

**DEVELOPING TOOLS TO SUPPORT THE SEARCH NEEDS OF NEWS
READERS AND NEWS WRITERS**

by
KENAN FAYOUMÍ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Computer Science

Sabancı University
January 2022

**DEVELOPING TOOLS TO SUPPORT THE SEARCH NEEDS OF NEWS
READERS AND NEWS WRITERS**

Approved by:



Date of Approval: January 12, 2022



KENAN FAYOUMI 2022 ©

All Rights Reserved

ABSTRACT

DEVELOPING TOOLS TO SUPPORT THE SEARCH NEEDS OF NEWS READERS AND NEWS WRITERS

KENAN FAYOUMI

MS in Computer Science THESIS, JANUARY 2022

Thesis Supervisor: Asst. Prof. REYYAN YENİTERZİ

Keywords: Wikification, Entity Linking, Entity Ranking, Background Linking,
Information Retrieval

The ongoing digitization of online news has changed and democratized the industry of news writing. The huge increase in the number of news sources has called for research on automated methods that link relevant news articles or entities that provide background information and enhance the reader's experience. In this work, we tackle three different tasks in the context of news articles: Wikification, Entity Ranking, and Background Linking. The work done on these tasks is in alignment with the tasks in News Track of Text REtrieval Conference (TREC). In Wikification, we detect a list of mentioned entities in articles, link them to their corresponding Wikipedia entry and rank the list of entities in terms of relevance to the article. Standalone Entity Ranking task is only concerned with the final ranking step of Wikification where the list mentioned entities are given. As for Background Linking, the task is to retrieve and rank a list of relevant articles given a query news article.

Our proposed solutions for these tasks are oriented towards deep modelling and using vector representations to estimate similarity and relevance. For Entity Ranking, we encode news articles and entities using Doc2Vec then use proximity between the pair to rank entities. As for Wikification, we use transformer-based architectures for detecting entity mentions and encoding mentions and entities into vector representations. These vectors are used for candidate retrieval and ranking as part of the entity linking pipeline. In Background Linking, we again use a transformer-based language model to encode news articles and fine-tune it for relevance ranking between articles.

For evaluation, we compare our approaches to classic information retrieval systems to analyze the quality or increase in performance brought by using deep complex architectures. Using Doc2Vec and Cosine similarity to measure relevance in a setting of perfect entity linking yields high performances. In Wikification, encoding mentions and performing dense vector search for candidate retrieval performs on-par with baseline. However, using contextual encoding for candidate entity ranking significantly improves the Wikification performance. The transformer-based re-ranker used in Background Linking does not improve over full-text search baseline but shows promising improvements in results when provided with more data for fine-tuning.



ÖZET

HABER OKUYUCULARI VE YAZARLARININ HABER ARAMA İHTİYAÇLARI İÇİN ARAÇLAR GELİŞTİRİLMESİ

KENAN FAYOUMI

BİLGİSAYAR MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, OCAK 2022

Tez Danışmanı: Asst. Prof. REYYAN YENİTERZİ

Anahtar Kelimeler: vikifikasyon, varlık sıralaması, geçmiş bağlantısı, doğal dil işleme, bilgi getirmesi ve çıkarımı

Çevrimiçi haberlerdeki dijitalleşme, haber yazma endüstrisini hem değiştiriyor hem de demokratikleştiriyor. Son yıllarda haber kaynaklarının sayısındaki büyük artış, ilgili haber makalelerini veya arka plan bilgisi sağlayan ve okuyucunun deneyimini geliştiren varlıkları birbirine bağlayan otomatik yöntemler üzerinde araştırma yapılmasını gerektirdi. Bu çalışmada, haber makaleleri bağlamında üç farklı görevi ele alıyoruz: Vikifikasyon, Varlık Sıralama ve Geçmiş Bağlantısı. Bu görevler üzerinde yapılan çalışma, News Track of Text Retrieval Conference (TREC) görevleriyle uyumludur. Vikifikasyonda, maddelerde adı geçen varlıkların bir listesini tespit eder, bunları ilgili Vikipedi girişlerine bağlar ve varlık listesini maddeyle alakalarına göre sıralarız. Bağımsız Varlık Sıralaması görevi, yalnızca listede belirtilen varlıkların verildiği Vikifikasyonun son sıralama adımı ile ilgilidir. Geçmiş Bağlantısında ise görev, bir sorgu haber makalesi için verilen ilgili makalelerin bir listesini almak ve sıralamaktır.

Bu görevler için önerilen çözümlerimiz, derin modellemeye ve benzerlik ve alaka düzeyini tahmin etmek için vektör temsillerini kullanmaya yöneliktir. Varlık Sıralaması için, Doc2Vec kullanarak haber makalelerini ve varlıkları kodlarız, ardından varlıkları sıralamak için çift arasındaki yakınlığı kullanırız. Vikifikasyona gelince, varlık ifadelerini tespit etmek ve bahsedeni ve varlıkları vektör temsillerine kodlamak için dönüştürücü tabanlı mimariler kullanıyoruz. Bu vektörler, varlığı bağlayan sistemin bir parçası olarak aday bulma ve sıralama için kullanılır. Geçmiş Bağlantısında, haber makalelerini kodlamak ve makaleler arasındaki alaka düzeyi sıralaması

için ince ayar yapmak için yine dönüştürücü tabanlı bir dil modeli kullanıyoruz.

Değerlendirme sırasında, derin karmaşık mimarileri kullanmanın getirdiği kaliteyi veya performans artışını analiz etmek için yaklaşımlarımızı klasik bilgi erişim sistemleriyle karşılaştırıyoruz. Varlık bağlama ortamında alaka düzeyini ölçmek için Doc2Vec ve Kosinüs benzerliğini kullanmanın yüksek performans sağladığı görülmüştür. Vikifikasyon'da, adayların belirlenmesi sırasında bağlamsal kodlamanın ile yoğun vektör araması yapmak, diğer yöntemlerle benzer performans göstermiştir. Bununla birlikte, aday varlık sıralaması için bağlamsal kodlamanın kullanılması, Vikifikasyon performansını önemli ölçüde artırır. Geçmiş Bağlantısında kullanılan dönüştürücü tabanlı yeniden sıralayıcı tam metin arama yöntemini iyileştirmemiştir, ancak ince ayar için daha fazla veri sağlandığında sonuçlarda umut verici gelişmeler gözlenmiştir.



ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Asst. Prof. Reyvan Yeniterzi for her continuous support, advice and patience during the course of my Master's studies. I am extremely grateful for her guidance and support throughout this journey.

I also would like to thank my colleagues Ali Eren Ak and Çağhan Köksal for their participation and efforts in this work.

Finally, my biggest thanks to my family for their love and support during this long journey of distance learning.



TABLE OF CONTENTS

LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF ABBREVIATONS	xiii
1. INTRODUCTION	1
1.1. Task Definition	2
1.1.1. Wikification	2
1.1.2. Entity Ranking	3
1.1.3. Background Linking	3
1.2. Research Questions	4
1.3. Thesis Outline	4
2. RELATED WORK	5
2.1. Entity Linking	5
2.1.1. Mention Detection	5
2.1.2. Candidate Generation/Retrieval	6
2.1.3. Candidate Ranking/Entity Disambiguation	6
2.1.4. Entity Representation	7
2.1.5. End-to-End Models	8
2.2. TREC News Wikification	8
2.3. Entity Ranking	9
2.4. Background Linking	11
3. DATASET	16
3.1. Washington Post News Corpus	16
3.2. Wikipedia	18
4. Wikification	19
4.1. Dataset	19

4.2. Methodology	21
4.2.1. Mention Detection	22
4.2.2. Candidate Generation	22
4.2.3. Candidate Ranking.....	24
4.2.4. Entity Ranking.....	25
4.3. Experiments	26
4.3.1. Mention Detection	26
4.3.2. Candidate Generation	28
4.3.3. Candidate Ranking.....	32
4.3.4. Entity Ranking.....	33
4.3.5. Other Participants Results	36
5. Entity Ranking.....	38
5.1. Dataset	38
5.2. Methodology	40
5.2.1. Doc2Vec.....	41
5.2.2. Vector Similarity and Ranking.....	41
5.2.3. Using Background Linked Articles	42
5.3. Results and Discussion.....	43
6. Background Linking	46
6.1. Dataset	46
6.2. Methodology	47
6.2.1. Baseline: Full-Text Search	48
6.2.2. Vector Similarity (Doc2Vec)	48
6.2.3. Neural Ranking for Ad-hoc Document Retrieval.....	48
6.3. Experiments and Results	50
6.3.1. Vector Similarity (Doc2Vec)	50
6.3.2. Neural Ranking for Ad-hoc Document Retrieval.....	51
6.3.3. Other Participants Results	51
7. Conclusion	53
7.1. Research Questions	54
7.2. Future Work	55
BIBLIOGRAPHY.....	56
APPENDIX A	61

LIST OF TABLES

Table 3.1. Washington Post Corpus Summary	16
Table 4.1. Wikification Scoring Methodology	20
Table 4.2. Mention Detection system scores on testing topics.....	26
Table 4.3. Recall@100 score for our different Elastic-Search query formulations.....	29
Table 4.4. Recall scores at different cutoffs for our retrieval models on 2020 and 2021 Test Topics.....	32
Table 4.5. Candidate Retrieval and Ranking performances reported in nDCG@5 and average minutes taken for inference	33
Table 4.6. Wikification’s nDCG@5 score for using the top K scoring entities for Entity Ranking	35
Table 4.7. nDCG@5 Wikification score for using Doc2Vec vector similarity in Entity Ranking. The list of document entities are linked by using our Cross-Encoder model.	35
Table 4.8. Wikification nDCG@5 score for 2020 and 2021 testing topics....	37
Table 5.1. Entity Ranking results using different Doc2Vec vector sizes	44
Table 5.2. Entity Ranking results compared to Other participants	44
Table 5.3. Entity Ranking Results Using Similar Articles.....	45
Table 6.1. Background Linking testing topics in different years.....	47
Table 6.2. Background Linking performance on different testing topics sets	50
Table 6.3. Background Linking performance of our approaches and different participants approaches on different testing topics sets.....	52

LIST OF FIGURES

Figure 1.1. Contextualization in News Articles	2
Figure 4.3. Entity Linking Pipeline	21
Figure 4.4. Full Wikification Pipeline	22
Figure 4.5. Organization name entity mentions that were detected by FLAIR but not with CoreNLP	27
Figure 4.6. Location-related entity mentions that were detected by FLAIR but not with CoreNLP	27
Figure 4.7. An example of incorrect annotations for mentions	28
Figure 4.8. Visualized entity embeddings space for 100 entities from 5 selected topics visualized using t-SNE	31
Figure 4.9. Wikification performance by using only the first N% of Entities for ranking	36
Figure 5.3. Entity Ranking Pipeline	41
Figure 6.3. Doc2Vec-based Background Linking Pipeline	48
Figure 6.4. Background Linking using BERT-based re-ranker	50
Figure A.1. WaPo article in the original Washington Post website	61
Figure A.3. Wikipedia page for "Alexander Mackenzie (politician)" entity ..	63

LIST OF ABBREVIATIONS

CG Candidate Generation.....	21, 24
CR Candidate Ranking.....	21
ER Entity Ranking.....	21
KB Knowledge Base.....	21, 22
MD Mention Detection.....	21, 22
nDCG Normalized Discounted Cumulative Gain.....	20, 40, 47
NER Named Entity Recognition.....	5, 22, 26
RM3 Relevance-Based Language Model.....	14, 15
RSV Robertson Term Selection Value.....	11, 13
SDM Sequential Dependency Model.....	14, 15

1. INTRODUCTION

Online news reporting services have changed the way individuals consume and exchange news dramatically over the last decade (Soboroff, Huang & Harman, 2018). According to a Pew Research poll from 2016¹, 38 percent of adult Americans get their news from a source on the internet.

News authors sometimes presume readers have some prior context or understanding of the topic or issue discussed in the news piece. However, this isn't the case for all the readers. A news article might be discussing an event or a certain topic and unless a certain piece of background information or context is provided, this can lead to a misinterpretation that can effect the reader's understanding of the bigger picture. Simply adding additional information to each article is not an ideal solution, as it may result in impractically long articles. As for the news writer's perspective, the writer might not know or remember all the background information when writing the article.

These reasons call for automating the process of news contextualization. Contextualization in news articles might come in two format. First, links to other news articles that give the essential background knowledge must be included in each article to give the reader the full picture. These articles might discuss specific topics, aspects or events that happened in the past and providing articles that discusses these therefore providing the necessary background information or context needed to fully comprehend the news article. Secondly, linking mentions of concepts, objects, and entities to internal or external sites that provides more detailed information that can assist the reader better understand the context of some events or aspects in the news article. These two formats are visualized in Figure 1.1 where a news article from The Washington Post is linked with other news articles that provides background information and mentions of entities are linked to their corresponding Wikipedia page where there are extended details about each entity.

Over the past 29 years, Text REtrieval Conference (TREC) has created a series

¹<https://www.pewresearch.org/journalism/2016/07/07/the-modern-news-consumer/>

of tracks aimed at encouraging research on information retrieval for large text collections. In 2018, TREC News track (Soboroff et al., 2018) was introduced with the motivation of encouraging research in the news domain. This track is focused on information retrieval or search tasks to help create tools that automates news contextualization and improves the news reading and writing experience. The track started with two task: Background Linking and Entity Ranking. Later, the Entity Ranking task was expanded into a full Wikification task.

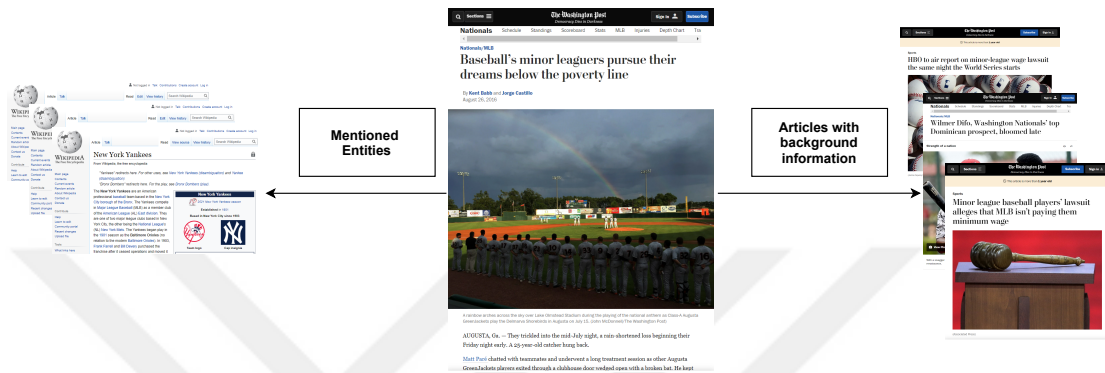


Figure 1.1 Contextualization in News Articles

1.1 Task Definition

The tasks offered in TREC News track focus on the retrieval of relevance information/document to enhance the readers experience. These tasks can be splitted into two main genres: relevant news article retrieval (Background Linking), and important entity retrieval such as Wikification and Entity Ranking. In this section, we discuss the formal task definition for each of the offered tasks in TREC News track.

1.1.1 Wikification

The terms wikification and entity linking are used interchangeably in the literature. The input for this task is a text document and a knowledge base containing a collection of pre-defined entities. The task is to recognize mentions of entities in the text and connect them to their corresponding entry in the knowledge base. In the

wikification case, Wikipedia is used as the knowledge base, and mentions of entities within the text are linked to the corresponding Wikipedia pages.

In the TREC News Wikification task, the task is not only about linking entities but also ranking them based on their relevance or importance to the news article. So given a document, a system should identify mentions, link them to their entities and then apply Entity Ranking to obtain a list of mentioned entities ordered in terms of importance.

A simplification of this task is shown in Figure 1.1 where systems detect and provide a list Wikipedia entities that were mentioned in an article. However, in Wikification task, systems should also provide a ranking for this mentioned entities.

1.1.2 Entity Ranking

Given a news article and a list of entities mentioned in that article, systems must return a rank for each of the provided entities which reflects the significance and relevance of each entity compared to the query article.

This task can be considered as the last step of Wikification where mentioned entities were already detected and linked such as in Figure 1.1 , and systems should only rank these entities in terms of relevance to the article.

1.1.3 Background Linking

In Background Linking, systems are provided with a news article as input and should retrieve a list of different news articles that give vital background and contextual information. The retrieved list of articles must be ordered in terms of importance and relevance to the query article. The retrieved articles should also be diverse and address different "topics".

An example of this task can be shown in the background linked article in Figure 1.1. It should be noted that these articles should also be ranked in terms of relevance to the query article.

