Vector Space Proximity Based Document Retrieval For Document Embeddings Built By Transformers

Pavel Khloponin

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

June 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Pavel Khloponin**

Entitled: **Vector Space Proximity Based Document Retrieval For Document Embeddings Built By Transformers**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

________________________________ Chair
Dr. Brigitte Jaumard

________________________________ Examiner
Dr. Essam Mansour

________________________________ Examiner
Dr. Brigitte Jaumard

________________________________ Supervisor
Dr. Leila Kosseim

Approved by ________________________________
Lata Narayanan, Chair
Department of Computer Science and Software Engineering

____________ 2022 ________________________
Mourad Debabbi, Dean
Faculty of Engineering and Computer Science

Abstract

Vector Space Proximity Based Document Retrieval For Document Embeddings Built By Transformers

Pavel Khloponin

Internet publications are staying atop of local and international events, generating hundreds, sometimes thousands of news articles per day, making it difficult for readers to navigate this stream of information without assistance. Competition for the reader's attention has never been greater. One strategy to keep readers' attention on a specific article and help them better understand its content is news recommendation, which automatically provides readers with references to relevant complementary articles. However, to be effective, news recommendation needs to select from a large collection of candidate articles only a handful of articles that are relevant yet provide diverse information.

In this thesis, we propose and experiment with three methods for news recommendation and evaluate them in the context of the NIST News Track. Our first approach is based on the classic BM25 information retrieval approach and assumes that relevant articles will share common keywords with the current article. Our second approach is based on novel document embedding representations and uses various proximity measures to retrieve the closest documents. For this approach, we experimented with a substantial number of models, proximity measures, and hyperparameters, yielding a total of 47,332 distinct models. Finally, our third approach combines the BM25 and the embedding models to increase the diversity of the results.

The results on the 2020 TREC News Track show that the performance of the BM25 model (nDCG@5 of 0.5924) greatly exceeds the TREC median performance (nDCG@5 of 0.5250) and achieves the highest score at the shared task. The performance of the embedding model alone (nDCG@5 of 0.4541) is lower than the TREC median and BM25. The performance of the combined model (nDCG@5 of 0.5873) is rather close to that of the BM25 model; however, an analysis of the results shows that the recommended articles are different from those proposed by BM25, hence may constitute a promising approach to reach diversity without much loss in relevance.

Acknowledgments

I would like to thank my mentor and thesis supervisor, Dr. Leila Kosseim, for all the help and guidance rendered during my study. Specifically, thank you for opening a brand new side of language and research by showing me how much effort should go into writing a paper, how sharp and focused words could be in prepared hands, and the excitement and satisfaction gained when presenting my work at a conference.

The friendly and competent atmosphere in the CLaC (Computational Linguistics at Concordia) lab set the level of expectation of my work and at the same time helped me gain valuable feedback from my peers, to whom I am forever grateful.

I would like to thank Dr. Brigitte Jaumard and Dr. Essam Mansour for finding the time to read this work and being on my examining committee. Your insightful questions and useful comments allowed me to look at my work in a different light.

The warmest thanks go to Nihatha Lathiff for her support. You helped me stay sane and motivated during my Master's degree. Your curious questioning on multiple aspects of the fields we studied together and fruitful discussions helped me discover details that I would never have noticed myself.

Lastly, I would like to thank my parents and brothers for giving me the ability and support to be where I am.

Contents

List of Figures **ix**

List of Tables **xi**

1 Introduction **1**

1.1 Historical Perspective 1

1.2 Motivation 2

1.3 TREC News Track 3

1.4 Goal of the Thesis 5

1.5 Contribution 5

1.6 Thesis Structure 6

2 Literature Review **7**

2.1 Vector Space Models in Information Retrieval 7

2.1.1 Boolean Retrieval 8

2.1.2 TF-IDF Approach 9

2.1.3 Okapi BM25 10

2.2 Graph-based Approach 11

2.3 Dense Document Representation 12

2.3.1 Transformers 12

2.3.2 Very Large Language Models 15

2.4 Approaches to News Background Linking 18

3 Our Approaches **22**

3.1 Data Processing 22

3.2 Doc2Vec Approach 25

3.2.1 Document Representation 25

3.2.2 Proximity Measure 25

3.2.3 Validation 25

3.2.4 Performance and Model Size 27

3.3 Deep Learning Approach 27

3.3.1 Document Representation 28

3.3.2 Chunking 29

3.3.3 Proximity Measures 30

3.3.4 Normalization 31

3.3.5 Validation 32

3.4 Combined Approach 34

4 Results and Analysis **36**

4.1 TREC Runs 36

4.2 TREC Results 37

4.3 Post-TREC 2020 Results 41

4.4 Analysis 42

4.5 Diversity 49

5 Conclusions and Future Work **53**

5.1 Summary 53

5.2 Contributions 53

5.3 Limitations and Future Work 54

Bibliography **56**

Appendix A Distance and Proximity Measures **68**

Appendix B Approaches Used at The TREC News Track (2018-2021) **72**

Appendix C Tools **81**

Appendix D Aggregated Experimental Results **83**

List of Figures

Figure 2.1 Graph constructed for an example article: "Online newspaper is the online version of a newspaper, either as a standalone publication or as the online version of a printed periodical", with a sliding window of size 2 (Essam and Elsayed (2019)). 12

Figure 2.2 Graph decomposition: (a) the main graph. (b) the 3-core decomposition. (c) the 3-truss decomposition. Note that node * is removed in the truss decomposition because its connecting edge is not part of a triangle (Essam and Elsayed (2019)). . 13

Figure 2.3 An example of attention weights for the English to French translation sentence pair (Bahdanau, Cho, and Bengio (2015)). 14

Figure 2.5 The Transformer architecture (Vaswani et al. (2017)) 16

Figure 2.6 Approaches used by participants at the TREC News Track from 2018 to 2021. Highlighted approaches yielded the highest results for the year. 19

Figure 3.1 Example of article representation in the Washington Post dataset 23

Figure 3.2 Length distribution of the articles in the collection 24

Figure 3.3 Architecture and data flow in the system 28

Figure 3.4 Example of the effect of different normalization techniques on two components of the `xlnet-large-cased` embeddings for the validation set. 31

Figure 3.5 Maximum nDCG@5 reached by each distance measure with the 2018 and 2019 data, independent of the embedding method. The top performing measure is marked with the a red "X". 34

Figure 4.1 Official results at the 2019 TREC News Track per topics compared to the min, max and median TREC performances 39

Figure 4.2 Official results at the 2020 TREC News Track per topics compared to the min, max and median TREC performances 40
Figure 4.3 Performance of all 47,332 model configurations on the 2018 queries against the 2019 queries 42
Figure 4.4 Performance of all 47,332 model configurations on the 2018 queries against the 2019 queries color-coded by embedding model family 43
Figure 4.5 Influence of the embedding model parameter on performance 44
Figure 4.6 Performance of GPT and GPT2 models 48
Figure 4.7 Influence of the windows size parameter 49
Figure 4.8 Influence of the normalization parameter 49
Figure 4.9 Influence of the pooling parameter 50
Figure 4.10 Influence of the proximity metric 51
Figure 4.11 Difference in nDCG@5 scores for two pairs of models showing the diversity of background links for each topic 52
Figure C.1 Example of Elasticsearch query with proximity based functionality and query vector (only first and last 4 components of query vector are displayed). `003d:3` is a code for specific proximity function, `embedding_state_last_hidden_mean` is a name of the field storing embedding vectors. 82

List of Tables

Table 3.1 Statistics of the 2020 TREC News document collection 24

Table 3.2 Hyperparameters of Doc2Vec of the top 2 models (`clac_100_cos` and `clac_300_cos`) with the validation set 27

Table 3.3 Top 5 performing models with the 2019 (top sub-table) and 2018 (bottom sub-table) validation datasets and their performance 33

Table 4.1 Official results of our runs at the 2019 and 2020 TREC News Tracks 37

Table 4.2 Results of the 4 submitted runs with the 2018 and 2019 data for validation purposes 38

Table 4.3 Number of topics ranked ≥ or < the collective median for each run 41

Table 4.4 Performance of the top configurations with the validation set, the BM25 model and a new combined model on the entire document collection 41

Table 4.5 Number of experiments with specific parameter fixed and the percentage of experiments which reached nDCG@5 of only 0.1 or less (lower is better). See Appendix D for a complete table. 46

Table D.1 Experiments with BERT as an embedding model and the portion of it which reached nDCG@5 of only 0.1 or less (lower is better). 83

Table D.2 Experiments with RoBERTa as an embedding model. 84

Table D.3 Experiments with PEGASUS as an embedding model. 84

Table D.4 Experiments with GPT as an embedding model. 84

Table D.5 Experiments with XLNet as an embedding model. 84

Table D.6 Experiments with different window size parameters. 85

Table D.7 Experiments with different normalization parameters. 85

Table D.8 Experiments with different pooling parameters. 85

Table D.9 Experiments with different output layers. 85

Table D.10 Experiments with different proximity measure. 88

Chapter 1

Introduction

And because the public good requires that you should be spared as much as possible from all trouble, I have subjoined to this epistle the contents of each of the following books, and have used my best endeavours to prevent your being obliged to read them all through. And this, which was done for your benefit, will also serve the same purpose for others, so that any one may search for what he wishes, and may know where to find it. This has been already done among us by Valerius Soranus, in his work which he entitled "On Mysteries."

— Gaius Plinius Secundus, *NATURALIS HISTORIÆ, AD 77*[a],

one of the first known description of the table of contents

[a]The Natural History. Pliny the Elder. John Bostock, M.D., F.R.S. H.T. Riley, Esq., B.A. London. Taylor and Francis, Red Lion Court, Fleet Street. 1855.

1.1 Historical Perspective

The moment humanity started accumulating information in the form of written texts, it looked for ways to create and improve methods of searching and navigating text collections. You might not realize how far back in history modern text searching concepts have their origins. For example, the term *index*, was used in ancient Rome to refer to a little note attached to a papyrus scroll on which the title and the author of the work were written (Wellisch (1995), p. 205). This allowed one to quickly go through the shelves of stacked scrolls and easily identify works without ever removing scrolls from the stack or opening them.

The first person known to use alphabetical order as a method of organization for words was Zenodotus of Ephesus (circa 325 – 270 BC), the first recorded head librarian of The Great Library of Alexandria, in which an estimated 40,000 to 400,000 papyrus scrolls were stored (Wiegand and Davis (2015), p. 20). The shelves and scripts in the library were ordered by the first letter of the author's name. However, it took about 400 years to use subsequent letters for further ordering, completing the lexicographic ordering we know today (Casson, Penn, and Davis (2001)).

According to Gaius Plinius Secundus (circa 23/24 AD – 24 August 79 AD), also known as Pliny the Elder, the first author to use *a table of contents* in Latin books was Quintus Valerius Soranus (circa 140-130 BC – 82 BC), a Latin poet and grammarian (Murphy and Secundus (2004)). Plinius himself was the author of the largest single work that survived from the Roman Empire, consisting of 37 books, where the entire first book was the table of contents and references to the sources for the rest of the work.

Before computers became available, these were the few tools available for information retrieval, supplemented by library directories and more detailed book indices. Without the help of a knowledgeable librarian, finding a relevant document in a large collection was prohibitive until computers came along.

Nowadays, with the available computational power, we have an abundance of tools and algorithms for information retrieval, with new approaches emerging almost daily. It is possible not only to look for a specific document by title or author name, but also to search by keywords and phrases, look for similar and related documents, automatically build directories, perform analysis of collections, look for anomalies, and automatically generate abstracts or even complete documents and stories.

1.2 Motivation

Given the sheer number of electronic sources of news available today and the variety of topics they cover, some of which could be completely unfamiliar to readers, it is important to develop approaches for the automatic recommendation of contextual information for users to better understand a news article. The recommended articles should be relevant, but also diverse. Indeed, it is also important to pay attention to the diversity of the recommended materials, which should provide richer and more informative contexts, increasing the chances that users will read and appreciate the readings recommended to them.

Readers might also benefit from such a recommendation system as it can help them understand an article on a less familiar subject and save time to manually research relevant content.

In order to address this need, in 2018, the National Institute of Standards in Technology (NIST) has originated an annual shared task called the TREC News Track.

1.3 TREC News Track

To address the need for better related news recommendations, since 2018, the News Track at TREC has proposed two related shared tasks: background linking and entity ranking (Soboroff, Huang, and Harman (2018), Soboroff, Huang, and Harman (2020a), Soboroff, Huang, and Harman (2020b)). The goal of the background linking task is to provide relevant background information to news articles through the identification of related articles. On the other hand, entity ranking focuses on providing a list of names, concepts, artifacts, etc. mentioned in news articles, which will help readers better understand the news. This thesis focuses on the first task: background linking.

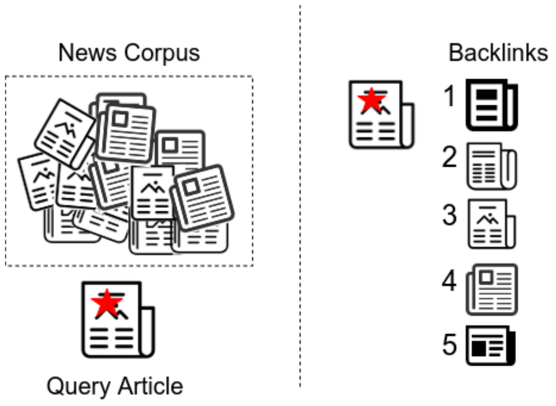


Figure 1.1: The TREC News Track task is focused on finding and ordering by relevancy, backlinks from the WSJ[1] news corpus for a given query article.

[1]The Wall Street Journal https://www.wsj.com/

The background linking task illustrated in Figure 1.1. NIST provides a large collection of news articles (see Section 3.1), and a set of query articles (or search topics) which are themselves articles from the collection. For each query article, participants need to select up to 100 related articles (or backlinks) from the collection and output them as a ranked list from the most related to the least related. Participants can submit several runs to present different systems or various configurations of the proposed system.

The top 50 backlinks from each participant run are pooled together for manual evaluation by NIST assessors. A 5 point score is manually assigned to each related article. The score is an integer between 0 (little or no useful information) and 4 (must appear in recommendations or critical context will be missed). The total score of the system is then computed using the nDCG@5 metric (Järvelin and Kekäläinen (2002)) as follows:

$$nDCG@5 = \frac{\sum_{d=1}^{5} \frac{2^{R(d)}-1}{\log(1+d)}}{IDCG@5} \tag{1}$$

where $R(d)$ is the rank that assessors gave to the document d, and IDCG@5 is the ideal nDCG@5, i.e., the best ranking possible for the query. IDCG@5 not only makes sure that the backlinks with the best scores have been returned, but also that these have been ideally ranked from the most relevant to least relevant. This makes the nDCG@n metric harder to improve compared to the Precision, Recall, and F1-measure.

One important objective of the task is the diversity of suggested articles. However, there are no clear definitions from TREC on what diversity is and how to measure it.

The organizers also provided query articles and their corresponding manually evaluated backlinks with a rank from 0 to 4 from TREC News 2018 (50 topics) and TREC News 2019 (60 topics), which use the same article collection. These previously evaluated topics represent a very small fraction of the collection and cannot be used for training purposes.

This task is performed on very large data collection with little to no labeled data. The proportion of irrelevant articles (99.99%) for a given query article greatly outweighs the number of relevant articles. Since the evaluated articles are pooled from the participant's submissions, if all participants missed a relevant article for a topic, this article will never be ranked. All this and the expectation to have background links in perfect order adds extra complexity to the task.

1.4 Goal of the Thesis

The main goal of this thesis is to develop a news recommendation system using state of the art document representations and compare its performance to classical information retrieval approaches within the context of the TREC News Track.

As our first step, we developed a baseline system using the classical information retrieval method Okapi BM25 (S. Robertson, Walker, Jones, Hancock-Beaulieu, and Gatford (1995)), which is a statistical approach relying on term frequencies in the query document and in the returned documents (see Section 2.1.3). Then we explored the performance of methods based on the latest advancement in deep learning, allowing to represent each document as a fixed size vector along with various proximity measures and how to improve this approach using vector-space normalization. Once we had established the document retrieval pipeline, we focused on the evaluation of the system results diversity, particularly the topic diversity.

1.5 Contribution

In this thesis, we developed and analysed a content recommendation system to assist readers better understand a news article by providing a diverse context in the form of related documents. This led to two conference papers at The TREC News Tracks (Khloponin and Kosseim (2019, 2020)). Our baseline system reached the highest score among all participants at the TREC 2020 News Track. In addition, a deeper analysis of our approaches was published in (Khloponin and Kosseim (2021)) at the NLDB-2021[2] conference.

This thesis presents a number of contributions:

- Through the exploration of a wide variety of proximity functions for vector-space document representation, we have identified more effective alternatives for the commonly used cosine similarity.

[2]http://nldb2021.sb.dfki.de/

- Using multiple modern pretrained deep neural network models, we have shown the difference in their performance and importance of the model size on the final performance.
- We have shown that scaling individual components of the embedding space to have the same amplitude reliably improves the performance of proximity measures. Further investigation helped us to enhance this result by discounting outliers with the help of nonlinear scaling function.
- We have shown that tuning our system parameters on a small amount of labeled data gives a predictable result for specific system configuration retrieving results for unseen queries.
- We have shown that the Okapi BM25 ranking function using the entire query article as a query is still the best approach in terms of performance and computational resources spent.

1.6 Thesis Structure

This chapter presented a historical perspective on the field of Information Retrieval, our motivation to work on the topic, a formal introduction to the shared task of the TREC News Track, our goal for the thesis and our contributions. Chapter 2 will describe work on classic and modern approaches in the field of Information Retrieval and Natural Language Processing that are relevant to our work, and how they were used by other participants in the TREC News Track over the years. Chapter 3 will present and justify the implementation of our approaches. Chapter 4 will show the results and analysis of all hyperparameters of our models and their influence on performance. Finally, Chapter 5 will summarize the thesis and present directions for future work.

Chapter 2

Literature Review

This chapter provides an overview of previous work related to our research. Despite the fact that the TREC News shared task started only in 2018 (Soboroff et al. (2018)), news background linking can be seen as a classic information retrieval problem. Indeed, both tasks involve retrieving a ranked list of documents related to a query from a large document collection. The main difference is that, in the news background linking, the information need is represented by an entire news article and not a hand-crafted user query. For this reason, Section 2.1 will review classic work on document retrieval; Section 2.2 will show a graph-based approach to keyword extraction; in Section 2.3 we will review work on the latest document representation techniques based on Deep Learning, and finally Section 2.4 will take a look at the approaches used at the TREC News Track.

2.1 Vector Space Models in Information Retrieval

According to Manning, Raghavan, and Schütze (2008) "Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)".

The following section will review Vector Space Models (VSM) approaches proposed in IR. The Vector Space Model is a family of models that represent documents and queries in a vector space. Vector components usually correspond to document terms (Manning et al. (2008)).

In the following sections, we will use the following notation:

- $\mathcal{T} = \{ t^1, t^2, ... \}$ is the set of all terms that can appear in a document or a query, where t^i denotes the individual term.
- $\mathcal{D} = \{ D_1, D_2, ... \mid D_j = \{ d_j^s : d^s \in \mathcal{T} \} \}$ is the set of all documents in the corpus, where D_j is a single document composed of individual terms denoted as d_j^s.
- $\mathcal{Q} = \{ Q_1, Q_2, ... \mid Q_z = \{ q_z^s : q^s \in \mathcal{T} \} \}$ is the set of all possible queries, where Q_z is a single query composed of individual terms denoted as q_z^s

For a document D_j, a vector $[w_j^1, w_j^2, ..., w_j^{|\mathcal{T}|}]$ can be constructed, where $w_j^i = 0$ if the term t^i is missing from the document, and $w_j^i > 0$ otherwise. Different weighting schemes can be used to calculate w_j^i. It could simply be 1 if the term appears at least once, this is called the Boolean model (see Section 2.1.1), or a simple count of how many times the term appears in the document (i.e., term frequency), or a more complex weighting scheme such as TF-IDF (see Section 2.1.2) or BM25 (see Section 2.1.3). Having documents and queries represented in a vector space gives us a simple way of comparing them using a variety of metrics, such as the cosine of the angle between them (see Section 3.2.2) or more complex measures (see Section 3.3.3).

2.1.1 Boolean Retrieval

One of the simplest approaches of the vector space model is the Boolean model (Lancaster and Fayen (1973)). The Boolean Retrieval Model treats documents and queries as sets of terms and applies Boolean logic to evaluate a query. The query will return only documents for which the Boolean query evaluates to 1. This makes the Boolean retrieval very simple to implement and easy to interpret its results.

However, this approach has several limitations, which make it not useful for our task directly. The most important limitation is the lack of a ranking of the results. Boolean retrieval simply provides a set of documents that match the query exactly. Another significant disadvantage is that all terms within the query and the document have equal weights. This, of course, is not ideal, as some words (such as stop words) carry less meaning than others (such as nouns or verbs). Finally, if used in the context of the TREC News Track, one would have to generate a query from the article text, which is not a trivial task. A naïve approach would be to use the entire article as the query, but this would likely lead to little to no results; or use something shorter like its title or its first paragraph, which on a large collection might return too many results. All of these leads us to look for more robust approaches devoid of these disadvantages.

2.1.2 TF-IDF Approach

More robust retrieval methods take advantage of the frequency of the query terms in the individual document (TF – term frequency) (Luhn (1957)), and the frequency of the documents in the entire collection that contain this term (DF – document frequency), or more often its inverse value (IDF – inverse document frequency) (Jones (1972)). If, for example, a term Q_1 from a search query appears in 50% of the documents, and term Q_2 only in 1% of the documents, it makes sense to give more priority to term Q_2, as it is more discriminating and will narrow down the results more than Q_1. In addition, if we have a document D_1 with only one occurrence of a query term, and document D_2 with multiple occurrences of the same term, then D_2 should be ranked higher for that term, as it describes the query terms in more detail. Combining these two principles leads to the TF-IDF weighting scheme (Salton and Buckley (1988)).

Given a query Q_z and a document collection $\mathcal{D}$, the document rank for document D_j is calculated by summing the individual ranks for each term q_z^s in query Q_z:

$$tfidf(Q_z, D_j, \mathcal{D}) = \sum_{q_z^s \in Q_z} tfidf(q_z^s, D_j, \mathcal{D}) = \sum_{q_z^s \in Q_z} tf(q_z^s, D_j) \times idf(q_z^s, \mathcal{D}) \tag{2}$$

where:

$$tf(q_z^s, D_j) = \frac{|\{ d_j^k \in D_j \mid d_j^k = q_z^s \}|}{|D_j|} \tag{3}$$

$$idf(q_z^s, \mathcal{D}) = log\left(\frac{|\mathcal{D}|}{|\{ D_j \in \mathcal{D} \mid \exists\, d_j^k \in D_j : d_j^k = q_z^s \}|} \right) \tag{4}$$

As we can see from Equations (2), (3) and (4) the TF-IDF retrieval model addresses some of the limitations of the Boolean retrieval. Instead of the binary decision boundary of Boolean retrieval, TF-IDF provides a continuous decision boundary (i.e. a ranking) for every document for a given query. It also gives different priorities to terms in the query, increasing weights higher for rare words in the collection, and words repeated more times in the document. This approach also allows us to build a search query from a query article by extracting only words with higher idf scores.

