Using Document Embeddings for Background Linking of News Articles

Pavel Khloponin and Leila Kosseim

ClaC Lab
Department of Computer Science and Software Engineering
Gina Cody School of Engineering and Computer Science
Concordia University
Montreal QC, Canada
p_khlopo@encs.concordia.ca, leila.kosseim@concordia.ca

Abstract. This paper describes our experiments in using document embeddings to provide background links to news articles. This work was done as part of the recent TREC 2020 News Track [26] whose goal is to provide a ranked list of related news articles from a large collection, given a query article. For our participation, we explored a variety of document embedding representations and proximity measures. Experiments with the 2018 and 2019 validation sets showed that GPT2 and XLNet embeddings lead to higher performances. In addition, regardless of the embedding, higher performances were reached when mean pooling, larger models and smaller token chunks are used. However, no embedding configuration alone led to a performance that matched the classic Okapi BM25 method. For our official TREC 2020 News Track submission, we therefore combined the BM25 model with an embedding method. The augmented model led to more diverse sets of related articles with minimal decrease in performance (nDCG@5 of 0.5873 versus 0.5924 with the vanilla BM25). This result is promising as diversity is a key factor used by journalists when providing background links and contextual information to news articles [27].

Keywords: Background linking · Document embedding · Proximity measures

1 Introduction

Given the sheer number of electronic sources of news available today, it is important to develop approaches for the automatic recommendation of contextual information for users to better understand a news article. In order to address this need, since 2018, the News Track at TREC has proposed two related shared tasks: background linking and entity ranking ([24], [25], [26]). The goal of the background linking task is to provide relevant background information to news articles through the identification of related articles. On the other hand, entity ranking focuses on providing a list of names, concepts, artifacts, etc. mentioned in news articles, which will help readers better understand the news. This paper focuses on the first task: background linking.

Natural Language Processing and Information Systems: 26th International Conference on Applications of Natural Language to Information Systems, NLDB 2021, Saarbrücken, Germany, June 23–25, 2021, Proceedings • Jun 2021 • Pages 317–329 • https://doi.org/10.1007/978-3-030-80599-9_28

For the background linking task, NIST provides a large collection of news articles (see Section 3.1), and a set of search topics which are themselves articles from the collection. For each search topic, participants need to select up to 100 related articles from the collection and output them as a ranked list from the most related to the least related. For evaluation purposes, a 5 point rank is manually assigned to the top 5 documents by NIST assessors during the evaluation phase. The score assigned to each search topic is an integer between 0 (little or no useful information) and 4 (must appear in recommendations or critical context will be missed). The total score of the system is then computed using the nDCG@5 metric ([9]) as follows:

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

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

Most previous approaches to the background linking task are based on information retrieval methods, and very few have investigated to use of neural language models for the task. Given the recent successes of neural language models such as GPT2 ([22]), XLNet ([32]) and BERT ([5]), we wanted to evaluate their possible contribution to the task either as an alternative or as a complement to classic information retrieval approaches. In this paper, through our experiments for the recent 2020 TREC News Track, we show that embeddings alone do not reach the performance of the BM25 model, but combining them to BM25 can lead to more diverse sets of related articles with minimal decrease in performance (nDCG@5 of 0.5873 versus 0.5924).

2 Previous Work

News background linking can be seen as a classic information retrieval (IR) problem, where systems need to retrieve from a large document collection, a ranked list of documents related to a query; but in the case of news background linking, the query itself is a news article. For this reason, most participants in the 2018, 2019 and 2020 News TREC used approaches based on IR.

Classic IR approaches such as TF-IDF, BM25 or combinations of these are typically used for news background linking (e.g. [31], [18]). Other approaches include relevance models (e.g. [18], [12]) and probabilistic models (e.g. [14]). Another successful common approach is to extract named entities from articles and use them as additional features to retrieve relevant documents (e.g. [15], [1]). Several models have also experimented with re-ranking the results using relevance feedback [20], [10] based on the idea that the end users' behavioural information can provide additional useful information to rank relevant documents. In particular, [19] used information on the frequency of user's clicks on the initially provided links to re-rank documents; while [1] used this data for collaborative

filtering. Although relevance feedback has been shown to improve background linking, in the context of the TREC News task, such information is not available. Another approach ([6]) that yielded a high performance at TREC (nDCG@5 of 0.5918 in 2019) is based on query construction using a graph-based approach. Bigrams taken from the query articles are used to construct a co-occurrence graph. After pruning the graph, keywords are extracted with weights associated to them. These weighted terms are then used to run a query on Apache Lucene.