1.2 Research Questions

In this work, we focus on developing search and retrieval tools for news articles domain. To solve our main tasks, we use vector representation and deep modelling based approaches as we aim to answer the following research questions:

RQ *How successful are vector representation and deep modelling based approaches compared to baseline information retrieval methods in the domain of news articles?*

This research question can be splitted into several sub-question related to our main tasks in this work:

RQ1 *How effective are document representations for embedding different topics or concepts in news articles into a singular vector representation?*

RQ2 *How effective are out-of-the-box pre-trained language models compared with classic Information Retrieval systems for the task of ranking news articles in terms of relevance?*

RQ3 *Is mutual vector modelling of news articles and entities effective for ranking entities in terms of relevance to an article?*

1.3 Thesis Outline

First chapter (this chapter) contains the introduction, task definitions, and a statement of research questions. Chapter 2 provides a review over the literature and previous work done on our main tasks. In chapter 3 we describe in detail the dataset and Knowledge base used in this work. Chapters 4, 5 and 6 are dedicated for the work on Wikification, Entity Ranking, and Background Linking respectively. In these separate chapters, we discuss task-specific data format, inputs, outputs, and evaluation metrics. We also describe our methodology and the experiments done in the scope of the task and then report and analyze the results obtained in our work. Finally, chapter 7 concludes the thesis by summarizing the findings of this work and discusses potential directions for future work.

2. RELATED WORK

In this chapter, we provide a literature overview of the main tasks discussed in this work: Wikification, Entity Ranking and Background Linking. We first go over previous work done on the different sub-tasks of Entity Linking. Then, we provide a summary on the work done in TREC News for the Wikification, Entity Ranking and Background Linking in that respective order.

2.1 Entity Linking

Generally, there are two types of Entity Linking (EL) systems. First, EL systems that contain 3 separate components: (1) a mention detection model, (2) a candidate generation or retrieval model and (3) a candidate ranking or Entity Disambiguation (ED) model. In this case, each model is trained separately and one model's output is sent to the next and so on. Second type of EL systems is end-to-end entity linking systems in which systems learn the sub-tasks of entity linking jointly. It's worth noting that the joint learning process usually only includes mention detection and entity disambiguation components.

2.1.1 Mention Detection

For such approaches, mention detection is mostly done by using Named Entity Recognition (NER) models. van Hulst, Hasibi, Dercksen, Balog & de Vries (2020); Wu, Petroni, Josifoski, Riedel & Zettlemoyer (2020) utilized state-of-the-art NER model Flair (Akbik, Bergmann, Blythe, Rasul, Schweter & Vollgraf, 2019) for detecting mentions. Another approach for generating candidate entity mentions is by

utilizing the pre-computed mention-to-entity mapping dictionary (Hoffart, Yosef, Bordino, Fürstenau, Pinkal, Spaniol, Taneva, Thater & Weikum, 2011) where any text span that retrieves a list of candidate entities is detected as a possible entity mention (Peters, Neumann, Logan, Schwartz, Joshi, Singh & Smith, 2019; Poerner, Waltinger & Schütze, 2020).

2.1.2 Candidate Generation/Retrieval

Knowledge bases (i.e., Wikipedia) are typically huge and contain millions of entities. Searching the entire entity universe can be very expensive and impractical. AIDA (Hoffart et al., 2011) an entity linking system, computes a dictionary based entity candidate set. Given a string, this dictionary returns a list of candidate entities and a list of prior-probabilities. This dictionary was calculated using features based on entity page titles, hyperlinks, nicknames, Wikipedia disambiguation pages and redirects. This dictionary set is used in most ED and end-to-end EL approaches as candidate generator. Also, as mentioned earlier, this dictionary is utilized for detecting candidate mentions.

BLINK entity linking system (Wu et al., 2020) utilizes more recent methods such as deep learning as they use BERT (Devlin, Chang, Lee & Toutanova, 2019) to encode and learn the optimal vector representations for mentions and Wikipedia pages then apply dense vector search FAISS (Johnson, Douze & Jégou, 2017) as a way for candidate retrieval.

2.1.3 Candidate Ranking/Entity Disambiguation

As for candidate ranking or ED, contextual information and textual features are usually used to help in finding the similarity and making the final prediction. After the recent success of BERT and newer transformer-based models in many different NLP tasks, there has been some work on utilizing BERT for entity representation and ED. In the approach proposed in Yin, Huang, Zhou, Li, Lan & Jia (2019), BERT’s Next Sentence Prediction task was used for ED. Sentence A contains the context around the mention and Sentence B was the entity Wikipedia page.

Yamada, Washio, Shindo & Matsumoto (2020) learn entities as special BERT tokens similar to their approach in LUKE (Yamada, Asai, Shindo, Takeda & Matsumoto, 2020). BERT is then trained to predict masked entities for entity disambiguation. The approach also utilizes a global iterative predicting mechanism which disambiguates entities with highest confidence first and use these in disambiguating more ambiguous mentions in the same document.

BLINK (Wu et al., 2020) encodes mention context and entity page information into a singular embedding by concatenating the two and obtaining BERT’s encoding and then calculates a ranking with a simple feed-forward neural network.

2.1.4 Entity Representation

As most ED methods require a way to represent entities. There has been different approaches to embed entity knowledge into numerical vectors. Wikipedia2Vec (Yamada, Asai, Sakuma, Shindo, Takeda, Takefuji & Matsumoto, 2020) learns word and entity vectors in a manner based on Word2Vec’s (Mikolov, Chen, Corrado & Dean, 2013) Skip-Gram model. This was done by jointly optimizing 3 Skip-Gram models trained on Wikipedia pages: (1) a model that predicts context around words (2) a model that predict context around entities, (3) a model that predict neighboring entities given an entity in a linked Wikipedia Graph. These representations have showed great performances when integrated in ED and other entity-related systems.

A more recent work LUKE (Yamada et al., 2020) utilized transformer-based model RoBERTa (Liu, Ott, Goyal, Du, Joshi, Chen, Levy, Lewis, Zettlemoyer & Stoyanov, 2019) to learn entity representation. In this approach, RoBERTa’s vocabulary is extended to include entities as special tokens. The model is trained on Wikipedia articles on a modified Masked Language Modelling where the model tries to predict masked words and entities.

Knowbert (Peters et al., 2019) worked on enhancing BERT’s knowledge for entities by injecting an entity linking component between BERT’s attention layers and pre-training BERT for entity-oriented Masked Language Modelling task. The entity linker takes BERT’s layer output, incorporates mention-span and entity information into the representations then passes it to the consequent BERT layer.

A more recent approach E-BERT (Poerner et al., 2020) propose a workaround for the expensive pre-training process needed by KnowBERT, by aligning pre-trained Wikipedia2Vec with BERT tokens. E-BERT learns a linear mapping be-

tween BERT’s and Wikipedia2Vec word vectors and applies that mapping to project Wikipedia2Vec’s entity vectors into BERT’s vector space. BERT is then fine-tuned on a special Masked Entity Modeling task where the model tries to predict the masked entity tokens.

2.1.5 End-to-End Models

Early work on end-to-end EL (Durrett & Klein, 2014; Nguyen, Theobald & Weikum, 2016) utilized hand-engineered features and set strong baseline results for EL.

A more recent approach by Kolitsas, Ganea & Hofmann (2018) was the first work to successfully use deep learning for E2E entity linking by outperforming previous SOA models and eliminating the need for hand-engineered features. In this work, the authors jointly optimize MD and ED by learning many components to represent characters, words, mentions, entities, local and global context utilizing LSTMs.

In the work of Broscheit (2019) the authors investigate how much entity knowledge is already learned in pre-trained BERT models. They approach E2E entity linking as a token classification problem where the number of token classes is the entity universe.

A more recent approach Chen, Zukov-Gregoric, Li & Wadhwa (2020) reported SOA results by jointly learning two task-specific layers on top of BERT (MD and ED). In their approach MD is handled as I-O-B token classification task, and ED layer tries to minimize the distance between golden Wikipedia2Vec entity vectors and mention-spans representations.

2.2 TREC News Wikification

As discussed earlier, the difference between TREC’s Wikification task and vanilla Entity Linking is the extra step in TREC News to rank the detected mention-entity pairs in terms of importance to the reader.

For the task of wikification, Ningtyas, El-Ebshihy, Piroi, Hanbury & Andersson (2020) first perform mention spotting by using a look-up dictionary that identifies

candidate text spans that represents an entity mention. This look-up dictionary is constructed with Aho-Corasick algorithm for tree-like string search by using the titles of Wikipedia articles. For candidate retrieval, Jelinek-Mercer smoothing is applied to the mention query and the top 10 candidate entities are retrieved. Then, a weighted-sum of three features (commonness, topic modeling and embedding similarity) is used for re-ranking the retrieved candidates. Commonness measures the probability this mention is used to link this entity while considering a mention can refer to other entities. Topic Modeling feature utilizes Latent Dirichlet Allocation for detecting topic distribution of the mention context and the first paragraph of the entity’s Wikipedia article. Hellinger Distance is then used to compare the two distributions. As for embeddings similarity feature, Doc2Vec is used to represent the query mention paragraph and entity Wikipedia article then the vectors are compared using cosine similarity. Then, probability of a mention to be linked to an entity in Wikipedia is used for the last step of ranking mention-entity pairs.

In the work of Ak, Köksal, Fayoumi & Yeniterzi (2020), mention detection is performed by utilizing Stanford CoreNLP (Manning, Surdeanu, Bauer, Finkel, Bethard & McClosky, 2014) tool for extracting named-entity mentions. To perform linking, Elastic-Search is used to retrieve entities by simply querying the detected mentions using the default scoring mechanism by just matching the Wikipedia page title. The retrieved (mention,entity) pairs are then ranked using their appearance in text. Meaning, first mention in the query article receives the highest rank, second mention is second highest and so on and so forth.

For the task of Entity Linking, many of the recent effective and state-of-the-art approaches utilize the knowledge of pre-trained language models and extend it to the domain of entity and KBs. Such approaches have proven very successful especially for Entity Disambiguation task when understanding context around mentions is crucial for entity prediction.

2.3 Entity Ranking

Signal (van der Sluis, Albakour & Martinez, 2018) participated in the first iterations of TREC News Track Entity Ranking task. The authors treat the task as an entity salience ranking task. Entity salience is a measure that determines the extent to which an entity stand out as more important than other entities in a text. For this

purpose, van der Sluis et al. (2018) use Salient Entity Linking algorithm introduced by Trani, Lucchese, Perego, Losada, Ceccarelli & Orlando (2018). SEL is a two step supervised algorithm for entity linking and salience ranking. As TREC News task only includes Entity Ranking, the authors adapt SEL architecture to use the given entities and rank them in terms of salience. To do so, they train a supervised regression using generated features. These features include position of mention span text, frequency and length. Other features include Wikipedia graph related features such as node (entity) degree, distance to other nodes, graph size, diameter...etc. Since this is a supervised approach, Dexter Entity Salience dataset ¹ is used to train the salience regression model.

TREMA-UNH and ICTNETAT (Ding, Lian, Zhou, Liu, Ding & Hou, 2019; Kashyapi, Chatterjee, Ramsdell & Dietz, 2018) follow simple approaches by using Okapi BM25 retrieval model with an index built on Wikipedia pages. Both teams experiment with different query formulations for retrieving ranked Wikipedia. These include using the Washington Post news article title, first paragraph or first 200 characters content and other combinations as queries. DMINR (Missaoui, MacFarlane, Makri & Gutierrez-Lopez, 2019) follow a similar path but uses an index built only on Washington Post documents. Then, they extract entity mention spans using spaCy tool ² and score these mentions against the document by using BM25.

CMU (Gonçalves, Magalhães & Callan, 2019) handle this task as a learning-to-rank (Liu, 2009) problem. They generate various features based on retrieval models such as BM25 and TF-IDF over a built index. They also use count and statistical features such as TF, number of unique surface forms, and count of entity pairs in a document. To train their model, the authors use previous year’s topics (TREC 2018 News Track Entity Ranking). In their work, Gonçalves et al. (2019) were aiming to validate the inverted pyramid writing scheme (Pöttker, 2003) in the context of entity ranking. This widely used scheme states that most significant and crucial content of a news article is written in the beginning of the article. This was reflected on their feature generation as they experiment with different models that utilize different parts of the article such as using the title, first 5 paragraphs or using the whole article.

Radboud (Kamphuis, Hasibi, de Vries & Crijns, 2019) experiment by applying different heuristics or simple statistics directly to decide on the final ranking. For instance, they use the order of appearance. Another method for ranking is using the number of times an entity is mentioned. Their best performing method is ranking entities using the number of tokens in the entity’s Wikipedia page where less tokens

¹<https://github.com/dexter/dexter-datasets/tree/master/entity-saliency>

²<https://spacy.io/>

means higher rank. Kamphuis et al. (2019) argue that having a shorter Wikipedia page are more likely to be relevant since they are more discriminative.

UQ (Le & Demartini, 2019) approach this task as a similarity problem. The authors use spaCy tool to identify entity mentions and extract sentences that contains a certain entity. Then they spaCy semantic similarity function, which is based on Word2Vec (Mikolov et al., 2013), to calculate similarity between entity sentences and the original news article or Wikipedia article.

DMINR (Missaoui et al., 2019) also follow a pipeline of entity linking followed by entity ranking. In this approach, the authors extract named-entity mentions from documents using spaCy tool. Then, they utilize Robertson Term Selection Value (RSV) (Chandra & Dwivedi, 2020) to map mentions to a list of similar entities in Wikipedia. By doing so, this list of similar entities act as a representation for mentioned entity. Finally, they use a probabilistic model to rank mentioned entities (representation) based on their relevance to the article.

To summarize, a number of different approaches were used for Entity Ranking. Many of the approaches utilize NLP tools to detect entity mentions. A couple of approaches use these detected mentions to simply query indices built on Wikipedia. Other approaches calculate representations for these mentions and estimate the relevance between mentions representation and Wikipedia entities to produce the final ranking.

2.4 Background Linking

A majority of the published work on background linking task utilize pre-built indexes for querying the Washington Post corpus and retrieving candidate background articles. Most of the approaches mentioned below include a pre-processing step or a candidate pruning step that utilizes the meta-data of the news articles. Articles that has the values “Opinion”, “Letters to the Editor”, or “The Post’s View” in the “kicker” field are not be linked as mentioned in TREC News’ guidelines. Also, retrieved article should be filtered using the date. A number of approaches also check retrieved candidates for duplicates by matching titles.

Anserini³ is an open-source toolkit based on Apache Lucene⁴ search engine library.

³<https://github.com/castorini/anserini>

⁴<https://lucene.apache.org/>

Anserini focuses on reducing the gap between academical Information Retrieval and real-world search solutions facilitate reproducing academical research results in Information retrieval. In their work for background linking (Yang, Lin & Cheriton, 2019), the authors build an index on Washington Post news corpus and experiment with different query expansions. One query formulation is using the top 1000 word in the query article using their TF-IDF score. Another formulation is using the article's first 5 paragraphs in 5 different queries and then selecting the top N final candidates out of all the queries results. The retrieved news articles are ranked using BM25.

SINAI (López-Úbeda, Díaz-Galiano, Valdivia & López, 2018) perform clustering to separate the document collection into different topical clusters. At query time, top documents are retrieved from the corresponding cluster and the documents are re-ordered using their relevance of the queried article . Each document is represented by TF-IDF vector of its title and abstract.

HTWSAAR (Bimantara, Blau, Engelhardt, Gerwert, Gottschalk, Lukosz, Piri, Shaft & Berberich, 2018) utilize Elastic-Search ⁵ for building an index over the news articles corpus and experiment with different querying methods. For example, using TF-IDF score to select top 20 keyword from query article, Another method is extract named entity mentions using Stanford CoreNLP (Manning et al., 2014) toolkit, and query using extracted mentions. Other experiment is to utilize TextRank Mihalcea & Tarau (2004) to extract key phrases from the query document and then use all key-phrases and entity mentions in the document to query the index. When queried, the detected phrases and mentions are assigned a boosting factor to give higher influence on the search. For instance, key-phrases and person-tagged mentions are given a higher boost factor than organization and location tagged mentions.

UdelFang (Lu & Fang, 2018) investigate the effect of indexing the news collection in two different ways. First method include the usual pre-processing and settings. In their other method, they apply entity linking using DBpedia Spotlight (Mendes, Jakob, Garcia-Silva & Bizer, 2011) entities and replace the mention keyword with the canonical form of the entity and then perform indexing while treating entities as a single keyword. In their more recent work (Lu & Fang, 2019) they continue their focus on utilizing entities as they try to estimate entity weights depending on its context. For this purpose, they generate two language models for the context around entity and the article document then use KL-Divergence as a way to estimate entity's importance and weight. Their work for background linking also includes using different paragraphs for querying the index and applying different methods for

⁵<https://github.com/elastic/elasticsearch>

ranking such as retrieval score order or number of times a document was retrieved out of different paragraph queries. In a continuation of their work on utilizing entities, their most recent work (Lu & Fang, 2020) revolves around identifying aspects or topics using an entity graph and leveraging individual topics for background linking. The authors create an entity graph for each document where each node is an entity mentioned in the document and the connection is the word distance between entities. The Louvain method (Blondel, Guillaume, Lambiotte & Lefebvre, 2008) for community graph segmentation is used to extract sub-graphs that represents topics. Then, individual topics keywords or entities are used to construct query and retrieved candidates for that specific topic. Lastly, to combine individual topics candidates, weights were given to topics by comparing the language models of the topics and the original document.

