4] [18] is the first work using the naming focused crawler. They are using general approach types of the focused crawler thus setting general convention and approach to the focused crawler area. They use naïve-Bayesian classifier to measure the relevance of a downloaded page to the target topic. Using the link information on selecting URLs is presented in their work which is inspired most of the later and even recent works such as our study. The idea of using

the link information is used in Rennie et al. [19] before the study of Chakrabarti et al. The link information can be summarized as follows:

- The URL of the link.
- The anchor text and the text near anchor text.

Although this information is simple, it is very important to catch the actual meaning of the link. Moreover, they use the category information from DMOZ which is important for our study since we also use a similar information on our crawler.

The work of Altingovde and Ulusoy [20] is a different approach to gather the pages related to the target topic. The main idea behind the work is to gather pages that can be located in an unrelated page. For example, page1 has a link to page2 and our crawler find page1 irrelevant, however page2 is highly relevant to our topic. They try to use probability of the page to be in the context of our topic and by creating rules they obtain the probability of getting correct information by following a path. Thus, their proposed method is able to select best path among the pages.

Wang et al. [21] are implementing a focused crawler that is applying relevance judgement using a Bayesian Classifier (as explained by Leung [22]). They are not using a scoring function such as BM25, instead they direcly use their classifier to find the relevance of a page to the target topic. Similar to Wang et al., Lu et al. [23] are improving their focused crawler using a classifier to detect irrelevant pages. However, for the scoring function they extend the page score with anchor text and URL text information like the practice of Chakrabarti et al. [4] and Pant et al. [5]. The workflow of their crawler is similar to our non diversification focused crawler workflow. Although, the general structure of their focused crawler is similar to our proposed model, they are not applying any diversification methods to the system. Again similar to our work, they create a focused crawler that supports for multiple topics.

Du et al. [24] are implementing 4 different focused crawlers that works for different target topics (where the first crawler is just breadth-first crawler). They are using $tf\text{-}idf$ and compute cosine similarity (just like our initial focused crawler) at their VSM crawler. Then they extend their model by adding semantic meaning to

the VSM crawler by calculating the semantic meaning between the terms using ontology. Their final crawler uses semantic vector space instead of semantic meaning between the terms.

Another approach from Farag et al. [25] is classifying the web pages to some predefined events since events might carry important information to a related topic or related locations. They extend a general purpose crawler to focus on a given event by using classification models. An event contains the topic (they use California shooting event), the date and the location information. By scoring the page using the event and page similarity they aim to find pages only relevant to the given event.

2.2 Diversification for Tweet Search Results

Microblogging sites and diversification on them are both interesting and challenging problems. The work of Ozsoy et al. [12] is applying diversification on a microblogging site, namely Twitter. They apply most well-known diversification algorithms such as MMR [13], Sy [14] [15] and xQuAD [3] [16] in order to re-rank the twitter search results by combining these methods with distance metrics such as Jaccard and Cosine Although the sentence length of each data is limited, their results are shown to be successful in terms of diversifying tweets.

Onal et al. [26] are using the same setup with Ozsoy et al. [12] and they use word embeddings in their work. Unlike our proposed method, they use word embeddings to expand the query and tweets. Using K-Nearest Neighbor expansion method together with word embeddings they increase the information available for each query and tweet, then they use Jaccard similarity to calculate the similarity. As a second expansion method, they use ConceptNet [1]. Moreover they use the most well-known diversification algorithms, namely MMR, Sy and xQuAD to diversify the tweet results. Their results show that expansion that is applied is giving better diversification results than the direct usage of the query and tweets.

Gollapudi et al. [27] propose methods to increase diversity on result sets. Their methods are trying to maximize: the sum of similarity and diversity (Max-Sum) and mini-

[1] https://api.conceptnet.io/

mum relevance and dissimilarity (Max-Min). They use greedy algorithms to achieve the two objectives. For another greedy approach, Vieira et al. [28] proposes 2 methods which are related to the diversification paradigm. GMC tries to maximize marginal contribution of each candidate which is similar to the MMR, however, the GMC uses a different ranking function. Second method, GNE they use the GRASP technique which is a randomized selection method. Then they use the same ranking function that they used on GMC to restrict the selection set. By changing then λ variable they change the randomness of the results.

Although it is not applied for tweet search results, Ulu et al. [29] are exploiting the word embeddings for web search results diversification. Since the web search results are large documents, they create document embeddings using the idea of De Boom et al. [30] and use the document embeddings on well-known diversification algorithm MMR. As they state in their experimental results, almost all of the sets document embeddings with MMR gives the best α-nDCG@10 scores.

Moreover, Text REtrieval Conference (TREC) [2] was organizing microblogging tracks from 2011 to 2015. Just like our study, microblogging tracks contains twitter data in order to be diversified by the contestants. Moreover, there are 23 submission on average to a single track and 40 submission to the track 2011 as peak. There are many different approach to the microblogging problem just from the TREC microblogging track.

[2] https://trec.nist.gov/

CHAPTER 3

PRELIMINARIES

In this section, the methods, techniques and the fundamentals used throughout this study is explained in detail.

3.1 Representing Documents

3.1.1 TF-IDF

The tf-idf weighting method, as explained in [31], aims to represent the importance of a word in a document. We can use the tf-idf to calculate the importance of a document on a corpus by finding the importance of all of the words inside the document. There are two parts of the method: term frequency and inverse document frequency.

Term frequency denotes the frequency of a word inside a document, not the whole corpus. This is used as a metric for the word's importance. The calculation of the term frequency is given in the Equation 3.1.

$$\text{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{3.1}$$

The inverse document frequency (IDF) gives insight on how unique or rare a word is in the corpus. The unique words are considered more important than the frequent ones such as stop words. The calculation for IDF is given in the Equation 3.2 where N is the number of documents in the corpus and D is the set of documents in the corpus.

$$\text{idf}(t) = \log(\frac{N}{1 + |d \in D : t \in d|}) \tag{3.2}$$

If the number of documents that contain the term t is lower, the idf value becomes higher in order to increase the importance of a rare word. The $tf\text{-}idf$ score of a word can be calculated as shown in Equation 3.3.

$$\text{tf-idf}(t, d) = tf(t, d) * idf(t) \tag{3.3}$$

Another usage for $tf\text{-}idf$ is to create $tf\text{-}idf$ vectors using the calculations above. Thus, a document can be represented by a vector of $tf\text{-}idf$ values computed for the terms in the document.

The vector representations are important for NLP, as well as for the current study since:

- A document vector can be used alongside a distance metric (e.g. cosine distance) to calculate the similarity between two documents.
- Similarly, a document vector can be used to calculate the similarity of the document to a given search query.

3.1.2 BM25

BM25 is a function which is widely used in information retrieval to rank a set of document in terms of their relevance to a given query without considering positions of the words inside the document or query. The calculation of BM25 score of a document for a given query is given in the Equation 3.4.

$$\text{score}(d, q) = \sum_{i=1}^{n} \text{idf}(q_i) \cdot \frac{\text{tf}(q_i, d) \cdot (k_1 + 1)}{\text{tf}(q_i, d) + k_1 \cdot (1 - b + b * \frac{|d|}{\text{avg}(|D|)})} \tag{3.4}$$

BM25 uses the idea behind the $tf\text{-}idf$ (Equation 3.1 and Equation 3.2) and expand with some parameters it to achieve better accuracy.

- $k_1 \in [1.2, 2]$

- $b = 0.75$
- avg(|D|): average length of the documents inside the corpus.

3.1.3 Word Embeddings

As explained in the above sections, many retrieval methods need to represent words and/or documents as numerical vectors. However, numerical representation of a word is a challenging problem for information retrieval. One of the solution to converting a document to a numerical representation is creating $tf\text{-}idf$ vectors as explained in the section 3.1.1. Another, recent approach is to find vector representation of a word, such as Word2Vec [32] [7], to represent a word. By using word embedding representations, we are able to achieve the following:

- Check similarity between two words using a distance metric (e.g. cosine distance)
- Combine these vectors to represent a document vector to find similarity between two documents or a document to a query.

There are many applications of word embeddings such as Word2Vec [32] [7], ELMo [10], GloVe [8] and FastText [9]. For the purposes of this study, we will explain only GloVe and FastText.

3.1.3.1 GloVe

Glove [8] model, stands for Global Vectors, is developed with an unsupervised learning algorithm to create word representations. The idea behind the algorithm is to represent words using their semantic distances and probability to be on the same context. For example, if we have 3 word representations for 1 capital ('Ankara') and 2 countries ('Turkey' and 'Italy'). 'Ankara' has two main properties: being a capital, being inside Turkey. If we remove the vector of Turkey from the vector of Ankara we obtain only the capital status of the word. Moreover, if we add Italy to the equation, we again add the belonging status to the word, however, this time for Italy and get the

capital of the Italy, Rome.

$$vector('Ankara') - vector('Turkey') + vector('Italy') \tag{3.5}$$

Their approach is to define the word representations such that after the operations in the Equation 3.5, resulting vector should be as close as possible to the vector of 'Rome'. Another example can be the relation between Prince and Princess. If we remove Man vector from the Prince vector and add Woman, we should be able to get a resulting vector which is very close to the vector of the Princess word. One vector's value is dependent to the other one. The main constraint for the model is given in the Equation 3.6, where w_i is the vector of the main word, w_j is the vector for the word in the context and b_i, b_j values are biases. Finally, X_{ij} gives the probability of the main word and the context word's appear together.

$$w_i^T w_j + b_i + b_j = \log(X_{ij}) \tag{3.6}$$

3.1.3.2 FastText

FastText [9] is a library created by Facebook AI team to create vector representations by taking morphology into account. Facebook made pre-trained word embeddings for 294 languages available on FastText. They are using the idea behind the Word2Vec (skip-grams) by expanding the idea with the usage of n-grams, thus separating the word into smaller chunks. This method allows model to be successful on the languages that are morphologically rich such as Turkish. For example, in Turkish we are using 'hecelemek', which can be directly defined in English as spell. However, the word 'hecelemek' has very similar meaning with its root 'hece', which is very common than the original word. The common words are easier to represent as vectors, since they are more likely to appear in the training set. However, the rare words can be never seen by the model, thus reducing the accuracy of it when it encounters such word. By taking n-grams the model is able to get the meaning behind the rare word by finding more common one.

Their proposed model is using n-grams, where $n \in [3, 6]$. An example for the n-gram can be the word 'abacus' (where $<$ and $>$ are boundary symbols):

```
2-grams of the word abacus:
    <a, ab, ba, ac, cu, us, s>
3-grams of the word abacus:
    <ab, aba, bac, acu, cus, us>
```

The methodology behind using n-grams is to find the word representation in terms of the word vectors of its n-grams. The general scoring function of the model is given in Equation 3.7, where u_{wt} is the vector representation of the term and v_{wc} is the vector representation of the context word.

$$s(w_t, w_c) = u_{wt}^T v_{wc} \tag{3.7}$$

However, after taking n-grams into account, the equation becomes the Equation 3.8, where G_w is the n-grams of the word and z_g is the vector representation of each n-gram.

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c \tag{3.8}$$

3.1.4 Sentence Transformers

In order to understand sentence transformers, we should have a look at Bidirectional Encoder Representations from Transformers (BERT) [1].

3.1.4.1

BERT BERT is a multi-layer bidirectional Transformer and unlike Word2Vec or GloVe, BERT is considering the context in which the word is found. For example, we know that the word 'application' has two different meaning in the following words:

```
    He is researching the application of machine learning
techniques on NLP.
    He has released his application on AppStore.
```

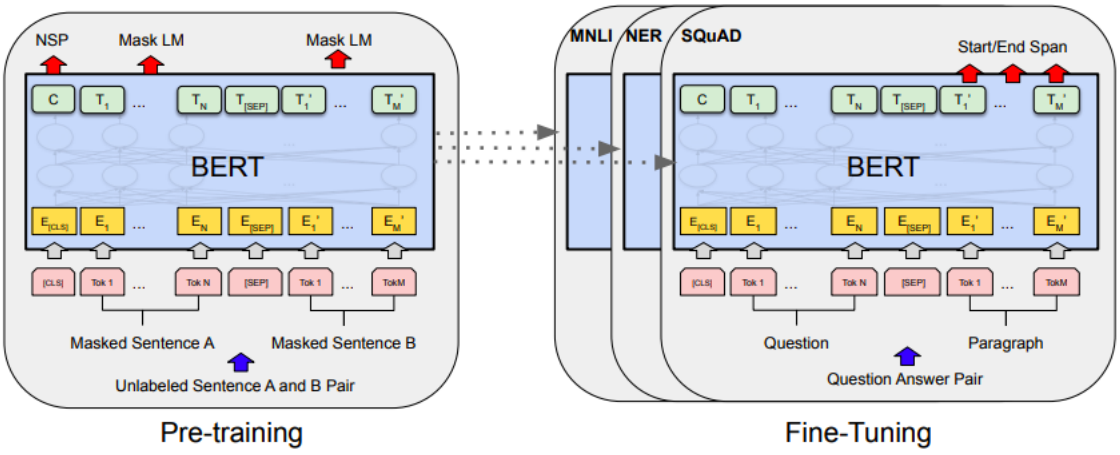

Figure 3.1: BERT Architecture (adapted from [1])

Another example can be the word 'saw', since saw can be a verb which is the past tense of see, or a carpenter tool that is used to cut trees, these two usages are very different from each other. For Word2Vec or GloVe representations these 'application's has the same vectors. However, BERT is considering the context of the sentence to define the word vectors. The word 'application' will have different word vectors for these usages as it should be. The architecture of BERT, as explained in the BERT paper [1], is given in the Figure 3.1

3.1.4.2 SBERT

Word embeddings are powerful tools to represent the words and find the relation between two words. Moreover, word embeddings can be converted using some techniques (explained in 4.2.3.3) to represent documents or sentences. However, these techniques are not accurate enough to find the actual document representation. Sentence Transformers (SBERT) [11] is developed for such need, to represent sentences in more accurate way. The main idea behind the SBERT is expanding the fine-tuning capabilities of BERT and modify it to obtain more accurate and meaningful sentence-embeddings. Both BERT and SBERT are developed by Google to apply machine learning techniques in the Natural Language Processing (NLP) area.

The drastic improvement in the performance of the fine-tuning BERT model as showed

in SBERT paper [11] is one of the main advantages of using sentence transformers. In this thesis, we also employ SBERT to obtain sentence embeddings to be used during focused crawling and ranking tasks. Note that, we essentially focus on the impact of such embeddings on the effectiveness (i.e., quality of the results) but not the efficiency.

3.2 Search Result Diversification

3.2.1 Maximal Marginal Relevance

Maximal Marginal Relevance (MMR) [13] is a greedy re-ranking algorithm designed to create diverse search results on a given result set for a given query. The typical approach to rank the documents for a given query is to rank the documents based on their similarity to the query. Although, ranking using the similarity between a query and a document is the part of the algorithm, MMR also checks the similarity between the current document and already selected documents. If current document is close to the already selected ones, we can reduce its ranking score, thus prevent duplicate, near duplicate or similar results on the result set. The algorithm's scoring function is given in the Equation 3.9:

$$\text{MMR} = \text{Arg} \max_{D_i \in R \setminus S} \left[\lambda(sim(D_i, Q)) - (1 - \lambda) \max_{D_j \in S} (sim(D_i, D_j)) \right] \tag{3.9}$$

In Equation 3.9, D_i denotes documents in the collections, Q denotes the query, R denotes the relevant documents in the collection and S denotes the already selected documents.