On the other hand, TF-IDF works well on documents of similar size (titles, abstracts, library index cards, etc.) but will produce less comparable ranks if some documents are very long or very short. This drawback is significant for us because in the TREC News Track, some articles can be 10 times longer than others (see Section 3.1). The Okapi BM25 model addresses this issue.

2.1.3 Okapi BM25

One of the most widely used weighting schemes in information retrieval is BM25, also known as Okapi BM25, named after the Okapi system it was initially implemented in (S. Robertson et al. (1995)). BM25 combines simplicity and high efficiency on different corpora at the same time. Being a "bag-of-words" approach, it does not know the internal structure of a document, but it manages to capture enough information to work very well without much tuning. Similarly to TF-IDF, for a given query, BM25 will assign a score to each document in the collection. Documents with higher scores are expected to be more relevant to the query. Given a query Q_z and a document D_j, its BM25 score will be computed using Formula 5,

$$BM25score(Q_z, D_j) = \sum_{q_z^s \in Q_z} idf(q_z^s) \times \frac{f(q_z^s, D_j)(k1 + 1)}{f(q_z^s, D_j) + k1(1 - b + b \times \frac{|D_j|}{avg(\{|D_s| : D_s \in \mathcal{D}\})})} \tag{5}$$

the IDF component is computed as:

$$idf(q_z^s) = log\left(1 + \frac{|\mathcal{D}| - f(q_z^s) + 0.5}{f(q_z^s) + 0.5}\right) \tag{6}$$

where $f(q_z^s, D_j)$ is the frequency of the token q_z^s in the document D_j, and $f(q_z^s)$ is the number of documents containing the token q_z^s.

Formulas (5) and (6) are more robust variations of Formulas (3) and (4) of the TF-IDF weighting scheme (see Section 2.1.2). Compared to TF-IDF, BM25 adds normalization to the average length of the document, applies smoothing for idf, and provides additional tuning coefficients such as $k1$

and b which are used to adjust the scoring function to the collection of documents. It might be useful to adjust them for collections of documents of unusual length or if the sensitivity to token frequency in documents needs to be adjusted.

Despite the popularity of BM25, it has its weaknesses. When formulating queries, we often use phrases or whole sentences. Just like the Boolean retrieval and the TF-IDF scheme, BM25 is a bag-of-words approach, which means that it does not take into account the position of the terms in the query and the documents; only their presence is considered, which makes it less useful to look for specific terms in context. In addition, BM25 only considers terms from the query. Query expansion with synonyms and spelling corrections are not considered. Finally, according to the work of (Lv and Zhai (2011)), BM25 could also overly penalize very long documents in the collection.

Given the success of BM25 in previous work (eg. Bimantara et al. (2018)), in this project we have used the Elasticsearch[1] implementation of BM25 as a baseline (see Section 4.1).

2.2 Graph-based Approach

Another approach that yielded a high performance at the TREC News Track (nDCG@5 of 0.5918 in 2019) is based on query construction using a graph-based system in combination with Apache Lucene[2](Essam and Elsayed (2019)). Bigrams taken from the query articles are used to construct a co-occurrence graph with weighted edges (see Figure 2.1). Every co-occurrence increases the weight of the edge between two words. The built graph is then pruned by removing edges with weights smaller than a given minimal threshold (called core decomposition, see Figure 2.2 (b)), or by removing nodes not at the apex of triangles (called truss decomposition, see Figure 2.2 (c)).

These operations filter out less important terms and preserve only cohesive terms, often clustered into several groups (cores or trusses). The preserved terms are extracted along with the sum of weights with its neighbors. These weighted query terms are then used to run a query on Apache Lucene. The hypothesis behind this approach is that the graph helps to find not only frequent terms but also co-occurring terms.

[1] https://www.elastic.co/elasticsearch/
[2] https://lucene.apache.org/

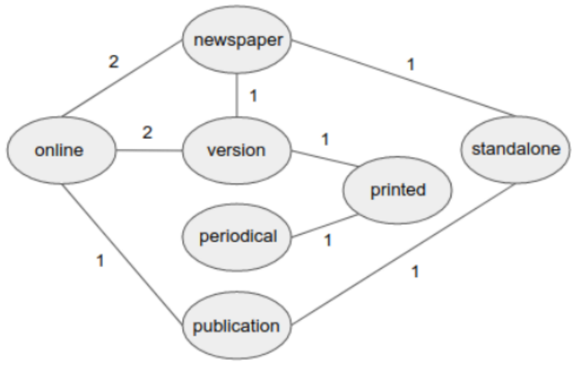


Figure 2.1: Graph constructed for an example article: "Online newspaper is the online version of a newspaper, either as a standalone publication or as the online version of a printed periodical", with a sliding window of size 2 (Essam and Elsayed (2019)).

2.3 Dense Document Representation

Section 2.1.1 has described the vector space model in which documents are represented as a sparse vector of size $|\mathcal{T}|$, where $\mathcal{T} = \{t^1, t^2, ...\}$ is the set of all terms t^i in the collection, and each component of the vector is the weight of a term t^i. This classic IR document representation method allows us to perform many operations on document collection. However, the document vectors built this way are very large and mostly contain zeros. More advanced dense vector representations were developed to represent words (eg. Word2Vec by Mikolov, Chen, Corrado, and Dean (2013)). This representation not only provides embeddings for individual words, but also able to distinguish words based on their context. Subsequent work by Mikolov, Sutskever, Chen, Corrado, and Dean (2013) allowed to build embeddings for phrases and then for an entire document (eg. Doc2Vec Le and Mikolov (2014)). More advanced document embedding approaches use Transformers (Vaswani et al. (2017)) and their derivatives.

2.3.1 Transformers

Attention-based models are becoming increasingly effective in a variety of NLP architectures (Galassi, Lippi, and Torroni (2021)). Attention in machine learning is designed to mimic cognitive attention by increasing the weights of important parts of the model's input and muting less relevant information. Attention was introduced in the field of NLP for the task of machine translation by Bahdanau et al. (2015) for the encoder-decoder family of models. In such models, the encoder processes

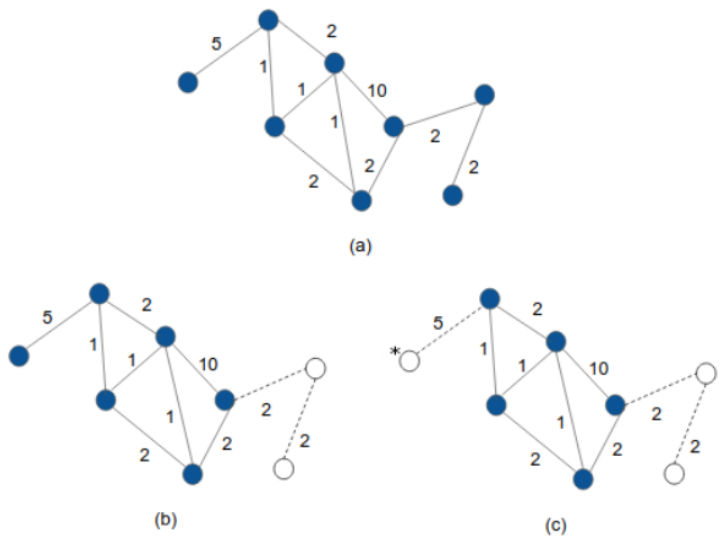


Figure 2.2: Graph decomposition: (a) the main graph. (b) the 3-core decomposition. (c) the 3-truss decomposition. Note that node * is removed in the truss decomposition because its connecting edge is not part of a triangle (Essam and Elsayed (2019)).

the input text sequence into a dense vector of fixed length, and then this vector is used by the decoder to generate the output sequence. Bahdanau et al. presumed that the fixed size vectors could be a bottleneck that limits further performance improvement of such models, and the models could benefit from allowing the decoder to automatically search for relevant words in the input text sequence when predicting a target word. Figure 2.3 illustrates the attention weights for aligned sentences in the task of the machine translation from English to French. An English sentence "`The agreement of the European Economic Area was signed in August 1992.`" is translated into French "`L'accord sur la zone économique européen a été signé en août 1992.`". Both sentences in the figure are aligned, and the brightness of the pixels corresponds to the attention weight the decoder applies to the English input sequence to generate the French output sequence. For this almost word-to-word translation example, the attention weight matrix is almost diagonal with the exception of 3 words, which have an inverse order in French compared to English.

Self-attention or intra-attention, on the other hand, relates words from the same sequence and provides a 2-D matrix for the sequence embedding instead of a vector. In particular, if we take a sentence "`The animal didn't cross the street because it was too tired`", attention weights for the word "`animal`" and for every word in the sentence compose one row of

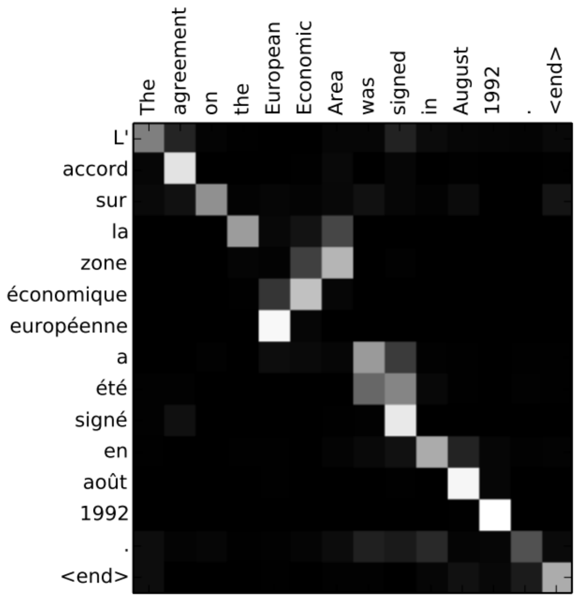


Figure 2.3: An example of attention weights for the English to French translation sentence pair (Bahdanau et al. (2015)).

the attention matrix where the brighter color indicates a higher attention weight. The self-attention matrix looks similar to the one depicted in Figure 2.4 but the input and output sequences will be the same, making the matrix square.

Self-attention, according to Lin et al. (2017), who introduced it in the sentence embedding task, improves the internal representation of the input sequence and makes it more interpretable. These facts will be important for the implementation of our approach, as we are directly relying on document embeddings to find relevant articles (see Section 3.3).

Vaswani et al. (2017) proposed the Transformer model, which is solely relying on an attention mechanism to draw global dependencies between input and output sequences. The Transformer architecture is shown in Figure 2.5. It consists of the encoder part on the left and the decoder part on the right.

Encoder The encoder receives token embeddings as its input, applies positional encoding, and passes the result through $N = 6$ identical layers. Each of these layers contains two sublayers. The first sublayer is a multi-head self-attention mechanism, which is just 8 independently trained self-attention heads. The second sublayer is a position-wise fully connected feedforward network. Both

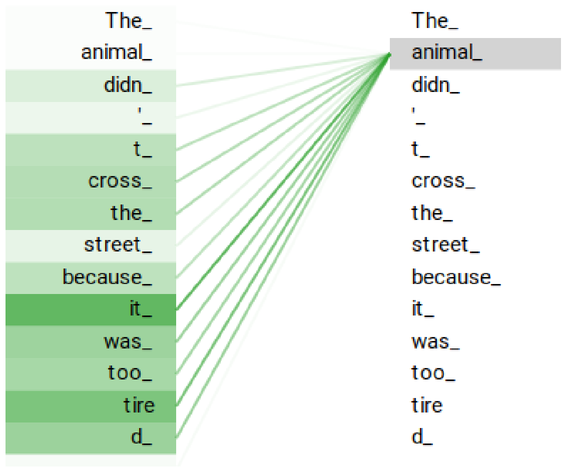


Figure 2.4: An example of self-attention weights for the word "`animal`" in the sentence "`The animal didn't cross the street because it was too tired`"[3]

parts have a residual connection followed by a normalization layer.

Decoder The decoder also consists of $N = 6$ identical layers, but in addition to the two sublayers described in the encoder part, it has a third sublayer, which applies multi-head attention to the encoder output. The decoder also benefits from having previously generated output token as its additional input.

The Transformer model and its derivatives have shown strong performance in many NLP tasks since they were introduced (Kalyan, Rajasekharan, and Sangeetha (2021)) and gave rise to a new generation of powerful language models.

2.3.2 Very Large Language Models

Having a large corpus of documents in a specific language allows one to build a model which can learn word distributions in that language. Very large language models have been created using Transformers methodology. The most important of such transformer-based language models are the following:

BERT stands for Bidirectional Encoder Representation from Transformers (Devlin, Chang, Lee, and Toutanova (2018)). BERT is almost identical to the Transformer model of Vaswani et al. (2017),

[3]Figure generated by code from Tensorflow project https://github.com/tensorflow/tensor2tensor

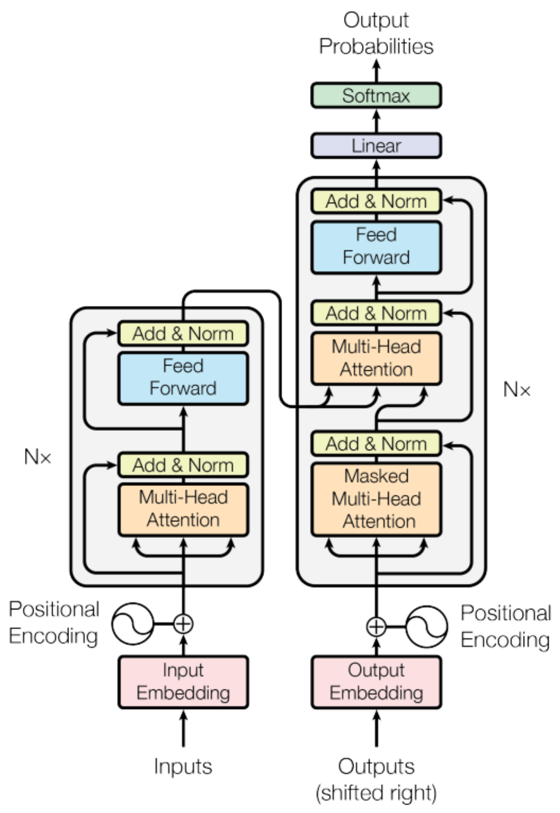


Figure 2.5: The Transformer architecture (Vaswani et al. (2017))

but uses only the encoder part of the Transformers architecture, and instead of transducing a sequence to a sequence, BERT is more focused on building an internal representation of an input text, which is then fed to a task-specific model. It is pretrained on unlabeled texts containing 800M words from BooksCorpus (Zhu et al. (2015)) and about 2.5B words from English Wikipedia. The pretraining consists of two tasks: randomly masking and predicting 15% of the words in the text and predicting whether two given sentences follow each other in the document or not. BERT also has a variety of pretrained models available for tuning on a custom task. In the context of the TREC News Track, BERT has been used by a number of participants, such as (Ak, ahan Kksal, Fayoumi, and Yeniterzi (2020); Ding et al. (2019); Essam and Elsayed (2021); Lirong, Joho, and Fujita (2021)).

GPT or Generative PreTraining (Radford and Narasimhan (2018)) is a Transformer-based model containing 117M parameters, 12 encoder-decoder layers, 12 attention heads, batch size of 64 tokens, and 512 token context window. First, the model is passed through unsupervised generative pretraining, when it generates the next word for a given text fragment on a corpus with long stretches of contiguous text. This allows the generative model to condition on information stretched over a long context. After the pretraining model passed through supervised task-specific fine-tuning.

GPT-2 (Radford et al. (2019)) is a successor of the GPT model and largely follows the details of the GPT model, with 1.5 billion parameters, 12 encoder-decoder layers, 12 attention heads, batch size of 512 tokens, and context window of 1024 tokens.

RoBERTa A Robustly Optimized BERT Approach. Liu et al. (2019) performed a replication study of the BERT model, carefully observing the impact of many key hyperparameters. Compared to BERT, RoBERTa is trained using 10 times more training data (about 160GB of text), with longer pretraining (500K steps) and a larger batch size (8K compared to 256 for BERT). Using the same architecture as BERT the authors were able to achieve state-of-the-art results on GLUE (Wang et al. (2019)), SQuaD (Rajpurkar, Jia, and Liang (2018); Rajpurkar, Zhang, Lopyrev, and Liang (2016)) and RACE (Lai, Xie, Liu, Yang, and Hovy (2017))

XLNet model (Z. Yang et al. (2019)) is based on Transformer-XL (Dai et al. (2019)) which enables the capture of longer-term dependencies, approximately 450% longer than the vanilla Transformers. XLNet also overcomes the issue of BERT relying on corrupting input with mask, ignoring not only the masked word but also the dependencies between the masked positions by using autoregression. The largest model ($XLNet_{Large}$) has 340M parameters, 24 encoder-decoder layers, 16 attention heads, batch size of 8192, and context window of 512 tokens.

PEGASUS or Pretraining with Extracted Gap-sentences for Abstractive Summarization. Zhang, Zhao, Saleh, and Liu (2020) proposed the PEGASUS Transformer-based encoder-decoder model pretrained on a massive text corpora: C4 (Colossal Cleaned Crawled Corpus) (Raffel et al. (2019)) of about 750GB of text, and HugeNews (Zhang et al. (2020)) of about 3.8TB of text; where important sentences are masked in an input document and generated as single-model output similar to an extractive summary, but without actually seeing these sentences in the input document.

$PEGASUS_{Large}$ was pretrained with 568M parameters, 16 encoder-decoder layers, 16 attention heads, batch size of 8192, and context window of 512 tokens.

In order to take advantage of all these large-scale language models, we experimented with each in our work (see Section 3.3).

2.4 Approaches to News Background Linking

A summary of all the approaches used at the TREC News Track over the years is presented in Figure 2.6. This figure groups the main approaches or algorithms used by all participants for each year and was built based on the 29 papers available in the proceedings of the shared tasks for years 2018 (Soboroff et al. (2018)), 2019 (Soboroff et al. (2020a)), 2020 (Soboroff et al. (2020b)) and 2021 (Soboroff (2021)). In Figure 2.6, the best performing approaches for each year are indicated in bold, while the approaches explored in our work are marked with (∗). The figure also shows which year a specific approach was first attempted and which year it was reused by any of the participants. Some connections between approaches have been omitted to simplify the diagram. A more detailed description of each approach shown in Figure 2.6 is available in Appendix B.

As Figure 2.6 shows, most of the 2018 approaches were based on already existing tools and classical information retrieval methods (see Section 2.1). For example, the use of BM25 from Elasticsearch or Lucene allowed to quickly setup a baseline for the task. Participants in the 2019 task explored graph methods (see Section 2.2), ways of turning words or documents into vectors, and using BERT (see Section 2.3.1) to assist with query evaluation, as well as reusing most of the previous year approaches. The year 2020 brought more variations of BERT based models, transformers (see Section 2.3.1), and more ways to encode documents in vector form. The year 2021 was more focused on reusing and recombining approaches from the previous years.

As shown in Figure 2.6, in the first year of the TREC News Track, the main focus was on classical IR approaches and widely available IR tools. Statistical-based approaches such as TF-IDF, BM25, or combinations of these were by (e.g.Bimantara et al. (2018), Naseri, Foley, and Allan (2018)). Most approaches also used off-the-shelf tools and libraries, explored feature extraction, reranking, and combining with other models.

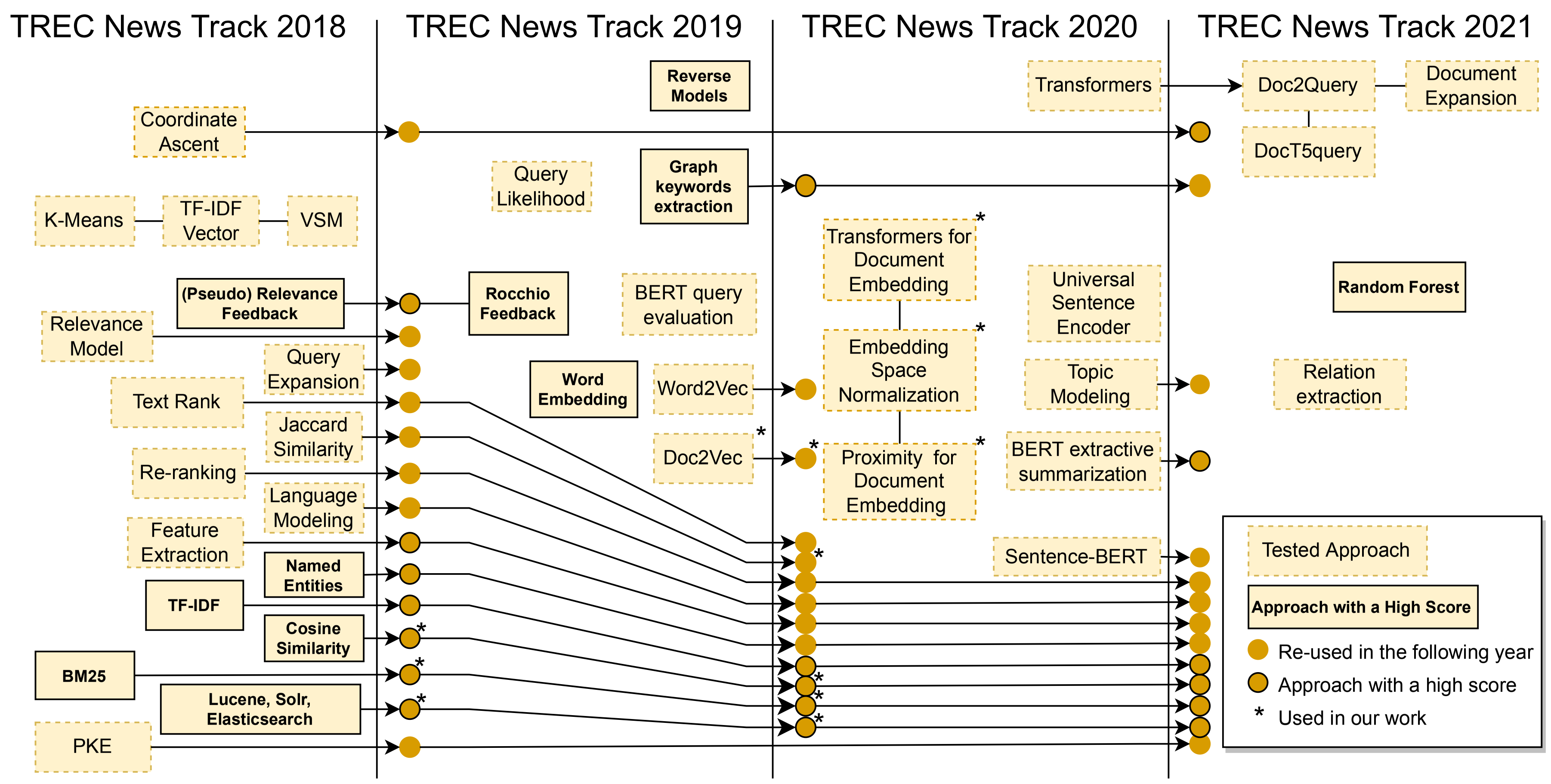


Figure 2.6: Approaches used by participants at the TREC News Track from 2018 to 2021. Highlighted approaches yielded the highest results for the year.