Similar to their approach for entity ranking, DMINR (Missaoui et al., 2019) use RSV (Chandra & Dwivedi, 2020) to extract the top K named entities that represents the queried news article. Then, using a Hill Climber iterative algorithm, they optimize to find the best named-entities. These best named entities are then used to query a Washington Post news articles index.

CLAC (Khloponin & Kosseim, 2019,2) hypothesize that background articles that should be linked are similar to the query article in document vector space compared to other articles. To test their hypothesis, the authors experiment with different methods for document representation like Doc2vec (Le & Mikolov, 2014) and Transformer-based pre-trained language models such as BERT (Devlin et al., 2019), XL-NET (Yang, Dai, Yang, Carbonell, Salakhutdinov & Le, 2020) , GPT-2 (Radford, Wu, Child, Luan, Amodei & Sutskever, 2019). Different distance or proximity measures where also used such as Cosine distance, Jaccard distance and Minkowsky L_p distance. To produce a ranking score, candidates were retrieved during using BM25 and then similarity functions were used to compare the vector representations and obtain a re-re-ranking for the candidates list.

Similarly, IRLabISI (Rahul Gautam, 2020) utilized Word2Vec to generate series of vectors representing each document. Then, for the final representation they sum up the vector of top 300 TF-IDF scoring words and then apply cosine similarity and search the entire documents space for retrieval and ranking. Their other experiments include extracting named-entity mentions using Stanford CoreNLP and utilize these and top TF-IDF scoring keywords to generate queries.

Another work that followed a vector similarity approach was OSC (Nathan Day, 2020). In their work, the authors utilize Sentence-BERT (Reimers & Gurevych, 2019) to obtain a vector representation for the first 3 paragraphs. Final document

vector is then calculated by summing up the 3 vectors. Then, cosine similarity is used to produce a re-ranked list of candidates retrieved by TF-IDF scored query.

ICTNETAT (Ding et al., 2019) follow the idea of relevance feedback to enhance queries. They use BM25 as their baseline model to retrieve ranked background articles. Rocchio relevance feedback algorithm is used with TF-IDF vector representations (Joachims, 1997) of the documents to expand the original query. Specifically, they label the top and bottom 20 BM25 retrieved documents as relevant and irrelevant documents respectively, and then query the system using the new optimized query. The authors also try learning-to-rank with BERT (Devlin et al., 2019). In that method, BERT is used to rank BM25 retrieved candidates by scoring query and candidate pairs as a next sentence prediction task. They create a fine-tuning dataset that consists of relevant and irrelevant documents by the same methodology used earlier for Rocchio query expansion.

IRQatar (Essam & Elsayed, 2019) use term co-occurrence graph to extract keywords and build search queries for background linking. They construct a graph for each article where each node represents a uni-gram and its connection represents the co-occurrences with other uni-grams in the document. These graphs are trimmed into segmented into smaller sub-graphs using graph decomposition methods (Rousseau & Vazirgiannis, 2015; Tixier, Malliaros & Vazirgiannis, 2016). The motivation behind this methodology is that useful keyword are usually at the core of sub-graphs and can reach many other nodes. These keyword are used for constructing a query and the sum of their neighbors weights is used as a weight to boost the terms in query.

In the work of Radboud (Boers, Kamphuis & de Vries, 2020), the authors follow a graph-based approach to enhance document representations for retrieval. In their work, they build a graph for each document where nodes represents terms and are weighted using their TF-IDF score and location in the documents. Terms are represented as vectors using Gerritse, Hasibi & de Vries (2020), and the cosine similarity between word vectors are used to weight connections in the graph. The authors also perform entity linking (van Hulst et al., 2020) to extract entities and populate the graph. To perform ranking, Greatest Maximum Common Sub-graph criterion (Bunke & Shearer, 1998) was calculated to measure relevance between articles and achieve the final ranking. Radboud (Kamphuis et al., 2019) also experiment with different ways of retrieval and re-ranking. Their retrieval methods include BM25, and Sequential Dependency Model (SDM). For re-ranking, they query Relevance-Based Language Model (RM3) (Lavrenko & Croft, 2001) with using different query formulations for instance: using the top 100 works in terms of TF-IDF score. Another re-ranking method is based on Entity Linking incorporated Retrieval Hasibi, Balog & Bratsberg (2016) where detected linked entities, using TAGME (Ferragina

& Scaiella, 2010), are incorporated into the queries to produce the final ranking.

SMITH (Foley, Montoly & Pena, 2019) handles this task as a learning-to-rank problem. The authors use a variety of features from different categories. These include retrieval model based features for the candidate article by querying the original article such as BM25, RM3, SDM, Query Likelihood and others. Reverse retrieval model features where the candidate article is used for querying and the query article is retrieved. Helper features such as article entropy, document length difference in publication time. Keywords and key-phrases are also extracted using different algorithms such TextRank, SingleRank (Wan & Xiao, 2008), and TopicRank (Bougouin, Boudin & Daille, 2013), then used as features. The authors also train other models to provide features such as click-bait probability, using a classifier trained on click-bait detection dataset. Another model is poetry category classifier where this model classifies texts into categories such as "Arts/Science", "Social Comment", "Activities" and other classes.

UNC (Qu & Wang, 2019) also take learning-to-rank approach for this task. In their work, they utilize similarity features to re-rank BM25 retrieved results. The similarity features are the cosine similarity of the query and candidate pairs using TF-IDF representation for different fields such as article title, content, named-entity mentions (spaCy) and category. The authors also experiment with ensembling BM25 and re-ranking scores to produce the final ranking.

In summary, a number of different approaches were used to solve the problem of Background Linking. A large number of these approaches can be classified into 3 different types. The first and most successful methods are classic information retrieval methods where different query expansions and boosting techniques are used to query search indices. Second, are vector similarity based approaches. Vector representation of articles learned or calculated and then proximity measures are used to estimate relevance and rank articles. Finally, Learn-To-Rank approaches where different types of features are used such as textual, similarity or graph-based features.

3. DATASET

In this chapter, we discuss and provide statistics and visualization for the datasets used in the different tasks addressed in this work. These comprise of the two main datasets: Washington News Post Corpus and Wikipedia.

3.1 Washington Post News Corpus

The work and experiments in TREC News track is in cooperation with The Washington Post (WaPo), one of the biggest news papers in The United States. The National Institute of Standards and Technology (NIST) provide a sizeable news corpus comprised of Washington Post news articles. This news corpus is to be used for TREC News track tasks such as Entity Ranking, Background Linking and Wikification.

In Table 3.1 we report the total number of news articles in Washington Post corpus over different TREC News iterations. On its first iteration (TREC 2018 News Track) the size of the corpus was 608,180 news articles. These articles contain duplicates such as when Washington Post republish an old article. A high percentage of these duplicates were removed in 2019 which results in a smaller corpus. After reducing the dataset in 2019 by removing duplicates, the corpus size grows significantly in 2020 and 2021 as the corpus got populated with much more recent news articles.

Track Year	Number of Articles	Duplicates Removed?
2018	608,180	Only Exact
2019	595,037	Exact&Near
2020	671,947	Exact&Near
2021	728,626	Exact&Near

Table 3.1 Washington Post Corpus Summary

Duplicates can be troublesome for uniformity in system evaluations (background linking) (Soboroff et al., 2018). System might rightfully retrieve duplicate articles that should not be in the articles pool in the first place. Exact duplicates where article content is matched were dropped in 2018’s corpus (Soboroff et al., 2018). However, the corpus still posses an amount of near-duplicates. The coordinators address this problem in the 2019 track iteration (Soboroff, Huang & Harman, 2019) by checking for similarity between articles using fast algorithms such as MinHashing (Broder, 2000) and Locality Sensitive Hashing (Datar, Immorlica, Indyk & Mirrokni, 2004) and eliminating documents above a certain Jaccard similarity threshold score. Hence, eliminating 13,143 duplicate articles to drop the corpus size to 595,037 news articles. Over the next TREC conference iterations (2020 and 2021), the corpus was expanded with more recent news articles while applying the same duplicate elimination method discussed earlier.

Dataset is provided as JSON-lines format, where each line is an article represented as a JSON object. Each article JSON object is comprised of 8 fields:

- id
- article_url
- title
- author
- published_date
- type: article type (article, blog post, etc.)
- source: article source (WaPo, Bloomberg News, etc.)
- content

The content of the news article is en-capsuled in the content field. This field contain text paragraphs, image captions and links, section headers. Textual data might include HTML tags. A simplified example JSON document is shown in Figure 3.1. The objects in the contents field are ordered as they appear in the news article (Figure A.1). Keeping the original ordering is important for Wikification task as relevance judgements (golden labels) are provided on the content block level. Original state of the JSON document is shown in Figure A.2.

```

"article_url": "https://www.washingtonpost.com/sports/nationals/the-minor-leagues-life-in-pro-baseballs-shadowy-corner/2016/08/26/96ab542e-6a07-11e6-ba32-5a4bf5aad4fa_story.html",
"author": "Kent Babb; Jorge Castillo",
"id": "96ab542e-6a07-11e6-ba32-5a4bf5aad4fa",
"published_date": 1472222644000,
"source": "The Washington Post",
"title": "Baseball's minor leaguers pursue their dreams below the poverty line",
"type": "article"
"contents": [
{"fullcaption": "A rainbow arches across the sky over Lake Olmstead Stadium during the playing of the national anthem as Class-- A Augusta GreenJackets play the Delmarva Shorebirds in Augusta on July 15. (John McDonnell/The Washington Post)"},
{"content": "AUGUSTA, Ga.-- They trickled into the mid-July night, a rain-shortened loss beginning their Friday night early. A 25-year-old catcher hung back."},
{"content": "Matt Pare chatted with teammates and underwent a long treatment session as other Augusta GreenJackets players exited through a clubhouse door wedged open with a broken bat. He kept himself awake with a marathon shower."}
]

```

Figure 3.1 WaPo article from Figure A.1 in reduced JSON format

3.2 Wikipedia

The famous Wikipedia is an open-source managed encyclopedia that contains millions of entries or articles. Wikipedia is used as a Knowledge Base for TREC News' entity related tasks such as Entity Ranking and Wikification. The data in Wikipedia is semi-structured. Textual data is stored in structures such as sections, lists, tables and information boxes. Articles are categorized with respect to their type, or topic. Relationship between different articles are defined through mentions, disambiguation pages, redirects. The Knowledge Base used in Entity Ranking tasks in TREC News Track 2018 (Soboroff et al., 2018) and 2019 (Soboroff et al., 2019) is TREC-CAR Wikipedia dump as of August 2017. After two iterations of Entity Ranking task, the track coordinators expand the task to full a task of Wikification. Wikification task in TREC News Track 2020 (Soboroff, Huang & Harman, 2020) and (Soboroff, 2021) uses the provided Wikipedia dump of January 2020 containing 7,893,275 entities. The dump is provided in Concise Binary Object Representation (CBOR) format. CBOR is a format for storing name-value paired data (like JSON) in binary format. To parse the corpus, we use the scripts provided by TREC CAR organizers¹. A sample entry is provided in Figure A.4. This entry was parsed from CBOR format into plain un-structured text. It should be noted that the parsed data still holds the structure information such as titles, headers, child sections, paragraphs, lists, etc. .

¹<https://github.com/TREMA-UNH/trec-car-tools>

4. Wikification

In this chapter, we discuss the dataset, as in inputs and outputs, used in the Wikification task. We also explain the methodology and algorithms used in tackling the different sub-tasks of Wikification task. We also discuss our experiments, results, and findings in the course of the work on this task.

4.1 Dataset

The task pipeline is composed of detecting mentions of entities and linking these mentions/anchors to their correspondent page in Wikipedia. Finally, the list of detected mention-entity pairs should be ranked in a manner similar to Entity Ranking. Wikification task appeared in TREC 2020 and 2021. Similar to Entity Ranking, the participants are provided with testing topics and are asked to submit a list of ranked linked-mentions for each topic. However, in Wikification mentions and entities are not provided unlike Entity Ranking. XML format of the test topics is shown in Figure 4.1.

```
<top>
  <num> Number: 886 </num>
  <docid>AEQZNZSVT5BGPPUTTJ07SNMOLE</docid>
  <url>https://www.washingtonpost.com/politics/2019/06/05/trump-says-transgender-
    troops-cant-serve-because-troops-cant-take-any-drugs-hes-wrong-many-ways/</
    url>
</top>
```

Figure 4.1 Wikification Test Topic

For submission, the systems should provide the text span location, length and the linked entity link as output. Exact submission format is shown below:

C1	C2	C3	C4	score
False	False	False	False	0
True	True	False	False	1
True	True	True	False	5
True	True	True	True	10

Table 4.1 Wikification Scoring Methodology

topicNum	score	cNum	start	len	link
936	17.86	4	94	6	enwiki:Ephron
936	18.56	4	201	8	enwiki:Silkwood
936	10.53	6	14	8	enwiki:Perelman

Figure 4.2 Wikification Output Format

Here "cNum" stands for the content block number. As shown in the JSON format of the articles in Figure A.2, articles are made up of content blocks where each block can be a paragraph, heading, image, caption, etc. "start" is the character index since the beginning of the content. Each "len" is the number of characters in the mention span. It's worth noting that for each detected mention span, systems should only submit the best predicted link rather than a list of ranked links.

To evaluate systems, assessors score each linked mention on 4 criteria:

- C1** Sensible mention span: this criteria judges whether it makes sense to have an entity link in this text span.
- C2** Useful mention link: this criteria judges whether it would help the reader to link this mention.
- C3** Correct link: this criteria judges whether the provided link matches the actual correct link.
- C4** Useful link content: this criteria judges whether the content in the linked entity is useful in providing context or background information to the reader.

These multiple binary criteria represent are converted to a singular numerical score as shown in Table 4.1. This criteria reflects the correctness and relevance of a mention. If all criteria are false, this means the relevance score of the mention is 0. If both the mention span and entity link are correct and relevant (all 4 criteria are satisfied) the relevance score of the mention is 10.

Similar to TREC News 2019's Entity Ranking task, the main metric to evaluate systems performances is nDCG but with a cutoff value of 10 (nDCG@10). 50 and 51 test topics were provided for 2020 and 2021 TREC News track iterations respectively. TREC also provides a Wikipedia dump as of 2020 as the KB for both 2020 and 2021

track iterations.

4.2 Methodology

The pipeline of our approach in the Wikification task contains 4 sub-tasks. First sub-task is Mention Detection (MD) where we find text spans that mention entity from the KB. We utilize the effective FLAIR NER tool (Akbiik et al., 2019) for the MD sub-task. Second sub-task is Candidate Generation (CG) or retrieval where we generate a list candidate entity to send for ranking. CG system utilizes search engines such as Elastic-Search for term-matching or FAISS for dense vector search. Third sub-task in our pipeline is Candidate Ranking (CR) where we predict the best candidate entity. In this sub-task we perform context encoding and apply neural ranking select the best candidate. In Figure 4.3 we demonstrate the general pipeline used in our first three sub-tasks. The input in the pipeline is a Washing Post (WaPo) document, and the output is a non-ranked list of entities detected in that document. Lastly, the final sub-task is the previously featured Entity Ranking (ER) task where the list of detected entities for a document is ranked. For this sub-task, we use a simple but effective method for ranking based on the order of appearance in text. Figure 4.4 summarizes the entire Wikification pipeline. Entity Linking model basically comprises of our first 3 models: Mention Detection, Candidate Generation and Candidate Ranking models.