We can set the lambda value to weigh the relevance and diversification parts of equation. As the lambda increases we give higher importance to the first part of the equation, which is relevance, similarly, if we decrease the lambda value we give higher importance to the second part of the equation, which is the diversification. As an example, assume that we have 5 documents and their relevance values to the query and similarity among each other given below:

	Doc_1	Doc_2	Doc_3	Doc_4	Doc_5		Query
Doc_1	1	0.8	0.3	0.6	0.1	Doc_1	0.9
Doc_2	0.8	1	0.7	0.4	0.5	Doc_2	0.7
Doc_3	0.3	0.7	1	0.1	0.4	Doc_3	0.6
Doc_4	0.6	0.4	0.1	1	0.3	Doc_4	0.5
Doc_5	0.1	0.5	0.1	0.3	1	Doc_5	0.2

Table 3.1: Example: Doc-doc and Doc-query Similarity Scores

If we want to select the top 3 documents using MMR algorithm and set the lambda value to 0.5, we can follow the steps given in the Equation 3.10

$$
\begin{aligned}
&Iteration1: \\
&\qquad \text{Doc}_1 \text{ is selected.} \\
&Iteration2: \\
&\qquad \text{Doc}_2 = 0.5 * 0.7 - 0.5 * 0.8 = -0.05 \\
&\qquad \text{Doc}_3 = 0.5 * 0.6 - 0.5 * 0.3 = 0.15 \\
&\qquad \text{Doc}_4 = 0.5 * 0.5 - 0.5 * 0.6 = -0.05 \\
&\qquad \text{Doc}_5 = 0.5 * 0.2 - 0.5 * 0.1 = 0.05 \\
&\qquad \text{Doc}_3 \text{ has the maximum score, thus selected.} \\
&Iteration3: \\
&\qquad \text{Doc}_2 = 0.5 * 0.7 - 0.5 * 0.8 = -0.05 \\
&\qquad \text{Doc}_4 = 0.5 * 0.5 - 0.5 * 0.6 = -0.05 \\
&\qquad \text{Doc}_5 = 0.5 * 0.2 - 0.5 * 0.2 = 0.00 \\
&\qquad \text{Doc}_5 \text{ has the maximum score, thus selected.} \\
&\qquad \text{RS} = [\text{Doc}_1, \text{Doc}_3, \text{Doc}_5].
\end{aligned}
\tag{3.10}
$$

Although D_2 and D_4 are more relevant than D_5, we've not selected them. The algorithm favored diversity to relevance. The pseudocode of MMR is given in the Algorithm 1.

Algorithm 1: MMR

Data: Docs, length, Q

Result: S

$D \leftarrow$ sort(Docs)

$S \leftarrow D[0]$

del $D[0]$ from D

while $LEN(S) < length$ **do**

  $X \leftarrow 0$

  $mSc \leftarrow -INTMAX$

  $i \leftarrow -1$

  while $X < LEN(D)$ **do**

    //calculate MMR score of $D[X]$ over Q and documents inside S

    $sc \leftarrow \lambda(sim(D[X], Q) - (1 - \lambda) \max(sim(D[X], S)))$

    if $sc > mSc$ **then**

      $mSc \leftarrow sc$

      $i \leftarrow X$

  push $D[i]$ to S

3.2.2 Sy

Sy method, as proposed in [14] [15], is a duplicate or near-duplicate elimination method which is used on tweet streams. According to their work there are 5 different levels of near-duplication:

- Exact Copy
- Nearly Exact Copy
- Strong near-duplicate
- Weak near-duplicate
- Low Overlapping

The levels are defined by the similarity between tweets in terms of syntactical similarity, contextual meaning and semantic meaning. Since the method is defined and

tested on Twitter environment it perfectly fits our experimental setup 5.4. Moreover, near-duplicate elimination can be used on diversification problem since the less duplicate in a result set results in a more diverse results. The method uses different types of distance metrics and methods such as Levenshtein distance, the difference in tweet lengths, overlap in WordNet [33] and WordNet concepts as applied in the work of Lin [34] and tweet features (e.g hashtags, URLs, etc.). By calculating these distances they eliminate tweets if they have near-duplicate tweets in the result set, thus reducing the number of duplicates to 0 (only the highest ranked one). For each tweet if the result set contains a duplicate of the current tweet, the algorithm rejects the tweet and only accepts tweets that are not have any duplicate on the selected result set. The algorithm is explained in the Algorithm 2 and Algorithm 3.

Algorithm 2: Sy

Data: Docs, length, Q

Result: S

$D \leftarrow$ sort(Docs)

$S \leftarrow$

$i \leftarrow 0$

while $LEN(S) < length$ **do**

    // Check whether the current document (tweet) has any duplicate in the

    // already selected result set.

    if *!duplicate(*$D[i]$*,* S*)* **then**

        push $D[i]$ to S

    $i \leftarrow i + 1$

Algorithm 3: Sy Duplicate Detection

Data: T, S, length

Result: B

$B \leftarrow False$

$i \leftarrow 0$

while $LEN(S) < length$ **do**

    // Check whether two tweets are duplicate **if** *duplicate(*T*,*$S[i]$*)* **then**

        $B \leftarrow True$

    $i \leftarrow i + 1$

In [15] [14] the authors define six variants of Sy algorithm that differs in the employed

similarity metric. The general algorithm is the same in all variants, however, the semantics or the contextual features are not common in all of them. The variants are listed below:

Sy	Syntactical features compared only
SySe	Syntactical features and semantic features (WordNet)
SyCo	Syntactical and contextual features
SySeCo	Syntactical, semantic and contextual features
SySeEn	Syntactical and semantical features
SySeEnCo	All features

3.2.3 xQuAD

Another result diversification method, the eXplicit Query Aspect Diversification [16] [3] (xQuAD), is an explicit diversification method which considers not only the query itself but also subqueries (i.e., aspects) of the query. xQuAD is a probabilistic model that is diversifying the results by using the knowledge of the aspects. The basis of the model can be seen in the Equation 3.11

$$(1 - \lambda)\mathrm{P(d|q)} + \lambda\mathrm{P(d,\ S|q)} \tag{3.11}$$

where P(d|q) is the probability of the document occur in the result set of the query q and P(d, S|q) is observing the document but not the documents in the set S. As can be seen in the formula, the P(d|q) is defining the relevance whereas the P(d, S|q) aims to achieve the diversification of the results. The λ value is defining the trade-off between these two components. Moreover, in their work, they employ the query aspects while computing the P(d, S|q). The algorithm of the method (which is directly adapted from [3]) is given in the Algorithm 4

Algorithm 4: xQuAD (adapted from [3])

Data: q, R, τ, λ

Result: S

$S \leftarrow \emptyset$

while $LEN(S) < \tau$ **do**

$d^* \leftarrow \arg\max_{d \in R \setminus S}(1-\lambda)\text{P(d|q)} + \lambda\text{P(d, S|q)}$

$R \leftarrow R \setminus \{d^*\}$

$S \leftarrow S \cup \{d^*\}$

3.3 Distance Metrics

Vector representations of words allow us to operate over numerical data, however, for some operations, such as finding similarity over two words, we need some distance metrics. There are some distance metrics such as Euclidean distance, Manhattan distance, Cosine distance for such purposes. In this study, we used cosine distance, Jaccard distance and word mover's distance for such operations.

3.3.1 Cosine Distance

Cosine distance is used as a similarity metric on data analysis purposes since the metric gives results between [-1, 1] regardless the magnitude of the vectors and accurate if the provided vectors are well defined. The metric calculates the angle between given two vectors as long as they are at the same dimension vector spaces. The similarity between two vectors becomes the cosine of the angle between them. On the Figure 3.2 the similarity of B and C vectors can be defined as $\cos(\theta)$.

If the vectors are proportional, i.e. they have the same unit vectors, their cosine similarity becomes 1. Similarly, if they are orthogonal, their cosine similarity becomes 0 and for opposite vectors, cosine similarity becomes -1.
Cosine similarity calculation is given in the Equation 3.12.

$$\cos\theta = \frac{A \cdot B}{|A||B|} \tag{3.12}$$

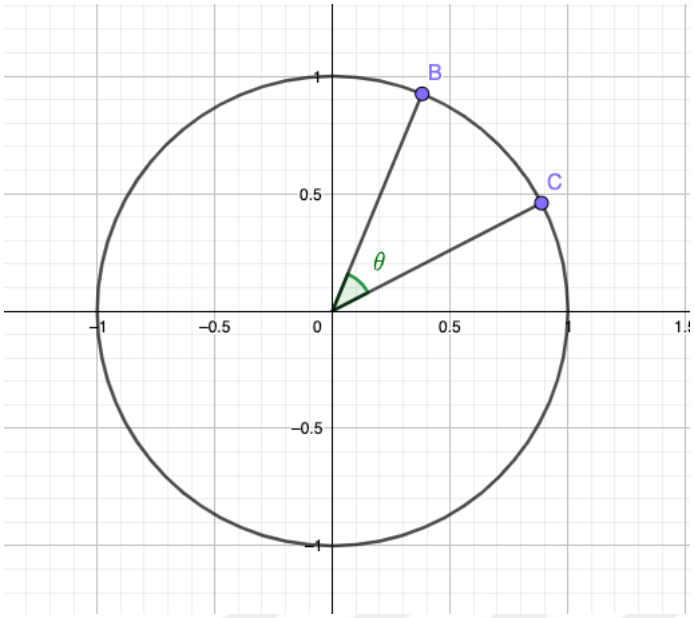

Figure 3.2: Cosine Distance

For this study, we will use the cosine similarity to find the distance/similarity between two document vectors.

3.3.2 Jaccard Distance

Jaccard Distance [35], also known as Jaccard index or Jaccard Similarity coefficient, is a statistical metric used to determine the similarity or diversity over given sets. Although the purpose of the metric is similar to the Cosine Distance, the use case of Jaccard is different. Instead of numerical representations, Jaccard can be applied directly on given two sets. The metric is simply finds a proportion of common elements of two sets over the unique elements in the union of them. The calculation of the metric is given in the Equation 3.13

$$J = \frac{|A \cap B|}{|A \cup B|} \tag{3.13}$$

Since we can think a document as a set of words, we can directly use Jaccard metric over given two documents in order to find their similarity.

Assume the following three documents:

```
D1 = Barack Obama was the president of the USA.
D2 = We can swim today, since its hot.
D3 = Kamala Harris become the vice president of the USA.
```

We can find the similarity between D1 and the others by applying Jaccard:

$$
\begin{aligned}
&\text{D1} \cap \text{D2} = \emptyset \\
&\text{D1} \cup \text{D2} = \{\text{Barack, Obama, was, the, president, of, USA, we, can, swim, today, since, its, hot}\} \\
&\text{J(D1,D2)} = \frac{|\text{D1} \cap \text{D2}|}{\text{D1} \cup \text{D2}} = \frac{0}{14} = 0.0 \\
&\text{D1} \cap \text{D3} = \{\text{the, president, of, USA}\} \\
&\text{D1} \cup \text{D3} = \{\text{Barack, Obama, was, the, president, of, the, USA, Kamala, Harris, become, vice}\} \\
&\text{J(D1,D3)} = \frac{|\text{D1} \cap \text{D3}|}{\text{D1} \cup \text{D3}} = \frac{4}{12} = 0.33
\end{aligned}
\tag{3.14}
$$

Jaccard similarity between D1 and D3 is higher than D1 and D2. As can be seen, the jaccard score is always normalized and in [0,1], so we don't need to apply any normalization to the score or the documents. Moreover, in this toy example, we have not applied any pre-processing to the documents, however, in real applications we need to. The pre-processing stages will be explained in the Section 4.1.1, by applying pre-processing we will not be biased neither by the stopwords, such as the, of, etc, nor the incorrectly spelled words.

3.3.3 Word Mover's Distance

Converting word embeddings into document representation is a challenging problem and there are some ways to solve that problem. Taking minimum, maximum or the average of each dimension of word embeddings of a document to represent the document itself is one of the ways. Moreover, sentence transformers can directly give the document embedding of a given document. We can use these document representations using distance metrics such as Cosine distance in order to find the similarity between two documents.

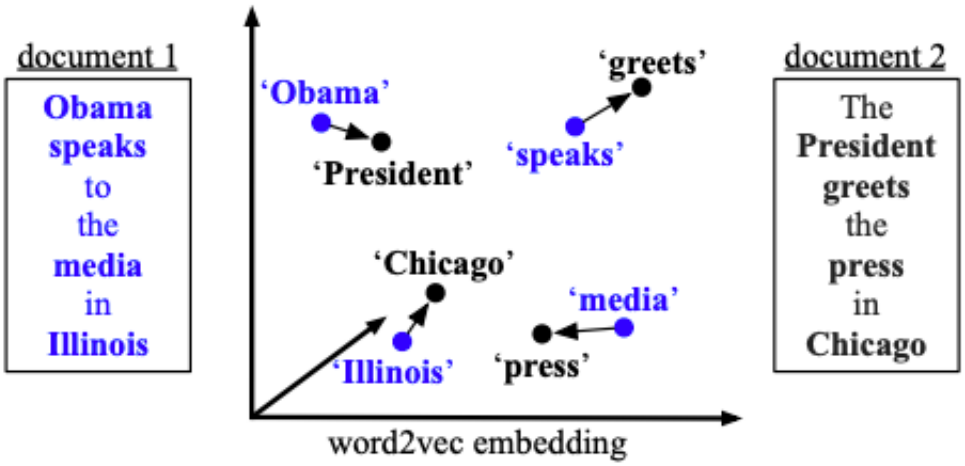

Figure 3.3: WMD Example (adapted from [2])

Word Mover's Distance (WMD) [2] is a distance metric which is directly calculating the distance between two documents by taking their word embeddings. The idea behind the WMD is to find the cumulative distance of the words inside the document to calculate the distance between the documents itself. As explained in the Section 3.1.3.1, adding or subtracting the correct meanings from the embedding, we can find another word's embeddings such as removing Paris from France and adding Ankara to it gives us Turkey. The WMD cumulatively calculates these distances and finds the distance between given documents. The explanation of the idea in 2D vector space from their work is given in the Figure 3.3 as an example.

CHAPTER 4

FOCUSED CRAWLING

One of the main contributions of this work is constructing a diversification-aware focused crawler, which is constantly collecting the web pages starting from a group of seed urls and trying to find most relevant pages to the target topic by moving on the urls inside the fetched ones.

In the following, we begin with describing Apache Nutch, an open-source general purpose crawler, which we extend to build a focused crawler (Section 4.1). Next, in Section 4.2, we present how we modified Nutch to build our baseline focused crawler, which employs classification and scoring components as in [4] [5]. In this section, we also discuss how we exploit word and sentence embeddings in this scenario. In Section 4.3, we propose a diversification-aware focused crawler which is aware of aspect information just like explicit diversification methods and again employ word and sentence embeddings to diversification-aware crawler

4.1 Apache Nutch

Apache Nutch is an open-source web crawler written in Java. We extended Nutch into a focus crawler for this study. In order to achieve that, we first analyzed the original structure of the Nutch, which executes the following steps in each iteration:

- Injecting
- Fetching
- Parsing

- Updating

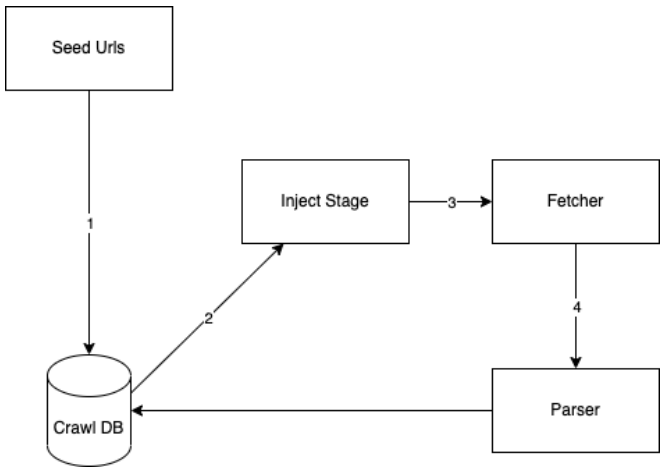

Figure 4.1: Workflow of General Nutch Crawler

The Nutch workflow can be explained as:

1 Adding the set of seed URLs to the crawl database in order to retrieve them later.

2 Getting top X URLs from the crawl database for traversal. Top X value is defined beforehand, which indicates the number of pages to be crawled on each iteration.

3 At this stage, injected URLs are ready to fetch. Fetcher crawls these URLs using multithreading in order to speed up this stage.

4 After all of the pages are fetched, parser starts to parse them. Some processing methods are applied here which are explained in detail in the Section 4.1.1.

4.1.1 Processing the Data

In this section, we briefly review the procedures applied in the parsing stage of Nutch. The web pages exhibit a high degree of variation in their content and style. In order to maintain consistency throughout our pipeline, we have to change the format of all the pages to fit a certain structure. In what follows, we briefly review the pre-processing

procedures applied in the parsing stage of Nutch. The procedures we followed to achieve this are:

- Decapitalizing the text
- Stop word elimination
- Stemming

4.1.1.1 Decapitalizing

Upper case and lower case characters are represented differently for machines, but they are easily discernable by humans. A person can see that "what" and "WhaT" are the same word. However, they are represented in different ASCII values. In most cases, the meaning of a sentence or a document is not affected by the case differences. Assume the following sentences:

```
A = the reasons of global warming.
B = The Reasons Of Global Warming.
```

Both of the sentences are the same. However, the Jaccard similarity between these two sentences are 0 as shown in the Example 4.1.

$$\begin{aligned} A \cap B &= \{\} \\ A \cap B &= \{\text{the, reasons, of, global, warming, The, Reasons, Of, Global, Warming}\} \\ J(A,B) &= \frac{0}{10} = 0.0 \end{aligned} \tag{4.1}$$

We can say that the second sentence is a title and the first sentence is not. The Jaccard similarity is not sufficient to anticipate the logical similarity between sentences when no preprocessing is used. In this study, we tackle this problem by lower casing every character on a document or a sentence.