As shown above, most previous work in background linking is based on information retrieval approaches. To our knowledge, very little work has investigated the use of novel language models for the background linking task. The expectation is that related articles would be closer to each other in vector space, and using language models for document representation, vector distance metrics would be a good approximation of document content relatedness. In that vein, [4] used BM25 for an initial retrieval of the documents, then used SBERT (Sentence BERT [23]) to perform semantic similarity reranking between the query article and the articles retrieved by BM25. For this, a list of keywords are extracted from each returned article and from the query article. These lists of keywords are then treated as sentences and SBERT embeddings are built for them. Finally, the cosine similarity is then computed between these embeddings and used to re-rank the returned articles. In [3], the authors used a similar approach by first retrieving initial results with the Elasticsearch implementation of BM25, and using SBERT to directly build embeddings for the first three paragraphs of each article before averaging them to get a final embedding for the article, and applying the cosine similarity for the final ranking. On the other hand, [17] reached promising results using BERT, ELMo and GloVe embeddings on the ad-hoc document ranking task. Based on the BM25 relevance score, they selected positive and negative pairs for training queries which were then used to fine-tune embedding models and train classifiers. Another interesting approach was used in the News2Vec system ([16]) where the authors proposed a distributed representation of news based on news-specific features such as named entities, sentiment, month, publication week and day, word count, paragraph count, etc.

Given the recent successes of neural language models such as GPT ([21]), GPT2 ([22]), XLNet ([32]), BERT ([5]) and RoBERTa ([13]), we experimented with different document embedding representations and proximity measures as an alternative or as a complement to classic information retrieval approaches.

3 Our Approach

3.1 Document Collection

The TREC News document collection consists of 671,934 articles from The Washington Post published between 2012 and 2019. This document collection is the same as the one provided in previous years (2018 and 2019) but with duplicate articles removed and with new articles from 2017 to 2019 added. NIST required participants to ignore wire articles, editorial content and opinion posts. Due to this, the initial set of 671,947 articles was reduced by 2,057 to 669,890 items. This is shown in Table 1.

Articles were pre-processed, HTML markup and other meta information was removed and only the article text was preserved. As shown in Table 1, the 671,947 documents considered have an average length of 10,391 characters prior to preprocessing, but only 4,533 after pre-processing.

Table 1: Statistics of the 2020 TREC News document collection

original number of articles	671,947
articles to ignore as per NIST requirements	2,057
articles used to build the models	669,890
average size of article before pre-processing (characters)	10,391
average size after pre-processing (characters)	4,533
average size after pre-processing (tokens)	945

3.2 Document Representation

After pre-processing, we experimented with 5 families of models to represent each document: GPT ([21]) & GPT2 ([22]), XLNet ([32]), BERT ([5]) and RoBERTa ([13]), PEGASUS ([33]) models trained on the Newsroom ([8]) and Multi-News ([7]) datasets. In addition, for each family of models, we experimented with a variety of specific pre-trained models without fine-tuning. In all cases, before creating the document vectors, meta-information from the text was removed and Unicode characters were normalised using the NFC form of the Unicode Standard. We used the following 19 models available from Hugging Face:

5 BERT models: `bert-base-multilingual-cased`, `bert-base-multilingual-uncased`, `bert-large-cased`, `bert-large-uncased`, and `bert-base-uncased`.
5 GPT & GPT2 models: `openai-gpt`, `gpt2`, `gpt2-large`, `gpt2-medium`, and `gpt2-xl`.
5 RoBERTa models: `roberta-large-openai-detector`, `roberta-base-openai-detector`, `distilroberta-base`, `roberta-base`, and `roberta-large`.
2 XLNet models: `xlnet-base-cased`, and `xlnet-large-cased`.
2 PEGASUS models: `google-pegasus-multi_news`, and `google-pegasus-newsroom`.

The above models have a maximum input sequence size and cannot receive entire articles as input. To overcome this limitation, the tokenized article content was split into chunks of sequential tokens. When a chunk border falls in the middle of a sentence, instead of splitting it across chunks and potentially loosing its meaning for the embedding, overlapping of 64 tokens was introduced. We built embeddings with chunks of 500 and 250 tokens. Model specific padding tokens were added to the last chunk to match the chunk size. The resulting chunks were used as input to the models. Once the embedding vectors for the chunks from an article were built, they were pooled together to create the final embedding vector for the entire article. We explored three pooling methods: min, max and

mean pooling. Given the deep convolution nature of the models, we evaluated the embeddings pulled from the last hidden layer of the models as well as from the pooler output layer of the BERT and RoBERTa models.