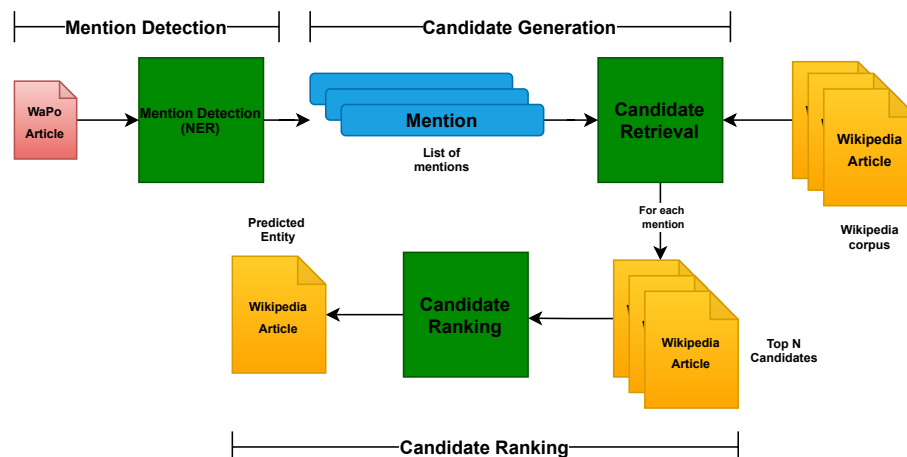


Figure 4.3 Entity Linking Pipeline

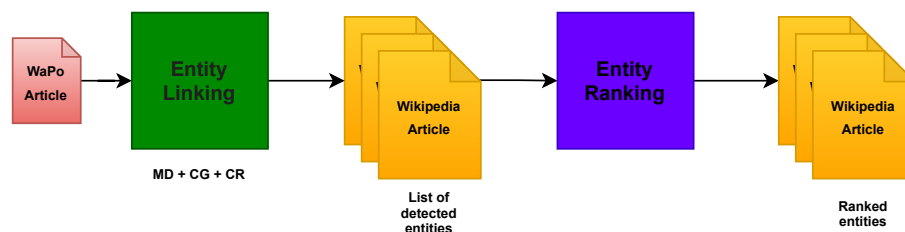


Figure 4.4 Full Wikification Pipeline

4.2.1 Mention Detection

Our first sub-task in the Wikification pipeline is MD. In this sub-task, our goal is to identify text spans that refer to named entities so we can later link these entries to our KB. We treat this task as a NER task. Namely, we utilize NLP tool FLAIR. At the time of its publication (Akbik et al., 2019), FLAIR has shown state-of-the-art results on NER benchmark datasets such as the famous CoNLL-2003 NER dataset (Tjong Kim Sang & De Meulder, 2003). FLAIR learns contextual character embeddings by performing neural character-level language modeling. These embeddings are used as sentence representation and are passed to BiLSTM-CRF (Huang, Xu & Yu, 2015) to perform NER as a sequence labeling task.

Furthermore, we experiment with another successful NLP tool, Stanford Core-NLP¹. Stanford Core-NLP is an NLP analysis tool based on statistical modeling. This tool provides many annotation services such as tokenization, Part-Of-Speech tagging, lemmatization, co-reference resolution, and NER.

We also experiment with the recent state-of-the-art transformer-based BERT (Devlin et al., 2019) with a token classification head fine-tuned on CoNLL 2003 dataset for NER².

4.2.2 Candidate Generation

Detected entity mentions from the previous step are used as queries to retrieve lists of candidate entities. KBs are naturally huge in size and contain millions of entries. Simply searching the entire entity universe for the golden entity is costly and impractical. This calls for fast methods to produce a smaller sub-set of entities

¹<https://stanfordnlp.github.io/CoreNLP/>

²<https://huggingface.co/dbmdz/bert-large-cased-finetuned-conll03-english>

that possibly contains the golden entity with high recall. In our work, we explore two different methods for generating candidates.

Our first method is a search index built on Wikipedia dump using Elastic-Search search engine identical to the one described in Ak et al. (2020). To construct the index, we parse the provided TREC CAR Wikipedia dump and use the following fields to construct the index:

- **Title:** Entity article page title
- **Content:** The textual content of the Wikipedia article concatenated into one text field
- **Redirect Names:** The list of surface forms or names that are used to re-direct to the entity page
- **Categories:** The list of wiki categories this article belongs to.
- **Headings:** The list of headings or section titles in the article.

Default Elastic-Search settings were used while building the index. To perform candidate retrieval, we experiment with different query formulation to search the index utilizing the detected mention text spans and the context around it. Query formulation include exact term matching, or fuzzy search with the title, content and disambiguation names fields. For scoring the candidates, we use Elastic-Search default scoring function BM25.

Our second candidate generation system utilizes vector similarity search to retrieve candidates. This approach is based on BLINK’s Bi-Encoder model (Wu et al., 2020). In this approach, mentions and entities are encoded using BERT to obtain a vector representation for each. Vector representation of mentions is obtained by encoding the mention as it appears in text along with its left and right context (in WaPo). The method below demonstrates how a sequence of tokens is created by using special tokens to signify the mention boundaries:

$[CLS]$ *left context* $[M_{start}]$ *mention text* $[M_{end}]$ *right context* $[SEP]$

$[M_{start}]$ and $[M_{end}]$ special token are used to signify the beginning and end mention and separate it from the preceding and following context. Last layer’s output representation for the first token in the sequence $[CLS]$ is used to represent the mention. The constructed token sequence is padded or cropped to match the maximum sequence length parameters of 256. Left and right context is limited to 128 tokens each.

Similarly, to obtain entity vectors, the corresponding Wikipedia article title and text is encoded in BERT by constructing the following sequence:

$$[CLS] \textit{title} [ENTITY] \textit{description} [SEP]$$

[*ENTITY*] special token is used to separate the title from the article content. Similarly, [*CLS*] token representation is extracted as used as a vector representation for entity. The constructed sequence is also limited to 256 tokens.

The vector pairs are passed into a simple single-layer neural network that scores the entity-mention vector pairs using their dot-product as a way to measure similarity.

A vast amount of mentions in Wikipedia articles are typically linked to their corresponding entity pages. This produces a large entity linking dataset that can be used to train entity linking components. BLINK (Wu et al., 2020) create a training set of 9 million examples out of Wikipedia’s set of linked mentions. This dataset is used to learn model parameters such as the representations of mentions and entities and optimize the weights of the scoring layer.

After training, the entirety of the entity universe is encoded, cached and used to build FAISS index (Johnson et al., 2017). FAISS is a library for efficient dense vector similarity search. Finally, to produce a list of candidates at inference time, the mention is encoded and used to query the vector search index (FAISS) and retrieve N closest candidates using Euclidean (L2) distance. These candidates can also be ranked using the scoring layer and make the final entity prediction based on the highest scoring entity.

The output of our CG methods , Elastic-Search and Bi-Encoder, is a list of 100 candidate entities for each detected mention. These candidates are then sent for the re-ranking model where the best candidate is selected as prediction.

4.2.3 Candidate Ranking

In this step, we use the list of candidates to make the final prediction for each detected mention. This step is optional, as our two retrieval methods already provide scores that can be used for ranking. However, the reduced entity space after candidate retrieval (100 compared to over 7 millions), now allows us to apply much complex and expensive methods and further incorporate the context around the mention to make a better entity link prediction.

For this purpose, we use the Cross-Encoder model described utilized in BLINK (Wu et al., 2020). Cross-Encoder follows the same encoding methodology as the Bi-Encoder. However, instead of encoding the mentions and entities as separate strings, the pair is used to construct a singular sequence as shown below:

$$[CLS] \textit{left context} [M_{start}] \textit{mention text} [M_{end}] \textit{right context}$$

$$[SEP] \textit{title} [ENTITY] \textit{description} [SEP]$$

The representation in a singular sequence allows the encoder to naturally apply attention mechanism between the mention’s and entity’s tokens. Similar to the case of Bi-Encoder, last layer’s output for the $[CLS]$ token is extracted. This is then passed to a linear layer that produces a ranking score. After producing a score for each mention-candidate pair from the candidate list, the candidate with the highest score is selected as predicted entity. Wikipedia’s linked mentions dataset is again used to optimize model’s parameters such as the encoder’s and ranking layer’s weights.

4.2.4 Entity Ranking

After making the final prediction for each mention-entity pair. We obtain a list of detected entities in each document. The final output for the Wikification task is to rank these entities in terms of importance and significance to the reader. To do this, we experiment with two methods of producing the final ranking. Our first method is best performing method 2020 Wikification task submissions (Ak et al., 2020). Where entities are simply ranked using the natural order entities appear in text. Our second method is to utilize our deep learning models scores from the previous step. These scores reflect the confidence made in the entity prediction. Hence, using these score tests whether similarity between document and entity text can be effective for ranking.

4.3 Experiments

System	2020			2021		
	Recall	Precision	F1	Recall	Precision	F1
Stanford CoreNLP	0.642	0.105	0.181	0.716	0.114	0.1973
FLAIR	0.782	0.131	0.225	1.0	0.158	0.274
BERT-Large-Ner	0.770	0.098	0.173	0.839	0.098	0.176

Table 4.2 Mention Detection system scores on testing topics

In this section, we discuss our experiments for the different sub-tasks of Wikification and report the results obtained by our different approaches.

4.3.1 Mention Detection

In our first sub-task in the Wikification pipeline, we use out-of-the-box NER models without performing any fine-tuning. This is due to our small labeled dataset size. To evaluate our standalone Mention Detection models, we extract the golden mentions from 2020 and 2021 QRELS and create a small testing dataset consisting of 422 and 481 mentions respectively. We use our three NER models to detect mentions in the testing topics. We report the models performances in terms of Recall, Precision and F-1 Score in Table 4.2. FLAIR model scores highest in terms of Recall on both testings sets where it shows a perfect recall score for 2021 set. BERT-based model scores 1% and 16% lower than FLAIR on 2020 and 2021 sets respectively. CoreNLP’s scores 14% and 28.4% worse than FLAIR.

As for Precision score, all of our NER models over-produce mentions with a high-rate of false positives which yields low Precision results. For the nature of our task, we believe the most important metric is Recall. We believe false positive examples can be filtered out during our Entity Ranking task at the end. However, undetected mentions or false negatives can have a higher direct impact on the final Wikification performance.

Error Analysis: Mention Detection

In our experiments we explore different components and methodologies for our different sub-tasks in Wikification. To better understand the differences in performance between different methodologies we analyze error cases where one system fails but

Over the course of the day, the 10 experts played the roles of U.S. officials in simulated National Security Council-convened meetings. They acted the parts of a national security adviser, the secretaries of health and human services, state, homeland security and defense, attorney general and the directors of the **Central Intelligence Agency** and **Centers for Disease Control and Prevention**. There were also two members of Congress. Inglesby played the national security adviser.

(a) Passage 1

On seeing that both the **National Symphony Orchestra** and the **Baltimore Symphony Orchestra** were presenting screenings with live orchestral accompaniment of “A New Hope” (the film we knew as the original “Star Wars”) in September, I did an informal online search and found “Star Wars” being played by at least 13 symphony orchestras this season alone, from Sydney to St. Louis. (The NSO is continuing the theme with “Return of the Jedi” next week and “The Force Awakens” Feb. 21 to 23.)

(b) Passage 2

Figure 4.5 Organization name entity mentions that were detected by FLAIR but not with CoreNLP

QUEEN MAUD GULF **NUNAVUT** CANADA — In 1846, in perilous seas to the north of here, the two steam-propelled ships of British explorer Sir James Franklin, the Erebus and the Terror, froze in the ice. It was the beginning of the end of the Franklin mission, in which more than 100 men perished in the cold despite the launch of scores of ships to try to rescue them — the greatest disaster in a long and troubled history of trying to uncover the **Northwest Passage**.

(a) Passage 3

Maryland Gov. Larry Hogan recently won a key vote for his traffic relief plan for the Washington suburbs by proposing to start with new toll lanes on **Interstate 270** in Montgomery County. Expanding I-270, he said, would be less controversial than his initial plan to first widen the **Capital Beltway** and its most crippling chokepoint at the American Legion Bridge.

(b) Passage 4

Figure 4.6 Location-related entity mentions that were detected by FLAIR but not with CoreNLP

the other alternative system doesn’t.

We analyze the performance differences between different Mention Detection models. Specifically we examine TREC 2020 golden mentions that were successfully detected by FLAIR but not with CoreNLP (63 mentions). In summary, we notice a big percentage CoreNLP undetected mentions belongs to organization names with long names (3 or more tokens). We report an example of these names in Passage 1 and 2 in Figure 4.5 . We also observe CoreNLP’s low performance on location-related entity mentions. This is more demonstrated in Figure 4.6 where we report 3 undetected entity mentions that refer to street names.

Error Analysis: Incorrect Annotations

While performing manual error analysis on our Mention Detection systems, we discovered several faulty annotated mention spans. TREC provides the golden entity mentions and link in their QRELS files. We don’t have the exact number of faulty samples. However, a number of miss-aligned annotations seems to happen in content blocks that contains HTML code. The mention start index provided in the QRELS seems to have incorrect values. An example of an incorrectly annotated mention is

“With social media and 24-hour cable and an environment in which experts and the value of science is increasingly questioned, we just can’t assume in a crisis that we can get up and talk to the American people,” said Margaret Hamburg, a former commissioner of the **Food and Drug Administration**, who played the health and human services secretary. Otherwise, she said, “it’s an environment where one thing that goes out in the media can suddenly mushroom, and before you know it, everything you’re doing in the most scientific way can be derailed.”

Provided annotation (QRELS): mention_start: 470, mention_length: 28, link: enwiki:Food%20and%20Drug%20Administration

Extracted mention using annotation: "nd Drug Administration, who "

Correct mention: "Food and Drug Administration"

Figure 4.7 An example of incorrect annotations for mentions

shown in Figure 4.7.

It is worth noting that for the rest of this work and experiments, we use the original dataset as it is with any manual corrections or modifications.

4.3.2 Candidate Generation

In our work, we explore two main methodologies for Candidate Generation: text-based and vector-based retrieval.

Elastic Search Query Formulations

Using the mentions surface form, we build different search queries by using different matching methods: Exact-Match and Fuzzy-Match. Fuzzy-match or approximate match is basically partial matching the search query.

We also explore different fields for matching: title, disambiguation names, and text content. Here is the list of different query formulations used for our Elastic-Search experiments:

- **EM-T:** Exact matching using the title field only.

Query Formulation	2020 Topics	2021 Topics
EM-T	0.905	0.906
EM-DN	0.689	0.792
EM-C	0.596	0.770
EM-ALL	0.933	0.979
EM-T+FZ-T	0.935	0.915
EM-ALL+FZ-T	0.950	0.977
EM-ALL+FZ-ALL	0.952	0.975

Table 4.3 Recall@100 score for our different Elastic-Search query formulations

- **EM-DN:** Exact matching using the disambiguation names field only.
- **EM-C:** Exact matching using the text content field only.
- **EM-ALL:** Exact matching using the title, disambiguation names, or the text content field (in that order) using the OR clause.
- **EM-T+FZ-T:** Exact or fuzzy matching using the title field only.
- **EM-ALL+FZ-T:** Same as EM-ALL but include another fuzzy matching clause with title field.
- **EM-ALL+FZ-ALL:** Same as EM-ALL but include 3 fuzzy matching clauses with the title, names, and text content fields.

To test our different query formulations, we extract the set of golden mentions from 2020’s QRELS file (422 mentions). We use the golden mentions spans as the query content for our queries.

We report the Recall@100 score for all query formulations in Table 4.3. We observe the highest gain in Recall for a single field goes to exact matching the title field (EM-T). Naturally, the title field is the most significant field for retrieval. We observe a good increase in performance when adding fuzzy matching to the title query (EM-T+FZ-T). Adding the disambiguation names and the text content fields provides a boost. The best performing query formulation is EM-ALL+FZ-ALL where we use exact and fuzzy matching for all 3 fields and we use this query for retrieval before Candidate Ranking.

Bi-Encoder Wikipedia Version

BLINK (Bi-Encoder and Cross-Encoder) uses English Wikipedia dump of August 2019 as the main knowledge base in their work (Wu et al., 2020). This Wikipedia

dump is used for learning entity's vectors and training the Bi-Encoder and Cross-Encoder. BLINK's authors provide their pre-trained models and entity universe vectors in their GitHub repository³. The authors report training setup and details in their paper. Using 8 Nvidia Volta v100 GPUs for training their models, training the Bi-Encoder took 70 hours while training the Cross-Encoder takes around 37.5 hours.

As reported earlier in Dataset Chapter, TREC News' 2020 and 2021 Wikification task uses TREC CAR's Wikipedia dump of January 2020. To validate our use of BLINK's entity universe, we use 2020 golden Wikification qrels to investigate if golden entities are found in BLINK's Wikipedia dump. Out of 422 golden linked entities, only 4 entities were not found in BLINK's Wikipedia dump.

The output of Bi-Encoder and Cross-Encoder is Wikipedia entity titles from BLINK's dump. To ensure the predicted the entities are mapped correctly to the equivalent TREC CAR's Wikipedia entry, we apply a simple alignment technique when making predictions using BLINK's models (Bi-Encoder or Cross-Encoder) . Our technique comprises of two steps to generate Wikification output in TREC's format of "enwiki:entry_id" from BLINK's outputs:

- Step 1** Use Elastic-Search to find TREC CAR's entity that matches the title returned by BLINK (exact match).
- Step 2** If step 1 fails, fetch BLINK's prediction entity's text content and use Elastic-Search to search the contents of TREC CAR's Wikipedia articles and match with highly similar articles.

Step 2 in our approach allows us to match entities for which titles where changed but the content is still very similar. For example: "Eleanor Butler Alexander-Roosevelt" entity page was re-named to "Eleanor Butler Roosevelt" in the newer Wikipedia.