4.1.1.2 Stop Word Elimination

According to the Oxford Dictionary, there are approximately 171,146 word on English that are currently in use [36]. The commonness of these words follow a long tail distribution: a large majority of these words are rare. However, some words appear in most of the sentences and they do not give a broad meaning to the sentence. These words are called stop words and they might cause misjudgements on our evaluations. Assume the following two sentences:

```
A = President of the USA is giving his speech to public.
B = He is called the secretary of department to get his ID.
```

These two words do not share a similar meaning, however if we apply Jaccard similarity (assuming we have lowered the cases), we will find that these two sentences are actually similar as shown in the Example 4.2

$$\begin{aligned} &\text{A} \cap \text{B} = \{\text{of, the, is, his, to}\} \\ &\text{A} \cap \text{B} = \{\text{president, of, the, USA, is, giving, his, speech, to, public, he, called,} \\ &\qquad \text{secretary, department, get, ID}\} \\ &\text{J(A,B)} = \frac{5}{16} = 0.31 \end{aligned} \tag{4.2}$$

A stop word list can be found on most NLP libraries or as plain text files on the web. In this study, we used Natural Language Toolkit [1] (NLTK) stop word list, which is given in the Table A.1.

4.1.1.3 Stemming

Eliminating stop words and lower casing the words solves most of the problems. There is another operation needs to be completed, which is stemming. Some words might share the same root or base, although their similarity on most of the metrics are

[1] https://www.nltk.org/

different. For example, “waiting”, “waited” and “waits” all have a common root of “wait”. If we apply Jaccard similarity to 2 sentences one of which contains “waited” and the other “waits”, we will find no similarity among them. However, in real life examples, they probably have similar meaning and we are losing that information. In order to avoid such situation we can apply stemming, which includes converting all the words to their corresponding stem words.

Stemming is a general linguistic problem, however first stemmer [37] is relatively new compared to the problem itself. There are many stemmers or stemming algorithms available. In this study, we have directly used the available libraries for this task which are: Java WordNet Library [2] (JWNL) and NLTK stemmer for Python.

4.2 Focused Crawler

The main idea of this study is to create a focused crawler that is successfully gathering pages in the context of the target topic and favoring different aspects of the target topic equally. The schema of the Nutch is given in the Figure 4.2 and main contribution to the system is that we extended the system by adding a classifier and the scoring function.

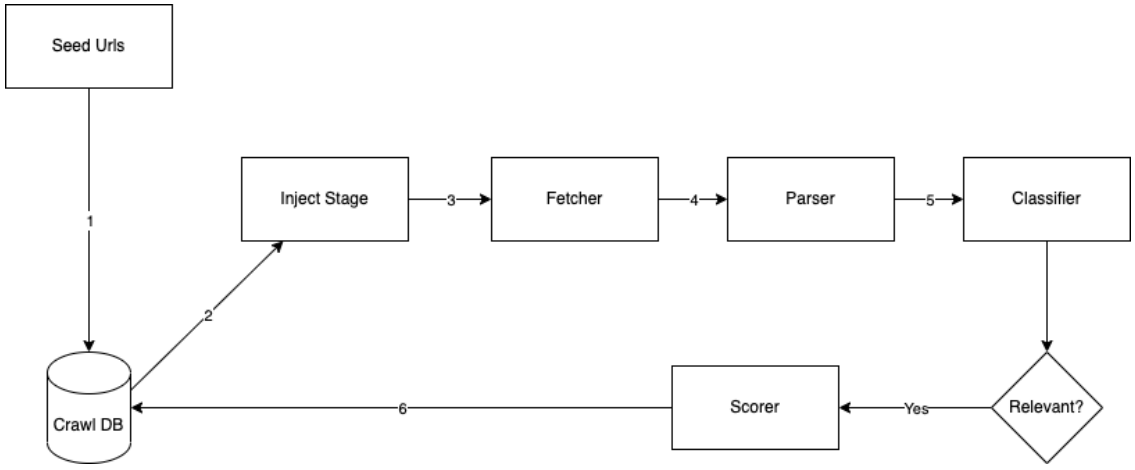

Figure 4.2: Workflow of Modified Nutch Focused Crawler

The new structure of our modified focused crawler is that after fetching and parsing a

[2] https://sourceforge.net/projects/jwordnet/

particular page, we send the content to our classifier which eliminates the pages that are not relevant to the target topic. After that, we pass the remaining scores to our scorer which scores the pages and the links inside the page. For each link inside the page, a weighted sum of original page score and the score that scorer give to the link is calculated. The scorer calculates the link score using the URL of the link, anchor text of the link and the text near link.

4.2.1 Classifier

The classifier is used for initial elimination process to remove the pages that are not relevant to the target topic from the list without using the scorer. It is a binary classifier that decides whether the page is relevant to the target topic or not.

4.2.1.1 Training Set

Since we want this crawler to be available for every target topic, we need a automatically working system. For this system we need a tool to create training set by just giving the context without need of a user judgement. Moreover, we can not find a data set for each target topic so we have to gather our own with the context of the target topic automatically.

In order to gather such data set, we've used our crawler, Google search engine and ODP [3] for gathering negative samples. The main idea is to reach the pages as in context of the target topic and similarly while finding pages that are distant from the target topic. For positive pages, we are providing a set of URLs that is taken from the Google's top X pages. The process of taking these URLs is to send the target topic as a query to the Google Search API and gathering the top pages that show up in results. For negative pages, we crawl the pages by checking the ODP categories. We take a set of irrelevant category contents that are selected by the human judges. An example of the ODP categories are listed below:

```
Arts/Animation/Anime
```

[3] https://dmoz-odp.org/

```
Arts/Art_History
Arts/Design/Fashion/
Arts/Literature
Arts/Music/
Arts/Performing_Arts
Business/Accounting/
Business/Aerospace_and_Defense
Business/Agriculture_and_Forestry
Business/Automotive/
Business/Chemicals/
Business/Construction_and_Maintenance
```

In order to comply with Google's API regulations, we used multiple account keys to gather the pages. We gather 1000 positive and 1000 negative pages (the values can be changed from the configuration file) from the Google Search API to create our data set.

On this training dataset, we employ Random Forest Classifier algorithm to train a model that will serve as an initial filter, i.e., to eliminate clearly irrelevant pages and keep others to be passed to the Scorer.

4.2.2 Modifying Nutch System

4.2.2.1 Adding the Classifier to the System

Nutch uses multithreaded fetchers and a database which contains the URLs that are fetched and needs to be fetched with their corresponding scores. We need to change the workflow of the system in order to merge our classifier to the system and apply our methodology fully. In order to do that, just before the scoring phase, at the parsing stage, we have added our classifier which checks whether the page is relevant or not. We have changed the parser by overriding the methods that sends the parsed content to the scorer. Before sending, we send the content to our classifier to check its relevance, relevant pages are scored and irrelevant pages are discarded.

4.2.2.2 Favoring Recently Extracted Links

After fetching a page, Nutch finds all the links inside the page and after scoring them, sends the links with their scores to its database. At the next inject stage, the URLs are sorted by their scores, regardless of their timestamp and the top N URLs are taken from the database to be fetched. However, we want to favor more recent URLs over older ones. For example, if a URL is found on the first iteration (i.e. a link inside the seed URL) and has not been fetched until iteration 10, we want to favor the URL found on the iteration 9 even if its score is lower than the older one. To achieve this, we have created a dummy database to apply our idea without changing the internal system too much. The scored links are sent to the dummy database instead of the original database. After that, instead of taking top N links from the database, we are taking top $\frac{N}{2}$ links from the original database and top $\frac{N}{2}$ links from the dummy database. Just before updating the database, we are merging dummy database into the original one and remove all the contents inside the dummy database. Thus, after each iteration, we return to our original state and gather half of the links from the older ones and other half from the recent ones.

4.2.3 Scoring System

The scoring system decides the relevance of the links inside the page to the target topic. Scoring the entire page content gives us insight about the relevance of the links inside the page, since a link to a target topic inside a relevant page has higher chance to be relevant to the target topic.

The important thing of scoring is to decide whether to fetch the link inside the page or not. In order to achieve that, we calculate the score of each link inside the page as shown in the Equation 4.3

$$\text{score} = \lambda * \text{score}(p, q) + (1 - \lambda) * \text{score}(l, q) \tag{4.3}$$

where:

- p : Current page
- q : Target topic
- l : Link to be scored
- λ : Constant, taken as 0.5

For the score function of the Equation 4.3, we tried different approaches. The simplest one is calculating the term frequency 3.1.1, however, in order to calculate the frequency using only target topic was not enough since we are trying to compare at least 500 words long file with 1 or 2 words topic similarity. To be more accurate, we need to expand the target topic with more meaningful words. However, expansion should be done for positive phrases of the target topic.

4.2.3.1 Selecting Positive Phrases

In order to expand the target topic for positive phrases, we used the data of Wikipedia by getting the content of the Wikipedia page of the target topic. We used a tool that takes every inlink from the Wikipedia page of the target topic and by manual evaluation, we are selecting the meaningful ones and thus creating the positive phrases of the topic.

We use these positive phrases in the link score calculation using the equation given in the Equation 4.3 which also considers the score of the link for the target topic. To calculate the score of the link we are checking following properties of the link:

- URL of the Link
- Anchor Text of the Link
- The text around the Link

The modified scoring function with link information above added is given in the Equation 4.4

$$\begin{aligned} \text{score} = \lambda * \text{score}(page) + (1 - \lambda) * (\text{score}(URL) \\ + \text{score}(anchor) + \text{score}(text))/3 \end{aligned} \tag{4.4}$$

The calculation gives the actual score of that link to be stored inside the database for next iterations of the crawl operation. For the Equation 4.4 we selected the scoring method as $tf\text{-}idf$ 3.1.1. To this end, we need to create a $tf\text{-}idf$ vector of both target topic and the current page in order to calculate the $tf\text{-}idf$ for that page and similarly, for the link, anchor and the text around the link.

4.2.3.2 TFIDF for Scoring

To start the $tf\text{-}idf$ calculation, we will not use the negative phrases since we are trying to find the similarity between two $tf\text{-}idf$ vectors. We need to create one vector that contains both the target topic term and the positive phrases (i.e. expanded target topic). Thus we merged these terms into a list of words. After that, we calculated the IDF score of each term using the IDF values taken from CLUEWEB-B [4] which contains more than 50 million pages. We do not need to check the IDF values of every single word on the pages but only those that appear inside the expanded target topic, since finding the IDF of every word inside the page will be costly and since we are calculating the cosine similarity, their effect on the result will be 0.

$$\begin{aligned} \text{linkScore} &= \big(tfidf(link, topic) + tfidf(anchor, topic) + tfidf(text, topic)\big)/3 \\ \text{score} &= \lambda * tfidf(page, topic) + (1 - \lambda) * \text{linkScore} \end{aligned} \tag{4.5}$$

After implementing $tf\text{-}idf$, we also implemented BM25 to score calculation as explained in the Section 3.1.2.

[4] https://lemurproject.org/clueweb09/

4.2.3.3 Word Embeddings for Crawler

As a part of the scoring system, we adapted word embeddings as a scoring function, just like $tf\text{-}idf$. To achieve this, we need the word embeddings for both expanded target topic phrases and the words inside the page. However, in order to compare the page similarity using word embeddings, we need to create a document embedding vector for the page and another vector for the target topic phrases. In order to convert word embeddings into documents embeddings there are some methods such as Ulu et al. [29] that we can follow which are for each dimension:

- Taking the minimum among all embeddings
- Taking the maximum among all embeddings
- Taking the average of the embeddings

In this work, we used the first option which is taking the minimum among the word embeddings for each dimension to represent the document using its word embeddings. For each word inside the target topic and positive phrases we get the word embeddings from the GloVe and by taking minimum of each dimension we convert the target topic and positive phrase embeddings into a document representation. Moreover for each page we fetched and parsed, we use the parsed content to get word embeddings of them and similar to the target topic and positive phrases we create a document representation. The process of getting document similarity is explained in Algorithm 5, which takes the page (D), the target topic (T) and positive phrases (P) as input and returns the similarity (R) using cosine as distance metric. We load the GloVe word embeddings to the Nutch system beforehand in order to get the word embeddings for each new page on the fly.

4.2.3.4 Sentence Transformers for Crawler

For our focused crawler, we apply the sentence transformers as a scoring function, just like we employ word embeddings. However, instead of converting word embeddings into document embeddings we convert sentence embeddings into document

embeddings. Sentence transformers have a limitation of 512 words per sentence, thus we might not represent a document as one sentence. For each aspect and the target topic we acquire a sentence embedding since the size of aspect and aspect phrases or topic and topic phrases are small. However, since a page that we fetch is generally longer than 512 words, we need more than one sentence embedding. The approach of creating document embedding using sentence embedding are similar to the word embedding to document embedding conversion. As explained in the previous section, we follow the practice of Ulu et al. [29] to get the document embedding from sentence embeddings.

Finally, we calculate cosine similarity between the document embedding of the page and the sentence embedding of the target topic and aspects and between the document embedding of the page and the sentence embedding of each aspect.

Algorithm 5: Calculating Similarity using Word Embeddings

Data: D, T, P

Result: R

u $\leftarrow \emptyset$

v $\leftarrow \emptyset$

foreach $w \in \mathcal{D}$ **do**
  push getWE (w) to u

foreach $w \in \mathcal{T} \cup P$ **do**
  push getWE (w) to v

$p \leftarrow$ calculateMin(u)

$q \leftarrow$ calculateMin(v)

$R \leftarrow$ cosine(p, q)

4.3 Diversification-Aware Focused Crawler

A target topic might have more than one meaning and searching the target topic directly might lead us to miss those meanings. For example, the word apple might be both a type of fruit and the company. Moreover, a word might have some aspects (i.e. aspects) which are slightly different from the general meaning of the word. For example, if we consider the coronavirus, there are lots of aspects behind the term,

symptoms or research towards coronavirus can be considered as aspects of coronavirus.

We assume that the aspects of the target topic are available to us and we adopt a strategy inspired from explicit search diversification methods (namely, CombSumDiv based approach introduced in [6]) To achieve that, we have to modify our current scoring system to score the links by not only the target topic but also the aspects. However, in order to change the scoring system, first we need the aspect phrases just like the target topic phrases.

4.3.1 Aspect Phrases

For each aspect, we need a set of phrases that explain the aspect itself. To achieve that, we are using Google Suggestions API, which provides suggestions just like the suggestions in Google Search. Sending the aspect to the Google Suggestions API and gathering the suggestions will provide a set of words that can be used as phrases. However, aspect word might not be directly related to the meaning of the target topic. For example, coronavirus has an aspect of research, while the word research itself is too general. Thus, before calling the API, we are merging the aspect to the target topic and gather the suggestions from Google. Examples of such suggestions are given in the Figure 4.3 and Figure 4.4

In order to gather more suggestions, we increased the dimension of the suggestions by adding every letter from alphabet at the end of the target topic. For example, we sent “coronavirus research x” to the API where x ranged from “a” to “z”. Thus, for each aspect, we gather 150-400 suggestions which creates 200-500 new words for each aspect.

Figure 4.3: Google Suggestions for Coronavirus Research

Figure 4.4: Google Suggestions for Coronavirus Symptoms

4.3.2 Scoring for the Diversification-Aware Focused Crawler

After gathering aspects, we calculate score of the link. For this purpose, we use CombSum [38], since its power for explicit diversification has been demonstrated as CombSumDiv [6]. The formula for the CombSumDiv is given in the Equation 4.6

$$S(q,d) = (1-\lambda)P(d|q) + \lambda \sum_{q_i \in T} P(q_i|q)P(d|q_i) \tag{4.6}$$

Up to this point we only expanded the target topic with aspects without applying diversification. In order to create a fair and diversified crawler, we consider the aspect relevance of the pages that we gathered. The λ value in the CombSumDiv Equation 4.6 is the same for all aspects, so we are not favoring any aspect over other. However, if we change the weights, we can favor less fetched ones over the more appeared ones, creating diversification. In order to achieve that, in a given iteration, we are calculating and storing the aspect score of each gathered page for each aspect. In the next iteration, we are using the previous aspect information to lower the impact of the most fetched aspects and improve the impact of the least fetched aspects to create fairness among all the aspects. To achieve that, we are using the formula given in the Equation 4.7.

$$\text{aspectWeight(x)} = (\sum_{a \in A}(score(a)) - score(x))/\sum_{a \in A}(score(a) \tag{4.7}$$

where A is all the aspects of the target topic and x $\in$ A is a particular aspect..