P. Yang and Lin (2018) focused on the standardization and reproducibility of the results. They created the open-source information retrieval toolkit, called Anserini, as part of their work. Despite the fact that they used the Lucene library and the BM25 ranking scheme as the backbone of their model, they explored a wide range of different query generation approaches. The simplest approach of selecting terms from a query article and forming a weighted query ended up being the most efficient one.

Other classic IR approaches included relevance models (e.g. Naseri et al. (2018), Lavrenko and Croft (2001)) and probabilistic models (e.g. Lu and Fang (2018), Lu and Fang (2019)).

Another successful common approach was to extract named entities from articles and use them as extra features to retrieve relevant documents (e.g., Boers, Kamphuis, and de Vries (2020); Cabrera-Diego, Boros, and Doucet (2021); Day, Worley, and Allison (2020); Engelmann and Schaer (2021)).

Several models have also experimented with reranking the results and using relevance feedback (Ak et al. (2020); Bimantara et al. (2018); Boers et al. (2020); Cabrera-Diego et al. (2021); Engelmann and Schaer (2021); Essam and Elsayed (2019, 2021); Koster and Foley (2021); Lirong et al. (2021); Qu and Wang (2019)) based on the idea that the end users' behavioural information can provide additional useful information to rank relevant documents. In particular, Okura et al. (2017) used information on the frequency of user clicks on the initially provided links to rerank documents; while Adomavicius et al. (2005) used these data for collaborative filtering. Although relevance feedback has been shown to improve background linking, in the context of the TREC News task, this information is not available.

Some teams applied language models and vector space representation of sentences or text, Deshmukh and Sethi (2020) used BM25 for an initial retrieval of the documents, then used SBERT (Sentence BERT Reimers and Gurevych (2019)) to perform semantic similarity reranking between the query article and the articles retrieved by BM25. For this, a list of keywords is extracted from each returned article and from the query article. These lists of keywords are then treated as sentences, and SBERT embeddings are built for them. Finally, the cosine similarity is then computed between these embeddings and used to rerank the returned articles. In Day et al. (2020), the authors used a similar approach by first retrieving the initial results with the Elasticsearch implementation of BM25, and using SBERT to directly build embeddings for the first three paragraphs of each article before averaging them to get a final embedding for the article and applying the cosine similarity for the final ranking. On the other hand, MacAvaney, Yates, Cohan, and Goharian (2019) reached promising results using BERT, ELMo and GloVe embeddings on the ad-hoc document ranking task. Based on the BM25 relevance score, they selected positive and negative pairs for training queries, which were then used to fine-tune the embedding models and train the classifiers. Another interesting approach was used in the News2Vec system (Ma et al. (2019)) where the authors proposed a distributed representation of news based on news-specific features such as: extracted named entities; sentiment; month, publication week and day; word count, paragraph count, etc.

As shown in this chapter, most previous work on background linking is based on information retrieval approaches. To our knowledge, very little work has investigated the use of novel language models for the background linking task. Given the recent successes of very large neural language models such as GPT (Radford and Narasimhan (2018)), GPT2 (Radford et al. (2019)), XLNet (Z. Yang et al. (2019)), BERT (Devlin et al. (2018)) and RoBERTa (Liu et al. (2019)) (see Section 2.3.2), we wanted to experiment with different document embedding representations and proximity measures as an alternative or a complement to classic information retrieval approaches. The expectation is that related articles would be closer to each other in embedding space, and using language models for document representation, vector distance metrics would be a good approximation of document content relatedness. In the next chapter, we will describe our approach in detail.

Chapter 3

Our Approaches

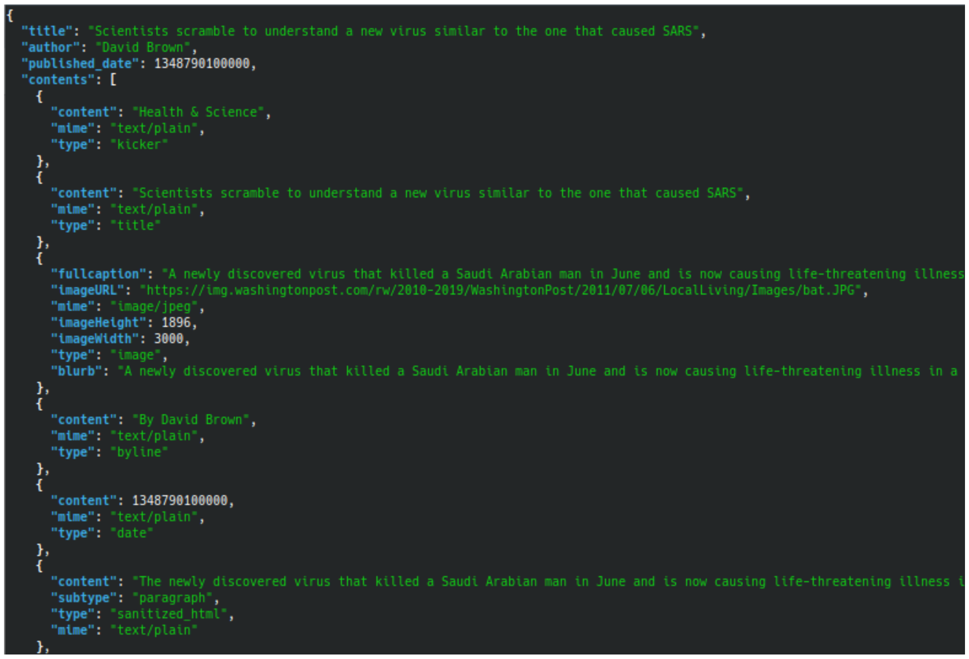
In this chapter, we will describe the details of the different approaches we have implemented for news recommendation. We will start with the steps that are common to all approaches, that is, data preprocessing and text extraction. Then we will focus on each approach separately.

3.1 Data Processing

The initial TREC News document collection consisted of 608,180 news articles from The Washington Post published between 2012 and 2017. The latest version of document collection is the same as the one provided in previous years (at the 2018 and 2019 editions), but with duplicate articles removed and with new articles from 2017 to 2019 added, and consists of 671,934 news articles. Each news article is stored in "JSON-lines" format and is represented as a single line of JSON[1]. Each document contains eight types of metainformation: `id, article URL, title, author, publication date, type` (blog post or article), `news source`, and `content field`.

Preprocessing Apart from the `id` field, which was used for identification purposes, we only considered the `article title` and the text extracted from the `content` field. The `title` was prepended to the content and processed as a single document. The `content` itself was stored in a form of content blocks, where each content block can be a text paragraph, an image, a video,

[1] https://developer.mozilla.org/en-US/docs/Glossary/JSON



```
{
  "title": "Scientists scramble to understand a new virus similar to the one that caused SARS",
  "author": "David Brown",
  "published_date": 1348790100000,
  "contents": [
    {
      "content": "Health & Science",
      "mime": "text/plain",
      "type": "kicker"
    },
    {
      "content": "Scientists scramble to understand a new virus similar to the one that caused SARS",
      "mime": "text/plain",
      "type": "title"
    },
    {
      "fullcaption": "A newly discovered virus that killed a Saudi Arabian man in June and is now causing life-threatening illness
      "imageURL": "https://img.washingtonpost.com/rw/2010-2019/WashingtonPost/2011/07/06/LocalLiving/Images/bat.JPG",
      "mime": "image/jpeg",
      "imageHeight": 1896,
      "imageWidth": 3000,
      "type": "image",
      "blurb": "A newly discovered virus that killed a Saudi Arabian man in June and is now causing life-threatening illness in a
    },
    {
      "content": "By David Brown",
      "mime": "text/plain",
      "type": "byline"
    },
    {
      "content": 1348790100000,
      "mime": "text/plain",
      "type": "date"
    },
    {
      "content": "The newly discovered virus that killed a Saudi Arabian man in June and is now causing life-threatening illness i
      "subtype": "paragraph",
      "type": "sanitized_html",
      "mime": "text/plain"
    },
```

Figure 3.1: Example of article representation in the Washington Post dataset

a tweet, a citation, etc. Each content block itself may contain metainformation (up to 133 different fields), such as `MIME-type, type, kicker (category), content, subtype, source, URLs`, etc. Based on this metainformation we identified blocks with frequently appearing content types and checked if they have any useful text descriptions, and kept for further processing only blocks with paragraphs, image captions, headers, and quotes. Blocks were further cleaned from embeds, links, images, and other HTML tags, preserving only plain text.

Statistics NIST required participants to ignore wire articles, editorial content and opinion posts, which have "Opinion", "Letters to the Editor", or "The Post's View" values in the content metainformation block with type "kicker". Due to this, the initial set of 671,947 articles was reduced by 2,057 to 669,890 items. This is shown in Table 3.1.

As also shown in Table 3.1, many sandbox articles (content previews or articles demonstrating website functionality) were discovered during data exploration: 1,112 articles with a URL path starting with "/test/wp/", presumably indicating content taken from the section of the website not intended to be public (content playground), and 86 documents with "Lorem ipsum..." (common

original number of articles	671,947
articles to ignore as per NIST requirements	2,057
Test articles discovered (preserved)	1,112
"Lorem ipsum" articles (preserved)	86
articles used to build the models	669,890
average size of article before preprocessing (characters)	10,391
average size after preprocessing (characters)	4,533
average size after preprocessing (tokens)	945

Table 3.1: Statistics of the 2020 TREC News document collection

placeholder text) content in the article text.

Articles were preprocessed, HTML markup, and other metainformation was removed and only article text was preserved. As shown in Table 3.1, the 669,890 documents considered have an average length of 10,391 characters prior to preprocessing, but only 4,533 after preprocessing. Figure 3.2 shows the distribution of the article lengths before and after preprocessing. Some articles are composed almost entirely of metainformation and have very little text inside (short statements, video players, cited tweets, etc.), while others have very long texts (transcripts of debates, conferences, testimony, crime reports). The majority of the articles have length between 1,000 and 10,000 characters.

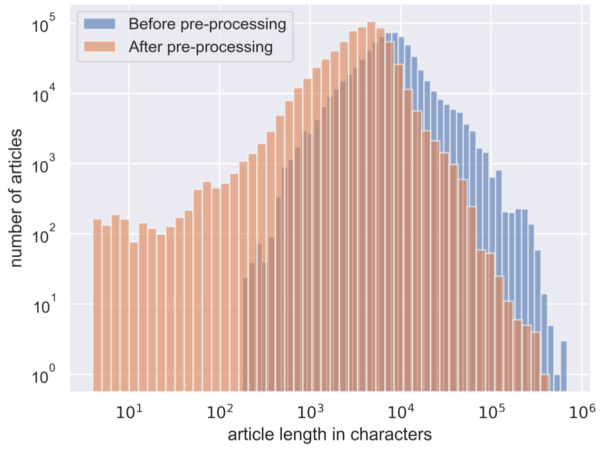


Figure 3.2: Length distribution of the articles in the collection

3.2 Doc2Vec Approach

3.2.1 Document Representation

After preprocessing, we used the Doc2Vec distributed representation Le and Mikolov (2014) to represent each document in the collection. The rationale for this choice was our expectation that related articles would be closer to each other in the embedding space than unrelated articles. Hence, the document embedding distance would be a good approximation of document content relatedness. Query articles and manually ranked documents from 2018 (see Section 3.2.3) were used as a validation set to select among different combinations of parameters for text processing and model parameters. For creating the document vectors: stopwords were filtered using the NLTK English stopwords list Bird, Klein, and Loper (2009), numbers were replaced with the "NUM" token, and the text was case folded. Embeddings were built using Doc2Vec PV-DM embedding with 10 epochs. Using a different size of embeddings significantly changed the produced backlinks and their position in the list. Two different sizes for embeddings were selected: 100 and 300.

3.2.2 Proximity Measure

After obtaining the embedding weights for each article, the proximity between two document vectors were computed. Due to the lack of resources, only the cosine distance was used. Given two document vectors α and β, the cosine of the angle θ between α and β is computed as:

$$cos(\theta) = \frac{\alpha \cdot \beta}{||\alpha||||\beta||} = \frac{\sum_{i=1}^{n} \alpha_i \beta_i}{\sqrt{\sum_{i=1}^{n} \alpha_i^2}\sqrt{\sum_{i=1}^{n} \beta_i^2}}$$

where values close to -1 will, hopefully, correspond to documents with opposite meaning, close to 0 to documents on uncorrelated topics and close 1 to documents with similar topics. The cosine value is directly used as the relevancy score in the final output of the system.

3.2.3 Validation

Several Doc2Vec hyperparameters had to be selected. These include: the training algorithm (such as distributed memory or distributed bag of words), the learning rate, the vector size, the maximum distance between the current word and predicted word within a sentence, the maximum number of epochs, the low-frequency words threshold, the high-frequency words down-sample, negative sampling, and the different tokenizer applied. In order to set the values of these hyperparameters, the models were evaluated with the 2018 TREC News document collection along with 50 query articles and their corresponding manually evaluated results (that is, all articles evaluated manually with a rank from 0 to 4). We used these 50 articles and 8508 evaluated backlinks for validation purposes. For each combination of parameters several epochs of training process performed. In most cases, 10 to 20 epochs of training led to the best result after 20 epochs model performance stopped increasing or deteriorated.

To validate our models, we generated the top 5 backlinks for each query article and computed nDCG@5 when compared with the 2018 dataset. If a generated backlink was not listed in the 2018 validation set, we used a strict evaluation and assigned it a rank of 0 (not relevant backlink) to calculate nDCG@5. As shown in Table 3.2 the model with embedding size of 100 achieved an average nDCG@5 of 0.3378, and with an embedding size of 300 achieved average an nDCG@5 of 0.3032. The fact that about 30% of the returned links did not appear in the validation set makes performance only lower bounds to the scores they would have achieved with an unseen test set. Unfortunately, these close scores and high number of missing backlinks do not give us confidence about which of the models would perform better with the 2019 query articles.

The optimisation was performed in several steps on the 2018 data using random search. At every step, the hyperparameters with the best score were selected and new hyperparameters were generated around them until improvement stopped. Finally, the 2 best models with the validation set were kept. These are referred to as `clac_100_cos` and `clac_300_cos` and are based on the models described in Section 3.2.1 and the proximity measure from Section 3.2.2 with embeddings sizes 100 and 300 respectively. Details of the hyperparameters used to build these models are shown in Table 3.2.

hyperparameters	**`clac_100_cos`**	**`clac_300_cos`**
training algorithm	distributed memory	distributed memory
learning rate	0.05	0.05
vector_size	100	300
maximum distance	5	5
epochs	10	10
low frequency words threshold	1	5
high frequency words down-sample	10^{-4}	10^{-4}
negative sampling	5	5
random seed	100	100
nDCG@5	0.3378	0.3032

Table 3.2: Hyperparameters of Doc2Vec of the top 2 models (`clac_100_cos` and `clac_300_cos`) with the validation set

3.2.4 Performance and Model Size

The models were built in Python 3 with the Gensim library on a desktop computer with an Intel® Core™ i7-7800X CPU @ 3.50GHz with 6 cores and 12 threads. One epoch of building the embeddings for the entire collection took about 14 minutes. Hence the total time for 10 epochs is about 2.5 hours. The size of the model on the disk is 1.2GB. Generating results for 60 query articles took 9 seconds, including 6 seconds for the reading model to memory. Generating 3 million backlinks (5 backlinks for each article of the 600,000 articles collection) took about 8 hours. On documents with precomputed embeddings, the model was able to generate backlinks for 28 query articles per second and 16 query articles per second for new articles, including the generation of dense vectors. These low resource requirements make the model applicable in a production environment.

3.3 Deep Learning Approach

In this section, we are covering in detail the implementation of our deep learning approach. Figure 3.3 displays its structure. On top of the figure we can see the data flow. It starts with JSON document collection and a query article (marked with star) as an input for our "Preprocessor" module, which extracts the plain text passed to the "Embedding" module. "Embedding" module, according to the configuration, builds a dense vector of fixed size for each document in the collection, including the query article. Having all documents mapped to the vector space allows "Ranking" module to apply proximity metrics to find the documents closest to the query article. Closest documents are sorted according to their proximity metric, which gives us a ranked list of background links for the query article. This is the system's goal output. Each module: "Preprocessing", "Embedding", and "Ranking" in Figure 3.3 have a set of parameters we want to explore to find the best configuration. For this purpose, we have the "Evaluation" module which calculates nDCG@5 for each combination of configuration parameters.

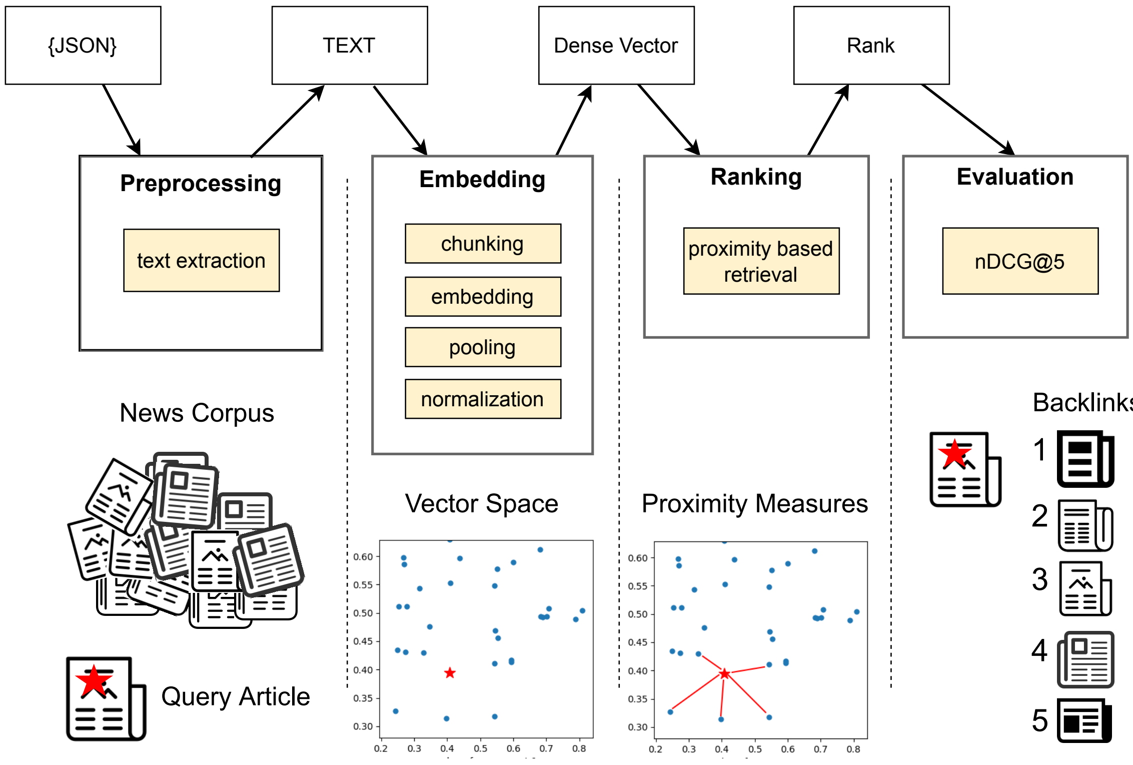


Figure 3.3: Architecture and data flow in the system

3.3.1 Document Representation

After preprocessing (see Section 3.1), we experimented with 5 families of models to represent each document: GPT (Radford and Narasimhan (2018)) & GPT2 (Radford et al. (2019)), XLNet (Z. Yang et al. (2019)), BERT (Devlin et al. (2018)) and RoBERTa (Liu et al. (2019)), PEGASUS (Zhang et al. (2020)) models trained on the Newsroom (Grusky, Naaman, and Artzi (2018)) and Multi-News (Fabbri, Li, She, Li, and Radev (2019)) datasets. In addition, for each family of models, we experimented with a variety of specific pretrained models without fine-tuning. In all cases, before creating the document vectors, metainformation from the text was removed and Unicode characters were normalized using the NFC form of the Unicode Standard. We used the following 20 models available from Hugging Face.

6 BERT models: `bert-base-multilingual-cased`, `bert-base-multilingual-uncased`, `bert-large-cased`, `bert-large-uncased`, `bert-base-cased`, and `bert-base-uncased`.

5 GPT & GPT2 models: `openai-gpt`, `gpt2`, `gpt2-large`, `gpt2-medium`, and `gpt2-xl`.

5 RoBERTa models: `roberta-large-openai-detector`, `roberta-base-openai- detector`, `distilroberta-base`, `roberta-base`, and `roberta-large`.

2 XLNet models: `xlnet-base-cased`, and `xlnet-large-cased`.

2 PEGASUS models: `google-pegasus-multi_news`, and `google-pegasus- newsroom`.

3.3.2 Chunking

The above models have a maximum input sequence size and cannot receive the entire article text as input in most cases. To overcome this limitation, the tokenized article content was split into chunks of sequential tokens. When a chunk border falls in the middle of a sentence, instead of splitting it across chunks and potentially losing its meaning for embedding, an overlapping of 64 tokens was introduced. It was sufficient to have the entire sentence in the next chunk, in case it was split in the previous one. We also experimented with sentence-aligned splitting when a chunk contains only whole sentences, but it did not improve performance. We build embeddings with chunks of 500 and 250 tokens. Model-specific padding tokens were added to some chunks to match the chunk size. The resulting chunks were used as input to the models. Once the embedding vectors for the chunks from a specific article were built, they were pooled to create the final embedding vector for the entire article. We explored three pooling methods: min, max, and mean pooling. Given the deep convolution nature of the models, we evaluated the embeddings pulled from the last hidden layer of the model, as well as from the pooler output layer of the BERT and RoBERTa models.