Entity Embeddings Visualization

Neural entity linking systems depend greatly on effective entity representations. These representations must encode semantic relatedness between entities in various aspects. To validate this aspect of BLINK's entity representation, we perform a simple visualization experiment. We select 5 different topics: "basketball", "politician", "animal wildlife", "programming language" and "deep learning". We retrieve

³<https://github.com/facebookresearch/BLINK>

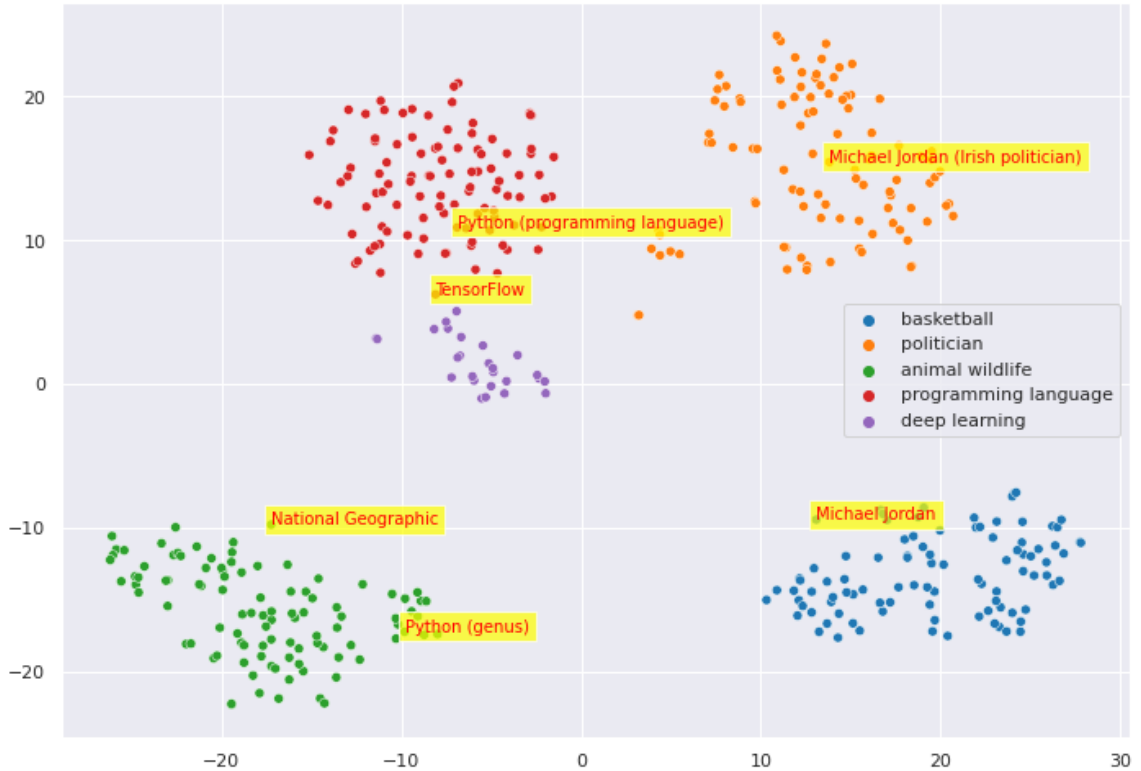


Figure 4.8 Visualized entity embeddings space for 100 entities from 5 selected topics visualized using t-SNE

100 entities related to these topics using Elastic-Search. We then extract the 768-dimensional vector representations of these entities and use t-SNE (van der Maaten & Hinton, 2008) for visualizing these entities in 2-D vectors space. The visualized entity embeddings are shown in Figure 4.8 . To test this, we use some entities as test cases and observe the relatedness and distance shown in the location of these entities. "Michael Jordan" the basketball player is located in the "basketball" cluster of vectors and away from "Michael Jordan (Irish politician)" and the "politician" cluster. The same can be observed with "Python (programming language)" and "Python (genus)". We also observe "TensorFlow" entity as being close to both "deep learning" and "programming language" clusters.

Candidate Generation Results

Furthermore, We analyze our two different Candidate Generation/Retrieval subsystems. We use the set of golden mentions from 2020's test topics. We perform candidate retrieval using our Bi-Encoder and Elastic-Search with the best query formulation (EM-ALL+FZ-ALL). We report the recall scores at different cutoffs:

	Model	R@1	R@5	R@10	R@25	R@50	R@100	R@1000
2020	ES-BEST	0.594	0.841	0.886	0.931	0.940	0.952	0.974
	Bi-Encoder	0.879	0.929	0.941	0.955	0.955	0.957	0.969
2021	ES-BEST	0.736	0.868	0.908	0.944	0.955	0.975	0.975
	Bi-Encoder	0.948	0.966	0.981	0.981	0.993	0.993	0.993

Table 4.4 Recall scores at different cutoffs for our retrieval models on 2020 and 2021 Test Topics

1,5,10,25,50,100 and 1000 in Table 4.4. For Elastic-Search, we observe high increase in recall when going from 1 candidate to 5 candidates. Elastic-Search provides the highest system recall score at 1000 candidates in 2020 topics while Bi-Encoder scores higher on 2021 topics. However, Bi-Encoder performances at all cutoff points except 1000 proves to be more effective than Elastic-Search. Since Bi-Encoder includes the context while encoding the mention, it can provide a good filtered list of candidates at a much lower cutoff. This can be seen in Bi-Encoder’s Recall@1 score which is significantly higher than Elastic-Search’s Recall@1 score and is even comparable to Elastic-Search’s Recall@10 and Recall@50 scores.

4.3.3 Candidate Ranking

After receiving a list of candidate entities for each mention, we apply Candidate Ranking to decide on the final prediction by linking an entity for each mention. We utilized two methods for ranking: retrieval models scores or BLINK’s Cross-Encoder. To evaluate the performances between the two methods, we perform candidate retrieval using Bi-Encoder and ES. Then, we use Cross-Encoder to apply re-ranking for the candidates list. We then measure the ranking performance for raw retrieval models scores and Cross-Encoder re-ranked scores. It should be noted that here we also use alignment methodology mentioned earlier to align BLINK’s entities into TREC Wikipedia names. We also measure the time efficiency of both methods. Re-ranking using the transformer-based Cross-Encoder is resource and time extensive so we would like to highlight the trade-off between our methodologies.

We report the nDCG@5 scores and total time taken (average) combined for retrieval and re-ranking in Table 4.5.

First, we note the low performance of Elastic-Search using the best query method (EM-ALL+FZ-ALL) without any re-ranking. This is shown as Elastic-Search (Non-Boosted) in Table 4.5. This low score is expected as we have seen earlier in Recall@1

Candidate Retrieval	Re-ranking	2020	2021	Avg Mins
Elastic-Search (Non-Boosted)	None	0.1716	0.3812	6
Elastic-Search (Boosted)	None	0.3483	0.6483	6
Bi-Encoder	None	0.3326	0.6422	11
Elastic-Search	Cross-Encoder	0.4306	0.7482	70
Bi-Encoder	Cross-Encoder	0.4223	0.7542	74

Table 4.5 Candidate Retrieval and Ranking performances reported in nDCG@5 and average minutes taken for inference

score for the best candidate retrieval query (EM-ALL+FZ-ALL). This query formulation retrieves a better candidate list. However, making the prediction as the candidate with the highest raw retrieval score does not perform well since we introduce fuzzy-search into our queries. Fuzzy-search can be helpful in retrieval when search terms only partially match. But, in doing so it also introduces some noise (irrelevant candidates). To counter this, we apply boosting to the non-fuzzy fields. Therefore, giving these fields such as title field more weight when scoring search results. Title field is given a weight of 10, disambiguation names is given a weight of 5 and other fields are weighted as 1. We observe noticeable improvements when apply boosting in the Candidate Ranking stage. Elastic-Search with boosted queries performs on-par with Bi-Encoder. Re-ranking using the Cross-Encoder significantly improves the final wikification performance.

As for computing performance, Elastic-Search is the most time and resource efficient. Both Bi-Encoder and Cross-Encoder are deep complex algorithms and require both memory and computation power for longer periods. Running the Cross-Encoder takes one hour (on average) longer on 2020 or 2021 testing topics.

4.3.4 Entity Ranking

After linking each detected mention to an entity, we would like to rank the list of linked mentions/entities in each document in terms of relevance. To perform ranking, we use two different methods: 1) Candidate Ranking model scores 2) Cosine similarity on Doc2Vec representations of articles and entities. Also, we reduce the list of retrieved linked mentions in a document to 100 to comply with the constraints of TREC evaluations. For this purpose we use the order of appearance in text to eliminate entities appearing later in text and reduce the list to 100 mention-entities pairs per-document.

Ranking: Model Scores

After obtaining the list of detected mention-entity pairs in each document. We get the model score of that was used in predicting that entity. This score can be obtained using Bi-Encoder, Elastic-Search or Cross-Encoder depending on which model was used to make the final entity prediction for each mention. This score represents the model's confidence in its prediction. In this approach, we use this score to rank mention-entity pairs in terms of relevance to the document.

Note that we use only the top scoring candidate score for each mention. We are not using different candidates for the same mention. Hence, the list of mention-entity pairs we use for Entity Ranking in each document looks like the following:

- "mention1 text", entityA, score
- "mention2 text", entityB, score
- "mention3 text", entityC, score
- "mention4 text", entityA, score

In Table 4.6 we report the NDCG@5 score of the final wikification utilizing only the top-K scoring entities in each document for our 3 different Candidate Ranking models.

Generally, using the highest scoring entity (top-1) in a document, achieves more than 50 percent of the performance that we get for using top-5 entities. This highlights the low-precision trait of our Mention Detection model. For 50 testing topics (2020 dataset), our MD system detects over 2500 mentions where only 421 of these are mentions link-able to Wikipedia (2020 topics). Using only 50 (top-1 in each document) of these 2500 accounts for over 50% of the full performance. This reflects on the idea of prediction confidence (model scores) and why it is useful in our case to eliminate the false-positive mentions produced by our MD model.

Ranking: Doc2Vec

In our TREC News 2019 participation, we utilized Doc2Vec and Cosine similarity for the task Entity Ranking. We re-visit this approach for the purpose of Entity Ranking sub-task as part of the Wikification task. We handle this task as a simple vector similarity task between the test article and the detected entities.

In Table 4.7 we report the final Wikification score of entities ranked using Cosine

Candidate Ranking	K	2020 Topics	2021 Topics
Cross-Encoder	5	0.4262	0.7461
Cross-Encoder	3	0.3774	0.6879
Cross-Encoder	2	0.3299	0.5402
Cross-Encoder	1	0.2625	0.4518
Bi-Encoder	5	0.3301	0.6392
Bi-Encoder	3	0.2980	0.5679
Bi-Encoder	2	0.2436	0.5003
Bi-Encoder	1	0.2040	0.3838
Elastic-Search	5	0.3271	0.6489
Elastic-Search	3	0.2950	0.5785
Elastic-Search	2	0.2407	0.5169
Elastic-Search	1	0.1906	0.3872

Table 4.6 Wikification’s nDCG@5 score for using the top K scoring entities for Entity Ranking

similarity between the article and entity Doc2Vec vectors. As seen in the results, Doc2Vec does not provide any improvements over Cross-Encoder. The methodology used in the Cross-Encoder already uses the principle of vector similarity but using a much more sophisticated representation for both mentions and entities, and uses an optimized ranking model to produce the similarity scoring. Also, it should be noted that in our previous Entity Ranking experiments, we use a manually validated golden list of entities. However, in this case, we are using a much larger list of detected entities that includes both true-positives and false-positives.

Entity Ranking method	2020 Test Topics	2021 Test Topics
Doc2Vec-Article	0.2294	0.3786
Doc2Vec-Paragraph	0.218	0.3560
Cross-Encoder Top-1	0.2625	0.4518

Table 4.7 nDCG@5 Wikification score for using Doc2Vec vector similarity in Entity Ranking. The list of document entities are linked by using our Cross-Encoder model.

Reducing the number of retrieved items

We reduce the number of retrieved mention-entity pairs in each document to 100 per document to comply with the TREC constraints.

In order to understand entity’s relevance to the article, we analyze the distribution of detected entities with respect to their location in the article. In Figure 4.9 where we report the Cross-Encoder wikification performance on 2020 topics while using

only the first $N\%$ entities in each document. We observe that using the first $\tilde{45}\%$ occurring entity mentions counts for most of the total Wikification performance 0.411 compared to 0.430 when retrieving all entities. We also observe that in some cases, retrieving more than first 45% mention can reduce the performance.

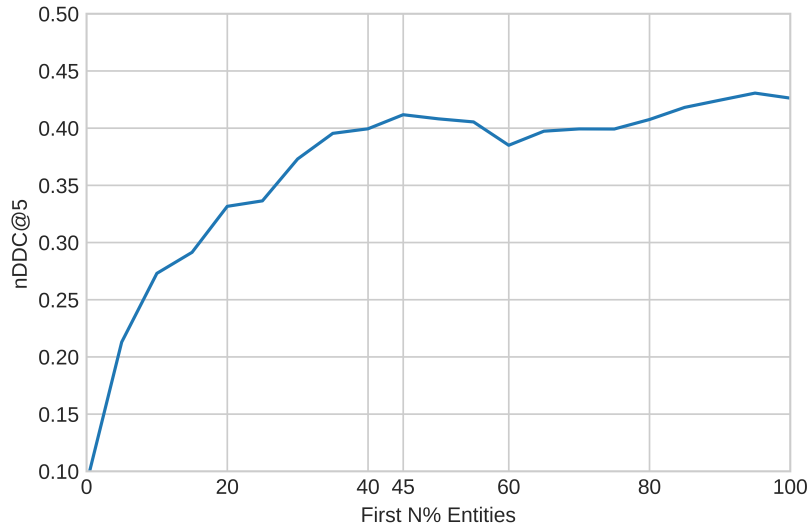


Figure 4.9 Wikification performance by using only the first $N\%$ of Entities for ranking

4.3.5 Other Participants Results

We report our final wikification results compared with 2020 Wikification participants in Table 4.8. All of our approaches score better compared to 2020’s participants. We also report our results on 2021 testing set. For our final results, we use the first 100 appearing mention-entity pairs in each document. These 100 pairs are ranked using their Candidate Ranking (model) scores.

We observe the Cross-Encoder effectiveness and improvement is consistent throughout this testing set. Elastic-Search only results proves better than Bi-Encoder. Bi-Encoder and Cross-Encoder combination scores slightly better than Elastic-Search and Cross-Encoder. However, this small improvement does not hold in the previous year. Elastic-Search shows a better overall performance as a Candidate Retrieval method.

Candidate Retrieval	Re-ranking	2020 Topics	2021 Topics
Elastic-Search	None	0.3483	0.6483
Bi-Encoder	None	0.3326	0.6422
Elastic-Search	Cross-Encoder	0.4306	0.7482
Bi-Encoder	Cross-Encoder	0.4223	0.7542
Ak et al. (2020)		0.3168	-
Ningtyas et al. (2020)		0.1191	-

Table 4.8 Wikification nDCG@5 score for 2020 and 2021 testing topics

5. Entity Ranking

In this chapter, we discuss the dataset, as in inputs and outputs, used in the Entity Ranking task. We also explain the methodology and algorithms used in tackling this task. It should be noted, this task was approached in 2019 before the Wikification task came out (in 2020) as part of our participation in TREC News 2019.

5.1 Dataset

In entity ranking task, the input is a document and the list of entities mentioned in that document. The task is to rank these entities in terms of importance to the reader and how much it will help the user in understanding the article or providing the needed context. To create the dataset, Stanford CoreNLP tool was used to detect named-entities. Then, the coordinators manually link each detected mention to its entry in the Wikipedia KB.

The testing topics are given in XML format as shown in Figure 5.1. The topic number "num" is used for easier reference to an article in the news corpus where the corresponding "docid" is also provided in the XML file. Original corresponding article is shown in Figure A.2 article The entities object contain a list of entities to be ranked by the system for that given article. Each entity is given an ID and the link for KB entry (enwiki:entity_link). Since an entity can be mentioned several times in article, different surface forms used to mention that entity are also provided. In 2018's track iteration, 50 test topics were provided. While there were 60 test topics provided for 2019 experiments. TREC also provides Wikipedia dump of 2017 as the KB for both 2018 and 2019 experiments. It's worth noting that this task was extended to a full Wikification task in 2020.

```

<top>
<num> Number: 826 </num>
<docid>96ab542e-6a07-11e6-ba32-5a4bf5aad4fa</docid>
<url>https://www.washingtonpost.com/sports/nationals/the-minor-leagues-life-in-
    pro-baseballs-shadowy-corner/2016/08/26/96ab542e-6a07-11e6-ba32-5
    a4bf5aad4fa_story.html</url>
<entities>
  <entity>
    <id> 826.1 </id>
    <mention>Richmond</mention>
    <link>enwiki:Richmond,%20Virginia</link>
  </entity>
  <entity>
    <id> 826.2 </id>
    <mention>Boston College</mention>
    <link>enwiki:Boston%20College</link>
  </entity>
  .
  .
  <entity>
    <id> 826.8 </id>
    <mention>San Francisco Giants</mention>
    <mention>Giants</mention>
    <link>enwiki:San%20Francisco%20Giants</link>
  </entity>
</entities>
</top>

```

Figure 5.1 Entity Ranking Input

For system evaluation, NIST assessors judge the entities for each test topic on a scale from 0 to 4 in terms of importance. The scoring criteria is the following:

- 4: Entity must be linked else crucial information is missing from the article.
- 3: Entity provides necessary context for the reader.
- 2: Entity provides important background information for the reader.
- 1: Entity might be helpful in understanding the larger picture or context of the article.
- 0: Entity does not provide any significant background information.