Finally, by changing the weight of the aspects on each iteration we are applying a favoring mechanism to be fair on every aspect and create diversification on our focused crawler.

We illustrate our method on a toy scenario as follows. Assume that we have a target topic of T and 3 aspects of the target topic, A_1, A_2, A_3 and in the first iteration we downloaded 3 documents, namely D_1, D_2 and D_3. Each document has 2 links inside them and the links are:

$D_1 \rightarrow D_5, D_8$

$D_2 \rightarrow D_6, D_7$

$D_3 \rightarrow D_4, D_9$

The Table 4.1 shows for each document the Cosine similarity score of target topic and each aspect, which may be calculated using distance of $tf\text{-}idf$ vectors.

	Target Topic	A1	A2	A3
D_1	0.7	0.8	0.1	0.1
D_2	0.6	0.2	0.9	0.1
D_3	0.5	0.1	0.2	0.9

Table 4.1: Example: Document to Topic and Aspect Relevances

We calculate the CombSumDiv [6] Score for all 3 documents as in the Equation 4.8 assuming the λ value is equal to 0.5. Note that, at the first crawling iteration, we use the same weight value, 1/3, for each aspect, as we have no prior information at this point.

$$\begin{aligned} D_1 &= 0.5 * 0.7 + 0.5 * (\frac{0.8}{3} + \frac{0.1}{3} + \frac{0.1}{3}) = 0.516 \\ D_2 &= 0.5 * 0.6 + 0.5 * (\frac{0.2}{3} + \frac{0.9}{3} + \frac{0.1}{3}) = 0.500 \\ D_3 &= 0.5 * 0.5 + 0.5 * (\frac{0.1}{3} + \frac{0.2}{3} + \frac{0.9}{3}) = 0.450 \end{aligned} \tag{4.8}$$

The highest scored document is D_1, thus we fetch the links inside the D_1 (for simplicity we are not using link score, as we normally apply like shown in Equation 4.5), namely D_5 and D_8.

After fetching D_5 and D_8 assume that the Table 4.2 shows the relevance table just before the second iteration.

Now, we can see that there is no equal Cosine similarity score in the aspects. The aspect A_3 is covered the most while the aspect A_2 has the least Cosine similarity score. Thus, we can recalculate the weights of each using the formula in the Equation 4.7 we can find the new weights given in the Equation 4.9.

	Target Topic	A1	A2	A3
D_1	0.7	0.8	0.1	0.1
D_2	0.6	0.2	0.9	0.1
D_3	0.5	0.1	0.2	0.9
D_5	0.4	0.7	0.3	0.5
D_8	0.6	0.4	0.2	0.8
Total		2.2	1.7	2.4

Table 4.2: Example: Document to Topic and Aspect Relevances

$$\begin{aligned} AspectWeight(A_1) &= (6.3 - 2.2)/6.3 = 0.65 \\ AspectWeight(A_2) &= (6.3 - 1.7)/6.3 = 0.73 \\ AspectWeight(A_3) &= (6.3 - 2.4)/6.3 = 0.62 \end{aligned} \tag{4.9}$$

Thus, in the third iteration, our method will be favoring the A_2 by increasing its weight and reducing the effect of other aspects on our calculation. The CombSumDiv scores of the pages before the third iteration is given in the Equation 4.10.

$$\begin{aligned} D_2 &= 0.5 * 0.6 + 0.5 * (0.2 * 0.65 + 0.9 * 0.73 + 0.1 * 0.62) = 0.724 \\ D_3 &= 0.5 * 0.5 + 0.5 * (0.1 * 0.65 + 0.2 * 0.73 + 0.9 * 0.62) = 0.634 \\ D_5 &= 0.5 * 0.4 + 0.5 * (0.7 * 0.65 + 0.3 * 0.73 + 0.5 * 0.62) = 0.692 \\ D_8 &= 0.5 * 0.6 + 0.5 * (0.4 * 0.65 + 0.2 * 0.73 + 0.8 * 0.62) = 0.751 \end{aligned} \tag{4.10}$$

Since the D_8 has the highest score among all 4 documents, we are fetching the documents inside the D_8 for the third iteration. This example is extended in our system by adding link scoring 4.5.

4.4 Experiments

For the experimental setup, we selected the target topic of "Covid-19" because of the ongoing pandemic. In our experiments, we compare the diversification aware focused

crawler and the focused crawler uses all the setup without considering a diverse result. Moreover, we compare the effect of the sentence transformers on the diversification aware focused crawler. First, we will explain the effect of the diversification, then we will explain the effect of the word embeddings and sentence transformers.

For both setups, we use the same classifier as explained in the Section 4.2.1 and for the classifier training set, as explained in the Section 4.2.1.1, we gather 100 positive and 100 pages from the Google Search API. Positive pages are related to the Covid-19 and directly taken from the Google. Negative pages are irrelevant to the Covid-19 as selected from the ODP categories.

4.4.1 Diversification on Focused Crawler

For this setup, we use $tf\text{-}idf$ vectors in order to score pages and links. In order to score a page or link, we need the following:

- Target topic, which is set to Covid-19.
- Phrases that are relevant to the target topic.
- Aspects of the target topic.
- Phrases that are relevant to each aspect.

The aspects of the topic Covid 19 are determined manually as follows:

- Vaccine
- Research
- Symptom
- Economic (impact)
- Prevention

Moreover, for both target topic and its aspects we gather the phrases that are relevant to them. For the target topic we are using the Wikipedia as the dataset as explained in

the Section 4.2.3.1. For the target topic Covid-19, we have 236 terms as topic phrases. For aspects, we gather the phrases using Google Suggestions API as explained in the Section 4.3.1. The number of terms for each aspect is given below:

- Vaccine: 367 terms
- Research: 134 terms
- Symptom: 321 terms
- Economic: 105 terms
- Prevention: 142 terms

Moreover, the aspect phrases Economic Impact, Symptoms, Research, Vaccine and Prevention are given in the Tables A.2, A.3, A.5, A.4, A.6 respectively.

For this experiment we compare 2 different focused crawlers. First one does not have any information about the aspects and their phrases, uses just the target topic and its positive phrases. Second crawler is diversification aware focused crawler, which considers diversification and has the information about both target topic and aspects as explained in the Section 4.3. In order to simplify the namings, we name these crawlers as follows: No diversification (NoDiv) crawler and Diversification aware (DivAware) crawler.

4.4.1.1 Deciding IDF Values

Calculation of $tf\text{-}idf$ requires idf value for each word. For this purpose, we need a dataset to decide on the idf values for the words. In particular, we are using CLUEWEB-B as explained in the Section 4.2.3.2. The dataset includes more than 50 million documents and the idf file is more than 50 GB. Using such a large file on the on the fly system is going to slow everything, however, we only need a portion of the data inside the file. Since, we are calculating the $tf\text{-}idf$ values for the target topic, aspects and their phrases the other words that are not part of this set will always give zero on the calculation. Thus, we shortened the file to a small portion of words which only contains target topic, aspects and phrases.

For each focused crawler and each batch of crawling, we are calculating the following in order to compare the success of each crawler.

- Cosine similarity between the $tf\text{-}idf$ vectors for the topic aspects and page.
- Number of totally irrelevant pages, i.e., those with zero Cosine similarity to the aspect.
- The weights given to each aspect.

Although the no diversification crawler does not have the aspect information on scoring, we apply two calculations, one for the actual scoring and the other for the evaluation purposes. We compute the aspect scores on the crawler, however, do not reflect the score to the actual computation.

For each setup, we calculate the Cosine similarity scores using the idf set and score both the page itself and its links in order to both decide whether the page is relevant and pass the score to the link in order to sort them. For a link, we consider the url, the anchor text of the link and the text surrounding the link.

4.4.2 Seed URLs

As a starting point, we define 20 seed URLs for the crawler. These URLs are selected from Google Search API and injected to the crawler at the initial fetching stage. The seed URLs are given below in A.1.

4.4.3 Comparison of the Crawlers

In this setup, we apply word embeddings and sentence embeddings as explained in Section 4.2.3.3 and Section 4.2.3.4 to the scoring function of the diversification aware focused crawler. Thus, we compare the performances of four different focused crawlers.

We calculate the Cosine similarity score in the no diversification focused crawler and diversification aware focused crawler. While the diversification aware focused

crawler also has the aspect information, we only calculate the Cosine similarity score of the page content for the sake of fair evaluation. Similarly, although the score calculations of word embedding applied diversification aware focused crawler and the sentence embedding applied one are different than $tf\text{-}idf$, we also calculate the Cosine similarity scores to compare all 4 crawlers.

In our experiments, we report the number of pages that are said to be relevant to each aspect, i.e., have a Cosine similarity score greater than a threshold. We consider these pages to relevant to that aspect. The threshold is set to 0.1 for this setup and the cumulative number of pages after each iteration are given for each crawler are given in the Tables B.1, B.2, B.4, B.3. For simplicity, we will examine the batches 20 and 45 for these experiments. For each crawler, the cumulative number of relevant pages at batch 20 is given in the Table 4.3 and at batch 45 is given in the Table 4.4.

	Vaccine	Symptoms	Prevention	Research	Economic
NoDiv	1351	4897	2177	2053	2022
DivAware	1658	5728	2569	2444	2452
DivAware+WE	1324	4869	2067	1966	1953
DivAware+SBERT	216	811	282	271	270

Table 4.3: Number of Relevant Pages for Each Aspect at Batch 20

	Vaccine	Symptoms	Prevention	Research	Economic
NoDiv	3302	12193	5015	4733	4677
DivAware	3644	13688	5531	5240	5327
DivAware+WE	3403	12333	5180	4982	4938
DivAware+SBERT	377	1801	506	486	483

Table 4.4: Number of Relevant Pages for Each Aspect at Batch 45

Each crawler is executed online for 45 batches. The number of pages downloaded at the end of the batch 45 are 33557 pages for no diversification crawler, 32546 pages for SBERT, 32972 pages for word embedding crawler and 33251 pages for diversification aware crawler. For each crawler, we compute the precision for each aspect of

the target topic and get the average of the latter values, which is equivalent to computing the well-known intent-aware precision (P-IA) [39] metric in the diversification literature. The P-IA values are reported in Table 4.5.

	P-IA
NoDiv	0.178
DivAware	**0.201**
DivAware+WE	0.187
DivAware+SBERT	0.022

Table 4.5: Precision-IA for each crawler setup at Batch 45

We see that the precision of the diversification aware focused crawler is better than all 3 crawlers. Moreover, the sentence transformers performs the worst among all crawlers. Except sentence transformers, we can say that precision is increasing if we apply diversification on the system. The Cosine similarity score of each aspect for each crawler at the last batch 45, given in the Table 4.6.

	Vaccine	Symptoms	Prevention	Research	Economic
NoDiv	0.110	0.407	0.167	0.158	0.156
DivAware	0.109	0.409	0.165	0.156	0.159
DivAware+WE	0.110	0.399	0.167	0.161	0.160
DivAware+SBERT	0.109	0.409	0.165	0.156	0.159

Table 4.6: Aspect Cosine Similarity Score Percentages at Batch 45

We calculate the standard deviation for each crawler which shows how the Cosine similarity score of each aspect differ from each other. On an ideal system, the standard deviation of the Cosine similarity scores of aspects on a crawler should be 0. The standard deviation results are given in the Table 4.7

We can say that no diversification crawler and diversification aware crawler is performing similarly and word embeddings has a better impact on the diversification results.

	STDEV
NoDiv	0.118
DivAware	0.119
DivAware+WE	**0.114**
DivAware+SBERT	0.164

Table 4.7: Standard Deviation of each crawler setup at Batch 45

Up to now, we presented our findings for the batch 45 which is the last batch of the crawling.

However, it would be useful to provide an insight for the trend in the earlier batches. Therefore, we also report the performance of all crawlers almost at the middle of our crawling cycle, namely, at the batch 20. In this case, the number of fetched pages for the no diversification crawler is 11809 while 12237 for diversification aware crawler, 11688 for word embeddings and 11456 for SBERT crawler. The P-IA scores for each crawler is given in the Table 4.8.

	P-IA
NoDiv	0.177
DivAware	**0.211**
DivAware+WE	0.182
DivAware+SBERT	0.032

Table 4.8: Precision-IA for each crawler setup at Batch 20

The results did not changed significantly at the batch 20, although there is a sharp increase in terms of precision. We also report the Cosine similarity scores and standard deviation at the batch 20 in Tables 4.9 and 4.10 respectively.

From Tables **??** and 4.10 , we see that this time the diversification aware focused crawler performed the best among all 4. In both of the batches 20 and 45 the no diversification crawler performed on average. Our diversification aware approaches (DivAware and DivAware+WE) performed the best and sentence embeddings per-

	Vaccine	Symptoms	Prevention	Research	Economic
NoDiv	0.110	0.407	0.167	0.158	0.156
DivAware	0.109	0.409	0.165	0.156	0.159
DivAware+WE	0.110	0.399	0.167	0.161	0.160
DivAware+SBERT	0.109	0.409	0.165	0.156	0.159

Table 4.9: Aspect Cosine Similarity Scores at Batch 20

	STDEV
NoDiv	0.115
DivAware	**0.110**
DivAware+WE	0.117
DivAware+SBERT	0.116

Table 4.10: Standard Deviation for each crawler setup at Batch 20

formed the worst in all two experiments.

Although SBERT fails to improve the results for the metrics reported up to this point, there is one case where it seems to be useful. This time we calculated the percentage of pages with zero similarity score to each aspect and report the cumulative number after each batch. For simplicity, again, we only cover the number of those pages at the Batch 45 for each crawler and each aspect is given in the Table 4.11 and the same approach for Batch 20 is given in the Table 4.12.

	Vaccine	Symptoms	Prevention	Research	Economic
NoDiv	0.542	0.543	0.548	0.549	0.549
DivAware	0.497	0.498	0.502	0.503	0.503
DivAware+WE	0.515	0.515	0.52	0.52	0.52
DivAware+SBERT	**0.365**	**0.366**	**0.372**	**0.386**	**0.385**

Table 4.11: Percentage of Zero Score Aspects at Batch 45

	Vaccine	Symptoms	Prevention	Research	Economic
NoDiv	0.542	0.543	0.548	0.549	0.549
DivAware	0.505	0.506	0.511	0.511	0.51
DivAware+WE	0.512	0.513	0.516	0.516	0.516
DivAware+SBERT	**0.417**	**0.416**	**0.424**	**0.435**	**0.437**

Table 4.12: Percentage of Zero Score Aspects at Batch 45

In both results, the SBERT outperforms all 3 methods. Although the diversification aware focused crawler using SBERT is giving worse in terms of P-IA and higher Cosine similarity scores, SBERT seems to be more successful in avoiding pages that are completely irrelevant, which is a promising finding that we plan to investigate deeper in our future work. Moreover, our diversification aware focused crawler using $tf\text{-}idf$ scoring is the second best in both cases while no diversification crawler is tend to fetch more pages with zero similarity score to target topic and its aspects than all the other crawlers.

Overall, we show that our DivAware crawler outperforms NoDiv crawler in terms of P-IA or Standard deviation. While tends are not conclusive, we also show that word embeddings further have the potential to improve the DivAware crawler. In contrast, we observed that sentence embeddings obtained via SBERT did not help improving performance. We think that SBERT may benefit from pre-training with a larger training set, so that embeddings can be more helpful to distinguish different aspects of a target topic.

CHAPTER 5

DIVERSIFICATION FOR TWEET SEARCH RESULTS

Twitter contains billions of tweets where any of these tweets can be a result to a search query. Our aim is to create a diverse search result for a given search query. This is a more challenging task, as tweets are much shorter than typical web pages over which diversification methods are typically applied. There are only a few studies that investigated the diversification over Tweet search results [12], which revealed that implicit methods perform better than explicit ones. In this chapter, we investigate the performance of both types of methods where similarity scores computed using word and sentence embeddings.

In what follows, we first describe how tweets are pre-processed and then in Section 5.2 we discuss their representation using embedding vectors for relevance ranking. Finally, in Section 5.3, we provide an in depth evaluation of div performance using embedding vectors.