3.3.3 Proximity Measures

After obtaining the embeddings for the articles, the proximity between two document vectors is computed. To do this, we explored a broad range of proximity measures. We experimented with all 62 different measures presented in Cha (2007). These measure are grouped into 9 families:

(1) L_p **Minkowsky** including Euclidean, Chebyshev …

(2) L_1 **family** including Srensen, Gower …

(3) **Intersection** including Wave Hedges, Czekanowski, Ruzicka …

(4) **Inner product** including Jaccard, Cosine, Dice …

(5) **Fidelity or Squared-chord families** including Bhattacharyya, Matusita …

(6) **Squared L_2 or χ^2 families** including Squared Euclidean, Pearson χ^2…

(7) **Shannons entropy family** including KullbackLeibler, Jeffreys …

(8) **Combinations** including Taneja, Kumar-Johnson …

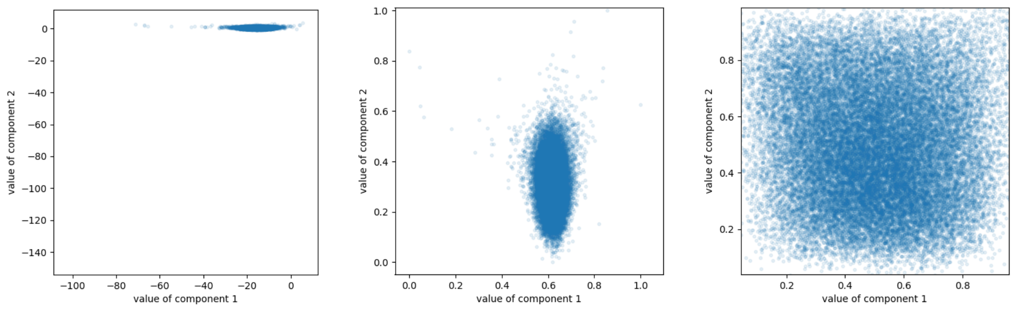
(9) **Vicissitude** including Vicis-Wave Hedges, VicisSymmetric χ^2 …

When taking into account the parameters in some of these measures, in total we experimented with 85 proximity measures. As indicated above, these include the standard Cosine measure, the Jaccard distance, as well as Pearson's χ^2 (see Equation 7) and the Dice distance (see Equation 8) distances:

$$\sum_{i=1}^{n} \frac{(p_i - q_i)^2}{q_i} \tag{7}$$

$$\frac{2\sum_{i=1}^{n} p_i q_i}{\sum_{i=1}^{n}(p_i^2 + q_i^2)} \tag{8}$$

where p and q are embedding vectors for the two documents and n is their vector size.



(a) original (b) amplitude (c) sigmoid

Figure 3.4: Example of the effect of different normalization techniques on two components of the `xlnet-large-cased` embeddings for the validation set.

3.3.4 Normalization

Because the off-the-shelf embeddings are not normalized, the amplitudes of the components in the computed document vectors can differ by several orders of magnitude. In that case, components with larger amplitudes dominate the distance measures and smaller components are not given an opportunity to influence the result. To avoid this problem, we experimented with normalization. Figure 3.4 shows scatter plots of only two components of a set of document embeddings computed from `xlnet-large-cased` embeddings. Figure 3.4(a) shows the original components, while Figure 3.4(b) and Figure 3.4(c) show the same components when the embeddings are normalized to values within the range $[0.0 - 1.0]$. As shown in Figure 3.4, we experimented with two types of normalization: amplitude normalization and sigmoid normalization.

Amplitude normalization To generate embeddings normalized by amplitude, we calculated the minimum value (min_i) and the amplitude ($max_i - min_i$) of each component for each embedding and scaled each value between 0 and 1 by deducting the corresponding minimum from each component and divided it by the amplitude of the component (see Equation 9).

$$vn_i = \frac{v_i - min_i}{max_i - min_i} \tag{9}$$

This type of normalization ensures that all components of the vectors have a value within [0,1], but as shown in Figure 3.4(b), outliers with a very large or very small component value will scale the vector components disproportionately, leaving most of the [0, 1] range unused.

Sigmoid normalization In order to avoid the influence of outliers, we also experimented with sigmoid as a normalization function. For this, we centered all components around their corresponding mean values and divided them by the standard deviations of the component before applying the sigmoid function (see Equation 10).

$$vn_i = \frac{1}{1 + exp(\frac{v_i - \bar{v_i}}{\sigma_i})} \tag{10}$$

As shown in Figure 3.4(c), sigmoid normalization allows for the components to be better spread over the [0,1] range.

For each model, we used the original embedding as well as the normalized embeddings. In total, we experimented with 558 types of embeddings per document. Overall, using different embedding models, pulling methods, output layers, and proximity measures gave us a total of 47,430 model configurations which we run on 2018 and 2019 validation datasets. Due to numerical issues, some combinations of parameters were incompatible leaving us with 47,332 evaluated models. To speed up the experiments, the proximity measures were implemented directly in Elasticsearch.

3.3.5 Validation

For validation purposes, NIST provided participants with the query articles and their corresponding manually evaluated results from the 2018 and 2019 TREC News tracks. From the 2018 edition, we had 50 topics and their manually ranked (from 0 to 4) background links, and from 2019, we had 57 topics and their ranked background links. Recall from Section 3.1 that the past document collection was very close to this year's, so they constituted a representative validation set. We used these 2018 and 2019 sets of topics (107 in total) and evaluated backlinks (22,338 in total) for validation purposes. Note that these backlinks constitute only $\approx$3% of the entire document collection of 669,890 articles (see Table 3.1). To evaluate our 47,332 models, we generated the top 5 backlinks for each topic and computed nDCG@5 with the 2018 and 2019 datasets.

	Embedding	Norm.	Chunk	Pooling	Distance	nDCG@5 2019	nDCG@5 2018
best 2019	`gpt2-xl`	sigmoid	250	mean	Pearson χ^2	**0.5071**	0.3107
	`gpt2-xl`	sigmoid	250	mean	Dice	0.5067	0.2916
	`gpt2-xl`	sigmoid	250	mean	Jaccard	0.5034	0.2919
	`gpt2-xl`	sigmoid	250	mean	Vicis-Symmetric χ^2	0.5018	0.2800
	`gpt2-xl`	sigmoid	250	mean	Probabilistic Symmetric χ^2	0.5004	0.2793
best 2018	`gpt2-medium`	sigmoid	500	mean	Cosine	0.4265	**0.3431**
	`xlnet-large-cased`	sigmoid	250	mean	Additive Symmetric χ^2	0.4345	0.3281
	`gpt2-large`	sigmoid	500	mean	Cosine	0.4868	0.3278
	`gpt2-xl`	sigmoid	250	mean	Cosine	0.4918	0.3269
	`xlnet-large-cased`	sigmoid	250	mean	Jaccard	0.4341	0.3266

Table 3.3: Top 5 performing models with the 2019 (top sub-table) and 2018 (bottom sub-table) validation datasets and their performance

Table 3.3 shows the results of the top models. Although the best performances with the 2018 data is significantly lower than with the 2019 data, comparisons across years cannot be done as the queries and the backlinks are different. Comparisons should be made within the same year. Among all embedding and similarity configurations, GPT2 embeddings outperformed all embeddings and dominated the top 261 best performing configurations with the 2019 dataset, and was among the leading models with the 2018 dataset (although XLNet did perform close to GPT2 models).

As seen from Table 3.3, all the best configurations for 2019 and 2018 have proximity metrics from the Squared L_2 and Inner product families (see Section 3.3.3). The Pearson χ^2 distance achieved the best nDCG@5 with the 2019 data; while the cosine measure dominated the top positions with the 2018 data.

In general, normalization seemed to improve performance. Although, for the best performing models, the improvement was small (≈6%), for some proximity measures, normalization did lead to a more important increase in performance. For example, the model that performed the best with the 2019 data, `gpt2-xl` with the Pearson χ^2 metric without normalization, reached only nDCG@5 of 0.0033 (not shown in the table), but with amplitude normalization it reached nDCG@5 of 0.4790, and with sigmoid normalization it reached 0.5071.

A further comparison of proximity measures is shown in Figure 3.5. The figure shows the maximum nDCG@5 reached by a distance measure regardless of the embedding method used. As the figure shows, the proximity measures seem to perform relatively similarly compared to each

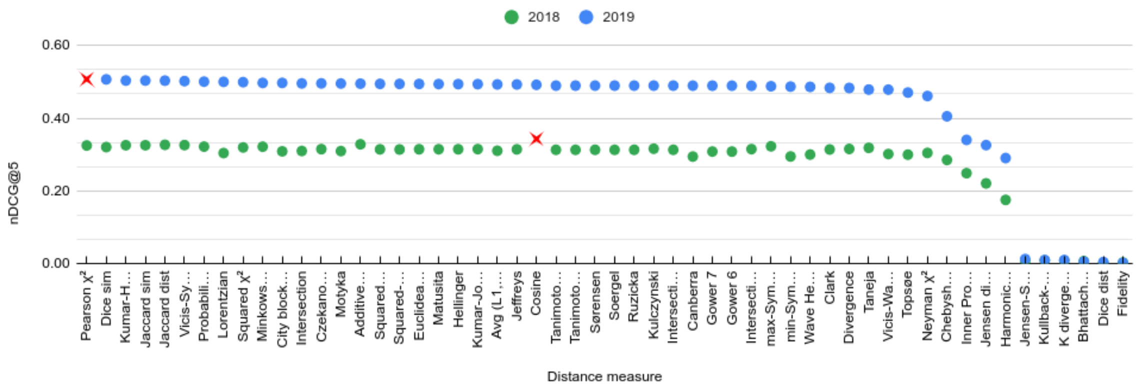


Figure 3.5: Maximum nDCG@5 reached by each distance measure with the 2018 and 2019 data, independent of the embedding method. The top performing measure is marked with the a red "X".

other on both the 2018 and 2019 datasets, but the top performing proximity measures are different.

Recall from Section 3.3.1, that three pooling methods were experimented with to create the document embeddings: min, max and mean pooling. As shown in Table 4.1, all top models use mean pooling. In addition, except for a few Google-PEGASUS models, all 47,430 configurations tested show significantly higher results across all proximity measures when mean pooling is applied.

Finally, we analyze the influence of the chunk size when creating the document embeddings. As indicated in Section 3.3.1, the articles were split into chunks to fit the models' requirements. We expected smaller chunks to decrease performance, because each chunk contain less information, but to our surprise, all but two models from Table 3.3 used the smallest chunk size (250 tokens).

3.4 Combined Approach

After considering the different nature of our baseline model and the deep learning approach we are exploring, we checked if merging them together can improve the result. The idea was to combine two models in such a way that will boost backlink weights on which both models are agreed and have equal votes to decide on backlinks on which they disagree. Taking into account the implementation limitations and performance requirements of Elasticsearch, and time limitation for the submission of the TREC News Track, we used the scoring function shown in Equation 11:

$$score_{combined}(D_j, Q_z) = (1 + score_{BM25}) \times log_{10}\left(1 + \frac{1}{1 + distance}\right) \tag{11}$$

where $score_{BM25} \in [0, \infty]$ is the raw score returned by BM25 implementation in Elasticsearch, and $distance \in [0, \infty]$ is the distance between document D_j and query Q_z from our deep learning approach. The score of the BM25 model and the distance from the proximity-based approach are modified to always be positive and, finally, combined to be weighting coefficients for each other.

Chapter 4

Results and Analysis

Notwithstanding the results presented in Section 3.3.5, the classic BM25 model outperformed all embedding models by reaching nDCG@5 measures of 0.7418 (for 2019) and 0.5289 (for 2018). Based on this, at the recent 2020 shared task, we submitted both an embedding method along with the classic BM25. In this chapter, we present the official results of our submissions to TREC News Track 2019 and 2020, describe the improved configurations we have found after the shared task completion, and analyze how every hyperparameter influences our system performance.

4.1 TREC Runs

All 47,430 models of Section 3.2.3 were not ready in time for the 2019 and 2020 TREC News tasks, therefore, we used the best embedding model found from a smaller subset of experiments.

We have submitted two runs for our participation in the TREC News Track 2019:

(1) `clac_100_cos` is a retrieval model based on Doc2Vec embedding of size 100 and cosine similarity as a proximity measure.

(2) `clac_300_cos` has the same configuration, but the embedding size is increased to 300.

Our official submission to the 2020 TREC News Track includes four runs:

(1) `gpt2_norm` is based only on the GPT2 amplitude normalised embedding with 250 token chunk size with mean pooling and the Minkowski L_3 proximity measure. This configuration

Runs 2019	**nDCG@5**
`clac_100_cos`	0.4057
`clac_300_cos`	0.4298
TREC max	0.7737
TREC median	0.5295
TREC min	0.1002
Runs 2020	**nDCG@5**
`es_bm25`	0.5924
`combined`	0.5873
`gpt2_norm`	0.4541
`d2v2019`	0.4481
TREC max	0.7914
TREC median	0.5250
TREC min	0.0660

Table 4.1: Official results of our runs at the 2019 and 2020 TREC News Tracks

was chosen because it was the best discovered among the embedding methods and proximity measures that we had explored at the time of submission.

(2) `es_bm25` is based on the Elasticsearch (Lucene) implementation of the Okapi BM25 ranking algorithm.

(3) `combined` is a combination of the previous two runs, where the BM25 score between the query and each target document is multiplied by the inverted distance between the corresponding GPT2-embeddings.

(4) `d2v2019` is based on Doc2Vec embeddings computed from News TREC 2019 and cosine similarity as a proximity measure. This model was used because we used this approach the previous year in our participation to the track (Khloponin and Kosseim (2019)), and, for the year 2020, we wanted to compare our 2019 method with the novel ones.

4.2 TREC Results

As indicated in Section 4.1, we have submitted two runs to the 2019 edition of TREC News Track and four runs to the 2020 edition. For each run and each topic, NIST provided us with our official score, as well as the collective minimum, maximum, and median scores. Table 4.1 shows

Run	nDCG@5 2019	nDCG@5 2018
`es_bm25`	0.5891	0.4622
`combined`	0.5668	0.3176
`gpt2_norm`	0.4660	0.2418
`d2v2019`	0.4280	0.2262
TREC max	0.7737	n/a
TREC median	0.5295	n/a
TREC min	0.1002	n/a

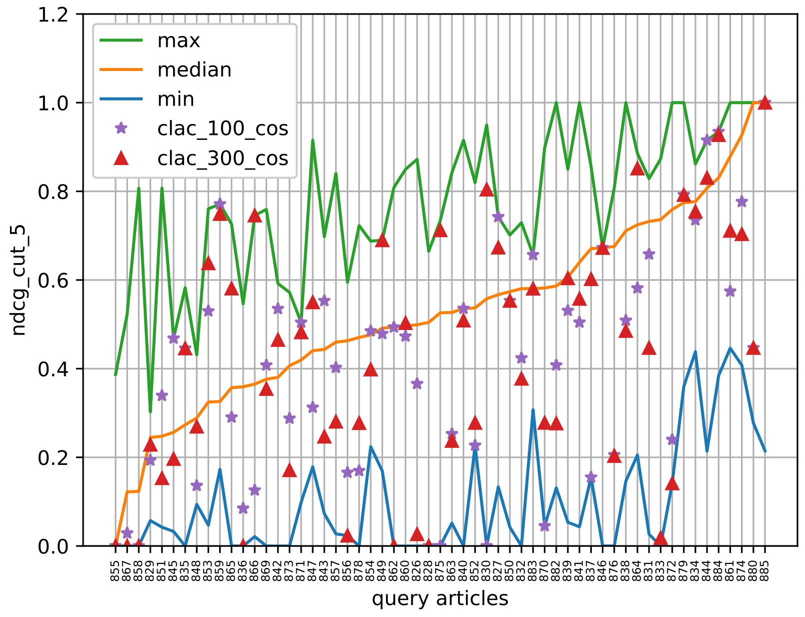
Table 4.2: Results of the 4 submitted runs with the 2018 and 2019 data for validation purposes

the official overall scores. Both our submissions for the 2019 year performed below the collective median; but two of our runs for the 2020 outperformed the collective median nDCG@5 of 0.5250. Among our submissions for 2020, `es_bm25` achieved the highest score with an nDCG@5 of 0.5924, `combined` performed slightly below with 0.5873, while `gpt2_norm` and `d2v2019` performed below the collective median with nDCG@5 of 0.4541 and 0.4481 respectively. These results are in line with our expectations from our validation runs on the 2019 and 2018 data (see Table 4.2).

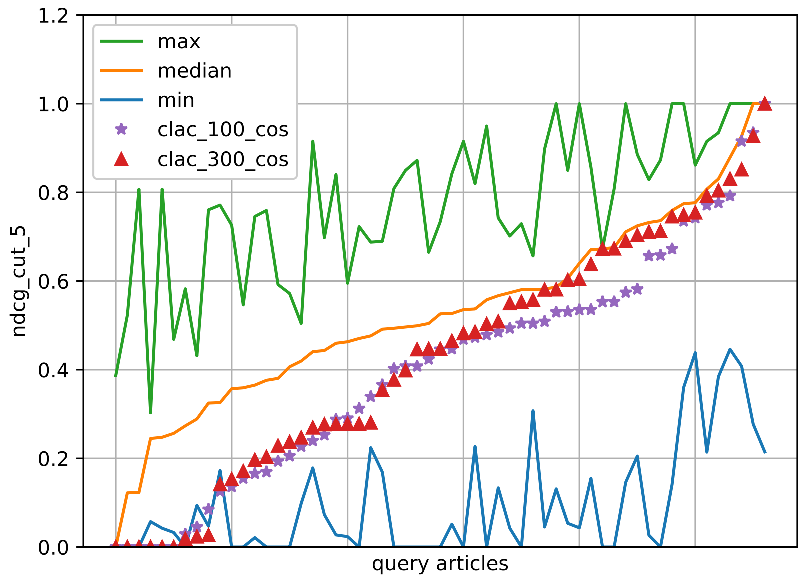
Figure 4.1 shows the results per topic of our 2019 submission compared to the collective minimum, maximum and median scores. Figure 4.1(a) shows the scores for each topic in decreasing order of the median score. Difficult topics for all participants appear on the left-hand side, while easier topics are on the right-hand side. Figure 4.1(b) shows our submission results sorted by nDCG@5 for each topic. This figure helps us to compare our runs with the collective median. We can see that both of our submissions for 2019 perform on average below the collective median.

Figure 4.2 shows the results for our 2020 submission. As shown in the figure, for a few topics our approach had the lowest performance among all runs (7 topics for `d2v2019` and `gpt2_norm`, 3 topics for `combined`, and only 2 topics for `es_bm25`), these are all situated on the minimum line. On the other hand, for a few other topics, all points are situated on the maximum line (4 topics for `d2v2019`, 6 for `gpt2_norm` and for `combined`, and 8 topics for `es_bm25`) hence we achieved the best performance among all submissions on these topics. Figure 4.2(b) shows the scores per topic in increasing order of nDCG@5, and we can see that `es_bm25` and `combined`, in general, perform better than `gpt2_norm`, `d2v2019`, and the collective median.

Table 4.3 shows over all topics the percentage of times that each run was above or equal to the median.

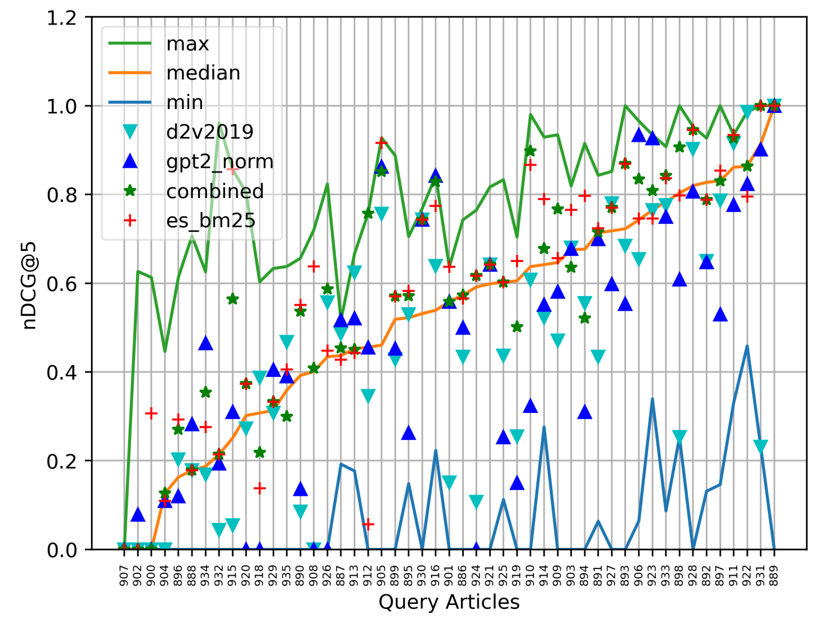


(a) 2019 results per topic in increasing order of median nDCG@5

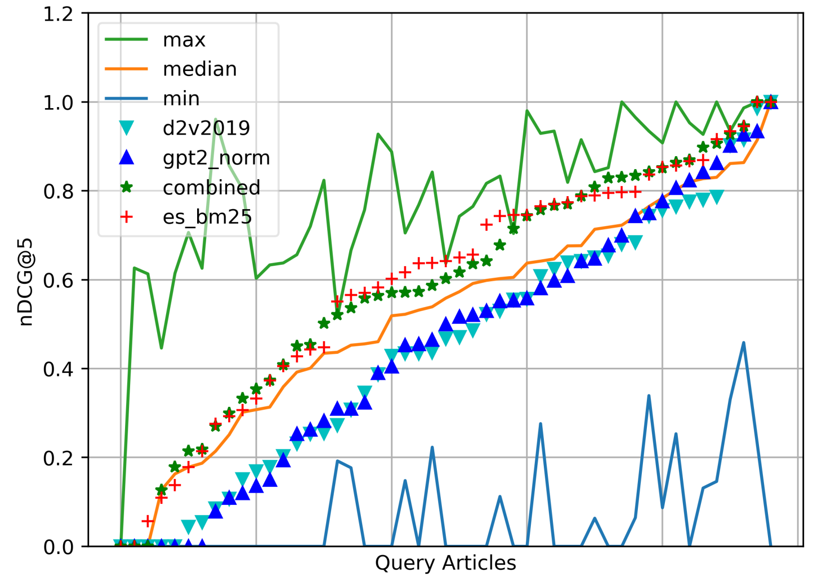


(b) 2019 results sorted individually by their nDCG@5. The topics do not necessarily correspond to the data points with the same abscissa.

Figure 4.1: Official results at the 2019 TREC News Track per topics compared to the min, max and median TREC performances



(a) 2020 results per topic in increasing order of median nDCG@5



(b) 2020 results sorted individually by their nDCG@5. The topics do not necessarily correspond to the data points with the same abscissa.