After the system evaluations of each year, the golden relevance judgements (qrels) used for evaluation are provided. These qrels provide the scoring for all test topic and entity pairs of that year.

The primary metric when this task was first released was Average Precision (AP) as shown in the following formula:

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

However, without a cut-off (@K), this metric takes all the list of entities where in a real-life application the ranking should be cut-off after some specific value. For better evaluation, 2019 News Track uses Normalized Discounted Cumulative Gain (nDCG) with cut-off @5 :

$$nDCG = \frac{DCG}{IDCG}$$

$$DCG = \sum_n \frac{rel_n}{\log_2(i+1)}$$

Where IDCG is the ideal discounted cumulative gain.

After performing ranking for each topic, participants should submit their results as a file where each line contains a tuple of test topics, mentioned entity and the ranking score:

topicNum	entityID	score
825	825.1	0.923
825	825.2	0.81

Figure 5.2 Entity Ranking Output Format (simplified)

5.2 Methodology

In this task, we hypothesize that the vector representation of important entities in a document should be close in vector space compared to the vector representation of the query document. To test this hypothesis, we learn the vector representation of Washington Post News and Wikipedia articles using Doc2Vec (Le & Mikolov, 2014). Then, we use cosine similarity as a proximity measure to produce a ranking for the list of entities. We provide a summary of the pipeline used in Figure 5.3 .

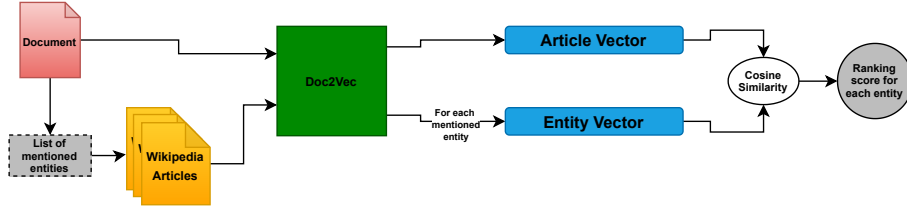


Figure 5.3 Entity Ranking Pipeline

5.2.1 Doc2Vec

Document distributed representations, or Doc2Vec (Le & Mikolov, 2014), is an unsupervised approach for learning a vector representation for phrases, paragraphs, or documents. Doc2Vec encapsulates both semantic and contextual into fixed-sized vectors for documents of all lengths. Doc2Vec is based on Word2Vec (Mikolov et al., 2013) algorithm for learning vector representation for words.

To train Doc2Vec, we use our provided corpora: Washington Post and Wikipedia. Training Doc2Vec only requires minimal pre-processing such as white-space tokenization. Embedding size was set to 200. As a training framework, we use the Distributed Memory (PV-DM) framework over the distributed Bag-Of-Words (dBOW) as PV-DM considers word ordering when learning the representations. After training Doc2Vec, we use the model to obtain vector representation for articles and each of their mentioned entities.

5.2.2 Vector Similarity and Ranking

After obtaining the vector representations for articles and entities, we calculate a proximity/similarity score between each article and mentioned entity. Our approach is based on the idea that articles and important entities share a good amount of keywords or phrases in their text. Therefore, the representations of the pair should be close to each other in vector space. To calculate vector similarity we use the Cosine Similarity.

Ranking score R for entity e_i mentioned in article D is calculated using the cosine similarity as the following:

$$(5.1) \quad R_{e_i}^D = \text{cos_sim}(V(D), V(e_i))$$

This score is calculated for every entity that is mentioned in the query article. The

full pipeline of ranking is summarized in Figure 5.3.

5.2.3 Using Background Linked Articles

We also extend our vector similarity-based hypothesis for Entity Ranking and investigate whether using similar articles such as Background Linked articles can benefit in Entity Ranking. The idea is that similar articles that share the same topic or important entities can have different keywords and phrases while discussing the topic. Therefore, including these similar articles while ranking entities might help in some cases.

To test this, we use Doc2Vec to encode all Washington Post articles into our 200-dimensional vector space. Then for each test article, we search the entire vector space and obtain the N most similar article using cosine similarity. These similar articles will now be used along with the original test topic to determine the ranking score for mentioned entities.

To perform the ranking, we explore different ways of including similar articles. We first use the cosine similarity from our original approach (see Equation 5.1) along with the cosine similarity between the top 1 similar article’s vector and each entity’s vector in a weighted fashion where the original score dominantly effects the final ranking. This approach is referred to as *orig + top1* . For top 1 similar article SIM_1^D , we calculate the ranking score using the following formula:

$$(5.2) \quad R_{e_i}^D = 0.9 * \text{cos_sim}(V(D), V(e_i)) + 0.1 * \text{cos_sim}(V(SIM_1^D), V(e_i))$$

We also retrieve the top 5 similar articles and incorporate their similarity score into the final ranking. This approach is referred to as *orig + top5* . Here, we using the following weighting :

$$\begin{aligned}
(5.3) \quad R_{e_i}^D = & 0.8 * \cos_sim(V(D), V(e_i)) \\
& + 0.05 * \cos_sim(V(SIM_1^D), V(e_i)) \\
& + 0.05 * \cos_sim(V(SIM_2^D), V(e_i)) \\
& + 0.04 * \cos_sim(V(SIM_3^D), V(e_i)) \\
& + 0.03 * \cos_sim(V(SIM_4^D), V(e_i)) \\
& + 0.03 * \cos_sim(V(SIM_5^D), V(e_i))
\end{aligned}$$

We also test the effect of using similar articles by just using the similarity of the similar article vectors and the entities without including the original test article. We experiment with using only the top 1 similar article. This approach is referred to as *top1*. We calculate the final ranking score as the following:

$$(5.4) \quad R_{e_i}^D = \cos_sim(V(SIM_1^D), V(e_i))$$

Also, we use the top 5 retrieved articles in an averaged fashion (approach is referred to as *top5*):

$$(5.5) \quad R_{e_i}^D = \frac{1}{5} \sum_{n=1}^5 \cos_sim(V(SIM_n^D), V(e_i))$$

5.3 Results and Discussion

In this section we discuss our experiments for choosing Doc2Vec’s vector dimension size. We also report and discuss the results for our similarity-based approach and compare to other participants in TREC News Entity Ranking task.

To select the optimal vector dimension size for the article and entity representation, we explore 3 different dimension sizes for our representations. We report the Entity Ranking performance using *orig* ranking method for the different vector sizes in Table 5.1. Based on the reported results, 50 and 500 dimensions proves too small

Dimension Size	TREC 2018		TREC 2019	
	MAP	nDCG@5	MAP	nDCG@5
50-D	0.7941	0.6982	0.6932	0.6384
200-D	0.8023	0.6913	0.7812	0.7509
500-D	0.7933	0.6960	0.7095	0.6650

Table 5.1 Entity Ranking results using different Doc2Vec vector sizes

Approach	TREC 2018		TREC 2019	
	MAP	nDCG@5	MAP	nDCG@5
<i>orig</i>	0.8184	0.7232	0.7906	0.7579
SIGNAL	0.7144	0.6084	-	-
TREMA-UNH	0.7859	0.4278	-	-
CMU	-	-	0.5778	0.4782
RUIR	-	-	-	0.6220

Table 5.2 Entity Ranking results compared to Other participants

or large for embedding news articles and entities. For the rest of Entity Ranking work, we utilize 200-D dimensional model to encode both articles and entities.

Table 5.2 reports the results for our Doc2Vec approach compared to other participants in 2018 and 2019 iterations of the Entity Ranking task. Our vector similarity based approach outperforms all participants on both 2018 and 2019 test topics. We observe good improvements over the salience ranking approach (SIGNAL) (van der Sluis et al., 2018) where they train their ranking model on Dexter ¹ dataset. Our approach is without the need for any fine-tuning using an external dataset. Our unsupervised approach only uses the provided text corpora for learning the vector representations and doesn’t require any labeled dataset for performing the ranking. Vector similarity approach proves to be much more effective than vanilla retrieval methods (TREMA-UNH) (Kashyapi et al., 2018).

We also report the results of our experiments using similar articles on 2018 and 2019 Entity Ranking topics in Table 5.3. We observe a slight increase in performance on 2018 topics when using 5 similar articles alongside the original article (*orig + top5*). However, this slight improvement does not translate into 2019 test topics. We observe that the original approach *orig* is still more consistent over the two testing sets. To further understand the effects of using similar articles, we observe the results of using only similar articles (*top1* and *top5*). We can see results very close to our original approach: similar articles are useful for overall Entity Ranking task. However, our argument that similar articles can bring improvements to the original approach does not hold. In our weighted scoring models (*orig + top1* and

¹<https://github.com/dexter/dexter-datasets/tree/master/entity-saliency>

Approach	TREC 2018		TREC 2019	
	MAP	nDCG@5	MAP	nDCG@5
<i>orig</i>	0.8184	0.7232	0.7906	0.7579
<i>top1</i>	0.8023	0.6913	0.7812	0.7509
<i>top5</i>	0.8154	0.6954	0.7785	0.7426
<i>orig + top1</i>	0.8143	0.7161	0.7551	0.7287
<i>orig + top5</i>	0.8199	0.7234	0.7561	0.7319

Table 5.3 Entity Ranking Results Using Similar Articles

(*orig + top5*) we do not observe improvements over (*orig*), but rather a decrease in performance. This means that the vector representation of the similar articles either doesn't provide any additional information or provides useless information that can hurt the final ranking. We further investigate this approach by using actual Background Linking articles extracted from QRELS files in our similar articles approaches. In other words, we use golden BL articles in the place of similar articles obtained by Doc2Vec. Using BL articles does not introduce any significant gain or loss in the results. In Chapter 6, we provide an analysis over the usage of Doc2Vec and similarity ranking for the task of Background Linking.

6. Background Linking

Background Linking is the task of retrieving and ranking articles in terms of providing background information to query article. In this chapter we discuss the dataset used for the Background Linking task. We also discuss our methodology, experiments and findings for this task.

6.1 Dataset

Background Linking task has featured in all 4 years of TREC News track versions. For this task, the participants are given test topics (WaPo articles) and are asked to retrieve a ranked list of news article that best provide background information and context to the reader. The input for this task is identical to the Wikification task where only topics are provided Figure 6.1.

```
<top>
  <num> Number: 886 </num>
  <docid>AEQZNZSVT5BGPPUTTJ07SNMOLE</docid>
  <url>https://www.washingtonpost.com/politics/2019/06/05/trump-says-transgender-
    troops-cant-serve-because-troops-cant-take-any-drugs-hes-wrong-many-ways/</
    url>
</top>
```

Figure 6.1 Background Linking Test Topic

The expected output is a ranked list of 100 Washington Post documents for each test topics in the following format:

	2018	2019	2020	2021
Background Linking Topics	50	60	50	51

Table 6.1 Background Linking testing topics in different years

topicNum	docID	score
825	2707e25a-cfaf-11e6-a87f-b917067331bb	0.923
825	513673ee-d003-11e6-b8a2-8c2a61b0436f	0.81

Figure 6.2 Background Linking Output Format (simplified)

Where docID is the background-linked news article identifier. Score is the relevance score used for ranking retrieved articles for each testing topic/query. Systems should retrieve documents using the entirety of Washington Post corpus. However, the coordinators provide some guidelines on articles linking (Soboroff et al., 2018). The articles should not be of news wire services nor should they be opinion or editorial articles. Another rule is that list of articles should be diverse, however the coordinators are still indecisive on the meaning of diversity in the context of news articles. News wire articles are already filtered from the corpus before release. However, it's the participants to filter out articles of types opinion and editorial. These special articles can be distinguished using the kicker field where articles with "Opinion", "Letters to the Editor" or "The Post's View" value should be excluded.

For system evaluations, coordinators use the same criteria used for Entity Ranking where articles are scored on a scale of 0-4 in terms of relevance to the core topic and importance to the reader. Assessors manually create relevance scores for the pooled document retrieved by systems. These are then used to evaluate systems using the nDCG@5 metric. Number of test topics for each year is provided in Table 6.1 .

6.2 Methodology

For all of our approaches, we apply a simple date filter where we only retrieve articles that were published before the query article.

6.2.1 Baseline: Full-Text Search

To evaluate our proposed approaches in Background Linking, we use a robust Full-Text search baseline that utilizes BM25 for ranking. This baseline is implemented using Elastic-Search index built over different versions of Washington Post corpus where we use all the content in articles as fields.

To perform Background Linking, we simply use all textual content of an article as query and retrieve the top 100 articles that were published before the query article.

6.2.2 Vector Similarity (Doc2Vec)

Similar to our Doc2Vec approach in Entity Ranking, we cast the task of Background Linking to a vector similarity problem. We hypothesize that articles that should be background linked share a set of keywords or phrases related to different aspects or topics. Therefore, vectors of background linked articles should be close in space to the query article. In this approach, we investigate Doc2Vec’s ability to encode different aspects or topics of news articles into vector representation.

The simple pipeline used is visualized in Figure 6.3. To encode the articles, we train the unsupervised Doc2Vec on Washington Post Corpus and obtain d-dimensional vector representation for all articles. At inference time, for a given test topic, we perform vector space search to retrieve N Washington Post articles with the highest Cosine similarity score with the query article vector.

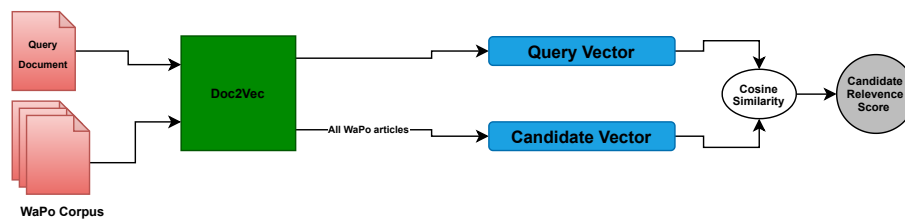


Figure 6.3 Doc2Vec-based Background Linking Pipeline

6.2.3 Neural Ranking for Ad-hoc Document Retrieval

In this approach of Background Linking, we apply neural ranking to rank a list of candidate documents retrieved by our baseline. We follow the approach in Contextualized Embeddings for Document Ranking (CEDR) (MacAvaney, Yates, Cohan

& Goharian, 2019). In their approach, MacAvaney et al. (2019) improve previously successful neural ranking architectures such Conv-KNRM (Dai, Xiong, Callan & Liu, 2018) and DRMM (Guo, Fan, Ai & Croft, 2016) by integrating BERT (Devlin et al., 2019) contextual word embeddings into the ranking architecture. Specifically, we utilize their proposed Vanilla BERT-based approach that uses Next Sentence Prediction (NSP) task model for ranking query-document pairs.

First, for each Background Linking test topic, we retrieve a list 100 candidates that will be re-ranked by BERT-based ranking model. For candidate retrieval, we use our Full-Text Search baseline .

After obtaining a list of candidate documents, we pass the candidates into the BERT-based re-ranking model. The re-ranking model is composed of BERT encoder with a sentence-pair classification layer on top. This architecture is used in Next Sentence Prediction task which is used in pre-training language models for sentence relation tasks such as Question Answering and Natural Language Inference (Devlin et al., 2019). Input of the sentence-pair classification task in BERT is modeled using the following formula:

$$[CLS] \textit{Sentence} - A [SEP] \textit{Sentence} - B$$

Where the task is to simply classify if Sentence-B comes after Sentence-A and the model output is a probability for the pair. MacAvaney et al. (2019) adapt this modeling for the purpose of Ad-hoc Document Ranking using the following formulation:

$$[CLS] \textit{Query} [SEP] \textit{Candidate Document}$$

Where the model output is a probability that represents the relevance of the document to the given query.

In our work, we use the same architecture proposed by CEDR to rank the list of candidates for each test topic. Sentence-A is the test topic and Sentence-B is the background linking candidate article retrieved by Elastic-Search.

A summary of our Background Linking pipeline is provided in Figure 6.4.