5.1 Pre-processing Tweets

Similar to the processing of the pages in focused crawler, we have to apply some pre-processing to the tweets in order to create a stable structure over all the dataset. We are applying all the procedures, explained in the Secion 4.1.1, are applied on focused crawler also to the tweets. Moreover, tweets has some special characters and words that are not considered as actual words and they reduce the accuracy of a vector representation. One of them is the word “RT” which means retweet. Although it shows that this tweet is a mention to another tweet, while creating a vector representation the word “RT” should better not have any meaning. Thus, we are eliminating all the

"RT" words inside a tweet. Finally, a user that is posting a tweet is given with the character "@", the username is not important for our vector representation thus we are again removing the character "@" and the username that comes after it.

Twitter contains excessive amount of duplicate or near duplicate tweets. In order to present a good result set to user, we need to consider two things:

- Initial Relevance Ranking
- Diversification of the initial ranking

The main idea is to rank these tweets on the context of the query while trying to achieve a diverse result set. This idea is similar to the idea behind the work of Ulu and Altingovde [29] which is using diversification methods along with embeddings in order to rank a set of documents by providing diverse results. However, the main difference we are using more shorter text while they are using long web pages.

5.2 Ranking

In order to create a ranking for each query, we need to sort the tweets by their similarity to the query. There are some similarity metrics such as Cosine Similarity (Subsection 3.3.1), Jaccard Similarity (Subsection 3.3.2) and Word Mover's Distance (Subsection 3.3.3).

For Jaccard similarity, we do not need any vector representation. We are directly taking the words inside the tweet and the query to compute the similarity. Using that information we are reranking all the tweets.

For Cosine Similarity and Word Mover's Distance, we need a vector representation for a given tweet. For that purpose we need one of the:

- TF vector
- tf-idf vector
- Word or Document representation

As long as we have vector representation we can compute the cosine similarity or calculate word mover's distance. Fortunately, tweets are shorter documents and can be considered as a sentence since it generally contains 1 to 3 sentences. Moreover, we can directly use SBert 3.1.4.2 in order to find sentence representation for a given tweet. Representing the tweets as the set of documents using $tf\text{-}idf$ or SBert allows us to use them in Cosine Similarity or Word Mover's Distance calculations. However, in order to use word embeddings to represent tweets, there are some methods that we can follow for such tasks, namely averaging word embeddings or getting maximum or minimum on each dimension of the word embeddings which are all explained and experimented in the work of De Boom et al. [30].

5.2.1 Averaging Word Embeddings

The sentence or document are represented by words as a sentence embedding can be represented by word embeddings. To achieve that, we can take the average of the embeddings of the words inside the sentence for each dimension to find an accurate corresponding sentence representation. Its application can be seen in the work of Kenter et al. [40] as they are also adding weight parameters to embedding values over a Siamese Network.

$$SE(j) = \frac{\sum_{i=0}^{N}(WE_i(j))}{N} \tag{5.1}$$

where:

- N: number of words inside the sentence.
- SE: sentence embedding
- WE: word embedding

The formula of the averaging word embeddings can be seen in the Equation 5.1 To understand the operations assume the basic words and word representations given in Example 5.2.

```
Cosine similarity used on vectors.
```

$$
\begin{aligned}
\text{Cosine} &= [0.6, 0.8, 1.0, 0.3, 0.5] \\
\text{similarity} &= [0.1, 0.8, 0.9, 0.4, 0.6] \\
\text{used} &= [0.3, 0.4, 0.4, 0.4, 0.4] \\
\text{vectors} &= [0.6, 0.9, 0.1, 0.8, 0.7] \\
\text{sentence} &= [0.4, 0.725, 0.6, 0.475, 0.55] \\
\text{sentence} &= [0.5, 0.7, 0.612, 0.456, 0.524]
\end{aligned}
\tag{5.2}
$$

5.2.2 Using Maximum and Minimum

Very similar to the averaging, we are getting the maximum and minimum of each dimension in order to convert word embeddings into sentence embeddings. If we modify the Equation 5.1 we can reach the following Equation 5.3 which is converting a word embedding into a sentence embedding using maximum function.

$$SE(j) = \max(WE_i(j)) \tag{5.3}$$

Similarly, if we change the $\max$ function with the $\min$ function we again can get the sentence embedding.

Another way of getting sentence embeddings is to use sentence transformers [11] which gives sentence representation of a given input as explained in the Section 3.1.4.2.

The main idea of the ranking is to find similarity between a given query (i.e. query) and a tweet (i.e. sentence) using the sentence representations with the similarity measures. However, another aspect of the task is creating diverse result set.

5.3 Diversification of Tweet Search Results Using Embeddings

In this thesis, we combine ranking and diversification methods to create a both accurate and diverse result set. To achieve this, the main idea is to combine the true

similarity metric with its corresponding representation. The step by stem explanation of the methodology is given below:

1 Obtain a candidate ranking of tweets (by applying methods in Section 5.2)

2 For diversification, obtain representation of the query (and its aspects, if available) and the tweets in the candidate ranking based on sentence embeddings (by applying methods in Section 3.1.4)

3 Apply one of the diversification algorthims (namely, MMR, Sy or xQuAD) discussed in Section 3.2.

Some diversification methods are more suitable to a vector representation method and to a similarity metric. In this study, we tried to apply most of the combinations in order to find out which one is the best. The combination of the diversification methods, vector representation type and similarity metric is given in the Table 5.1.

Diversification Method	Vector Representation	Similarity Metric
MMR	FastText	Word Mover's Distance
MMR	Sentence Transformers	Cosine
Sy	FastText	Word Mover's Distance
Sy	Sentence Transformers	Cosine
xQuad	FastText	Word Mover's Distance
xQuad	Sentence Transformers	Cosine

Table 5.1: Combination of Method and Metrics

Moreover, for xQuad, we use the subqueries of each query in order to take aspects into account. There are approximately 8 aspect for each query on the dataset. The aspects of the query "hillary clinton resign" is given in below:

```
positive aspects of Clinton's reign
who follows Clinton as secretary of state
what may be next for Hillary Clinton
details of resignation
```

```
political positions Hillary Clinton
details during time as secretary of state
```

Similar to query processing operations on ranking and diversification, we are applying the same process for each aspect. The vector representation of each aspect is created in order to calculate the similarity of the tweet to the aspect. Each different approach applied on xQuad (as given in the Table 5.1) is also applied to the aspect-tweet comparison.

For each method, we calculate tweet to tweet similarity, using two dimensional array and store the similarity of each tweet to another. Moreover, on the cases that we apply sentence transformers, we fine-tune the pretrained BERT model in order to get better results. The details of the fine-tuning is explained in the Section 5.4.3.

5.4 Experiments

5.4.1 Dataset

For this study, we will use a dataset specifically designed for twitter diversification which is Tweets2013. We used the same setup as Ozsoy et al. [12] On the setup, we have:

- 50 query
- Top 100 tweet for each query (83 on average)
- Top 500 tweet for each query (403 on average)

As they did, we removed queries with ids 7, 8, 14, 43, 46 and 47 from the top 100 tweets and queries with ids 3, 5, 7, 9, 14, 28, 37, 43, 45, 46 and 47 from the top 500 tweets since their lack of relevant tweets.

A tweet has the following properties:

- tweetId

- timestamp
- content
- URL

Similarly, a query has the following properties:

- queryId
- queryText
- timestamp

Since the dataset is specifically created for search result diversification, there is a ground truth table for each query and tweet relevances.

5.4.2 Experimental Setup

Using the Twitter2013 dataset, explained in 5.4.1, we are trying to achieve a diversified results. To compare our results we are using Ndeval [1] which is a program that is used for diversity measuring. The tool is measuring some metrics that are: ERR-IA, nERR-IA, α-DCG, α-nDCG, NRBP, nNRBP, MAP-IA, P-IA and strec.

For each method we are trying to calculate these values using the tool for each query. An example result, for some queries, is given in the Figure 5.2

It can be seen that for different queries the results are highly differs. For example, on the Table 5.2 query 3 has higher α-nDCG value which shows that the results are diverse, however, for query 13 this value is much smaller. Thus, we can not rely on one query, we are using the average results over all queries in order to find more accurate results. For each method on and setup we are using in this study, we are using the average result for all queries as our resulting metric. Moreover, we will consider α-nDCG@10, α-nDCG@20, P-IA@10, P-IA@20, strec@10 and strec@20 as evaluation metrics.

[1] `https://trec.nist.gov/data/web/10/ndeval.c`

query	alpha-nDCG@10	alpha-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
1	0.185117	0.258361	0.066667	0.050000	0.333333	0.666667
2	0.249394	0.223563	0.037500	0.018750	0.375000	0.375000
3	0.645926	0.646851	0.080000	0.050000	0.600000	0.600000
4	0.000000	0.201473	0.000000	0.050000	0.000000	0.400000
6	0.203367	0.307841	0.050000	0.050000	0.500000	0.750000
10	0.373040	0.363358	0.062500	0.043750	0.375000	0.375000
12	0.271388	0.285282	0.062500	0.050000	0.375000	0.375000
13	0.071342	0.079267	0.010000	0.010000	0.100000	0.100000
15	0.510305	0.471262	0.133333	0.075000	0.500000	0.500000

Table 5.2: Ndeval Output for Twitter Diversification

For each setup we gather top 30 tweets for a given query and use qrels file (which indicates query relevance judgements) as our ground truth file. The qrels file is included in the dataset and we are directly using it on ndeval evaluation without additional judgement.

We will explain the setup, the parameters and the results of each diversification method one by one and then compare them with the original results from the work of Ozsoy et al. [12].

5.4.3 Fine-Tuning

Sentence transformers provide more than 10 pretrained models on their work. We selected two of these models and used on our experiments. The pretrained models that we used on this study is given below:

- all-MiniLM-L6-v2
- all-mpnet-base-v2

Moreover, we fine-tuned these models using the Tweets 2013 (Internet Archive) [41] dataset, which contains similar structure of tweets with our own data. There are

252713133 tweets and 60 queries on the dataset which are different from our original dataset. We create tweet pairs using the relevance information on the dataset for each query and label each pair with a relevance percentage. We use the new tweet pairs which contains 18000 tweet pairs to fine tune the pretrained models above.

5.4.4 MMR

For the MMR experiments, we use the system that is provided by the work of [12] and add sentence transformers and FastText to the system. We try different lambda values for MMR on each experiment and select the best result among them. An example result of a lambda value is given below for the metrics that are explained in the Section 5.4.2, namely, α-nDCG@10, α-nDCG@20, P-IA@10, P-IA@20, strec@10 and strec@20. These metrics and setup are same for all other methods on this study. The Table 5.3 shows the results of an example MMR run. Each row defines a different query and the last row is the average results.

As in the Table 5.3, we calculate the evaluation metrics, using Ndeval, for each query. Since we apply the experiments for multiple λ values, to select the best setup, we take the last row, which is the average value for our queries. You can see the average results for different λ values for the same setup in the Table 5.4.

query	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
1	0.448	0.490	0.100	0.067	0.500	0.667
2	0.000	0.066	0.000	0.013	0.000	0.250
3	0.458	0.470	0.060	0.040	0.600	0.600
4	0.406	0.405	0.080	0.060	0.200	0.200
6	0.362	0.362	0.100	0.050	0.750	0.750
10	0.134	0.182	0.025	0.031	0.250	0.375
11	0.210	0.220	0.040	0.030	0.400	0.400
12	0.000	0.075	0.000	0.013	0.000	0.250
13	0.450	0.393	0.050	0.030	0.200	0.200
15	0.389	0.370	0.050	0.033	0.333	0.333

Continued on next page

Table 5.3 – *Continued from previous page*

query	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
16	0.144	0.190	0.027	0.041	0.182	0.273
17	0.237	0.249	0.023	0.031	0.231	0.231
18	0.480	0.494	0.057	0.079	0.571	0.571
19	0.542	0.568	0.150	0.150	0.500	0.500
20	0.000	0.078	0.000	0.020	0.000	0.200
21	0.000	0.059	0.000	0.008	0.000	0.105
24	0.177	0.265	0.055	0.068	0.273	0.455
26	0.568	0.612	0.090	0.075	0.500	0.700
27	0.665	0.663	0.250	0.138	1.000	1.000
29	0.158	0.162	0.100	0.075	0.250	0.250
30	0.368	0.373	0.100	0.079	0.429	0.429
31	0.209	0.272	0.025	0.025	0.250	0.500
32	0.243	0.239	0.040	0.040	0.200	0.200
33	0.377	0.457	0.140	0.120	0.800	1.000
34	0.447	0.510	0.100	0.090	0.600	0.800
36	0.137	0.157	0.044	0.044	0.222	0.333
37	0.386	0.493	0.070	0.060	0.400	0.700
38	0.238	0.255	0.033	0.028	0.333	0.444
39	0.596	0.712	0.160	0.120	0.600	0.800
40	0.216	0.424	0.075	0.150	0.500	1.000
41	0.064	0.123	0.014	0.021	0.143	0.286
42	0.260	0.239	0.044	0.044	0.333	0.333
44	0.375	0.412	0.055	0.055	0.455	0.545
45	0.000	0.189	0.000	0.030	0.000	0.600
48	0.466	0.542	0.083	0.083	0.667	0.833
49	0.404	0.416	0.100	0.060	0.600	0.600
50	0.432	0.434	0.067	0.042	0.500	0.667
amean	0.298	0.341	0.065	0.058	0.372	0.497

Continued on next page

Table 5.3 – *Continued from previous page*

query	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20

Table 5.3: MMR Single Lambda Example

runid	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
0.80	0.297	0.334	0.061	0.051	0.374	0.507
0.80	0.294	0.335	0.062	0.054	0.360	0.490
0.35	0.288	0.327	0.058	0.050	0.361	0.493
0.25	0.259	0.292	0.049	0.040	0.312	0.430
0.60	0.298	0.341	0.065	0.058	0.372	0.497
0.10	0.235	0.275	0.039	0.035	0.277	0.418
0.20	0.267	0.303	0.052	0.043	0.327	0.455
0.60	0.293	0.334	0.062	0.054	0.360	0.490
0.75	0.281	0.326	0.056	0.051	0.348	0.500
0.15	0.243	0.282	0.042	0.036	0.288	0.426
0.60	0.297	0.334	0.061	0.051	0.374	0.507
0.40	0.295	0.332	0.062	0.055	0.363	0.481
0.35	0.269	0.306	0.053	0.045	0.332	0.462
0.35	0.299	0.340	0.066	0.059	0.372	0.488
0.90	0.281	0.326	0.056	0.051	0.348	0.500
0.55	0.293	0.335	0.062	0.054	0.360	0.494
0.85	0.298	0.341	0.065	0.058	0.372	0.497
0.85	0.297	0.334	0.061	0.051	0.374	0.507
0.60	0.281	0.326	0.056	0.051	0.348	0.500
0.75	0.297	0.334	0.061	0.051	0.374	0.507
0.55	0.298	0.341	0.065	0.058	0.372	0.497
0.30	0.301	0.339	0.066	0.058	0.375	0.488
0.55	0.295	0.334	0.060	0.051	0.372	0.507
0.55	0.281	0.325	0.056	0.051	0.348	0.494
0.25	0.297	0.339	0.066	0.057	0.368	0.496

Continued on next page

Table 5.4 – *Continued from previous page*

runid	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
0.50	0.290	0.329	0.060	0.050	0.363	0.499
0.40	0.288	0.330	0.059	0.051	0.361	0.505
0.10	0.244	0.283	0.046	0.038	0.276	0.412
0.20	0.294	0.336	0.064	0.057	0.363	0.488
0.40	0.299	0.341	0.065	0.058	0.372	0.494
0.85	0.281	0.326	0.056	0.051	0.348	0.500
0.65	0.297	0.334	0.061	0.051	0.374	0.507
0.45	0.277	0.317	0.055	0.048	0.343	0.483
0.80	0.298	0.341	0.065	0.058	0.372	0.497
0.70	0.298	0.341	0.065	0.058	0.372	0.497
0.30	0.291	0.332	0.062	0.055	0.355	0.484
0.50	0.281	0.325	0.056	0.050	0.348	0.494
0.70	0.297	0.334	0.061	0.051	0.374	0.507

Table 5.4: MMR Average results for different lambda values

We will present the best λ value for each experimental setup and compare the results using the best lambda value and the average result among all queries. For each setup, we use a set of parameters in order to achieve the best result. The parameter types are given below:

- λ (Real): Constant used on MMR calculation. The formula can be seen in the Section 3.2.1
- Hashtags (Bool): Defines whether to use hashtags of the tweet or not.
- Timestamp (Bool): Similar to the hashtag, defines the usage of timestamp of the tweet.
- EdgeSimilarity (Function): The similarity function used on calculating the similarities between the tweets.