Figure 4.2: Official results at the 2020 TREC News Track per topics compared to the min, max and median TREC performances

Run	**≥ median**	**< median**	**percentage ≥ median**
`combined`	42	7	85.71%
`es_bm25`	39	10	79.59%
`d2v2019`	22	27	44.90%
`gpt2_norm`	20	29	40.81%

Table 4.3: Number of topics ranked ≥ or < the collective median for each run

	Model (embedding+norm+chunk+distance)	**nDCG@5 2020**	**nDCG@5 2019**	**nDCG@5 2018**
best 2019	#1 `gpt2-xl`+sigmoid+250+mean+Pearson χ^2	0.4882	0.4157	**0.2424**
	#2 `gpt2-xl`+sigmoid+250+mean+Dice	0.4908	**0.4184**	0.2350
	#3 `gpt2-xl`+sigmoid+250+mean+Jaccard	0.4905	0.4127	0.2338
	#4 `gpt2-xl`+sigmoid+250+mean+Vicis-Symmetric χ^2	**0.4950**	0.4126	0.2277
	#5 `gpt2-xl`+sigmoid+250+mean+Probabilistic Symmetric χ^2	0.4895	0.4144	0.2269
best 2018	#6 `gpt2-medium`+sigmoid+500+mean+Cosine	0.4239	0.3836	0.2394
	#7 `xlnet-large-cased`+sigmoid+250+mean+Additive Symmetric χ^2	0.4295	0.3539	0.2334
	#8 `gpt2-large`+sigmoid+500+mean+Cosine	0.4409	0.4001	0.2415
	#9 `gpt2-xl`+sigmoid+250+mean+Cosine	0.4409	0.4001	0.2415
	#10 `xlnet-large-cased`+sigmoid+250+mean+Jaccard	0.4422	0.3435	0.2206
	#11 `es_bm25`	**0.5924**	**0.5514**	**0.3011**
	#12 `es_bm25`+`gpt2-xl`+sigmoid+250+mean+Vicis-Symmetric χ^2	0.5737	0.5125	0.2878
	TREC median	**0.5250**	**0.5295**	**n/a**[1]

Table 4.4: Performance of the top configurations with the validation set, the BM25 model and a new combined model on the entire document collection

4.3 Post-TREC 2020 Results

Once the validation of all models of Section 3.2.3 on the 2018 and 2019 queries was complete, we simulated the official 2020 TREC News shared task using the top 5 models of each year (see Table 4.4). We applied the methods to the entire 2020 document collection (669,890 articles, see Table 3.1) and used the official TREC scorer. Table 4.4 shows the top final scores of these methods with the 2020, 2019 and 2018 queries.

As the complete validation suggested, these top-performing configurations outperformed the embedding method submitted to the shared task, `gpt2_norm` (`gpt2` embedding with mean pooling, amplitude normalization, chunk size of 250) which achieved an nDCG@5 of 0.4541 (see Table 4.1) but still performed below the overall TREC median of 0.5250 and `es_bm25` (0.5924), and achieved an nDCG@5 of 0.4950. The new combined model #12, based on `es_bm25` and model #4

[1]TREC median for 2018 available only to the participants

(see Section 4.1), achieved an nDCG@5 of 0.5737; ranking lower than the `es_bm25` model but higher than model (4) and the TREC median.

4.4 Analysis

Figure 4.3 shows a scatter plot of the nDCG@5 scores of all the configurations of our deep learning model. Each point (47,332 in total) represents a specific configuration with a fixed embedding model, chunk size, pooling method, normalization technique, output layer, and proximity measure. The horizontal axis corresponds to the nDCG@5 score of the model configuration for the 2018 queries, and the vertical axis corresponds to the same model nDCG@5 score but for the 2019 queries.

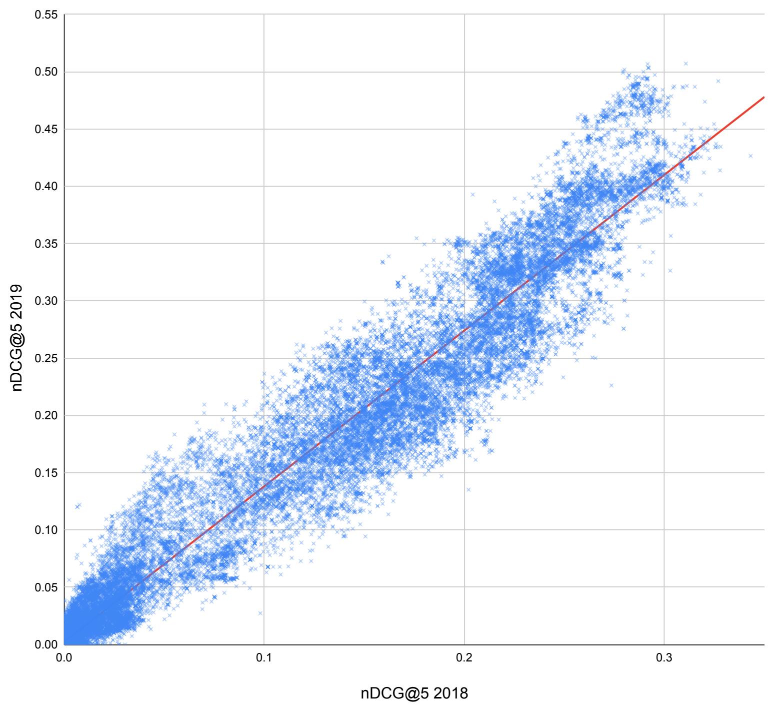


Figure 4.3: Performance of all 47,332 model configurations on the 2018 queries against the 2019 queries

One of the concerns during our approach implementation was whether the model parameters tuning on a very small portion of data (50-60 queries for one year) will adequately generalize for

2020. Because all configurations were evaluated independently of each other and all input variables except the evaluated hyperparameters are identical for all configurations, and as seen in Figure 4.3 the data clearly show a linear dependency, we can conclude that the performance of a selected configuration of our model on a small set of labeled queries could be a good predictor of this configuration performance on new queries from the same corpus not evaluated previously.

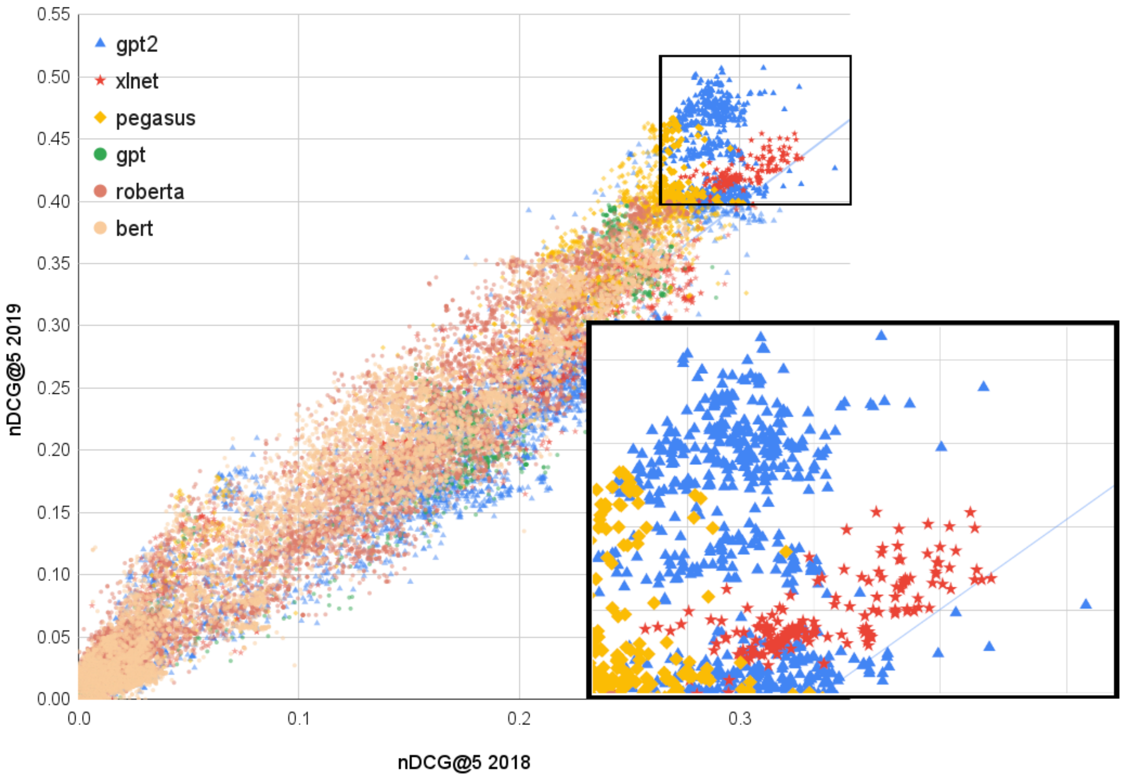
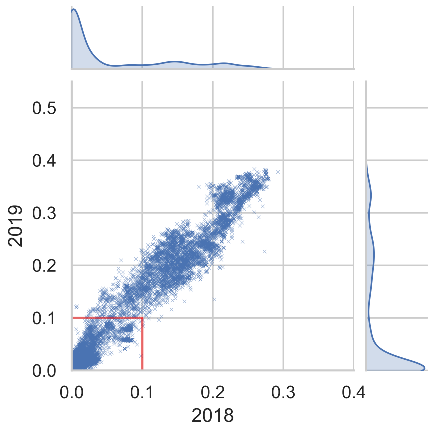


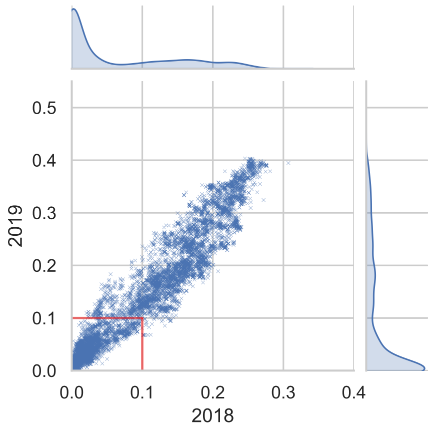
Figure 4.4: Performance of all 47,332 model configurations on the 2018 queries against the 2019 queries color-coded by embedding model family

Figure 4.4 shows the same scatter plot as in Figure 4.3, but each point is marked according to the embedding model used. The first two models in the annotation of Figure 4.4 (▲`gpt2` and ★`xlnet`) reached the maximum score in the 2018 data, and the first model along with the third model (▲`gpt2` and ♦`pegasus`) reached the maximum score in the 2019 data.

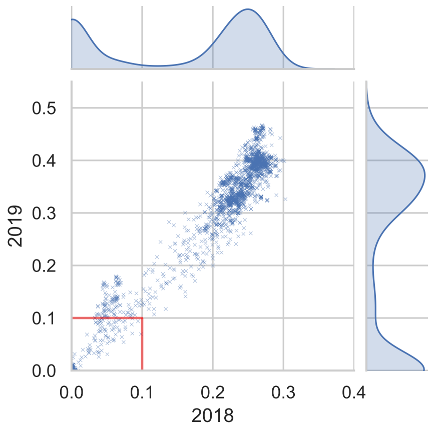
A more detailed performance analysis of each embedding model is shown in Figure 4.5, where the points on the scatter plot are a portion of all configurations in Figure 4.3 with a specific parameter fixed (the embedding model in this case). For example, Figure 4.5 (a) shows all configurations with



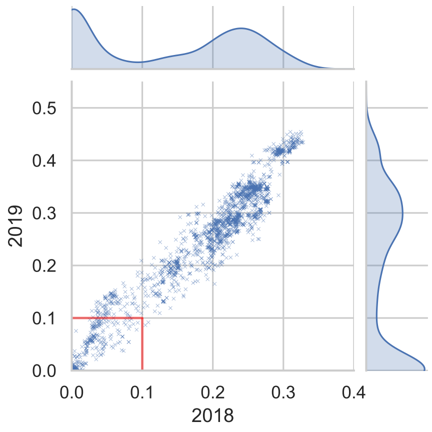
(a) BERT models



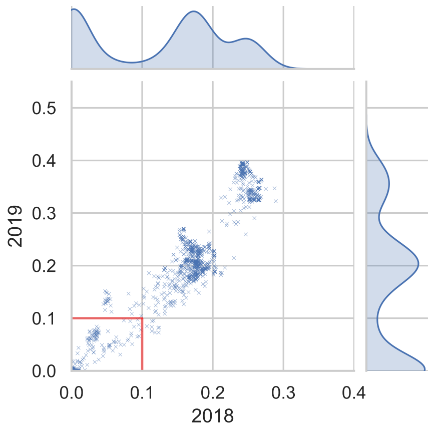
(b) RoBERTa models



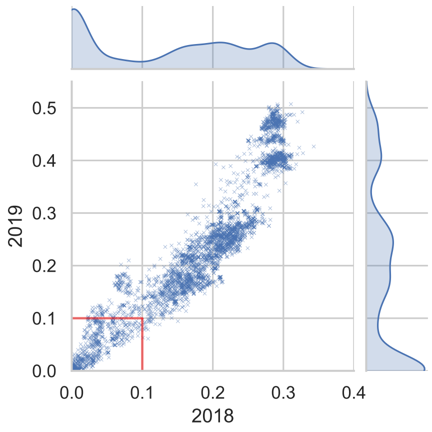
(c) PEGASUS models



(d) XLNet models



(e) GPT models



(f) GPT2 models

Figure 4.5: Influence of the embedding model parameter on performance

BERT models. Because data points are very close to each other and it is difficult to see the actual number of the points in certain areas of the chart, the density distributions of the points are shown on the top and right of each scatter plot.

With 47,332 total configurations and 18,326 configurations specifically for BERT, one would expect the result to look as an almost continuous blob of points stretching from the origin of the axes with very low-performing configurations to the top right side of the scatter plot for the top-performing configurations. However, as Figure 4.5 (a) shows, there is a clear ceiling for BERT models, and they are not among the top performing configurations. Furthermore, as seen in Table 4.5, almost 70% of the configurations with BERT as the embedding model produce nDCG@5 for 2018 and 2019 below 0.1. We arbitrarily selected the threshold of 0.1 which is close to 0, but we used it for all the parameters in Table 4.5 to have a consistent comparison. More detailed information could be found in Appendix D which presents the influence of every hyperparameter on performance.

RoBERTa configurations (see Figure 4.5 (b)) are very similar to the BERT, but achieved higher scores and have 65% of all configurations below the threshold. BERT and RoBERTa scatter plots have more points on the scatter plots compared to other model configurations because we explored more pretrained models using these specific embeddings. Recall from Section 3.3.1 that we used 6 models for BERT, 5 models for RoBERTa, 4 for GPT2, 2 for XLNet, 2 for PEGASUS, and only 1 for GPT.

The PEGASUS models (Figure 4.5 (c)) are particularly interesting. Although they did not reach the highest score, only 35% of the configurations fell below the threshold. This is the lowest percentage among all models, and most of the configurations for PEGASUS models are clustered in the upper right part of the scatter plot, making models based on this embedding the highest proportion of high performing configurations. Both pretrained PEGASUS embedding models we explored are trained on news datasets, which along with the abstractive summarization nature of the model might contribute to such results.

XLNet models (Figure 4.5 (d)) had only 40% of the configurations that performed below the threshold and reached the second highest score for 2018.

models	**total**	**≤ 0.1 / %**
BERT	18,326	12,732 / 69%
RoBERTa	15,300	9,885 / 65%
PEGASUS	3,060	1,067 / 35%
XLNet	3,028	1,212 / 40%
GPT	1,530	580 / 38%
GPT2	6,088	2,521 / 41%
window size	**total**	**≤ 0.1 / %**
250	23,649	13,876 / 59%
500	23,683	14,027 / 59%
normalization	**total**	**≤ 0.1 / %**
plain	15,746	10,716 / 68%
amplitude	15,776	8,711 / 55%
sigmoid	15,810	8,476 / 54%
pooling	**total**	**≤ 0.1 / %**
max	15,810	9,424 / 60%
mean	15,810	8,917 / 56%
min	15,712	9,562 / 61%
output layer	**total**	**≤ 0.1 / %**
last hidden	30,536	14,048 / 46%
pooler output	16,796	14,069 / 84%
proximity metric	**total**	**≤ 0.1 / %**
Cosine	558	454 / 81%
Jaccard	558	200 / 36%
Euclidean	558	197 / 35%
Manhattan	558	182 / 33%
Vicis-Symmetric χ^2	556	257 / 46%
Dice	558	199 / 36%
Pearson χ^2	556	240 / 43%
Czekanowski	558	263 / 47%
Lorentzian	558	180 / 32%
Total	**47,332**	**27,997 / 59%**

Table 4.5: Number of experiments with specific parameter fixed and the percentage of experiments which reached nDCG@5 of only 0.1 or less (lower is better). See Appendix D for a complete table.

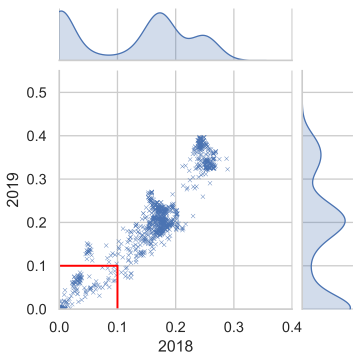
The performance of the GPT and GPT2 models looks similar, but moving from the smaller version of the GPT `openai-gpt` with 117M parameters to larger versions of the GPT2 `gpt2` with 124M parameters, `gpt2-medium` with 354M parameters, to even larger versions of `gpt2-large` with 774M parameters, and finally to the largest `gpt2-xl` with 1,558M parameters, the performance of our model steadily increases (see Figure 4.6) keeping the shape of the scatter plots very similar to each other and mostly stretching it towards better performance.

As seen in Table 4.5, the proportion of configurations falling below the 0.1 threshold for different window sizes is very close, but looking at Figure 4.7 we can see that almost all the top performing configurations for 2018 and all the top configurations for 2019 queries have smaller window sizes. It is not clear why a smaller window size is preferable. We suspect that it might be related to the specific structure of the news articles or the average length of the sentences in the news.

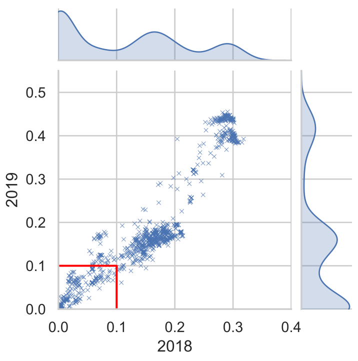
Looking at the influence of the normalization parameter (see Section 3.3.4) in Figure 4.8, applying any normalization improves the maximum performance achieved. From Table 4.5, we can see that normalization not only improves maximum performance but also moves the 2005 configurations for the amplitude normalization and the 2240 configurations for the sigmoid normalization above the 0.1 threshold.

The performance of the pooling parameter (see Section 3.3.2) is shown in Figure 4.9. Despite the fact that the `mean` pooling leads only to a slight improvement compared to the `max` and `min` pooling (from 60% to 56%) for the number of configurations that are below the 0.1 threshold (see Table 4.5), it gives a major boost to the configurations above the threshold, contributing to all the best performing configurations.

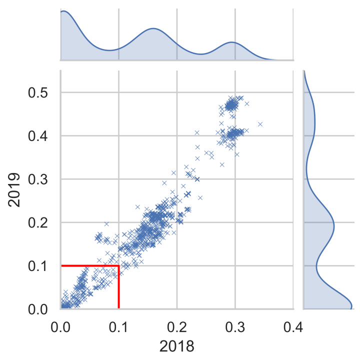
If we check the performance of the proximity metrics individually (see Figure 4.10 and Table 4.5), it is clear that they behave differently. The cosine similarity, although leading to the best configuration for the year 2018 (see Table 4.4), has one of the highest percentages (81%) of the number of configurations below the 0.1 threshold and very few top-performing configurations. The Jaccard, Euclidean and Manhattan metrics perform much better in this regard with, respectively, 36%, 35%, and 33% of the configurations below the threshold and more top-performing results. Among the top 1000 configurations we observed 45 different proximity measures. Even though



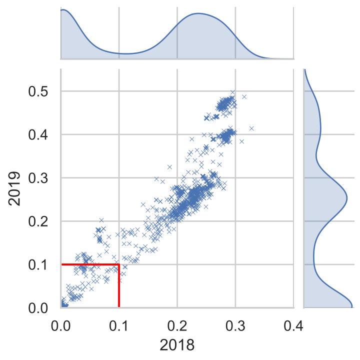
(a) openai-gpt



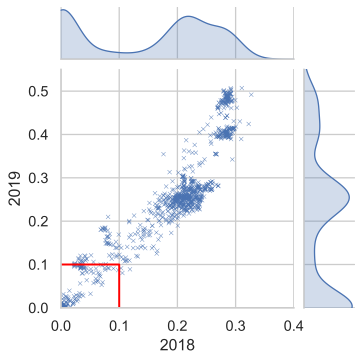
(b) gpt2



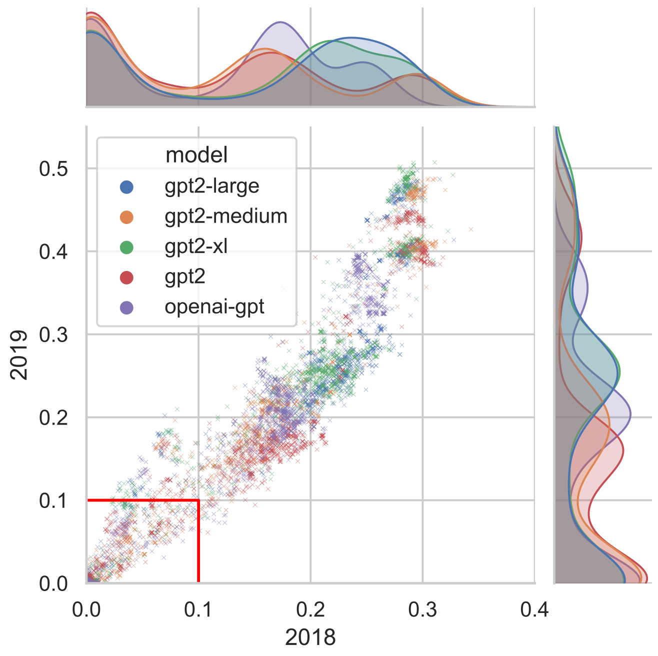
(c) gpt2-medium



(d) gpt2-large

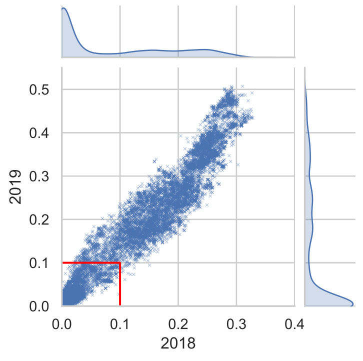


(e) gpt2-xl

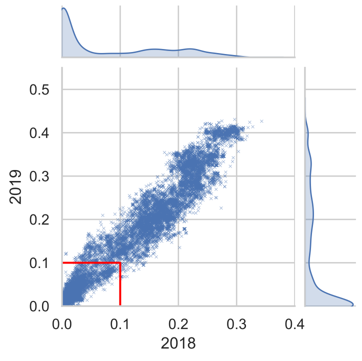


(f) all gpt models

Figure 4.6: Performance of GPT and GPT2 models

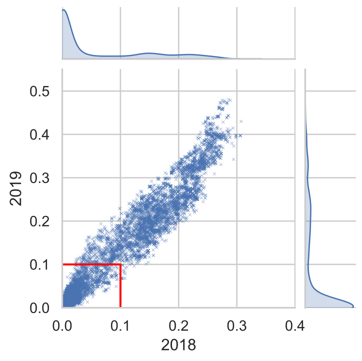


(a) window size 250

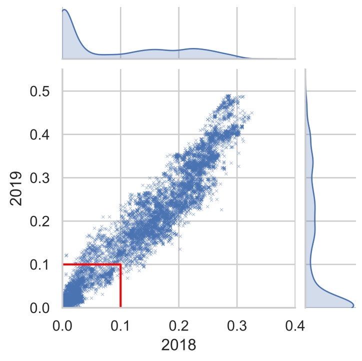


(b) window size 500

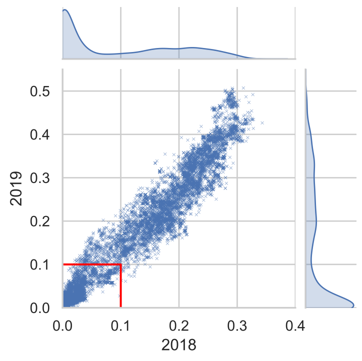
Figure 4.7: Influence of the windows size parameter



(a) without normalization



(b) amplitude normalization



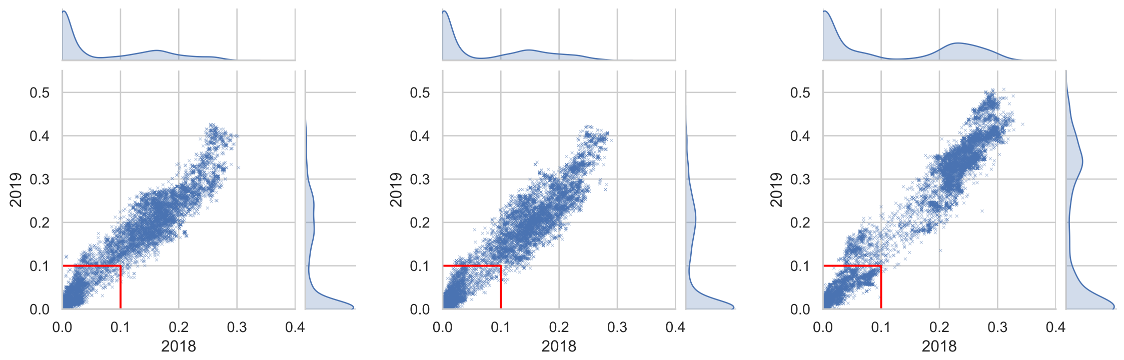
(c) sigmoid normalization

Figure 4.8: Influence of the normalization parameter

some proximity metrics perform better on average, they are not the decisive factor for the model performance especially if we can adjust other parameters.