6.3 Experiments and Results

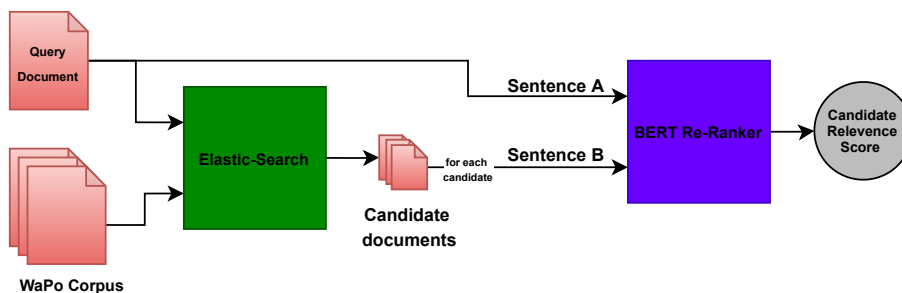


Figure 6.4 Background Linking using BERT-based re-ranker

Model	2018 Topics	2019 Topics	2020 Topics	2021 Topics
fullText-Search	0.4151	0.5727	0.5328	0.3849
Doc2Vec 50-D	0.1576	0.2202	0.1936	0.1864
Doc2Vec 200-D	0.2624	0.3620	0.3548	0.2393
Doc2Vec 500-D	0.2370	0.3421	0.3776	0.2162
CEDR-Model-1	0.3421	0.3595	0.3919	0.2871
CEDR-Model-2	Used in training	0.3670	0.3981	0.2903
CEDR-Model-3	Used in training	Used in training	0.4559	0.3101
CEDR-Model-4	Used in training	Used in training	Used in training	0.3361

Table 6.2 Background Linking performance on different testing topics sets

In this section, we report our experiments and results, and analyze the findings in our different approaches for Background Linking. Our Full-Text Search baseline results as-well as other approaches’ results are reported in Table 6.2.

6.3.1 Vector Similarity (Doc2Vec)

Similar to our experiments in Entity Ranking, we test Doc2Vec’s vector size parameter with three different values: 50,200,500. We then report the Background Linking performance using these different vector sizes in Table 6.2. Vector size of 200 again proves to be a better selection for representing articles where the Background Linking performance constantly improves over other sizes except in singular case where 500-D slightly improves over 2020 testing topics.

As for the overall performance, Doc2Vec performance is very low compared to our Full-Text search baseline. One trait for the retrieval system that should be reflected on the list retrieved articles is diversity. We believe simply using Doc2Vec with Cosine similarity does not enable us to retrieve a diverse list of articles on different topics but rather a list of articles sharing the same main topic of the query.

6.3.2 Neural Ranking for Ad-hoc Document Retrieval

As the base encoder, we use pre-trained BERT (bert-base-uncased). CEDR (MacAvaney et al., 2019) apply fine-tuning to optimize their ranking model performance for the task of relevance ranking. To understand the effect of fine-tuning, we test our model with, and without fine-tuning on our dataset. For fine-tuning, we use the provided relevance judgements (QRELS) for different years Background Linking testing topics. We use Adam optimizer (Kingma & Ba, 2017) with a learning rate of 1e-6 to optimize our model using pairwise ranking loss. Moreover, our inputs, query article and candidate article, are cropped or padded to 256 word-pieces each to conform to BERT’s maximum sequence length of 512 word-piece tokens.

In inference time, we obtain 100 candidates retrieved by our baseline for each test topic and re-rank them using our neural model to obtain the final ranked list of relevant articles.

In Table 6.2 we report the nDCG@5 score for the Background Linking performance on different testing sets. Neural ranking models are referred to as CEDR in the table. Our aim here to is to observe the effects of fine-tuning. Model-1 was not fine-tuned. Model-2 was fine-tuned on 2018’s topics and tested on other topics. Model-3 was fine-tuned on 2018’s and 2019’s topics and tested on other topics. Model-4 was fine-tuned on 2018’s, 2019’s and 2020’s topics and tested on 2021 topics. In Model-2, we observe minimal increase in performance compared to our non fine-tuned model (Model-1). While Model-3 shows promising increase in performance on 2020 topics and a small increase 2021 topics while trained on a bigger dataset compared to performance without fine-tuning (Model-1). We also observe another small boost in performance in Model-4 on 2021 testing topics.

6.3.3 Other Participants Results

Furthermore, we analyze the performance of our Background Linking approaches by comparing with other participants in the Background Linking task over its different iterations and the median of each year’s submitted runs by all participants as reported in Table 6.3. Our Baseline approach still out-performs the median and most approaches in Background Linking.

Khloponin & Kosseim (2020) explored different encoders to represent articles and different proximity measures to rank them. Their best representation-based ap-

Model	2018 Topics	2019 Topics	2020 Topics
fullText-Search	0.4151	0.5727	0.5328
Doc2Vec 200-D	0.2624	0.3620	0.3548
Neural Re-Ranker	0.3421	0.3670	0.4559
TREC Median	0.2792	0.5295	0.5250
Yang et al. (2019)	0.3293	0.4785	0.5231
Bimantara et al. (2018)	0.4619	-	-
Qu & Wang (2019)	-	0.5502	-
Lu & Fang (2019)	-	0.6064	-
Khloponin & Kosseim (2020)(GPT)	-	0.4660	0.4541
Khloponin & Kosseim (2020)(FT)	-	-	0.5924

Table 6.3 Background Linking performance of our approaches and different participants approaches on different testing topics sets

proach utilizes GPT-2 (Radford et al., 2019) (referred to as (GPT) in table) and shows a significant improvement over our Doc2Vec-based approach. However, it still doesn't compare to full-text search Baseline.

Qu & Wang (2019) perform Learn-To-Rank on a set of TF-IDF based features such as title, terms, and mentions similarity. This approach shows effective performance on 2019 dataset when compared with our fine-tuning based approach.

Yang et al. (2019) reports a baseline with top TF-IDF terms and BM25 for ranking. This approach shows better performance than our Doc2Vec and neural re-ranking methods throughout different testing topics. Best performing approach in 2018 (Bimantara et al., 2018) uses TF-IDF to rank queries with the extracted entities and key-phrases. Highest scoring approach in 2019 Lu & Fang (2019) builds a search index that utilizes linked entity mentions (using DBpedia Spotlight) and uses weighted entities and keywords for querying. Similarly for 2020, Khloponin & Kosseim (2020) use Full-Text search with additional terms boosting (referred to as (FT) in table) which shows an additional boost in performance over our vanilla Full-Text search baseline.

At the time of writing this work, the official publications of TREC News 2021 Background Linking task were not made public yet. Therefore, we do not report any work on Background Linking on 2021 testing topics.

7. Conclusion

In this work, we address the tasks of Entity Ranking, Wikification and Background Linking in the context of news articles. The aim in this thesis was to develop information retrieval tools to enhance the writing and reading of news articles. In doing so, we explore the use of deep architectures for modelling, retrieval, and ranking purposes. Moreover, we compare the results obtained with classic information retrieval baselines to validate our usage for deep architectures. It should be noted that most of work described in this paper was submitted in different iterations of TREC News Track earlier in 2019 (Fayoumi & Yeniterzi, 2019), 2020 (Ak et al., 2020) and 2021 Fayoumi & Yeniterzi (2021).

In Entity Ranking we cast this problem as similarity ranking task while using Doc2Vec to obtain vector representations of articles and entities. In Wikification, we address the different sub-tasks of the Wikification pipeline using different methodologies and models. We use effective and robust NLP tools for Mention Detection. For Candidate Generation we use search engines built using entity text contents and encoded vector representations of entities. For Candidate Ranking, we utilize deep transformer-based encoder and ranking model to provide a ranking for candidate entities. As for Wikification’s Entity Ranking sub-task, we apply simple yet robust ranking methods using deep learning model’s probabilities and order-of-appearance in text. As for Background Linking, we explored a transformer-based re-ranking model and fine-tune it for the task of relevance ranking.

Our results show Doc2Vec’s effectiveness for Entity Ranking in a perfect entity linking setting where our approach ranks first in Entity Ranking on 2018 and 2019 test topics. However, when used in Wikification pipeline where entity links are not given but rather predicted, Doc2Vec shows sub-par performances compared to baseline. Using transformer-based models for the Wikification sub-tasks of Mention Detection and Candidate Ranking proves very successful with high performance. As for Candidate Retrieval, using a dense vector search engine does not show any significant improvement in retrieval performance when compared to retrieval baseline. Using a transformer-based re-ranking model for relevance ranking without any task specific

fine-tuning yields low results. Fine-tuning on the small TREC dataset provides a boost in performance but still couldn't surpass a full-text search baseline as available dataset size is limited. Utilizing vector representation for relevant document ranking is helpful but does not surpass information retrieval baselines performances such as Full-Text search as seen in our work on Doc2Vec and other works in the literature that utilizes SOA transfer learning model for representations.

7.1 Research Questions

To conclude our work, we re-visit our research questions and discuss our finding to answer these questions:

RQ *How successful are vector representation and deep modelling based approaches compared to baseline information retrieval methods in the domain of news articles?*

In the course of work, we observed success of deep modelling and vector based approaches mainly in the context of entity-related tasks. Modelling articles/-mentions and entities in the same vector space and performing similarity search (Entity Ranking) or ranking (Wikification) showed high results comparing to information retrieval baselines.

As for Background Linking task, using different vector representations or proximity measures as seen in our experiments and the works of authors does not exceed the performance of robust information retrieval baselines such as Full-Text Search.

As for our research sub-questions:

RQ1 *How effective are document representations for embedding different topics or concepts in news articles into a singular vector representation?*

In our Doc2Vec experiments, we learned that vectors representations can be useful in encoding some concepts or subjects in a document such as entities as seen in our Entity Ranking results.

However, as we have seen in our Background Linking results, these vector representations are not enough to capture different abstract topics and it shown very little diversity in the retrieved results.

RQ2 *How effective are out-of-the-box pre-trained language models compared with classic Information Retrieval systems for the task of ranking news articles in terms of relevance?*

In our work for Background Linking, we learn that language models may contain useful information but are not very effective in relevance ranking out-of-the-box (without fine-tuning) or with fine-tuning on small datasets. However, this approach shows promising results when using larger dataset for fine-tuning

RQ3 *Is mutual vector modelling of news articles and entities effective for ranking entities in terms of relevance to an article?*

In our work for Wikification we apply neural ranking on vector representations of mentions and entities and observed high results in terms Wikification performance.

This is also supported by our results on Entity Ranking task where mutually model the pair using Doc2Vec and use Cosine similarity to measure relevance.

7.2 Future Work

As we have seen and concluded in this work, classic information retrieval baseline are very effective. Utilizing SOA Entity Linking systems to link entities and inject entity knowledge when building indices can increase performance in relevance retrieval tasks such as Background Linking. EL systems can also be utilized for query expansion and boosting by using model predictions and probabilities.

Also, utilizing the relationships between entities such as distance and hierarchy in the knowledge base in the form of graphs-based features could improve the Wikification performance.

Finally, fine-tuning on larger dataset for similar tasks can improve our relevance neural ranking model for the task of Background Linking as we have seen limited but promising improvements when utilizing bigger dataset for fine-tuning.

BIBLIOGRAPHY

- Ak, A. E., Köksal, Ç., Fayoumi, K., & Yeniterzi, R. (2020). SU-NLP at TREC NEWS 2020. In Voorhees, E. M. & Ellis, A. (Eds.), *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).
- Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., & Vollgraf, R. (2019). Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, (pp. 54–59).
- Bimantara, A., Blau, M., Engelhardt, K., Gerwert, J., Gottschalk, T., Lukosz, P., Piri, S., Shaft, N. S., & Berberich, K. (2018). htw saar @ trec 2018 news track. In *TREC*.
- Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. volume 2008, (pp. P10008). IOP Publishing.
- Boers, P., Kamphuis, C., & de Vries, A. P. (2020). Radboud university at trec 2020. In *TREC*.
- Bougouin, A., Boudin, F., & Daille, B. (2013). TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, (pp. 543–551)., Nagoya, Japan. Asian Federation of Natural Language Processing.
- Broder, A. Z. (2000). Identifying and filtering near-duplicate documents. In Giancarlo, R. & Sankoff, D. (Eds.), *Combinatorial Pattern Matching*, (pp. 1–10)., Berlin, Heidelberg. Springer Berlin Heidelberg.
- Broscheit, S. (2019). Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, (pp. 677–685)., Hong Kong, China. Association for Computational Linguistics.
- Bunke, H. & Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. volume 19, (pp. 255–259)., USA. Elsevier Science Inc.
- Chandra, G. & Dwivedi, S. K. (2020). Query expansion based on term selection for hindi – english cross lingual ir. *Journal of King Saud University - Computer and Information Sciences*, 32(3), 310–319.
- Chen, H., Zukov-Gregoric, A., Li, X. D., & Wadhwa, S. (2020). Contextualized end-to-end neural entity linking.
- Dai, Z., Xiong, C., Callan, J., & Liu, Z. (2018). Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM '18*, (pp. 126–134)., New York, NY, USA. Association for Computing Machinery.
- Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry, SCG '04*, (pp. 253–262)., New York, NY, USA. Association for Computing Machinery.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training

- of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 4171–4186)., Minneapolis, Minnesota. Association for Computational Linguistics.
- Ding, Y., Lian, X., Zhou, H., Liu, Z., Ding, H., & Hou, Z. (2019). Ictnet at trec 2019 news track. In *TREC*.
- Durrett, G. & Klein, D. (2014). A joint model for entity analysis: Coreference, typing, and linking. volume 2, (pp. 477–490).
- Essam, M. & Elsayed, T. (2019). bigir at trec 2019: Graph-based analysis for news background linking. In *TREC*.
- Fayoumi, K. & Yeniterzi, R. (2019). Ozu-nlp at trec news 2019: Entity ranking. In *TREC*.
- Fayoumi, K. & Yeniterzi, R. (2021). Su-nlp at trec news 2021. In *TREC*.
- Ferragina, P. & Scaiella, U. (2010). Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10*, (pp. 1625–1628)., New York, NY, USA. Association for Computing Machinery.
- Foley, J., Montoly, A., & Pena, M. (2019). Smith at trec2019: Learning to rank background articles with poetry categories and keyphrase extraction. In *TREC*.
- Gerritse, E. J., Hasibi, F., & de Vries, A. P. (2020). Graph-embedding empowered entity retrieval. (pp. 97–110). Springer International Publishing.
- Gonçalves, G., Magalhães, J., & Callan, J. (2019). Proximity-based entity ranking. In Voorhees, E. M. & Ellis, A. (Eds.), *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*, volume 1250 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).
- Guo, J., Fan, Y., Ai, Q., & Croft, W. B. (2016). A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16*, (pp. 55–64)., New York, NY, USA. Association for Computing Machinery.
- Hasibi, F., Balog, K., & Bratsberg, S. E. (2016). Exploiting entity linking in queries for entity retrieval. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval, ICTIR '16*, (pp. 209–218)., New York, NY, USA. Association for Computing Machinery.
- Hoffart, J., Yosef, M. A., Bordino, I., Fürstenauf, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., & Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, (pp. 782–792)., Edinburgh, Scotland, UK. Association for Computational Linguistics.
- Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging.
- Joachims, T. (1997). A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97*, (pp. 143–151)., San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with

- gpus.
- Kamphuis, C., Hasibi, F., de Vries, A. P., & Crijns, T. (2019). Radboud university at TREC 2019. In Voorhees, E. M. & Ellis, A. (Eds.), *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*, volume 1250 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).
- Kashyapi, S., Chatterjee, S., Ramsdell, J. S., & Dietz, L. (2018). Trema-unh at trec 2018: Complex answer retrieval and news track. In *TREC*.
- Khloponin, P. & Kosseim, L. (2019). The clac system at the trec 2019 news track. In *TREC*.
- Khloponin, P. & Kosseim, L. (2020). The clac system at the TREC 2020 news track. In Voorhees, E. M. & Ellis, A. (Eds.), *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).
- Kingma, D. P. & Ba, J. (2017). Adam: A method for stochastic optimization.
- Kolitsas, N., Ganea, O.-E., & Hofmann, T. (2018). End-to-end neural entity linking.
- Lavrenko, V. & Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01*, (pp. 120–127)., New York, NY, USA. Association for Computing Machinery.
- Le, Q. V. & Mikolov, T. (2014). Distributed representations of sentences and documents.
- Le, V. A. & Demartini, G. (2019). UQ at the TREC 2019 news track. In Voorhees, E. M. & Ellis, A. (Eds.), *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*, volume 1250 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).
- Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3), 225–331.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.
- López-Úbeda, P., Díaz-Galiano, M. C., Valdivia, M. T. M., & López, L. A. U. (2018). Using clustering to filter results of an information retrieval system. In *TREC*.
- Lu, K. & Fang, H. (2018). Paragraph as lead - finding background documents for news articles. In *TREC*.
- Lu, K. & Fang, H. (2019). Leveraging entities in background document retrieval for news articles. In *TREC*.
- Lu, K. & Fang, H. (2020). Aspect based background document retrieval for news.
- MacAvaney, S., Yates, A., Cohan, A., & Goharian, N. (2019). Cedr. ACM.
- Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, (pp. 55–60)., Baltimore, Maryland. Association for Computational Linguistics.
- Mendes, P. N., Jakob, M., Garcia-Silva, A., & Bizer, C. (2011). Dbpedia spotlight: shedding light on the web of documents. In *I-Semantics '11*.