- NodeSimilarity (Function): The similarity function used on calculating the similarity of the tweet and the query.

First, we reproduced the results of the work of Ozsoy et al. [12], since the dataset and the setup was similar on both their and our work. We experimented the MMR results with both Ratio-H and Cosine as similarity function their corresponding results are given in the Table 5.5.

Method	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
MMR_Cosine	0.313573	0.351419	0.064177	0.054150	0.386365	0.506234
MMR_Ratio-H	0.335558	0.373629	0.063934	0.053914	0.408797	0.546616

Table 5.5: MMR – Ratio-H Cosine Comparison

The best setup of Cosine similarity uses the following parameters: takes hashtag into consideration, while timestamps are not taken into account and λ is 0.8. Similarly, for the best setup of Ratio-H similarity the following parameters applied: both hashtag and timestamp information is not used and λ is 0.3.

After reproducing the baseline results, we tried both sentence transformers and the FastText for the same setup. The results is given in the Table 5.6.

Method	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
MMR_Cosine	0.313573	0.351419	0.064177	0.054150	0.386365	0.506234
MMR_Ratio-H	**0.335558**	**0.373629**	0.063934	0.053914	0.408797	**0.546616**
MMR_SBERT	0.298548	0.341170	0.065329	0.058335	0.372206	0.494316
MMR_FastText	0.328084	0.364173	**0.069257**	**0.058380**	**0.417546**	0.541950

Table 5.6: MMR – Comparison of SBERT and FastText with baseline results

The MMR parameters for sentence transformers and FastText are for their best setup is: both FastText and SBERT uses hashtag information and does not use timestamp information. The λ value for FastText is 0.95, while for SBERT is 0.8. We use Cosine similarity for SBERT similarity calculations while we use Word Mover's Distance for FastText setup.

As can be seen in the Table 5.6, although the sentence transformers and FastText are not performing better than the baseline results in much cases, in some cases the methods are the best such as P-IA10, P-IA20 and strec@10.

5.4.5 Sy

For Sy method, we use a similar environment as the MMR experiments 5.4.4. We change the parameter types along with the method and again we generate the baseline results again for Sy. The parameters used in the Sy method are given below:

- λ (Bool): The value that is the threshold of the Sy calculation, which is explained in the Section 3.2.2.
- Hashtags (Bool): Similar to the MMR experiments 5.4.4, defines whether to use the hashtags of the tweet in calculation or not.
- Timestamp (Bool): Defines whether to use the timestamp of the tweet or not.
- TweetSimilarity (Function): Similarity function used on calculation.

After reproducing the baseline results, we applied sentence transformers and FastText word embeddings to the system, just as the MMR experiments. The comparison between the baseline results of the Sy and our contributions to the system are given in the Table 5.7. As shown in the table, we can say that the SBERT gives competitive results in the Sy setup. Moreover, similar to the MMR method, in some cases SBERT outperforms the baseline results.

Method	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
Sy_Baseline	**0.348171**	**0.383123**	0.082955	0.068976	**0.418546**	**0.541773**
Sy_SBERT	0.340802	0.376743	**0.084510**	**0.076604**	0.409548	0.522357
Sy_FastText	0.339251	0.377139	0.078713	0.066695	0.400925	0.524407

Table 5.7: Sy – Baseline Result

The parameters of each method as follows: all 3 methods uses both hashtag and timestamp informations. Baseline and SBERT uses Cosine for similarity calculations

while FastText is using Word Mover's Distance. And finally, the λ values are 0.5 for baseline, 0.9 for SBERT and 0.23 for FastText.

Second, we fine-tuned the SBERT in order to achieve better results. To fine-tune the pretrained model, we used the Tweets 2013 (Internet Archive) [41]. There are two different fine-tuned models: one is trained with 1500 tweets (modelSmall) and the other is trained with 20000 tweets (modelBig). The results of the fine-tuned models are given with the baseline results in the Table **??**

Method	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
Sy_SBERT	**0.340802**	0.376743	**0.084510**	**0.076604**	**0.409548**	0.522357
Sy_SBERTsmall	0.333754	0.375599	0.078393	0.068258	0.397143	**0.531646**
Sy_SBERTbig	0.338858	**0.379366**	0.079021	0.068536	0.399502	0.530725

Table 5.8: Sy – SBERT Fine-Tuning Comparison

After fine-tuning with the bigger dataset, we find out an increase in the α-nDCG@20 value which is one of the most important metrics on our experimental setup. All the other metrics, except strec@20, shows slight decrease in their values. Although there is a decrease in other metrics, the increase in the α-nDCG@20 is sharper than these decreases which leads us to use the fine-tuned model SBERTbig as our main model for the Sy setup.

Finally, we observe that in some of the results some retweets appear with their original content, although the diversification algorithm is applied. Since a retweet is the same as its original except the user who tweets it, these values should not be appear in the result set. We come up with two solutions to this problem: completely remove all retweets by not accepting them in the result set and increasing their similarity result by a constant to reduce their likelihood of appearing in the result set. Increasing the similarity result is achieved by multiplying the original tweet to tweet similarity by a constant value as shown in the Equation 5.4.

$$\text{sim(t1,t2)} = \text{sim(t1,t2)} * \lambda \tag{5.4}$$

First, we applied the retweet removal and impact lowering to the SBERT fine-tuned big model and these experiment's results are given in the Table 5.9.

Method	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
Baseline	0.348171	**0.383123**	0.082955	**0.068976**	0.418546	0.541773
Sy_SBERTbig	0.338858	0.379366	0.079021	0.068536	0.399502	0.530725
RTRemove	0.348957	0.382893	0.080867	0.066000	0.426521	**0.550268**
RTLowerImpact	**0.355180**	0.382546	**0.082293**	0.065531	**0.434649**	0.531329

Table 5.9: Sy – Retweet Operation Results

Again these operations for SBERT gives the best results on the parameters of the original fine-tuned model's parameters. Moreover for lowering the impact of the retweets, the best result is found for the λ constant 1.35 for the similarity increase function. The results again shows sharp increase in the α-nDCG@20 metric and models gives better result in almost every metric, except the P-IA@20. Moreover, we see that the results are very competitive against the baseline results. Especially lowering the impact of the retweets gives better results than the baseline in the most of the cases. Although the results are competitive and better in some cases the baseline results still slightly higher than in the α-nDCG@20 metric.

Finally, we apply both retweet operations to the baseline method in order to find out whether there is a similar increase in the results or not. The results is given in the Table 5.10 with the SBERT retweet operation results for comparison.

Method	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
Baseline	0.348171	0.383123	0.082955	**0.068976**	0.418546	0.541773
Baseline_RTRemove	0.349991	0.380891	0.083069	0.063479	0.416875	0.541669
Baseline_RTLower	**0.360097**	**0.388982**	**0.084718**	0.064304	0.433368	0.549153
SBERTBig-RTRemove	0.348957	0.382893	0.080867	0.066000	0.426521	**0.550268**
SBERTBig-RTLower	0.355180	0.382546	0.082293	0.065531	**0.434649**	0.531329

Table 5.10: Sy – SBERT and Baseline Retweet Operation Result Comparison

There is a drastic increase in the baseline results after applying retweet lowering op-

eration, especially in terms of α-nDCG@10 and α-nDCG@20 metrics. Although retweet lowering causes sharp increase, removing retweets decreased the baseline results. Moreover, we can see that the SBERT still gives competitive results and even better in strec metrics.

5.4.6 xQuAD

xQuAD, as explained in the Section 3.2.3, is a probabilistic model which need the subquery information. The subqueries that are used are also taken from the Dataset 5.4.1. We apply both SBERT and FastText to the baseline setup in both parts of the xQuAD equation, which are tweet to query and tweet to aspect similarity. For each tweet, query and aspect we calculate SBERT embeddings and FastText word embeddings. For SBERT we use Cosine distance and for FastText we apply Word Mover's Distance. Initially, we present 3 results in the Table 5.11 which contains baseline, SBERT and FastText results for xQuAD. The SBERT model that we use is explained in the Section 3.2.2, we are using the fine-tuned big model.

Method	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
Baseline	**0.248999**	**0.295140**	**0.054159**	**0.050697**	**0.331457**	**0.478965**
FastText	0.187580	0.244771	0.036369	0.039731	0.228067	0.418430
SBERTBig	0.155725	0.198940	0.032212	0.035855	0.221816	0.351632

Table 5.11: xQuAD – SBERT, FastText and Baseline Results

We only change lambda values and the distance metric that is used in similarity calculation. For each method these parameters are: baseline and SBERT uses Cosine and FastText uses Word Mover's Distance for similarity calculations. The λ values are 1.6 for baseline, 0.8 for SBERT and 0.2 for FastText.

As can be seen in the Table 5.11, both SBERT and FastText results are significantly lower than the baseline results. Although, SBERT and FastText performs similar in previous approaches, for xQuAD the is no promising result from these methods. Moreover, similar to the Sy and MMR experiments, we apply retweet removing operation during the ranking stage for xQuAD. The parameters for the best results are

not changed for all 3 methods and the results are given in the Table 5.12.

Method	α-nDCG@10	α-nDCG@20	P-IA@10	P-IA@20	strec@10	strec@20
Baseline_RTRemove	**0.273648**	**0.319169**	**0.059554**	**0.054961**	**0.362647**	**0.509575**
FastText_RTRemove	0.201773	0.264507	0.042329	0.045081	0.254832	0.461445
SBERT_Big_RTRemove	0.187580	0.244771	0.036369	0.039731	0.228067	0.418430

Table 5.12: xQuAD – SBERT, FastText and Baseline Retweet Removal Results

Comparison of Table 5.11 and Table 5.12 shows that retweet removing is significantly increases the results of all 3 methods. However, similar to the initial results, the baseline methods are still performs the best among all 3 results.

5.5 On-the-Fly Ranking

Finally, we present an on-the-fly ranking system to the Twitter system. We gather tweets from the Twitter from the Twitter API using the developer system. Moreover, we rank the search results using MMR and SBERT to create diverse search results and present the results to the user. In order to convert this to a live tool, we created a Django application using Python, in which we gather query and number of tweets from the user and gather that many result from the Twitter API, then using SBERT and MMR we sort the results and present to the user. In order to make comparison, we present the results in a two column based page, in one side there is original Twitter results and on the second column, we present the diverse results.

The system is currently working on the local machine, but can be served to a Web Server and can be opened to the public use. The application is simple, consists of 3 pages:

- Login page
- Query-Count Input page
- Results page

Examples from the application are given in the Figures C.1 C.2 C.3.

Moreover, the system has another component which allow us to make human judgements. The system is creating a database using the already retrieved tweets and for each query we define aspects of the query. System assigns a set of tweets to each user and collect their annotation about how relevant given tweet is to the query and how relevant is tweet to its aspects. The judgements are stored in our local database for creating a our own dataset of tweet search and diversification.

An example from the evaluation page of the system is given in the Figures C.5.

CHAPTER 6

CONCLUSION

In this thesis, as our first contribution, we proposed a diversification aware focused crawler which aims to not only fetch the pages relevant to a given target topic but also provide a balanced coverage of the aspects of the topic among those pages. To achieve this goal, we converted a general purpose crawler into a focused crawler by extending the module using a Random Forest Classifier [42] and a scorer following the practice of Chakrabarti et al. [4] and Pant et al. [5]. Next, we modified the scorer function with a diversification strategy inspired from the approach in [6], and by using manually determined aspects and extending them using Google API, we constructed a diversification aware focused crawler. We also extended the diversification aware crawler using word and sentence embeddings, to investigate the impact of such alternative representations on a focused crawler.

To evaluate the performance of the proposed approaches, we conducted several crawling sessions for the target topic "Covid 19". Our experimental results showed that the diversification aware focused crawler outperforms the typical focused crawler in terms of the P-IA metric and standard deviation. We also found that word embeddings have the potential to further improve the performance of our crawler. In contrast, sentence embeddings only helped reducing the number of totally irrelevant pages, i.e., with zero similarity to the target topic and its aspects.

As our second contribution in this thesis, we exploited the word and sentence embeddings to improve the performance of diversification on Tweet search results. In particular, we adapted the evaluation methodology in [12] and exploited both word embeddings and sentence transformers using well-known implicit and explicit diversification algorithms. We evaluated the performance over the standard datasets and

found that both word embeddings and sentence embeddings yield promising results. We observed that especially sentence embeddings perform better than word embeddings and may improve the diversification performance in terms of P-IA and ST-recall metrics. This is an intuitive finding, as tweets, including no more than few hundred characters, are likely to be appropriately represented by sentence embeddings.

As future work, we are planning to apply Doc2Vec [43] to both diversification aware focused crawler and tweet search result diversification. Moreover, we will explore alternative aspect weighting functions that may favor the uncovered aspects more strongly. Finally, we plan to fine-tune our sentence transformer models with a larger dataset to see its impact for both of the diversification scenarios.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[2] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, (Lille, France), pp. 957–966, JMLR.org, July 2015.

[3] R. L. Santos, C. Macdonald, and I. Ounis, "Exploiting query reformulations for web search result diversification," in *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, (New York, NY, USA), pp. 881–890, Association for Computing Machinery, Apr. 2010.

[4] S. Chakrabarti, M. V. D. Berg, and B. Dom, "Focused crawling: A new approach to topic-specific web resource discovery," in *Computer Networks*, pp. 1623–1640, 1999.

[5] G. Pant and P. Srinivasan, "Link contexts in classifier-guided topical crawlers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 107–122, Jan. 2006.

[6] A. M. Ozdemiray and I. S. Altingovde, "Explicit search result diversification using score and rank aggregation methods," *J. Assoc. Inf. Sci. Technol.*, vol. 66, no. 6, pp. 1212–1228, 2015.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning*

Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (Y. Bengio and Y. LeCun, eds.), 2013.

[8] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, 2014.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, 2017.

[10] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)* (M. A. Walker, H. Ji, and A. Stent, eds.), pp. 2227–2237, Association for Computational Linguistics, 2018.

[11] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019* (K. Inui, J. Jiang, V. Ng, and X. Wan, eds.), pp. 3980–3990, Association for Computational Linguistics, 2019.

[12] M. G. Ozsoy, K. D. Onal, and I. S. Altingovde, "Result Diversification for Tweet Search," in *Web Information Systems Engineering – WISE 2014* (B. Benatallah, A. Bestavros, Y. Manolopoulos, A. Vakali, and Y. Zhang, eds.), Lecture Notes in Computer Science, (Cham), pp. 78–89, Springer International Publishing, 2014.

[13] J. Carbonell and J. Goldstein, "The use of MMR, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, (New York, NY, USA), pp. 335–336, Association for Computing Machinery, Aug. 1998.

[14] K. Tao, C. Hauff, and G.-J. Houben, "Building a Microblog Corpus for Search Result Diversification," in *Information Retrieval Technology* (R. E. Banchs, F. Silvestri, T.-Y. Liu, M. Zhang, S. Gao, and J. Lang, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 251–262, Springer, 2013.

[15] K. Tao, F. Abel, C. Hauff, G.-J. Houben, and U. Gadiraju, "Groundhog day: Near-duplicate detection on Twitter," in *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, (New York, NY, USA), pp. 1273–1284, Association for Computing Machinery, May 2013.

[16] R. L. T. Santos, J. Peng, C. Macdonald, and I. Ounis, "Explicit search result diversification through sub-queries," in *Advances in Information Retrieval, 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010. Proceedings* (C. Gurrin, Y. He, G. Kazai, U. Kruschwitz, S. Little, T. Roelleke, S. M. Rüger, and K. van Rijsbergen, eds.), vol. 5993 of *Lecture Notes in Computer Science*, pp. 87–99, Springer, 2010.

[17] F. Menczer, "ARCCHNID: adaptive retrieval agents choosing heuristic neighborhoods," in *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997* (D. H. Fisher, ed.), pp. 227–235, Morgan Kaufmann, 1997.

[18] S. Chakrabarti, K. Punera, and M. Subramanyam, "Accelerated focused crawling through online relevance feedback," in *Proceedings of the 11th international conference on World Wide Web*, WWW '02, (New York, NY, USA), pp. 148–159, Association for Computing Machinery, May 2002.

[19] J. Rennie and A. McCallum, "Using reinforcement learning to spider the web efficiently," in *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999* (I. Bratko and S. Dzeroski, eds.), pp. 335–343, Morgan Kaufmann, 1999.