4.5 Diversity

As shown in Table 4.4, `es_bm25` and model #12 are rather similar in terms of overall median nDCG@5, however, when looking at individual topics, they return significantly different background links. Figure 4.11 shows this diversity graphically. Figure 4.11(a) shows the difference in performance between `es_bm25` and model #4 for each search topic. If the bar is in the upper half of the figure, `es_bm25` reaches higher performance then model #4, if it is in the bottom half, then



(a) min pooling (b) max pooling (c) mean pooling

Figure 4.9: Influence of the pooling parameter

model #4 performed better for the topic. The size of the bar indicates the magnitude of the difference. Similarly, Figure 4.11(b) shows the difference in performance for `es_bm25` and the combined model #12.

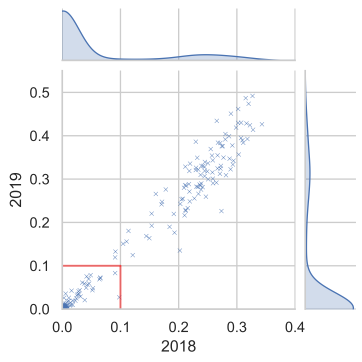
From Figure 4.11(a) we can conclude that `es_bm25` performs better on most topics (see the upward bars) but significantly decreases performance on certain queries for which model #4 is very successful (see the long downward bars). For example, model #4 returned the best result on all runs submitted for topic #912 where `es_bm25` missed the most relevant article. The query article [2] and the most relevant backlink [3] have a substantial word overlap related to cooking, so `es_bm25` returned many cooking articles, but missed the main idea of camping cooking. Model #4, on the other hand, which works on different principles, was able to return the most relevant backlink in the first position.

Figure 4.11(b) shows the per topic difference between `es_bm25` and model #12. As the figure shows, model #12 improved on most of the topics for which `es_bm25` outperformed model #4 without dropping in performance on most topics for which `es_bm25` did not produce the highest results.

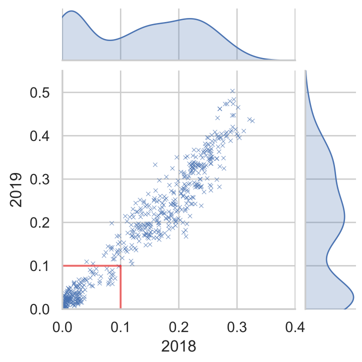
In this chapter, we discuss the results of our submissions at the TREC News Track shared task

[2]https://www.washingtonpost.com/news/voraciously/wp/2018/09/28/how-to-ditch-the-boring-trail-mix-and-eat-well-while-camping-out/

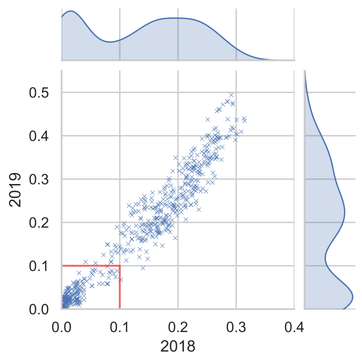
[3]https://www.washingtonpost.com/lifestyle/travel/move-beyond-skewers-and-smores-on-your-next-camping-trip/2018/09/13/9cb69b12-b2df-11e8-aed9-001309990777_story.html



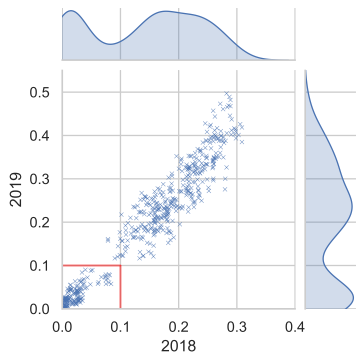
(a) Cosine



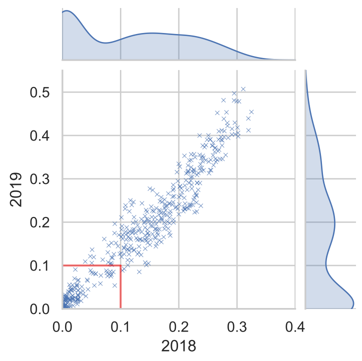
(b) Jaccard



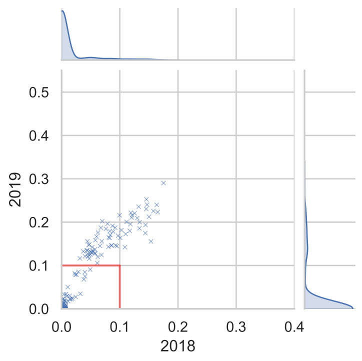
(c) Euclidean



(d) Manhattan



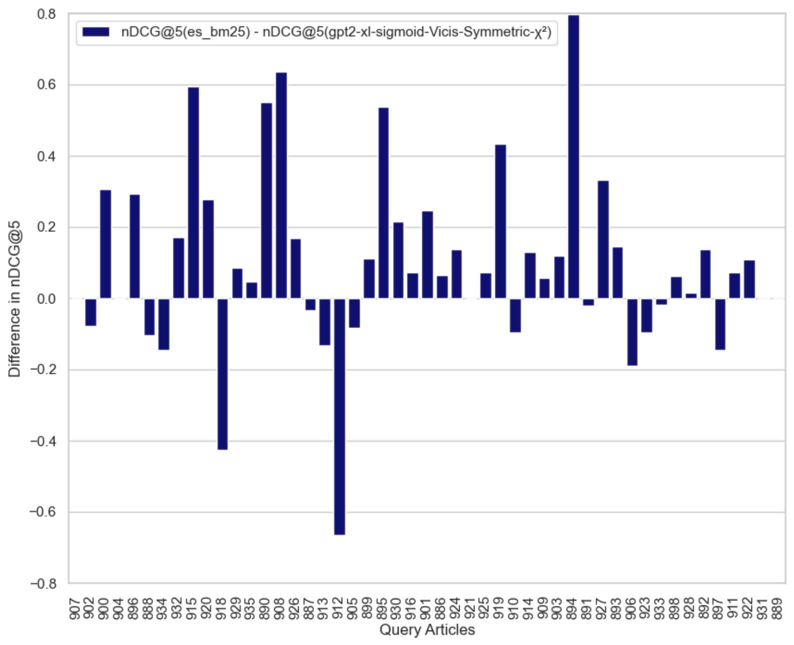
(e) Pearson χ^2



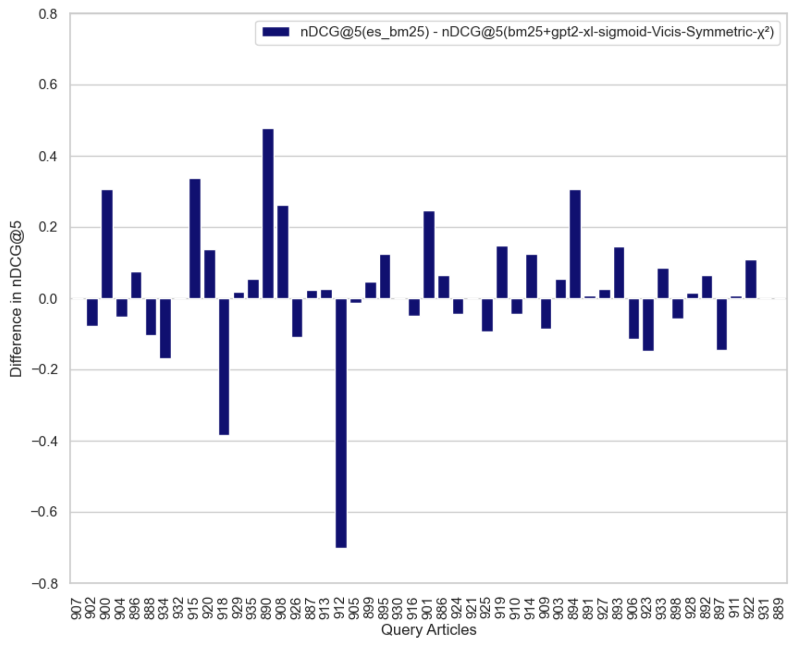
(f) Harmonic mean

Figure 4.10: Influence of the proximity metric

for the years 2019 and 2020. We show the results we achieved after completion of the shared tasks; then we analyzed the individual parameters of our deep learning model and their influence on model performance. We also analyzed the performance of our `combined` approach, showing that on average it performed very close to `es_bm25`. The performance per topic is rather different, and combining models based on different principles could help improve the diversity of the results. In the next chapter, we will present our overall conclusions and future work.



(a) es_bm25 versus the model #4



(b) es_bm25 versus the combined model (12)

Figure 4.11: Difference in nDCG@5 scores for two pairs of models showing the diversity of background links for each topic

Chapter 5

Conclusions and Future Work

5.1 Summary

In this thesis, we developed a number of news recommendation approaches using state-of-the-art document representations and compared their performance to classical information retrieval approaches within the context of the TREC News Track. In Chapter 2 we reviewed classic and modern approaches in information retrieval. We also reviewed all approaches participants used at the TREC News Track starting from 2018 to 2021, which was the last year this shared task took place.

In Chapter 3, we present details of our system implementation. We have developed several news recommendation systems. Our baseline system, based on BM25 reached the highest score among all participants at the TREC News Track 2020. We also describe our systems for the year 2019, which is based on Doc2Vec implementation, and for the year 2020, which is based on deep learning embedding models using transformers.

Finally in Chapter 4 we performed a detailed analysis of the results and analyzed the influence of all hyperparameters of our deep learning system and their influence on performance.

5.2 Contributions

This thesis presented a number of contributions, including:

- We have identified a more effective alternative to the commonly used proximity functions for vector-space document representations.

- We have demonstrated the difference in modern pretrained deep neural network models performance and importance of the model size.
- We have shown that scaling individual components of embedding space to have the same amplitude reliably improves the performance of proximity measures, and discounting outliers with the help of a non-linear scaling function gives additional boost in performance.
- We have demonstrated that our system being tuned on a small portion of labeled data produces a predictable result for previously unseen queries.
- We have shown that the Okapi BM25 ranking function that uses the entire query article as a search query is still the best approach in terms of performance and computational resources spent.
- Combining the Okapi BM25 model with our embedding approach can improve the system output on topics, on which BM25 performed poorly, without losing performance on other topics.

5.3 Limitations and Future Work

Many avenues of research are still to be investigated. In particular, we used the embedding models off-the-shelf with no fine-tuning. Tuning the models for our specific dataset and task itself might improve the representation of the documents in vector space, potentially providing a better ranking for backlinks.

More specialized embedding models might help improve the results. For example, we noticed an unusually high number of high-performing configurations of our system using the PEGASUS model as a pretrained embedding model. It might be due to the task this model is designed, which is abstractive text summarization, or the fact that it was pretrained on a news dataset.

We did not use any external knowledge in our systems and did not augment the article text with additional information, synonyms, information from knowledge graphs, etc. The use of such external resources could be beneficial in uncovering hidden connections between news and topics.

In addition, it would be interesting to evaluate the behavior of the approaches on a historical corpus of news with known links between events and settled priority for backlinks, and a corpus with newly emerging topics and dynamically changing connections between events and backlinks priorities, as is the case in a real-world scenario. The approaches we have developed would benefit from such an on-line learning component.

Finally, we would like to explore different ways of combining BM25 and embedding methods, in particular to better leverage the diversity of the results, instead of favoring common results.

References

Adomavicius, G., et al. (2005, January). Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. *ACM Transactions on Information Systems (TOIS)*, *23*(1), 103-145.

Ak, A. E., ahan Kksal, Fayoumi, K., & Yeniterzi, R. (2020). SU-NLP at TREC NEWS 2020. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 29*th* *Text REtrieval Conference (TREC 2020),* https://trec.nist.gov/pubs/trec29/trec2020.html. NIST.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Retrieved from http://arxiv.org/abs/1409.0473

Bialecki, A., Muir, R., & Ingersoll, G. (2012). Apache lucene 4. In A. Trotman, C. L. A. Clarke, I. Ounis, J. S. Culpepper, M. Cartright, & S. Geva (Eds.), *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval, OSIR@SIGIR 2012, Portland, Oregon, USA, 16th August 2012* (pp. 17–24). University of Otago, Dunedin, New Zealand.

Bimantara, A., et al. (2018). htw saar @ TREC 2018 News Track. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 27*th* *Text REtrieval Conference (TREC 2018),* https://trec.nist.gov/pubs/trec27/trec2018.html. NIST.

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python* (1st ed.). O'Reilly Media, Inc.

Boers, P., Kamphuis, C., & de Vries, A. (2020). Radboud University at TREC 2020. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 29*th* *Text REtrieval Conference (TREC*

2020), https://trec.nist.gov/pubs/trec29/trec2020.html. NIST.

Boudin, F. (2016, December). PKE: an open source python-based keyphrase extraction toolkit. In *Proceedings of the 26th International Conference on Computational Linguistics: System Demonstrations COLING 2016* (pp. 69–73). Osaka, Japan. Retrieved from http://aclweb.org/anthology/C16-2015

Cabrera-Diego, L. A., Boros, E., & Doucet, A. (2021). Elastic Embedded Background Linking for News Articles with Keywords, Entities and Events. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 30^{th} *Text REtrieval Conference (TREC 2021),* https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Carpineto, C., & Romano, G. (2012, jan). A survey of automatic query expansion in information retrieval. *ACM Computing Surveys*, *44*(1). Retrieved from https://doi.org/10.1145/2071389.2071390 doi: 10.1145/2071389.2071390

Casson, L., Penn, J., & Davis, T. (2001). *Libraries in the ancient world.* Yale University Press. Retrieved from https://books.google.ca/books?id=ECBkVPQkNSsC

Cha, S.-H. (2007, 01). Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, *1*, 300–307.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., & Salakhutdinov, R. (2019, July). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 2978–2988). Florence, Italy: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P19-1285 doi: 10.18653/v1/P19-1285

Day, N., Worley, D., & Allison, T. (2020). OSC at TREC 2020 - News tracks Background Linking Task. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 29^{th} *Text REtrieval Conference (TREC 2020),* https://trec.nist.gov/pubs/trec29/trec2020.html. NIST.

Deshmukh, A. A., & Sethi, U. (2020). IR-BERT: Leveraging BERT for Semantic Search in Background Linking for News Articles. *arXiv*. Retrieved from https://arxiv.org/abs/2007.12603

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*. Retrieved from https://arxiv.org/abs/1810.04805

Ding, Y., Lian, X., Zhou, H., Liu, Z., Ding, H., & Hou, Z. (2019). ICTNET at TREC 2019 News Track. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 28*th* *Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

Engelmann, B., & Schaer, P. (2021). Relation-based re-ranking for background linking. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 30*th* *Text REtrieval Conference (TREC 2021),* https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Essam, M., & Elsayed, T. (2019). bigIR at TREC 2019: Graph-based Analysis for News Background Linking. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 28*th* *Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

Essam, M., & Elsayed, T. (2021). bigIR at TREC 2021: Adopting Transfer Learning for News Background Linking. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 30*th* *Text REtrieval Conference (TREC 2021),* https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Fabbri, A., Li, I., She, T., Li, S., & Radev, D. (2019, July). Multi-News: A Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model. In *Proceedings of The Association for Computational Linguistics (ACL 2019)* (pp. 1074–1084). Florence, Italy.

Foley, J., Montoly, A., & Pena, M. (2019). Smith at TREC2019: Learning to Rank Background Articles with Poetry Categories and Keyphrase Extraction. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 28*th* *Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

Galassi, A., Lippi, M., & Torroni, P. (2021, Oct). Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(10), 42914308. Retrieved from http://dx.doi.org/10.1109/TNNLS.2020.3019893 doi: 10.1109/tnnls.2020.3019893

Gautam, R., Mitra, M., & Roy, D. (2020). TREC 2020 NEWS Track Background Linking Task. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 29^{th} *Text REtrieval Conference (TREC 2020),* https://trec.nist.gov/pubs/trec29/trec2020.html. NIST.

Grusky, M., Naaman, M., & Artzi, Y. (2018, June). Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies. In *Proceedings of North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2018)* (pp. 708–719). New Orleans.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., & Daumé III, H. (2015, July). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL/IJCNLP 2015) (Volume 1: Long Papers)* (pp. 1681–1691). Beijing, China: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P15-1162 doi: 10.3115/v1/P15-1162

Järvelin, K., & Kekäläinen, J. (2002, October). Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS)*, *20*(4), 422–446.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*, 11–21.

Kalyan, K. S., Rajasekharan, A., & Sangeetha, S. (2021). *Ammus : A survey of transformer-based pretrained models in natural language processing.* arXiv. Retrieved from https://arxiv.org/abs/2108.05542 doi: 10.48550/ARXIV.2108.05542

Khloponin, P., & Kosseim, L. (2019). The CLaC System at the TREC 2019 News Track. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 28^{th} *Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

Khloponin, P., & Kosseim, L. (2020). The CLaC System at the TREC 2020 News Track. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 29^{th} *Text REtrieval Conference (TREC 2020),* https://trec.nist.gov/pubs/trec29/trec2020.html. NIST.

Khloponin, P., & Kosseim, L. (2021). Using Document Embeddings for Background Linking of News Articles. In E. Métais, F. Meziane, H. Horacek, & E. Kapetanios (Eds.), *Natural Language Processing and Information Systems - 26th International Conference on Applications* *of Natural Language to Information Systems, NLDB 2021, Saarbrücken, Germany, June 23-25, 2021, Proceedings* (Vol. 12801, pp. 317–329). Springer. Retrieved from https://doi.org/10.1007/978-3-030-80599-9_28 doi: 10.1007/978-3-030-80599-9_28

Kim, S. N., Medelyan, O., Kan, M.-Y., & Baldwin, T. (2010, July). SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval 2010)* (pp. 21–26). Uppsala, Sweden: Association for Computational Linguistics. Retrieved from https://aclanthology.org/S10-1004

Koster, C., & Foley, J. (2021). Middlebury at TREC News 21 Exploring Learning to Rank Model Variants. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 30^{th} *Text REtrieval Conference (TREC 2021)*, https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Lai, G., Xie, Q., Liu, H., Yang, Y., & Hovy, E. (2017, September). RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 785–794). Copenhagen, Denmark: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D17-1082 doi: 10.18653/v1/D17-1082

Lancaster, F., & Fayen, E. (1973). *Information retrieval: On-line*. Melville Publishing Company. Retrieved from https://books.google.ca/books?id=FmdkAAAAMAAJ

Lavrenko, V., & Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th ACM Special Interest Group on Information Retrieval (SIGIR 2001) Conference* (p. 120-127). New York, NY.

Le, Q., & Mikolov, T. (2014, 22–24 Jun). Distributed representations of sentences and documents. In E. P. Xing & T. Jebara (Eds.), *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)* (Vol. 32, pp. 1188–1196). Bejing, China: PMLR. Retrieved from http://proceedings.mlr.press/v32/le14.html

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017, apr). A Structured Self-Attentive Sentence Embedding. In *Conference Track Proceedings, 5th International Conference on Learning Representations (ICLR 2017)* (pp. 24–26). OpenReview.net.

Retrieved from https://openreview.net/forum?id=BJC_jUqxe

Lirong, Z., Joho, H., & Fujita, S. (2021). TKB48 at TREC 2021 News Track. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 30th *Text REtrieval Conference (TREC 2021),* https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Liu, Y., et al. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv*. Retrieved from https://arxiv.org/abs/1907.11692

Lopez-Ubeda, P., Diaz-Galiano, M. C., Martin-Valdivia, M.-T., & Urena-Lopez, L. A. (2018). Using clustering to filter results of an Information Retrieval system. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 27th *Text REtrieval Conference (TREC 2018),* https://trec.nist.gov/pubs/trec27/trec2018.html. NIST.

Lu, K., & Fang, H. (2018). Paragraph as Lead - Finding Background Documents for News Articles. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 27th *Text REtrieval Conference (TREC 2018),* https://trec.nist.gov/pubs/trec27/trec2018.html. NIST.

Lu, K., & Fang, H. (2019). Leveraging Entities in Background Document Retrieval for News Articles. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 28th *Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

Lu, K., & Fang, H. (2020). Aspect Based Background Document Retrieval for News Articles. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 29th *Text REtrieval Conference (TREC 2020),* https://trec.nist.gov/pubs/trec29/trec2020.html. NIST.

Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, *1*(4), 309-317. doi: 10.1147/rd.14.0309

Lv, Y., & Zhai, C. (2011). When documents are very long, BM25 fails! In *Proceedings of the 34th international acm special interest group on information retrieval (sigir 2011) conference on research and development in information retrieval* (p. 11031104). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2009916.2010070 doi: 10.1145/2009916.2010070

Ma, Y., et al. (2019, November). News2vec: News Network Embedding with Subnode Information. In *Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing* *and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)* (p. 4843-4852). Hong Kong: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D19-1490 doi: 10.18653/v1/D19-1490

MacAvaney, S., Yates, A., Cohan, A., & Goharian, N. (2019). CEDR: Contextualized Embeddings for Document Ranking. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, 11011104. Retrieved from https://doi.org/10.1145/3331184.3331317 doi: 10.1145/3331184.3331317

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 281–297).