- Mihalcea, R. & Tarau, P. (2004). TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, (pp. 404–411)., Barcelona, Spain. Association for Computational Linguistics.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space.
- Missaoui, S., MacFarlane, A., Makri, S., & Gutierrez-Lopez, M. (2019). Dminr at trec news track. In *TREC*.
- Nathan Day, Dan Worley, T. A. (2020). OSC at TREC 2020 - news track’s background linking task. In Ellen M. Voorhees, A. E. (Ed.), *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).
- Nguyen, D. B., Theobald, M., & Weikum, G. (2016). J-NERD: Joint named entity recognition and disambiguation with rich linguistic features. volume 4, (pp. 215–229).
- Ningtyas, A. M., El-Ebshihy, A., Piroi, F., Hanbury, A., & Andersson, L. (2020). Tuw-ifs at trec news 2020 : Wikification task. In *TREC*.
- Peters, M. E., Neumann, M., Logan, R., Schwartz, R., Joshi, V., Singh, S., & Smith, N. A. (2019). Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (pp. 43–54)., Hong Kong, China. Association for Computational Linguistics.
- Poerner, N., Waltinger, U., & Schütze, H. (2020). E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, (pp. 803–818)., Online. Association for Computational Linguistics.
- Pöttker, H. (2003). News and its communicative quality: the inverted pyramid—when and why did it appear? *Journalism Studies*, 4, 501–511.
- Qu, J. & Wang, Y. (2019). Unc sils at trec 2019 news track. In *TREC*.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.
- Rahul Gautam, Mandar Mitra, D. R. (2020). Trec 2020 news track background linking task. In *TREC*.
- Reimers, N. & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.
- Rousseau, F. & Vazirgiannis, M. (2015). Main core retention on graph-of-words for single-document keyword extraction. In Hanbury, A., Kazai, G., Rauber, A., & Fuhr, N. (Eds.), *Advances in Information Retrieval*, (pp. 382–393)., Cham. Springer International Publishing.
- Soboroff, I. (2021). Trec 2021 news track overview. In *TREC*.
- Soboroff, I., Huang, S., & Harman, D. (2019). Trec 2019 news track overview. In *TREC*.
- Soboroff, I., Huang, S., & Harman, D. (2020). TREC 2020 news track overview. In Voorhees, E. M. & Ellis, A. (Eds.), *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

- Soboroff, I., Huang, S., & Harman, D. K. (2018). Trec 2018 news track overview. In *TREC*.
- Tixier, A., Malliaros, F., & Vazirgiannis, M. (2016). A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (pp. 1860–1870)., Austin, Texas. Association for Computational Linguistics.
- Tjong Kim Sang, E. F. & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, (pp. 142–147).
- Trani, S., Lucchese, C., Perego, R., Losada, D. E., Ceccarelli, D., & Orlando, S. (2018). Sel: A unified algorithm for salient entity linking. *Computational Intelligence*, 34(1), 2–29.
- van der Maaten, L. & Hinton, G. (2008). Visualizing data using t-sne. volume 9, (pp. 2579–2605).
- van der Sluis, D., Albakour, M., & Martinez, M. (2018). Signal at trec 2018 news track. In *TREC*.
- van Hulst, J. M., Hasibi, F., Dercksen, K., Balog, K., & de Vries, A. P. (2020). Rel: An entity linker standing on the shoulders of giants. ACM.
- Wan, X. & Xiao, J. (2008). Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2, AAAI’08*, (pp. 855–860). AAAI Press.
- Wu, L., Petroni, F., Josifoski, M., Riedel, S., & Zettlemoyer, L. (2020). Zero-shot entity linking with dense entity retrieval. In *EMNLP*.
- Yamada, I., Asai, A., Sakuma, J., Shindo, H., Takeda, H., Takefuji, Y., & Matsumoto, Y. (2020). Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (pp. 23–30). Association for Computational Linguistics.
- Yamada, I., Asai, A., Shindo, H., Takeda, H., & Matsumoto, Y. (2020). Luke: Deep contextualized entity representations with entity-aware self-attention.
- Yamada, I., Washio, K., Shindo, H., & Matsumoto, Y. (2020). Global entity disambiguation with pretrained contextualized embeddings of words and entities.
- Yang, P., Lin, J. J., & Cheriton, D. R. (2019). Anserini at trec 2018 : Centre , common core , and news tracks. In *TREC*.
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2020). Xlnet: Generalized autoregressive pretraining for language understanding.
- Yin, X., Huang, Y., Zhou, B., Li, A., Lan, L., & Jia, Y. (2019). Deep entity linking via eliminating semantic ambiguity with bert. volume 7, (pp. 169434–169445).

APPENDIX A


The screenshot shows the Washington Post website interface. At the top, the navigation bar includes 'Sections', 'Schedule', 'Standings', 'Scoreboard', 'Stats', 'MLB', 'Injuries', 'Depth Chart', and 'Transactions'. The main headline is 'Baseball's minor leaguers pursue their dreams below the poverty line'. The byline reads 'By Kent Babb and Jorge Castillo' with a date of 'August 26, 2016'. A large photograph shows a baseball team on a field at night with a rainbow in the sky. Below the photo is a caption: 'A rainbow arches across the sky over Lake Olmstead Stadium during the playing of the national anthem as Class-A Augusta GreenJackets play the Delmarva Shorebirds in Augusta on July 15. (John McDonnell/The Washington Post)'. The article text begins with 'AUGUSTA, Ga. — They trickled into the mid-July night, a rain-shortened loss beginning their Friday night early. A 25-year-old catcher hung back.' and continues with a paragraph about Matt Paré. A sidebar on the right features 'Most Read Sports' with five items, a 'Post REPORTS' section with an article 'Why a jury acquitted Kyle Rittenhouse', and a 'Monday Morning Post Up newsletter' sign-up form. The footer contains copyright information: '© 1996-2021 The Washington Post'.

Nationals Schedule Standings Scoreboard Stats MLB Injuries Depth Chart Transactions

Nationals/MLB

Baseball's minor leaguers pursue their dreams below the poverty line

By **Kent Babb** and **Jorge Castillo**
August 26, 2016



A rainbow arches across the sky over Lake Olmstead Stadium during the playing of the national anthem as Class-A Augusta GreenJackets play the Delmarva Shorebirds in Augusta on July 15. (John McDonnell/The Washington Post)

AUGUSTA, Ga. — They trickled into the mid-July night, a rain-shortened loss beginning their Friday night early. A 25-year-old catcher hung back.

Matt Paré chatted with teammates and underwent a long treatment session as other Augusta GreenJackets players exited through a clubhouse door wedged open with a broken bat. He kept himself awake with a marathon shower.

By the time Paré dressed, he had the clubhouse to himself, along with what was left of the GreenJackets' postgame buffet. "What have we got, Sarge?" he called toward clubhouse manager Kristopher Nichols, a former Army drill sergeant who appreciates — and usually rewards — Paré's resourcefulness.

Paré is the oldest player for the San Francisco Giants' Class A affiliate, and he begins most homestands by waiting out his teammates in order to stockpile free food.

Most Read Sports

- 1 We all thought that he would be the one
- 2 **Perspective** Reserve your nostalgia for your favorite teams, not their office supply sponsors
- 3 Why the Williams family finally decided to tell their story in 'King Richard'
- 4 **Analysis** Michigan State finds out Ohio State is still Ohio State (college football winners and losers)
- 5 Washington Spirit defeats Chicago Red Stars to win first National Women's Soccer League title

Post REPORTS

Latest episode
Why a jury acquitted Kyle Rittenhouse
▶ Listen 14:52
Unparalleled reporting. Expert insight. Clear analysis. Everything you've come to expect from the newsroom of The Post — for your ears.

Monday Morning Post Up newsletter
All the NBA news and commentary you need, every Monday morning.

E-mail address

washingtonpost.com
© 1996-2021 The Washington Post
Help
Policies and Standards
Terms of Service

Figure A.1 WaPo article in the original Washington Post website

```

{
  "article_url": "https://www.washingtonpost.com/sports/nationals/the-minor-leagues-life-in-pro-baseballs-shadowy-corner/2016/08/26/96ab542e-6a07-11e6-ba32-5a4bf5aad4fa_story.html",
  "author": "Kent Babb; Jorge Castillo",
  "id": "96ab542e-6a07-11e6-ba32-5a4bf5aad4fa",
  "published_date": 1472222644000,
  "source": "The Washington Post",
  "title": "Baseball\u2019s minor leaguers pursue their dreams below the poverty line",
  "type": "article"
  "contents": [
    {
      "blurb": "A rainbow arches across the sky over Lake Olmstead Stadium during the playing of the national anthem as Class-A Augusta GreenJackets play the Delmarva Shorebirds in Augusta on July 15. (John McDonnell/The Washington Post)",
      "fullcaption": "A rainbow arches across the sky over Lake Olmstead Stadium during the playing of the national anthem as Class-A Augusta GreenJackets play the Delmarva Shorebirds in Augusta on July 15. (John McDonnell/The Washington Post)",
      "imageHeight": 2363,
      "imageURL": "https://img.washingtonpost.com/rw/2010-2019/WashingtonPost/2016/08/20/Sports/Images/a01sharp1471656070.jpg",
      "imageWidth": 3184,
      "mime": "image/jpeg",
      "type": "image"
    },
    {
      "content": "<span>AUGUSTA, Ga. \u2014 They trickled into the mid-July night, a rain-shortened loss beginning their Friday night early. A 25-year-old catcher hung back.",
      "mime": "text/html",
      "subtype": "paragraph",
      "type": "sanitized_html"
    },
    {
      "content": "<a href='\"http://www.milb.com/player/index.jsp?sid=t478&player_id=613557\"' shape='\"rect\"'>Matt Par\u201c chatted with teammates and underwent a long treatment session as other Augusta GreenJackets players exited through a clubhouse door wedged open with a broken bat. He kept himself awake with a marathon shower.",
      "mime": "text/html",
      "subtype": "paragraph",
      "type": "sanitized_html"
    },
    {
      "content": "By the time Par\u201c dressed, he had the clubhouse to himself, along with what was left of the GreenJackets\u2019 postgame buffet. \u201cWhat have we got, Sarge?\u201d he called toward clubhouse manager Kristopher Nichols, a former Army drill sergeant who appreciates \u2014 and usually rewards \u2014 Par\u201c\u2019s resourcefulness.",
      "mime": "text/plain",
      "subtype": "paragraph",
      "type": "sanitized_html"
    },
    {
      "content": "Par\u201c is the oldest player for the San Francisco Giants\u2019 Class A affiliate, and he begins most homestands by waiting out his teammates in order to stockpile free food.",
      "mime": "text/plain",
      "subtype": "paragraph",
      "type": "sanitized_html"
    },
    ],
  ],
  ....
}

```

Figure A.2 WaPo article from Figure A.1 in JSON format

Alexander Mackenzie (politician)

From Wikipedia, the free encyclopedia

For other people see Alexander McKenzie (politician) and Alexander Mackenzie (disambiguation).

Alexander Mackenzie, PC (January 28, 1822 – April 17, 1892) was a Canadian politician who served as the second **prime minister of Canada**, in office from 1873 to 1878.

Mackenzie was born in Logierait, Perthshire, Scotland. He left school at the age of 13, following his father's death to help his widowed mother, and trained as a stonemason. Mackenzie immigrated to Canada when he was 19, settling in what became Ontario. His masonry business prospered, allowing him to pursue other interests – such as the editorship of a pro-Reformist newspaper called the *Lambton Shield*.^[2] Mackenzie was elected to the Legislative Assembly of the Province of Canada in 1862, as a supporter of George Brown.

In 1867, Mackenzie was elected to the new House of Commons of Canada for the Liberal Party. He became leader of the party (thus **Leader of the Opposition**) in mid-1873, and a few months later succeeded John A. Macdonald as prime minister, following Macdonald's resignation in the aftermath of the **Pacific Scandal**. Mackenzie and the Liberals won a clear majority at the 1874 election. He was popular among the general public for his humble background and apparent democratic tendencies.

As prime minister, Mackenzie continued the nation-building programme that had been begun by his predecessor. His government established the Supreme Court of Canada and Royal Military College of Canada, and created the District of Keewatin to better administer Canada's newly acquired western territories. However, it made little progress on the transcontinental railway, and struggled to deal with the aftermath of the Panic of 1873. At the 1878 election, Mackenzie's government suffered a landslide defeat. He remained leader of the Liberal Party for another two years, and continued on as a Member of Parliament (MP) until his death, due to a stroke.

Contents [hide]

- Early life
- Early political involvement
- Prime Minister (1873–1878)
 - Supreme Court appointments
- Later life
- Character
- Legacy
 - Namesakes
 - Other honours
- See also
- References
 - Citations
 - Works cited
 - General sources
- Further reading
- External links

Early life [edit]

Mackenzie was born on 28 January 1822 in Logierait, Perthshire, Scotland, the son of Mary Stewart (Fleming) and Alexander Mackenzie, Sr (born 1794), who were married in 1817.^[2] The site of his birthplace is known as Clais-'n-deoir "The Hollow of the Weeping", where families said their goodbyes as the convicted were led to nearby Gallows Hill. The house in which he was born was built by his father and is still standing in 2019. He was the third of 10 boys, seven of whom survived infancy.^[2] Alexander Mackenzie, Sr., was a carpenter and ship's joiner who had to move around frequently for work after the end of the Napoleonic Wars in 1815. Mackenzie's father died on 7 March 1836 and at the age of 13, Alexander Mackenzie, Jr., was thus forced to end his formal education to help support his family. He apprenticed as a stonemason and met his future wife, Helen Neil, in Irvine, where her father was also a stonemason.^[2] The Neils were Baptist and shortly thereafter, Mackenzie converted from **Presbyterianism** to **Baptist** beliefs.^[2] Together with the Neils, he immigrated to Canada in 1842 to seek a better life. Mackenzie's faith was to link him to the increasingly influential **temperance cause**, particularly strong in Canada West where he lived, a constituency of which he was to represent in the Parliament of Canada.^[2]

	
<div><div></div><div><div>The Honourable</div><div>Alexander Mackenzie</div><div>PC</div></div></div>	
 <div>Mackenzie in 1878</div>	
2nd Prime Minister of Canada	
 <div>In office</div>	November 7, 1873 – October 8, 1878
Monarch	Victoria
Governor	The Earl of Dufferin
General	
Preceded by	John A. Macdonald
Succeeded by	John A. Macdonald
Leader of the Liberal Party	
 <div>In office</div>	March 8, 1873 – May 4, 1880
Preceded by	Edward Blake
Succeeded by	Edward Blake
Member of the House of Commons of Canada	
 <div>In office</div>	September 20, 1867 – April 17, 1892
 <div>More...</div>	
Personal details	
Born	January 28, 1822 Logierait, Scotland
Died	April 17, 1892 (aged 70) Toronto, Ontario, Canada
Resting place	Lakeview Cemetery, Samia, Ontario

Figure A.3 Wikipedia page for "Alexander Mackenzie (politician)" entity

<p>Alexander Mackenzie (politician)</p> <p>Alexander Mackenzie, (January 28, 1822April 17, 1892) was a Scottish–Canadian politician who served as the second prime minister of Canada, in office from 1873 to 1878.</p> <p>Mackenzie was born in Logierait, Perthshire, Scotland. He left school at the age of 13, following his father's death to help his widowed mother, and trained as a stonemason. Mackenzie immigrated to Canada when he was 19, ..., as a supporter of George Brown.</p> <p>In 1867, Mackenzie was elected to the new House of Commons of Canada for the Liberal Party. Mackenzie and the Liberals won a clear majority at the 1874 election. He was popular among the general public for his humble background and apparent democratic tendencies.</p> <p>As prime minister, Mackenzie continued the nation–building programme that had been begun by his predecessor. His government established the Supreme Court of Canada and Royal Military College of Canada,... Mackenzie's government suffered a landslide defeat. He remained leader of the Liberal Party for another two years, and continued on as a Member of Parliament until his death, due to a stroke.</p> <p>Early life</p> <p>Mackenzie was born on 28 January 1822 in Logierait, Perthshire, Scotland, the son of Mary Stewart (Fleming) and Alexander Mackenzie, Sr., who were married in 1817. The site of his birthplace is known as Clais-'n-deoir "The Hollow of the Weeping", where families said their goodbyes as the convicted were led to nearby Gallows Hill. The house in which he was born was built by his father and is still standing in 2019. He was the third of 10 boys, seven of whom survived infancy. Alexander Mackenzie, Sr....</p>
--

Figure A.4 Wikipedia page extracted content for "Alexander Mackenzie (politician)" entity