[20] I. Altingovde and O. Ulusoy, "Exploiting interclass rules for focused crawling," *IEEE Intelligent Systems*, vol. 19, pp. 66–73, Nov. 2004.

[21] W. Wang, X. Chen, Y. Zou, H. Wang, and Z. Dai, "A Focused Crawler Based on

Naive Bayes Classifier," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pp. 517–521, Apr. 2010.

[22] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[23] H. Lu, D. Zhan, L. Zhou, and D. He, "An Improved Focused Crawler: Using Web Page Classification and Link Priority Evaluation," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–10, 2016.

[24] Y. Du, W. Liu, X. Lv, and G. Peng, "An improved focused crawler based on Semantic Similarity Vector Space Model," *Applied Soft Computing*, vol. 36, pp. 392–407, Nov. 2015.

[25] M. M. G. Farag, S. Lee, and E. A. Fox, "Focused crawler for events," *International Journal on Digital Libraries*, vol. 19, pp. 3–19, Mar. 2018.

[26] K. D. Onal, I. S. Altingovde, and P. Karagoz, "Utilizing word embeddings for result diversification in tweet search," in *Information Retrieval Technology - 11th Asia Information Retrieval Societies Conference, AIRS 2015, Brisbane, QLD, Australia, December 2-4, 2015. Proceedings* (G. Zuccon, S. Geva, H. Joho, F. Scholer, A. Sun, and P. Zhang, eds.), vol. 9460 of *Lecture Notes in Computer Science*, pp. 366–378, Springer, 2015.

[27] S. Gollapudi and A. Sharma, "An axiomatic approach for result diversification," in *Proceedings of the 18th international conference on World wide web*, WWW '09, (New York, NY, USA), pp. 381–390, Association for Computing Machinery, Apr. 2009.

[28] M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. Traina, and V. J. Tsotras, "On query result diversification," in *2011 IEEE 27th International Conference on Data Engineering*, pp. 1163–1174, Apr. 2011. ISSN: 2375-026X.

[29] Y. B. Ulu and I. S. Altingovde, "Predicting the Size of Candidate Document Set for Implicit Web Search Result Diversification," in *Advances in Information Retrieval* (J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva,

and F. Martins, eds.), Lecture Notes in Computer Science, (Cham), pp. 410–417, Springer International Publishing, 2020.

[30] C. De Boom, S. Van Canneyt, T. Demeester, and B. Dhoedt, "Representation learning for very short texts using weighted word embedding aggregation," *Pattern Recognition Letters*, vol. 80, pp. 150–156, Sept. 2016.

[31] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.

[32] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, vol. 32 of *JMLR Workshop and Conference Proceedings*, pp. 1188–1196, JMLR.org, 2014.

[33] G. A. Miller, "WordNet: A lexical database for English," *Communications of the ACM*, vol. 38, pp. 39–41, Nov. 1995.

[34] D. Lin, "An Information-Theoretic Definition of Similarity," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, (San Francisco, CA, USA), pp. 296–304, Morgan Kaufmann Publishers Inc., July 1998.

[35] P. Jaccard, "The Distribution of the Flora in the Alpine Zone.1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[36] B. Sagar-Fenton and L. McNeill, "How many words do you need to speak a language?," *BBC News*, June 2018.

[37] J. B. Lovins, "Development of a stemming algorithm," *Mech. Transl. Comput. Linguistics*, vol. 11, pp. 22–31, 1968.

[38] J. A. Shaw and E. A. Fox, "Combination of multiple searches," in *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994* (D. K. Harman, ed.), vol. 500-225 of *NIST Special Publication*, pp. 105–108, National Institute of Standards and Technology (NIST), 1994.

[39] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, "Diversifying search results," in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, (New York, NY, USA), pp. 5–14, Association for Computing Machinery, Feb. 2009.

[40] T. Kenter, A. Borisov, and M. de Rijke, "Siamese CBOW: optimizing word embeddings for sentence representations," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, The Association for Computer Linguistics, 2016.

[41] R. Sequiera and J. Lin, "Finally, a Downloadable Test Collection of Tweets," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, (Shinjuku Tokyo Japan), pp. 1225–1228, ACM, Aug. 2017.

[42] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[43] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.), vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 1188–1196, PMLR, 22–24 Jun 2014.

APPENDIX A

FOCUSED CRAWLER SETUP

Listing A.1: Seed URLs

```
https://coronavirus.jhu.edu/map.html
https://scdhec.gov/covid19
https://www.usa.gov/coronavirus
https://www.coronavirus.in.gov/
https://www.aap.org/en/pages/2019 novel coronavirus covid 19
infections/children and covid 19 state level data report/
https://www.osha.gov/coronavirus
https://ldh.la.gov/coronavirus/
https://www.whitehouse.gov/covidplan/
https://floridahealthcovid19.gov/
https://www.michigan.gov/coronavirus/
https://www.hhs.gov/coronavirus/community based testing sites/ind
ex.html
https://www.austintexas.gov/covid19
https://covid19.ncdhhs.gov/dashboard
https://dshs.texas.gov/covidvaccine/
https://covid19.ca.gov/
https://studentaid.gov/announcements events/coronavirus
https://coronavirus.idaho.gov/
https://www.ssa.gov/coronavirus/
https://www.publix.com/covid vaccine
https://coronavirus.iowa.gov/
```

i	me	my	myself	we	our	ours
ourselves	you	you’re	you’ve	you’ll	you’d	your
yours	yourself	yourselves	he	him	his	himself
she	she’s	her	hers	herself	it	it’s
its	itself	they	them	their	theirs	themselves
what	which	who	whom	this	that	that’ll
these	those	am	is	are	was	were
be	been	being	have	has	had	having
do	does	did	doing	a	an	the
and	but	if	or	because	as	until
while	of	at	by	for	with	about
against	between	into	through	during	before	after
above	below	to	from	up	down	in
out	on	off	over	under	again	further
then	once	here	there	when	where	why
how	all	any	both	each	few	more
most	other	some	such	no	nor	not
only	own	same	so	than	too	very
s	t	can	will	just	don	don’t
should	should’ve	now	d	ll	m	o
re	ve	y	ain	aren	aren’t	couldn
couldn’t	didn	didn’t	doesn	doesn’t	hadn	hadn’t
hasn	hasn’t	haven	haven’t	isn	isn’t	ma
mightn	mightn’t	mustn	mustn’t	needn	needn’t	shan
shan’t	shouldn	shouldn’t	wasn	wasn’t	weren	weren’t
won	won’t	wouldn	wouldn’t			

Table A.1: NLTK Stopwords

map	payments	consequences	forum	europe	recession
injury	cases	19	economy	damage	causes
outlook	united	act	statistics	post	covid
aid	relief	slowdown	supporting	live	update
latest	articles	depression	coronavirus	city	loss
affect	analysis	challenges	fund	uk	crisis
inequality	affected	effects	essay	collapse	world
blog	package	italy	covid-19	recovery	fallout
stimulus	problems	policy	application	paper	updates
pakistan	tax	alternatives	price	documentation	bangladesh
measures	affects	corona	germany	global	us
reset	development	measure	effect	south	bill
times	war	china	disaster	country	shock
payment	strategy	approach	downturn	support	swiss
response	2022	growth	economic	socio	sweden
forecast	kavach	covid19	plan	news	login
canada	loan	factors	states	impact	india

Table A.2: Covid-19 – Economic Impact Phrases

leg	just	precautions	19	signs	medicine
stuffy	covid	relief	latest	nose	late
head	groin	rate	2020	progression	cold
stomach	common	days	muscle	sores	fully
mayo	babies	urgent	confusion	pregnant	child
quarantine	hunger	questionnaire	vaccinated	fatigue	baby
negative	jaw	quick	mouth	knee	people
going	away	infection	nowadays	young	quotes
gone	county	come	onset	hospital	eye
list	excessive	care	unable	hot	clinic
smell	stages	wikipedia	initial	tongue	dataset
thirst	vertigo	light	throwing	francais	heartbeat
rash	relapse	teens	kids	headache	vision
pressure	burning	test	sore	employees	worse
night	flashes	flu	bleed	pfizer	jour
waves	upper	sweats	legs	gastrointestinal	blocked
chart	heartburn	adults	watch	temperature	taste
yellow	peeing	range	swollen	gum	skin
peak	last	wise	ear	update	eyes
joint	newborn	emergency	corona	heart	checker
eyesight	ray	recently	elderly	cure	toddler
kidney	many	variant	urine	ears	timeline
show	brain	sneezing	phlegm	appetite	dogs
rapid	nodes	time	exposure	boosted	shot
rib	en	early	versus	color	pain
india	nasal	wave	pregnancy	day	shivering
gastro	diary	redness	headed	aches	gassy
dizzy	hallucinations	remain	mucus	first	throat
chest	diarrhea	recent	coronavirus	fever	belegte
per	vomiting	treatment	allergies	unvaccinated	getting
related	sleep	migraine	vaccine	covid-19	sweating
recovery	zunge	pink	booster	man	delta
earache	red	bangla	orange	glands	qld
hearing	order	symptoms	gov	cough	second
2021	tested	step	sinus	english	reviews
omicron	2022	children	covid19	mild	restless

Table A.3: Covid-19 – Symptoms Phrases

safe	benefits	scan	percentage	19	ladies
hand	registration	zonder	issues	walk	jersey
covid	certificate	distribution	long	ludhiana	vaccinations
ny	nj	kinds	registry	research	kroger
babies	reactions	download	pregnant	made	helsinki
verification	malaysia	report	jobs	rate	status
guidelines	online	immunocompromised	details	jacksonville	vero
quantity	patients	ingredients	numbers	infection	number
news	quotes	types	gone	county	hospital
graph	questionnaire	youtube	vaasa	hub	yonkers
shots	region	third	wala	term	gravid
uk	safety	drawing	wikipedia	effects	toll
application	usa	espoo	perth	nz	soopers
database	groningen	pfizer	helpline	under	los
zurich	queue	investigation	logo	nuksan	yuma
tylenol	which	fda	vaccine.gov.lk	without	vaccines
netherlands	company	cancer	working	names	equity
greensboro	recommendations	side	kisne	japan	az
breastfeeding	stroke	template	verify	chart	testing
training	inventor	travel	tidsbestilling	diego	vial
name	nikale	haven	united	appointment	questions
journal	old	target	breast	aid	bloomberg
paso	approval	wiki	den	rite	qualifications
corona	evidence	poster	development	free	timeline
walgreens	animal	efficacy	cdc	rates	dosage
sites	variants	work	map	vaccination	finder
waiver	eligibility	update	risks	osha	virus
wrong	statistics	qr	giant	live	omicron
moderna	mandatory	effective	coronavirus	approved	per
indiana	coronavaccin	denmark	kaise	portal	information
washington	pakistan	contraindications	booster	yakima	makati
younger	tracker	viagra	paralysis	entry	groups
effectiveness	haag	country	francisco	worldwide	manitoba
mandate	fact	info	lawsuit	reviews	trials
dose	2022	children	covid19		

Table A.4: Covid-19 – Vaccine Phrases

smoking	who	sbi	partnership	19.clinical	quantitative
near	good	range	19	curfew	medicine
antibody	education	funding	students	questions	jhu
journal	kara	recipe	covid	coalition	examples
cans	centre	update	blood	john	nl
latest	result	nih	findings	volunteers	coronavirus
title	recent	wuhan	university	mit	proposal
treatment	analysis	uk	tagalog	results	queso
ideas	related	philippines	effects	information	vaccine
uf	covid-19	problem	qatar	example	hopkins
topic	paper	papers	washington	net	data
booster	research	registry	price	article	blvd
database	psychology	coronanet	boulevard	sample	corona
center	important	report	about	jobs	disease
drug	livestock	brainly	fonds	south	csir
test	study	solar	review	topics	literature
project	studies	harrison	institute	group	company
community	denver	donation	trials	eu	researchgate
omicron	design	toyota	statement	games	clinical
paid	qualitative	medical	dose	pdf	type
during	jardin	russia	hindi	stock	canada
oxford	luxembourg	history	super	grants	impact

Table A.5: Covid-19 – Research Phrases

urdu	slogan	safety	food	role	cartoon
chart	who	gif	training	vaccination	guidance
pills	preventing	materials	malayalam	checklist	precautions
workplace	shopify	epidemic	btp	medicine	icon
questions	virus	camp	clipart	pics	covid
signs	vector	dress	notice	bengali	pictures
website	articles	webinar	coronavirus	tips	california
dental	video	treatment	marathi	rate	against
tagalog	swiggy	drawing	quiz	wikipedia	essay
vaccine	machine	shop	prevent	telugu	control
tamil	quebec	covid-19	rules	devices	images
kit	equipment	ways	contre	policy	school
method	methods	cpt	cure	industry	research
organization	babies	measures	message	corona	malaysia
global	exercises	prevention	activities	report	spray
steps	slogans	discharge	rice	poster	guidelines
disease	drug	speech	logo	flyer	symptoms
nepali	tienda	tablets	slideshare	animation	meds
cdc	appendix	english	network	strategies	prevention.org
trials	program	infection	covid19	natural	medication
plan	code	pdf	tools	quotes	messages
ucsd	hindi	schools	questionnaire	office	protocol
home	and	symbol	awareness	nasal	ppt

Table A.6: Covid-19 – Prevention Phrases

APPENDIX B

FOCUSED CRAWLER EXPERIMENTAL RESULTS

Batch	Vaccine	Symptoms	Prevention	Research	Economic
1	4	6	4	4	4
2	48	78	53	52	52
3	79	169	93	90	88
4	121	382	159	150	151
5	176	579	249	224	224
6	252	871	371	337	329
7	315	1161	489	444	429
8	361	1429	615	559	546
9	404	1693	706	642	638
10	477	2005	847	773	768
11	550	2292	967	888	878
12	637	2610	1120	1031	1016
13	713	2899	1242	1148	1137
14	801	3183	1373	1277	1259
15	857	3427	1475	1372	1354
16	930	3706	1583	1476	1462
17	996	3956	1684	1573	1556
18	1083	4223	1806	1690	1672
19	1217	4561	1992	1872	1847
20	1351	4897	2177	2053	2022
21	1479	5193	2357	2224	2185
22	1587	5496	2508	2377	2336

Continued on next page

Table B.1 – *Continued from previous page*

Batch	Vaccine	Symptoms	Prevention	Research	Economic
23	1726	5825	2689	2553	2514
24	1827	6098	2813	2670	2635
25	1934	6414	2967	2822	2777
26	2030	6713	3094	2944	2902
27	2120	7016	3201	3051	3004
28	2199	7321	3311	3160	3108
29	2290	7626	3433	3273	3222
30	2375	7892	3527	3364	3316
31	2450	8159	3639	3469	3424
32	2526	8449	3752	3570	3531
33	2591	8735	3852	3660	3624
34	2639	8999	3942	3742	3705
35	2710	9325	4057	3852	3810
36	2766	9671	4153	3941	3888
37	2813	9948	4227	4010	3954
38	2887	10224	4329	4108	4047
39	2959	10523	4440	4212	4150
40	3018	10796	4535	4295	4229
41	3076	11079	4626	4370	4311
42	3141	11367	4737	4471	4410
43	3198	11647	4822	4554	4494
44	3241	11909	4907	4634	4571
45	3302	12193	5015	4733	4677

Table B.1: Number of Relevant Pages in No Diversification Focused Crawler

Batch	Vaccine	Symptoms	Prevention	Research	Economic
1	4	6	4	4	4
2	45	81	53	52	52
3	81	181	94	91	91
4	129	413	163	152	157
5	194	674	269	244	250
6	258	960	390	350	359
7	335	1267	539	485	502
8	417	1595	684	619	642
9	474	1905	789	717	739
10	562	2255	949	869	887
11	666	2632	1122	1033	1047
12	751	2963	1257	1165	1179
13	854	3321	1418	1326	1336
14	960	3679	1573	1471	1485
15	1067	4018	1726	1620	1632
16	1156	4333	1859	1745	1760
17	1275	4694	2048	1928	1941
18	1396	5076	2225	2106	2118
19	1516	5379	2378	2255	2269
20	1658	5728	2569	2444	2452
21	1781	6050	2718	2583	2594
22	1906	6435	2898	2750	2773
23	1976	6752	3018	2863	2890
24	2052	7078	3145	2983	3014
25	2141	7413	3272	3103	3138
26	2228	7783	3408	3226	3264
27	2316	8126	3538	3352	3385
28	2407	8427	3692	3499	3526
29	2458	8727	3793	3593	3622
30	2510	9025	3879	3676	3709