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge, UK: Cambridge University Press. Retrieved from http://nlp.stanford.edu/IR-book/information-retrieval-book.html

Maron, M. E., & Kuhns, J. L. (1960, jul). On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, *7*(3), 216244. Retrieved from https://doi.org/10.1145/321033.321035 doi: 10.1145/321033.321035

Metzler, D., & Bruce Croft, W. (2007, jun). Linear feature-based models for information retrieval. *Inf. Retr.*, *10*(3), 257274. Retrieved from https://doi.org/10.1007/s10791-006-9019-z doi: 10.1007/s10791-006-9019-z

Mihalcea, R., & Tarau, P. (2004, July). TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)* (pp. 404–411). Barcelona, Spain: Association for Computational Linguistics. Retrieved from https://aclanthology.org/W04-3252

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In Y. Bengio & Y. LeCun (Eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings.* Retrieved from http://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International*

Conference on Neural Information Processing Systems (NeurIPS 2013) - Volume 2 (p. 3111-3119). Red Hook, NY, USA: Curran Associates Inc.

Miller, D. (2019). Leveraging BERT for extractive text summarization on lectures. *CoRR*, *abs/1906.04165*. Retrieved from http://arxiv.org/abs/1906.04165

Missaoui, S., MacFarlane, A., Makri, S., & Gutierrez-Lopez, M. (2019). DMINR at TREC News Track. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 28th *Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

Muresan, G., & Harper, D. J. (2004). Topic modeling for mediated access to very large document collections. *Journal of the American Society for Information Science and Technology*, *55*(10), 892-910. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20034 doi: https://doi.org/10.1002/asi.20034

Murphy, T., & Secundus, G. (2004). *Pliny the elder's natural history: The empire in the encyclopedia*. Oxford University Press. Retrieved from https://books.google.ca/books?id=6NC_T_tG9lQC

Naseri, S., Foley, J., & Allan, J. (2018). UMass at TREC 2018: CAR, Common Core and News Tracks. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 27th *Text REtrieval Conference (TREC 2018),* https://trec.nist.gov/pubs/trec27/trec2018.html. NIST.

Nogueira, R., & Lin, J. (2019). From Doc2Query to DocTTTTTQuery. *Online preprint*. Retrieved from https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-latest.pdf

Nogueira, R., Yang, W., Lin, J. J., & Cho, K. (2019). Document expansion by query prediction. *ArXiv*, *https://arxiv.org/abs/1904.08375*.

Okura, S., et al. (2017, August). Embedding-Based News Recommendation for Millions of Users. In *Proceedings of the 23rd ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD 2017) Conference* (pp. 1933–1942). New York.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999, November). *The PageRank Citation Ranking: Bringing Order to the Web.* (Technical Report No. 1999-66). Stanford InfoLab.

Retrieved from http://ilpubs.stanford.edu:8090/422/ (Previous number = SIDL-WP-1999-0120)

Qu, J., & Wang, Y. (2019). UNC SILS at TREC 2019 News Track. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 28^{th} *Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

Radford, A., & Narasimhan, K. (2018). Improving language understanding by generative pre-training. *Preprint*. Retrieved from https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Radford, A., et al. (2019). Language models are unsupervised multitask learners. *Preprint*. Retrieved from https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., . . . Liu, P. J. (2019). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.* arXiv. Retrieved from https://arxiv.org/abs/1910.10683 doi: 10.48550/ARXIV.1910.10683

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., et al. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, *21*(140), 1-67. Retrieved from http://jmlr.org/papers/v21/20-074.html

Rajpurkar, P., Jia, R., & Liang, P. (2018). *Know What You Don't Know: Unanswerable Questions for SQuAD.* arXiv. Retrieved from https://arxiv.org/abs/1806.03822 doi: 10.48550/ARXIV.1806.03822

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). *SQuAD: 100,000+ Questions for Machine Comprehension of Text.* arXiv. Retrieved from https://arxiv.org/abs/1606.05250 doi: 10.48550/ARXIV.1606.05250

Reimers, N., & Gurevych, I. (2019, November). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)* (p. 39823992). Hong Kong.

Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995, January).

Okapi at trec-3. In *Overview of the third text retrieval conference (trec-3)* (Overview of the Third Text REtrieval Conference (TREC3) ed., p. 109-126). Gaithersburg, MD: NIST. Retrieved from https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, *3*(4), 333-389. Retrieved from http://dx.doi.org/10.1561/1500000019 doi: 10.1561/1500000019

Robertson, S. E., & Jones, K. S. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, *27*(3), 129-146. Retrieved from https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.4630270302 doi: https://doi.org/10.1002/asi.4630270302

Salton, G., & Buckley, C. (1988, jan). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. Retrieved from https://doi.org/10.1016%2F0306-4573%2888%2990021-0 doi: 10.1016/0306-4573(88)90021-0

Sethi, U., & Deshmukh, A. A. (2021). Semantic Search for Background Linking in News Articles. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 30th *Text REtrieval Conference (TREC 2021),* https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Soboroff, I. (2021, November). TREC 2021 News Track Overview. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 30th *Text REtrieval Conference (TREC 2021),* https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Soboroff, I., Huang, S., & Harman, D. (2018, November). TREC 2018 News Track Overview. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 27th *Text REtrieval Conference (TREC 2018),* https://trec.nist.gov/pubs/trec27/trec2018.html. NIST.

Soboroff, I., Huang, S., & Harman, D. (2020a, March). TREC 2019 News Track Overview. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 28th *Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

Soboroff, I., Huang, S., & Harman, D. (2020b, November). TREC 2020 News Track Overview. In TREC (Ed.), *NIST Special Publication: Proceedings of the* 29th *Text REtrieval Conference (TREC 2020),* https://trec.nist.gov/pubs/trec29/trec2020.html. NIST.

TREC (Ed.). (2018). *NIST Special Publication: Proceedings of the 27th Text REtrieval Conference (TREC 2018),* https://trec.nist.gov/pubs/trec27/trec2018.html. NIST.

TREC (Ed.). (2019). *NIST Special Publication: Proceedings of the 28th Text REtrieval Conference (TREC 2019),* https://trec.nist.gov/pubs/trec28/trec2019.html. NIST.

TREC (Ed.). (2020). *NIST Special Publication: Proceedings of the 29th Text REtrieval Conference (TREC 2020),* https://trec.nist.gov/pubs/trec29/trec2020.html. NIST.

TREC (Ed.). (2021). *NIST Special Publication: Proceedings of the 30th Text REtrieval Conference (TREC 2021),* https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., et al. (2017). *Attention is all you need* (Vol. 30). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wagenpfeil, S., Kevitt, P. M., & Hemmje, M. (2021). University of Hagen @ TREC2021 News Track. In TREC (Ed.), *NIST Special Publication: Proceedings of the 30th Text REtrieval Conference (TREC 2021),* https://trec.nist.gov/pubs/trec30/trec2021.html. NIST.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International conference on learning representations.* Retrieved from https://openreview.net/forum?id=rJ4km2R5t7

Wellisch, H. (1995). *Indexing from A to Z.* New York, Dublin: H.W. Wilson. Retrieved from https://books.google.ca/books?id=_KQ5AQAAMAAJ

Wiegand, W., & Davis, D. (2015). *Encyclopedia of library history*. Taylor & Francis. Retrieved from https://books.google.ca/books?id=YZpsBgAAQBAJ

Yang, P., & Lin, J. (2018). Anserini at TREC 2018: CENTRE, Common Core, and News Tracks. In TREC (Ed.), *NIST Special Publication: Proceedings of the 27th Text REtrieval Conference (TREC 2018),* https://trec.nist.gov/pubs/trec27/trec2018.html. NIST.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XL-Net: Generalized Autoregressive Pretraining for Language Understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems (!NeurIPS!)* (Vol. 32). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020, 13–18 Jul). PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)* (Vol. 119, pp. 11328–11339). PMLR. Retrieved from https://proceedings.mlr.press/v119/zhang20ae.html

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015, dec). Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *2015 IEEE International Conference on Computer Vision (ICCV 2015)* (p. 19-27). Los Alamitos, CA, USA: IEEE Computer Society. Retrieved from https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.11 doi: 10.1109/ICCV.2015.11

Appendix A

Distance and Proximity Measures

In the table below, we list all distance and proximity functions we used in our work. In functions formulas Q stands for query embedding vector, P for article embedding vector, Q_i and P_i are respectively for embedding vector components, d used to denote distance, and s for proximity.

#	Label	Distance or Similarity Formula
1	Euclidean L2	$d_{Euc} = \sqrt{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert^2}$
2	Manhattan L1	$d_{CB} = \sum_{i=1}^{d} \lvert P_i - Q_i \rvert$
3	Minkowski Lp	$d_{Mk} = \sqrt[p]{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert^p}$
4	Chebyshev L_∞	$d_{Cheb} = \max_i \lvert P_i - Q_i \rvert$
5	Sørensen	$d_{sor} = \frac{\sum_{i=1} \lvert P_i - Q_i \rvert}{\sum_{i=1}^{d} (P_i + Q_i)}$
6	Gower	$d_{gow} = \frac{1}{d} \sum_{i=1}^{d} \lvert P_i - Q_i \rvert$
7	Soergel	$d_{sg} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}{\sum_{i=1}^{d} \max(P_i, Q_i)}$
8	Kulczynski	$d_{kul} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}{\sum_{i=1}^{d} \min(P_i, Q_i)}$
9	Canberra	$d_{Can} = \sum_{i=1}^{d} \frac{\lvert P_i - Q_i \rvert}{P_i + Q_i}$

10	Lorentzian	$d_{Lor} = \sum_{i=1}^{d} \ln\left(1 + \lvert P_i - Q_i \rvert\right)$
11	Intersection (1)	$s_{is} = \sum_{i=1}^{d} \min\left(P_i, Q_i\right)$
12	Intersection (2)	$d_{non-is} = 1 - s_{is}$
13	Intersection (3)	$d_{non-is} = \frac{1}{2} \sum_{i=1}^{d} \lvert P_i - Q_i \rvert$
14	Wave Hedges (1)	$d_{WH} = \sum_{i=1}^{d} \left(1 - \frac{\min(P_i,Q_i)}{\max(P_i,Q_i)}\right)$
15	Wave Hedges (2)	$d_{WH} = \sum_{i=1}^{d} \frac{\lvert P_i - Q_i \rvert}{\max(P_i,Q_i)}$
16	Czekanowski	$s_{Cze} = \frac{2\sum_{i=1}^{d} \min(P_i,Q_i)}{\sum_{i=1}^{d}(P_i+Q_i)}$
17	Czekanowski	$d_{Cze} = 1 - s_{Cze} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}{\sum_{i=1}^{d}(P_i+Q_i)}$
18	Motyka	$s_{Mot} = \frac{\sum_{i=1}^{d} \min(P_i,Q_i)}{\sum_{i=1}^{d}(P_i+Q_i)}$
19	Motyka	$d_{Mot} = 1 - s_{Mot} = \frac{\sum_{i=1}^{d} \max(P_i,Q_i)}{\sum_{i=1}^{d}(P_i+Q_i)}$
20	Kulczynski	$s_{Kul} = \frac{1}{d_{Kul}} = \frac{\sum_{i=1}^{d} \min(P_i,Q_i)}{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}$
21	Ruzicka	$s_{Ruz} = \frac{\sum_{i=1}^{d} \min(P_i,Q_i)}{\sum_{i=1}^{d} \max(P_i,Q_i)}$
22	Tanimoto	$d_{Tani} = \frac{\sum_{i=1}^{d} P_i + \sum_{i=1}^{d} Q_i - 2\sum_{i=1}^{d} \min(P_i,Q_i)}{\sum_{i=1}^{d} P_i + \sum_{i=1}^{d} Q_i - \sum_{i=1}^{d} \min(P_i,Q_i)}$
23	Tanimoto	$d_{Tani} = \frac{\sum_{i=1}^{d}(\max(P_i,Q_i) - \min(P_i,Q_i))}{\sum_{i=1}^{d} \max(P_i,Q_i)}$
24	Inner Product	$s_{IP} = P \cdot Q = \sum_{j=1}^{d} P_i Q_i$
25	Harmonic mean	$s_{HM} = 2\sum_{i=1}^{d} \frac{P_i Q_i}{P_i + Q_i}$
26	Cosine	$s_{cos} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sqrt{\sum_{i=1}^{d} P_i^2}\sqrt{\sum_{i=1}^{d} Q_i^2}}$
27	Kumar-Hassebrook (PCE)	$s_{kh} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$

28	Jaccard sim	$s_{Jac} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$
29	Jaccard dist	$d_{Jac} = 1 - s_{Jac} = \frac{\sum_{i=1}^{d} (P_i - Q_i)^2}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$
30	Dice	$s_{Dice} = \frac{2 \sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2}$
31	Dice	$d_{Dice} = 1 - s_{Dice} = \frac{\sum_{i=1}^{d} (P_i - Q_i)^2}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2}$
32	Fidelity	$s_{Fid} = \sum_{i=1}^{d} \sqrt{P_i Q_i}$
33	Bhattacharyya	$d_B = - \ln \sum_{i=1}^{d} \sqrt{P_i Q_i}$
34	Hellinger	$d_H = \sqrt{2 \sum_{i=1}^{d} \left(\sqrt{P_i} - \sqrt{Q_i} \right)^2}$
35	Hellinger	$d_H = 2 \sqrt{1 - \sum_{i=1}^{d} \sqrt{P_i Q_i}}$
36	Matusita	$d_M = \sqrt{\sum_{i=1}^{d} \left(\sqrt{P_i} - \sqrt{Q_i} \right)^2}$
37	Matusita	$d_M = \sqrt{2 - 2 \sum_{i=1}^{d} \sqrt{P_i Q_i}}$
38	Squared-chord	$d_{sqc} = \sum_{i=1}^{d} \left(\sqrt{P_i} - \sqrt{Q_i} \right)^2$
39	Squared-chord	$s_{sqc} = 2 \sum_{i=1}^{d} \sqrt{P_i Q_i} - 1$
40	Squared Euclidian	$d_{sqe} = \sum_{i=1}^{d} (P_i - Q_i)^2$
41	Pearson χ^2	$d_P(P, Q) = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{Q_i}$
42	Neyman χ^2	$d_N(P, Q) = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i}$
43	Squared χ^2	$d_{SqChi} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i + Q_i}$
44	Probabilistic Symmetric χ^2	$d_{PChii} = 2 \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i + Q_i}$
45	Divergence	$d_{Div} = 2 \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{(P_i + Q_i)^2}$

46	Clark	$d_{Clk} = \sqrt{\sum_{i=1}^{d} \left(\frac{\lvert P_i - Q_i \rvert}{P_i + Q_i} \right)^2}$
47	Additive Symmetric χ^2	$d_{AdChi} = \sum_{i=1}^{b} \frac{(P_i - Q_i)^2 (P_i + Q_i)}{P_i Q_i}$
48	Kullback-Leibler	$d_{KL} = \sum_{i=1}^{d} P_i \ln \frac{P_i}{Q_i}$
49	Jeffreys	$d_J = \sum_{i=1}^{d} (P_i - Q_i) \ln \frac{P_i}{Q_i}$
50	K divergence	$d_{Kdiv} = \sum_{i=1}^{d} P_i \ln \frac{2P_i}{P_i + Q_i}$
51	Topsøe	$d_{Top} = \sum_{i=1}^{d} \left(P_i \ln \left(\frac{2P_i}{P_i + Q_i} \right) + Q_i \ln \left(\frac{2Q_i}{P_i + Q_i} \right) \right)$
52	Jensen-Shannon	$d_{JS} = \frac{1}{2} \left[\sum_{i=1}^{d} P_i \ln \left(\frac{2P_i}{P_i + Q_i} \right) + \sum_{i=1}^{d} Q_i \ln \left(\frac{2Q_i}{P_i + Q_i} \right) \right]$
53	Jensen difference	$d_{JD} = \sum_{i=1}^{b} \left[\frac{P_i \ln P_i + Q_i \ln Q_i}{2} - \left(\frac{P_i + Q_i}{2} \right) \ln \left(\frac{P_i + Q_i}{2} \right) \right]$
54	Taneja	$d_{TJ} = \sum_{i=1}^{d} \left(\frac{P_i + Q_i}{2} \right) \ln \left(\frac{P_i + Q_i}{2\sqrt{P_i Q_i}} \right)$
55	Kumar-Johnson	$d_{KJ} = \sum_{i=1}^{d} \left(\frac{\left(P_i^2 - Q_i^2 \right)^2}{2 (P_i Q_i)^{3/2}} \right)$
56	Avg (L_1, L_∞)	$d_{ACC} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert + \max_i \lvert P_i - Q_i \rvert}{2}$
57	Vicis-Wave Hedges	$d_{e1} = \sum_{i=1}^{d} \frac{\lvert P_i - Q_i \rvert}{\min(P_i, Q_i)}$
58	Vicis-Symmetric χ^2	$d_{e2} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{\min(P_i, Q_i)^2}$
59	Vicis-Symmetric χ^2	$d_{e3} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{\min(P_i, Q_i)}$
60	Vicis-Symmetric χ^2	$d_{e4} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{\max(P_i, Q_i)}$
61	max-Symmetric χ^2	$d_{e5} = \max \left(\sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i}, \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{Q_i} \right)$
62	min-Symmetric χ^2	$d_{e6} = \min \left(\sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i}, \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{Q_i} \right)$

Appendix B

Approaches Used at The TREC News Track (2018-2021)

Below are descriptions of the approaches taken by participants at the TREC News Track shared task for years 2018 (Soboroff et al. (2018)), 2019 (Soboroff et al. (2020a)), 2020 (Soboroff et al. (2020b)), 2021 (Soboroff (2021)) shown in Figure 2.6. The approaches are listed alphabetically.

BERT See Section 2.3.1

BERT extractive summarization was used by (Ak et al. (2020)) and performs document sentence embedding using the BERT and K-Means clustering algorithm to group all sentences in a document and determine the sentences closest to each group centroid (Miller (2019)). Summarizing a document with a desired number of its own sentences.

BERT query evaluation was used by (Sethi and Deshmukh (2021)) together with query generation to run a semantic analysis of candidate queries and select the closest ones to the document.

BM25 was used by the following participants (Boers et al. (2020); Ding et al. (2019); Engelmann and Schaer (2021); Foley, Montoly, and Pena (2019); Gautam, Mitra, and Roy (2020); Khloponin and Kosseim (2019, 2020); Koster and Foley (2021); Lirong et al. (2021); Missaoui, MacFarlane, Makri, and Gutierrez-Lopez (2019); Naseri et al. (2018); Qu and Wang (2019); Sethi and Deshmukh (2021); P. Yang and Lin (2018)) For more details, see Section 2.1.3.

Coordinate ascent was used by (Koster and Foley (2021)), it is an optimization technique of multivariate objective functions (Metzler and Bruce Croft (2007)). It is used to directly optimize the evaluation function by performing a series of one-dimensional searches, fixing all parameters of the evaluation function except the one being optimized and repeating it for all parameters.

Cosine similarity is one of the most commonly used similarity functions for vectors in information retrieval. In the TREC News Track alone, it was used by 13 out of 24 proceedings papers (Ak et al. (2020); Boers et al. (2020); Cabrera-Diego et al. (2021); Day et al. (2020); Foley et al. (2019); Gautam et al. (2020); Khloponin and Kosseim (2019, 2020); Koster and Foley (2021); Lirong et al. (2021); Naseri et al. (2018); Qu and Wang (2019); Sethi and Deshmukh (2021)). The Cosine similarity, along with many other measures is defined in Appendix A. It is often considered the go-to similarity metric because it is easy to compute, it is indifferent to the vector lengths and conveniently evaluated to the range $[-1; 1]$ where 1 denotes maximum similarity, -1 maximum dissimilarity or opposite meaning, and 0 for orthogonal vectors. However, as our work shows (see Section 3.3.3) the cosine similarity is not always the most precise metric.

Doc2Query was used by (Lirong et al. (2021)), it is a supervised sequence–to–to–sequence transformer model which for a given document predicts questions to which the document can potentially answer (Nogueira, Yang, Lin, and Cho (2019)). The fact that the model is working in a continuous vector space allows it to return questions with relevant keywords potentially not seen in the document. These questions could be used for document expansion, providing relevant keywords and enriching the document.

Doc2Vec was used by (Ak et al. (2020); Khloponin and Kosseim (2019)), it is very similar to Word2Vec, but during training, in addition to word context, Doc2Vec contains a document ID which allows model to capture which words are more probable for a given document, and which documents are more probable for a given word, leading to a vector representation for the document. See Section

3.2.1 for more details.

DocT5query was used by (Lirong et al. (2021)) in the context of the TREC News Track, it is the latest model of Doc2Query family based on T5 (Text-To-Text Transfer Transformer, Raffel et al. (2020)). Replacing a simple transformer in Doc2Query architecture with T5 model helped improve the performance of the model (Nogueira and Lin (2019)).

Document expansion was used at the TREC News Track by (Lirong et al. (2021)). Document expansion is similar to query expansion, but in this case the document text is augmented. In addition to the techniques described for query expansion, good results were obtained by the same team at TREC News for documents expanded by the corresponding search queries generated with Doc2Query, initially proposed by (Nogueira et al. (2019)).

ElasticSearch is an enterprise level, distributed search engine based on the Lucene library. Elasticsearch[3] compared to Lucene does not require any programming for indexing and retrieval. Nevertheless, it exposes all the details of the Lucene library necessary for configuration and tuning. Elasticsearch also implements many functions that Lucene does not have, for example, dense vector fields and proximity-based similarity for them. Our work specifically used these characteristics and extended them with additional proximity measures (see Chapter 3.3).

Feature extraction includes any features specifically created for a document collection or task. In the case of the TREC News Track, several teams (Bimantara et al. (2018); Essam and Elsayed (2021); Sethi and Deshmukh (2021); Wagenpfeil, Kevitt, and Hemmje (2021)) tried to use the publication date to filter out backlinks or author names, news categories, or any other keyword extraction technique.

Named Entities is one of the most commonly used feature in the TREC News Track, it was used by (Boers et al. (2020); Cabrera-Diego et al. (2021); Day et al. (2020); Engelmann and Schaer (2021)). News articles are focused on events, persons, companies and brands, dates, and locations. One should think that relevant articles have at least partial overlap in these concepts, and giving different priorities to different named entity types could help with the ranking.

PKE (Boudin (2016)) is an open source keyphrase extraction toolkit based on Python and was used in the TREC News Track by (Bimantara et al. (2018); Foley et al. (2019); Lirong et al. (2021)) for feature extraction. Keyphrases are words and phrases that capture the main ideas of a document useful for summarization, question-answering, and information retrieval. PKE in the same pipeline can use and combine algorithms and pretrained models based on statistical, graph-based, and feature-based methods. It makes it easy to use in a broad range of NLP tasks. Default models supplied with PKE are trained on SemEval-2010 dataset (Kim, Medelyan, Kan, and Baldwin (2010)).

Although feature extraction is commonly used, it does not solve the task by itself and must be used in combination with other approaches.

Jaccard similarity (see Appendix A) is another commonly used similarity measure in information retrieval for real and binary vectors. Measures how large the overlap between two vectors is. It was used by (Khloponin and Kosseim (2020); Qu and Wang (2019)) in the context of the TREC News Track.

K-Means was used by (Lopez-Ubeda, Diaz-Galiano, Martin-Valdivia, and Urena-Lopez (2018)), it is an iterative clustering algorithm for vector data (MacQueen et al. (1967)). For a given k it starts by randomly initializing k cluster centers, associates all data points with the closest cluster center, calculates the new cluster centers with the minimal sum of squared Euclidean distance from the associated data points. These steps are repeated for the updated cluster centers until all data points remain in the same cluster.

Language modeling was used by the following teams (Ding et al. (2019); Lopez-Ubeda et al. (2018); Lu and Fang (2019, 2020); Sethi and Deshmukh (2021); P. Yang and Lin (2018)) in the context of the TREC News Track. Having a large corpus of documents in a specific language allows us to build a model that can learn word distributions in that language. It is called a *unigram* model (Manning et al. (2008)) if all tokens in a sequence (it could be characters, subwords, or words, depending on the model) are considered independent of each other. The weight of each token in this case only depends on the probability of this token in a document or a corpus. *N-gram* model in contrast, relies on the probability of n sequential tokens. *N-gram* models compared to *unigram* models have some notion of context.

Lucene is an open source search engine library [1], originally written in Java and ported to many other languages. It was released in 1992 and today has become the de facto standard for implementing full text search for document collection. Many popular databases such as MongoDB Atlas[2], Elasticsearch[3], Apache Solr[4] are built around the Lucene library (Bialecki, Muir, and Ingersoll (2012)). At least 18 of the 24 proceedings papers (Soboroff (2021); Soboroff et al. (2018, 2020a, 2020b)) of TREC News Track have used the Lucene library or a tool based on it.

Query expansion was used in the context of the TREC News Track by (Lirong et al. (2021); Missaoui et al. (2019); P. Yang and Lin (2018)). Query expansion is a process of enhancing or reorganizing a query to improve the retrieved results (Maron and Kuhns (1960)). This process could be as simple as spelling correction, adding morphological forms of words in the query, adding synonyms and antonyms or adding common bigrams seen in the corpus. More advanced forms of query expansion include, adding information associated with named entities in the query: geographical locations for events, positions for political figures, sports and team names for athletes, etc. (Carpineto and Romano (2012)). Query expansion improves recall by adding words not present in a query directly but probably assumed by a user.

Query likelihood combines language modeling and Bayes' rule to rank documents for a specific query q. If a language model M_d is build for each document d, the probability of the query q for this model $P(q \mid d)$ can be calculated and used to rank documents by applying Bayes' rule:

$$P(d \mid q) = P(q \mid d)P(d)/P(q) \tag{12}$$

Since $P(q)$ has the same value for all documents, it can be omitted. $P(d)$ can either be treated as a uniform distribution and ignored leading to $P(d \mid q) \sim P(q \mid d)$, or can be given a prior

[1]https://lucene.apache.org/
[2]https://www.mongodb.com/atlas
[3]https://www.elastic.co/
[4]https://solr.apache.org/

probability $P(d)$ for each document, leading to $P(d \mid q) \sim P(d)P(q \mid d)$, where $P(d)$ could serve as a weighting coefficient. This coefficient could be calculated based on document authority, PageRank, length, freshness, etc. (Manning et al. (2008)). Query likelihood was used by (Foley et al. (2019); Lu and Fang (2020); Naseri et al. (2018); P. Yang and Lin (2018)) at the TREC News Track.

Random forest was used by (Koster and Foley (2021)), it is a classification or regression machine learning approach based on a set of decision trees. In the classification task, the class is decided based on the votes of individual trees from the ensemble. In the regression task, we take the average of the individual trees votes.

Relation extraction is the task of identifying relations and attributes of relations for entities in a sentence or a text. For example, the subject-object relationship for two entities in a sentence. Due to the lack of labeled data and the noisy nature of the labels which do exist, some participants (eg. Engelmann and Schaer (2021)) used the verb connecting two entities as a label for the relation.

Relevance feedback and pseudo relevance feedback was used by (Ding et al. (2019); Missaoui et al. (2019); Naseri et al. (2018)) at the TREC News shared task. If a user is given an opportunity to provide feedback on the initial results returned from his query, the model can compute a better representation of the information need and improve the query results by boosting relevant documents and penalizing irrelevant ones. This is called relevance feedback. (Ding et al. (2019)) used the specific relevance feedback method called Rocchio feedback which modifies the initial query vector by moving it in vector space away from irrelevant documents and closer to the relevant ones, improving the retrieval system performance (Manning et al. (2008)). The TREC News Track task, does not provide feedback information, but some teams assumed that the top k results returned by the initial retrieval are relevant and use them instead of user relevance feedback; this is called pseudo relevance feedback (Manning et al. (2008)).

Relevance model was used by (Ding et al. (2019); Foley et al. (2019); Naseri et al. (2018)), it is estimating the similarity for each query-document pair and uses this similarity for document ranking with the most similar on top (S. E. Robertson and Jones (1976) ; S. Robertson and Zaragoza (2009)). Similarity is defined as shown in equation 13, where R is an ideal set of documents user would like to receive in response of query q.

$$sim(D_i, q) = \frac{P(R \mid D_i)}{P(\bar{R} \mid D_i)} \tag{13}$$

Re-ranking is an approach that combines two ranking systems working sequentially. Usually, the first system is used to retrieve an ordered or unordered set of relevant documents; when the second system is used to create the final order of the results. Re-ranking is beneficial for large collections where a less accurate but faster first system is used to reduce the collection to a size suitable for the more precise reranking system, which is usually more complex or more computationally demanding. Re-ranking was used at the TREC News Track by several teams including (Ak et al. (2020); Bimantara et al. (2018); Boers et al. (2020); Cabrera-Diego et al. (2021); Engelmann and Schaer (2021); Essam and Elsayed (2019, 2021); Koster and Foley (2021); Lirong et al. (2021); Qu and Wang (2019)).

Reverse models are appropriate when the query and the documents are of similar size. These models consider a potential document as a query and the original query as a document to rank. Reverse models rely on the assumption that most retrieval systems are not symmetric and switching queries and documents helps to get additional information about document similarity. It also might help with document or query expansion. In the TREC News Track (Foley et al. (2019)) used reverse models.

Sentence-BERT Sentence-pair regression task, like semantic textual similarity (STS), with BERT architecture requires two sentences for which similarity is calculated to be fed to the model simultaneously, causing computational overhead and leading to $O(n^2)$ operations of the number of sentences to be performed in a task like clustering. Even for a relatively small n of 10,000 sentences it might take ~ 65 hours on a modern V100 GPU. SBERT model (Reimers and Gurevych (2019)) using pooling operation on the output of BERT model, provides a fixed size single sentence embedding, allowing to move the sentence similarity calculation in a vector space, drastically reducing the calculation overhead (from 65 hours to 5 seconds for an example above). Sentence-BERT was used at the TREC News Track by (Cabrera-Diego et al. (2021); Day et al. (2020); Sethi and Deshmukh (2021)).

TextRank was used by (Bimantara et al. (2018); Boers et al. (2020); Foley et al. (2019)), it is a graph-based unsupervised text ranking algorithm (Mihalcea and Tarau (2004)) used for keywords and sentence extraction, inspired by PageRank (Page, Brin, Motwani, and Winograd (1999)) algorithm. It requires to select the text unit best suited for the task (word, title, sentence, paragraph, chapter, etc.) which will represent graph nodes; identify the relations between text units (cooccurrence, similarity, etc.) which will represent graph edges, edges could be weighted or unweighted, directed or undirected; iterate the graph-based ranking algorithm on resulted graph until it converges; sort graph nodes based on their score.

TF-IDF For more details see Section 2.1.2. TF-IDF was used at the TREC News Track by (Day et al. (2020); Ding et al. (2019); Foley et al. (2019); Gautam et al. (2020); Koster and Foley (2021); Lopez-Ubeda et al. (2018); Qu and Wang (2019); Sethi and Deshmukh (2021); Wagenpfeil et al. (2021); P. Yang and Lin (2018)).

Topic modeling was applied by (Wagenpfeil et al. (2021)) in the context of the TREC News Track, it is a statistical generative model that discovers abstract topics from observations (Muresan and Harper (2004)). This model can explain observations, like a certain probability of a word being in a document by this document being marked by a mixture of a small number of topics. Once the topics have been established, they can be used to group similar documents.

Transformers is a family of NLP and computer vision deep neural network models based on attention mechanism (Vaswani et al. (2017)). See Section 2.3.1 for more details. Transformers were use at the TREC News Track by (Ak et al. (2020); Cabrera-Diego et al. (2021); Day et al. (2020); Lirong et al. (2021)).

Universal sentence encoder was used by (Ak et al. (2020)) in the context of the TREC News Track, it is an architecture for encoding sentences with a focus on transfer learning to other NLP tasks. It has two alternative implementations with a trade-off between accuracy and performance. More accurate transformer based architecture is based on the encoding subgraph of the transformer model (Vaswani et al. (2017)), which uses an attention mechanism to calculate the contextualized representation of words in a sentence relying both on the ordering and identity of all other words. Element wise sum of the representation at each word position is calculated giving a sentence encoding. More computationally efficient one, based on Deep Averaging Networks (Iyyer, Manjunatha, Boyd-Graber, and Daumé III (2015)), in which input word embeddings are averaged first and then passed through a feedforward deep neural network, producing sentence embeddings.

Vector Space Model (VSM) see Section 2.1.

Word2Vec is a family of shallow neural network models to learn word embeddings from a large corpus of text (Mikolov, Chen, et al. (2013); and Mikolov, Sutskever, et al. (2013)). There are two main approaches: CBOW (continuous bag of words) and skipgram. In the CBOW architecture, the target is to predict a word using its context; whereas in the skip-gram model, the context of a given word is predicted given the word. Both of these approaches learn an internal representation for words during the training phase, which can be word vectors. (Gautam et al. (2020)) used Word2Vec embedding for the TREC News Track.

Word embedding was used by (Boers et al. (2020); Foley et al. (2019); Naseri et al. (2018)), it groups various techniques to represent words as real value vectors. Typically, words that are close to each other in the vector space have related meanings. Word embeddings could be built using neural networks, dimensionality reduction techniques, word co-occurrence matrices, etc.

Appendix C

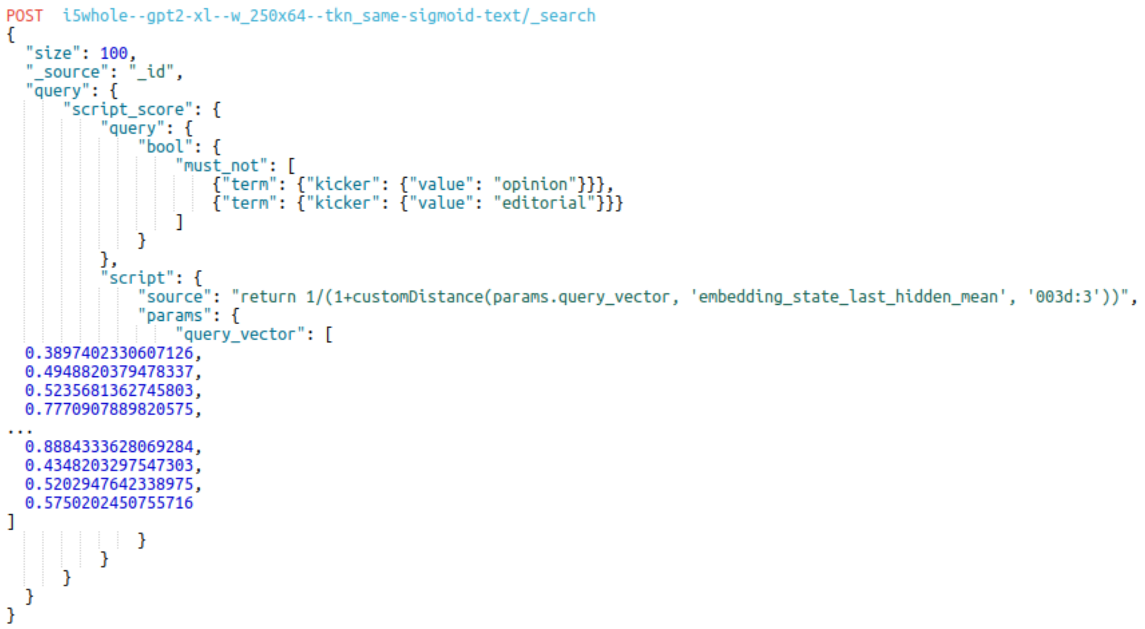
Tools

This work would not be possible without the tools developed over the years and open sourced by their creators. All the work to read, write, preprocess, index and rank the documents was run inside the GNU/Linux[1] environment. Gzip[2] utility provided easy compression and decompression of raw data, preprocessed texts, document embeddings, and ranking results, keeping all of this within 2TB of disk space. Parallel[3] utility helped us to speed up most of the operation at least 10 times by using all available cores on the machine that runs the experiments. Python[4], Pytorch[5] and Transformers[6] from Huggingface[7] provided us with the implementation of the deep learning models and the interpreter to execute Python code. GNU Bash[8] helped to organize a seamless pipeline to run all the steps from reading the raw text of the articles to obtaining the ranking results for every system configuration. OpenRefine[9] helped us to sift through tens of thousands of ranking results of different system configurations and to build valuable insights.

Elasticsearch[10] provided us with a BM25 implementation which was used for the baseline retrieval model. The Elasticsearch core source code was modified to add all necessary proximity

[1] https://www.gnu.org/gnu/linux-and-gnu.en.html
[2] https://www.gnu.org/software/gzip/
[3] https://www.gnu.org/software/parallel/
[4] https://www.python.org/
[5] https://pytorch.org/
[6] https://huggingface.co/docs/transformers/index
[7] https://huggingface.co/
[8] https://www.gnu.org/software/bash/
[9] https://openrefine.org/
[10] https://www.elastic.co/elasticsearch/

functions (see Appendix A) and custom ranking functionality to extend document retrieval in vector space. All modifications to the Elasticsearch source code are publicly available in the GitHub repository [11].



```
POST  i5whole--gpt2-xl--w_250x64--tkn_same-sigmoid-text/_search
{
  "size": 100,
  "_source": "_id",
  "query": {
    "script_score": {
      "query": {
        "bool": {
          "must_not": [
            {"term": {"kicker": {"value": "opinion"}}},
            {"term": {"kicker": {"value": "editorial"}}}
          ]
        }
      },
      "script": {
        "source": "return 1/(1+customDistance(params.query_vector, 'embedding_state_last_hidden_mean', '003d:3'))",
        "params": {
          "query_vector": [
  0.3897402330607126,
  0.4948820379478337,
  0.5235681362745803,
  0.7770907889820575,
...
  0.8884333628069284,
  0.4348203297547303,
  0.5202947642338975,
  0.5750202450755716
]
          }
        }
      }
    }
}
```

Figure C.1: Example of Elasticsearch query with proximity based functionality and query vector (only first and last 4 components of query vector are displayed). `003d:3` is a code for specific proximity function, `embedding_state_last_hidden_mean` is a name of the field storing embedding vectors.

Because modifications were applied in the core of the Elasticsearch database and precomputed document embeddings vectors were indexed directly in the database, it was possible to use native Elasticsearch syntax to query ranked documents from raw ebmeddings. Example of such a query presented in Figure C.1. As you can see, it is possible to add extra filters, for example exclude opinion or editorial articles.

[11] https://github.com/pkhlop/nist-news-trec/

Appendix D

Aggregated Experimental Results

models	**total**	**≤ 0.1 / %**
bert-base-cased	3,060	2125 / 69%
bert-base-multilingual-cased	3,060	2185 / 71%
bert-base-multilingual-uncased	3,060	2113 / 69%
bert-base-uncased	3,043	2062 / 68%
bert-large-cased	3,060	2106 / 69%
bert-large-uncased	3,043	2141 / 70%
all BERT models	**18,326**	**12,732 / 69%**

Table D.1: Experiments with BERT as an embedding model and the portion of it which reached nDCG@5 of only 0.1 or less (lower is better).

models	**total**	**≤ 0.1 / %**
distilroberta-base	3,060	1238 / 40%
roberta-base	3,060	1287 / 42%
roberta-base-openai-detector	3,060	3046 / 99%
roberta-large	3,060	1287 / 42%
roberta-large-openai-detector	3,060	3027 / 99%
all RoBERTa models	**15,300**	**9,885 / 65%**

Table D.2: Experiments with RoBERTa as an embedding model.

models	**total**	**≤ 0.1 / %**
google-pegasus-multi_news	1,530	516 / 34%
google-pegasus-newsroom	1,530	551 / 36%
all PEGASUS models	**3,060**	**1,067 / 35%**

Table D.3: Experiments with PEGASUS as an embedding model.

models	**total**	**≤ 0.1 / %**
openai-gpt / GPT	1,530	580 / 38%
gpt2	1,530	707 / 46%
gpt2-large	1,530	560 / 36%
gpt2-medium	1,498	686 / 46%
gpt2-xl	1,530	568 / 37%
all GPT2 models	**6,088**	**2,521 / 41%**

Table D.4: Experiments with GPT as an embedding model.

models	**total**	**≤ 0.1 / %**
xlnet-base-cased	1,530	607 / 40%
xlnet-large-cased	1,498	605 / 40%
all XLNet models	**3,028**	**1,212 / 40%**

Table D.5: Experiments with XLNet as an embedding model.

window size	**total**	**≤ 0.1 / %**
250	23,649	13,876 / 59%
500	23,683	14,027 / 59%

Table D.6: Experiments with different window size parameters.

normalization	**total**	**≤ 0.1 / %**
plain	15,746	10,716 / 68%
amplitude	15,776	8,711 / 55%
sigmoid	15,810	8,476 / 54%

Table D.7: Experiments with different normalization parameters.

pooling	**total**	**≤ 0.1 / %**
max	15,810	9,424 / 60%
mean	15,810	8,917 / 56%
min	15,712	9,562 / 61%

Table D.8: Experiments with different pooling parameters.

output layer	**total**	**≤ 0.1 / %**
last hidden	30,536	14,048 / 46%
pooler output	16,796	14,069 / 84%

Table D.9: Experiments with different output layers.

proximity metric	**total**	**≤ 0.1 / %**
Minkowski $L_{p=-0.1}$	558	558 / 100%
Minkowski $L_{p=-0.3}$	558	558 / 100%
Minkowski $L_{p=-0.5}$	558	558 / 100%
Minkowski $L_{p=-0.7}$	558	558 / 100%
Minkowski $L_{p=-0.9}$	558	558 / 100%
Minkowski $L_{p=-1}$	558	558 / 100%

Minkowski $L_{p=-2}$	558	558 / 100%
Minkowski $L_{p=-3}$	558	558 / 100%
Minkowski $L_{p=-4}$	558	558 / 100%
Minkowski $L_{p=-5}$	558	558 / 100%
Minkowski $L_{p=-6}$	558	558 / 100%
Minkowski $L_{p=-7}$	558	558 / 100%
Minkowski $L_{p=0.1}$	558	382 / 68%
Minkowski $L_{p=0.3}$	558	368 / 66%
Minkowski $L_{p=0.5}$	558	362 / 65%
Minkowski $L_{p=0.7}$	558	243 / 44%
Minkowski $L_{p=0.9}$	558	182 / 33%
Minkowski $L_{p=1}$, Manhattan	558	182 / 33%
Minkowski $L_{p=2}$, Euclidean	558	198 / 35%
Minkowski $L_{p=3}$	558	207 / 37%
Minkowski $L_{p=4}$	558	224 / 40%
Minkowski $L_{p=5}$	558	268 / 48%
Minkowski $L_{p=6}$	558	295 / 53%
Minkowski $L_{p=7}$	558	330 / 59%
Minkowski Lp	**13,392**	**9,937 / 74%**
Chebyshev L_{∞}	558	412 / 74%
Sørensen	558	186 / 33%
Gower	558	182 / 33%
Soergel	558	201 / 36%
Kulczynski (1)	558	200 / 36%
Kulczynski (2)	558	200 / 36%
Canberra	556	250 / 45%
Lorentzian	558	180 / 32%
Intersection (1)	558	471 / 84%

Intersection (2)	558	182 / 33%
Intersection (3)	558	186 / 33%
Intersection (4)	558	231 / 41%
Wave Hedges (1)	556	256 / 46%
Wave Hedges (2)	556	256 / 46%
Czekanowski	558	263 / 47%
Motyka	558	263 / 47%
Ruzicka	558	252 / 45%
Tanimoto (1)	558	201 / 36%
Tanimoto (2)	558	201 / 36%
Inner Product	558	485 / 87%
Harmonic mean	556	482 / 87%
Cosine	558	454 / 81%
Kumar-Hassebrook (PCE)	558	199 / 36%
Jaccard dist	558	198 / 35%
Jaccard sim	558	199 / 36%
Dice dist	558	558 / 100%
Dice sim	558	199 / 36%
Fidelity	554	554 / 100%
Bhattacharyya	554	554 / 100%
Hellinger (1)	554	192 / 35%
Hellinger (2)	554	554 / 100%
Matusita	554	191 / 34%
Matusita	554	554 / 100%
Squared-chord (1)	554	191 / 34%
Squared-chord (2)	554	554 / 100%
Squared Euclidian	558	198 / 35%
Pearson χ^2	556	238 / 43%

Neyman χ^2	556	297 / 53%
Squared χ^2	556	249 / 45%
Probabilistic Symmetric χ^2	556	249 / 45%
Divergence	556	230 / 41%
Clark	556	225 / 40%
Additive Symmetric χ^2	556	274 / 49%
Kullback-Leibler	554	554 / 100%
Jeffreys	554	194 / 35%
K divergence	554	554 / 100%
Topsøe	554	323 / 58%
Jensen-Shannon	554	554 / 100%
Jensen difference	554	487 / 88%
Taneja	554	296 / 53%
Kumar-Johnson	554	193 / 35%
Avg (L_1, L_∞)	558	182 / 33%
Vicis-Wave Hedges	556	263 / 47%
Vicis-Symmetric χ^2 (1)	556	261 / 47%
Vicis-Symmetric χ^2 (2)	556	259 / 47%
Vicis-Symmetric χ^2 (3)	556	257 / 46%
max-Symmetric χ^2	556	262 / 47%
min-Symmetric χ^2	556	273 / 49%
Total	**47,332**	**27,997 / 59%**

Table D.10: Experiments with different proximity measure.