Continued on next page

Table B.2 – *Continued from previous page*

Batch	Vaccine	Symptoms	Prevention	Research	Economic
31	2579	9302	3984	3775	3812
32	2657	9569	4099	3886	3929
33	2760	9908	4239	4017	4073
34	2837	10195	4345	4117	4179
35	2925	10483	4439	4206	4283
36	3006	10789	4546	4303	4386
37	3085	11094	4666	4417	4502
38	3185	11422	4809	4557	4637
39	3289	11753	4955	4701	4773
40	3357	12075	5050	4790	4866
41	3415	12371	5153	4884	4971
42	3491	12732	5270	4993	5080
43	3548	13053	5369	5089	5172
44	3589	13372	5444	5157	5243
45	3644	13688	5531	5240	5327

Table B.2: Number of Relevant Pages in Diversification Aware Focused Crawler Results

Batch	Vaccine	Symptoms	Prevention	Research	Economic
1	4	6	4	4	4
2	45	78	53	52	52
3	78	169	92	89	88
4	116	376	151	141	143
5	182	622	242	220	220
6	233	890	329	299	293
7	286	1177	430	393	384
8	346	1461	556	508	500

Continued on next page

Table B.3 – *Continued from previous page*

Batch	Vaccine	Symptoms	Prevention	Research	Economic
9	405	1721	675	620	612
10	499	2019	825	765	748
11	580	2321	970	902	891
12	663	2618	1096	1023	1017
13	735	2869	1210	1133	1124
14	809	3127	1325	1244	1230
15	887	3392	1430	1347	1337
16	954	3635	1533	1451	1430
17	1060	3924	1676	1586	1569
18	1155	4247	1806	1714	1692
19	1248	4553	1935	1838	1820
20	1324	4869	2067	1966	1953
21	1430	5175	2209	2100	2080
22	1509	5472	2321	2208	2189
23	1615	5808	2470	2352	2324
24	1698	6106	2589	2469	2439
25	1797	6410	2717	2599	2568
26	1868	6679	2815	2691	2666
27	1945	6938	2920	2791	2758
28	2023	7233	3026	2891	2858
29	2097	7532	3144	3006	2970
30	2179	7842	3273	3132	3085
31	2272	8148	3422	3278	3221
32	2326	8416	3517	3369	3310
33	2405	8744	3637	3489	3427
34	2499	9036	3764	3616	3554
35	2592	9330	3896	3749	3682
36	2671	9603	4017	3864	3796
37	2759	9880	4160	3997	3938

Continued on next page

Table B.3 – *Continued from previous page*

Batch	Vaccine	Symptoms	Prevention	Research	Economic
38	2854	10183	4293	4123	4064
39	2924	10511	4424	4249	4189
40	3010	10801	4565	4389	4333
41	3115	11110	4706	4530	4478
42	3200	11424	4835	4657	4604
43	3276	11759	4971	4783	4733
44	3333	12038	5075	4879	4836
45	3403	12333	5180	4982	4938

Table B.3: Number of Relevant Pages in Diversification Aware Word Embedding Focused Crawler

Batch	Vaccine	Symptoms	Prevention	Research	Economic
1	4	5	4	4	4
2	48	77	53	52	52
3	48	84	56	53	53
4	50	95	59	56	56
5	53	118	65	60	59
6	84	233	113	103	102
7	84	236	113	103	102
8	86	248	115	105	105
9	89	256	119	109	109
10	91	268	122	111	111
11	139	432	186	174	176
12	139	435	187	175	176
13	139	456	188	175	176
14	185	644	245	233	232
15	185	647	245	233	232

Continued on next page

Table B.4 – *Continued from previous page*

Batch	Vaccine	Symptoms	Prevention	Research	Economic
16	215	799	281	270	269
17	215	801	281	270	269
18	216	805	282	271	270
19	216	808	282	271	270
20	216	811	282	271	270
21	255	1022	329	318	321
22	255	1023	329	318	321
23	255	1029	330	319	321
24	255	1031	330	320	321
25	255	1040	333	323	323
26	257	1053	336	326	326
27	258	1061	340	330	328
28	261	1076	343	333	331
29	315	1293	427	411	402
30	316	1295	427	411	402
31	317	1305	428	411	402
32	317	1305	428	411	402
33	343	1484	466	446	438
34	343	1487	466	446	438
35	343	1489	466	446	438
36	356	1644	479	459	453
37	356	1645	479	459	453
38	356	1646	479	459	453
39	356	1648	479	459	453
40	356	1651	479	459	453
41	358	1655	480	460	454
42	358	1657	480	460	454
43	358	1661	480	460	455
44	377	1799	506	486	483

Continued on next page

Table B.4 – *Continued from previous page*

Batch	Vaccine	Symptoms	Prevention	Research	Economic
45	377	1801	506	486	483

Table B.4: Number of Relevant Pages in Diversification Aware SBERT Focused Crawler

APPENDIX C

ON THE FLY TWEET DIVERSIFICATION

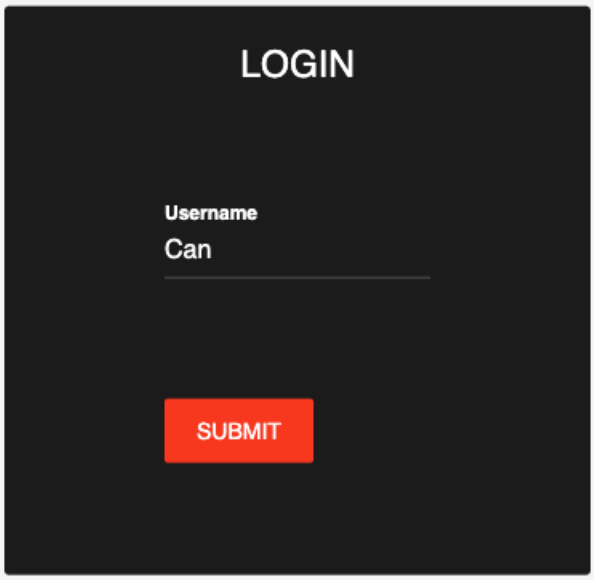

Figure C.1: Twitter Application – Login Page

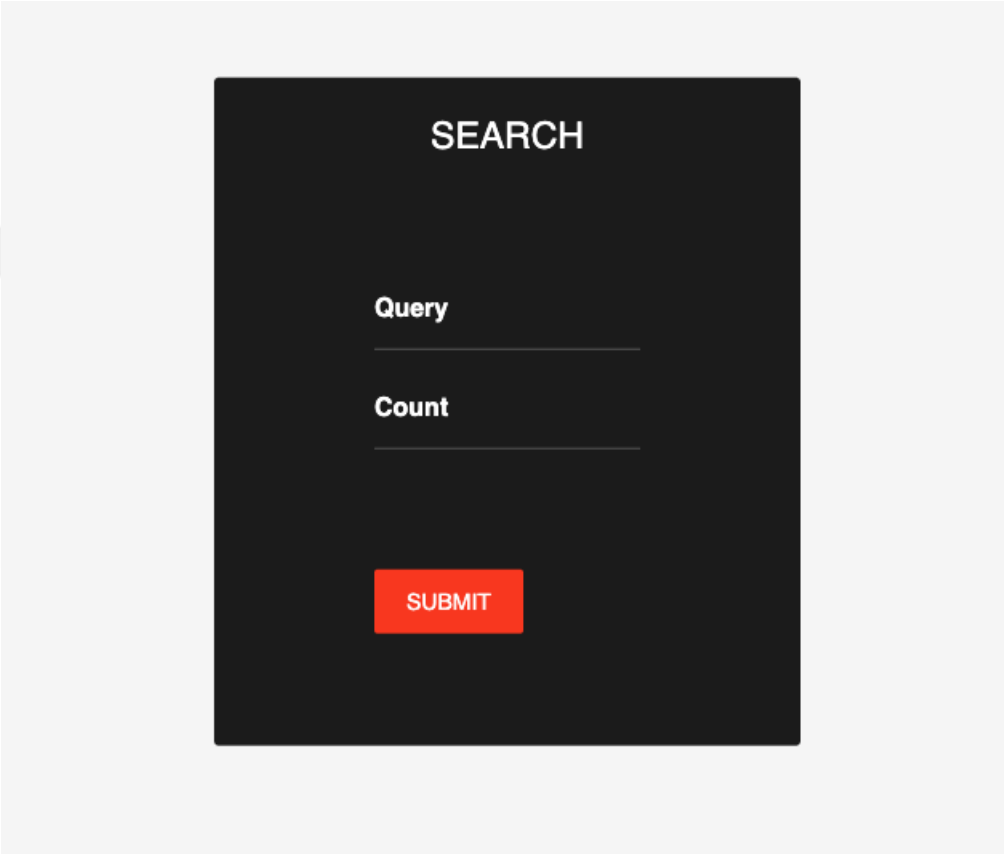

Figure C.2: Twitter Application – Search Page

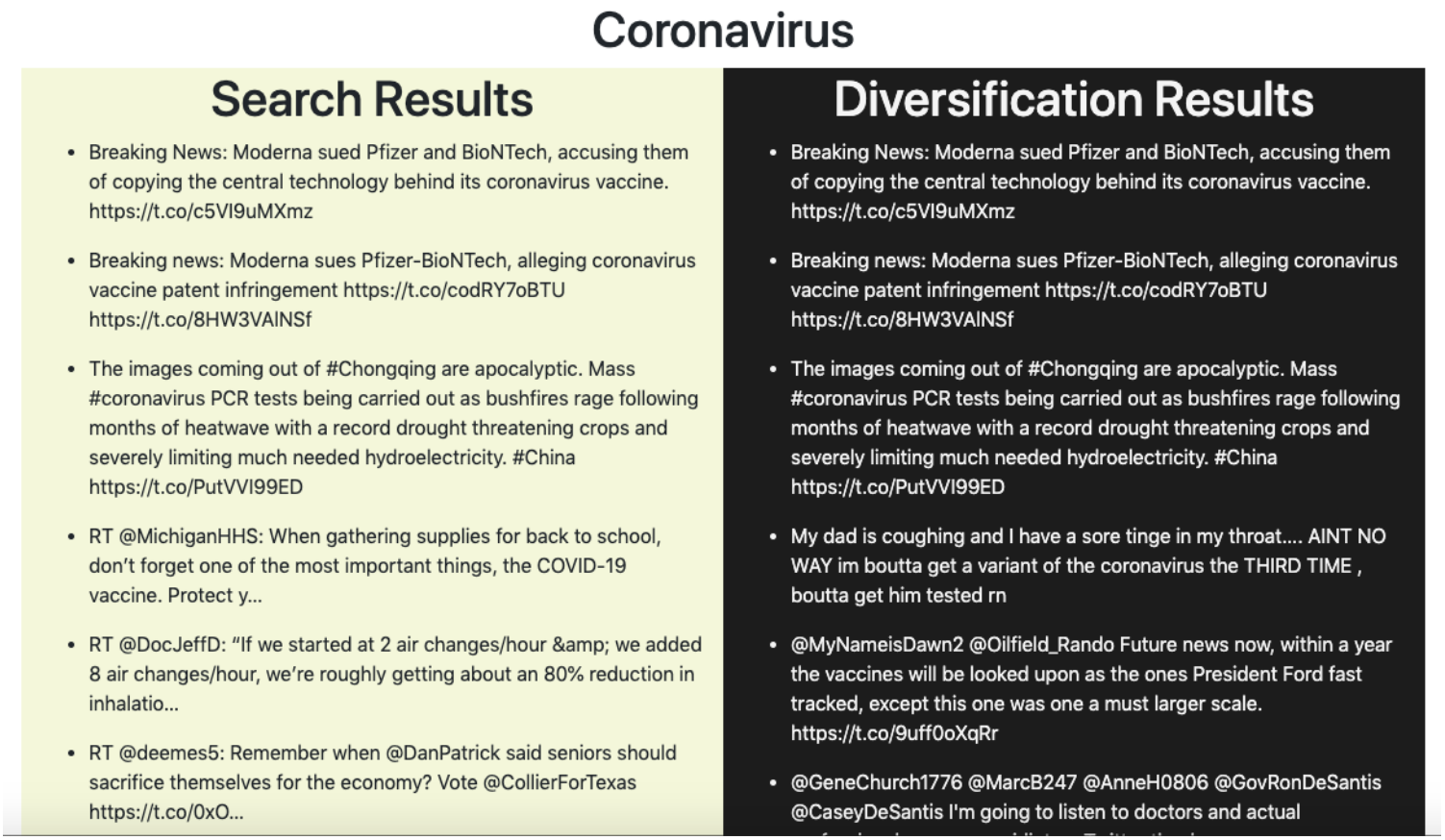

Figure C.3: Twitter Application – Results Page - Coronavirus

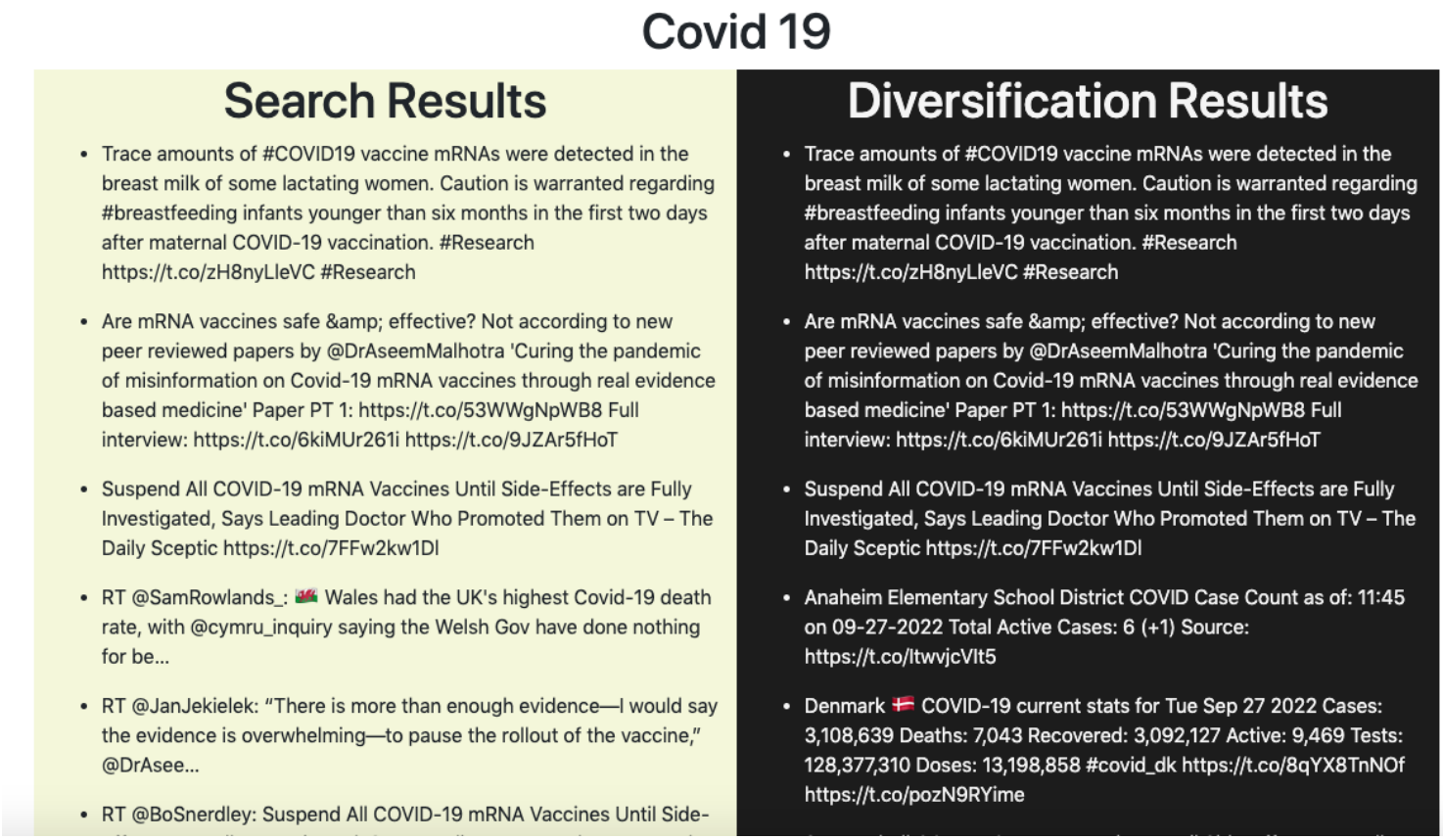

Figure C.4: Twitter Application – Results Page - Covid 19

Figure C.5: Twitter Application – Annotation Page