3.3 Proximity Measures

After obtaining the embeddings for the articles, the proximity between two document vectors is computed. To do this, we explored a broad range of proximity measures. We experimented with all 62 different measures presented in [2]. These measures are grouped into 9 families:

1. L_p **Minkowsky** including Euclidean, Chebyshev ...
2. L_1 **family** including Sørensen, Gower ...
3. **Intersection** including Wave Hedges, Czekanowski, Ruzicka ...
4. **Inner product** including Jaccard, Cosine, Dice ...
5. **Fidelity or Squared-chord families** including Bhattacharyya, Matusita ...
6. **Squared** L_2 **or** χ^2 **families** including Squared Euclidean, Pearson χ^2...
7. **Shannon's entropy family** including Kullback–Leibler, Jeffreys ...
8. **Combinations** including Taneja, Kumar-Johnson ...
9. **Vicissitude** including Vicis-Wave Hedges, VicisSymmetric χ^2 ...

When taking into account the parameters in some of these measures, in total we experimented with 85 proximity measures.

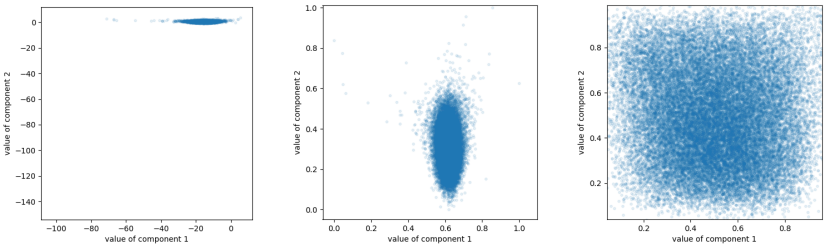
3.4 Normalisation

Because the off-the-shelf embeddings are not normalised, the amplitude of the components in the computed document vectors can differ by several orders of magnitude. In that case, components with larger amplitudes dominate the distance measures and smaller components are not given an opportunity to influence the distance measures. To avoid this problem, we experimented with normalisation. Figure 1 shows scatter plots of only two components of a set of document embeddings computed from `xlnet-large-cased` embeddings. Figure 1(a) shows the original components, while Figure 1(b) and Figure 1(c) show the same components when the embeddings are normalised to values within the range [0.0–1.0]. As shown in Figure 1, we experimented with two types of normalisation: amplitude normalisation and sigmoid normalisation.

Amplitude normalisation To generate embeddings normalised by amplitude, we calculated the minimum value (min_i) and the amplitude ($max_i - min_i$) of each component i for each embedding and scaled each value between 0 and 1 by deducting the corresponding minimum from each component and divided it by the amplitude of the component (see Equation 2).

$$vn_i = \frac{v_i - min_i}{max_i - min_i} \tag{2}$$

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



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

Fig. 1: Example of the effect of different normalisation techniques on two components of the `xlnet-large-cased` embeddings for the validation set.

Sigmoid normalisation In order to avoid the influence of outliers, we also experimented with sigmoid as a normalisation function. For this, we centered all components around their corresponding mean values and divided them by the component's standard deviations before applying the sigmoid function (see Equation 3). As shown in Figure 1(c), sigmoid normalisation allows for the components to be better spread over the [0,1] range.

$$vn_i = \frac{1}{1 + exp(\frac{v_i - \bar{v_i}}{\sigma_i})} \tag{3}$$

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

3.5 Validation

For validation purposes, NIST provided participants with the search topics and their corresponding manually evaluated results from the 2018 and 2019 TREC News tracks. From the 2018 edition, we had 50 topics and their manually ranked (from 0 to 4) background links, and from 2019, we had 57 topics and their ranked background links. Recall from Section 3.1 that the past document collection was very close to this year's so they constituted a representative validation set. We used these 2018 and 2019 sets of topics (107 in total) and evaluated backlinks (22,338 in total) for validation purposes. Note that these backlinks constitute only ≈3% of the entire document collection of 669,890 articles (see Table 1). To evaluate our 47,430 models, we generated the top 5 backlinks for each topic and computed nDCG@5 with the 2018 and 2019 datasets.

Table 2 shows the results of the top models. Although the best performances with the 2018 data are significantly lower than with the 2019 data, comparisons across years cannot be done as the queries and the backlinks are different.

Comparisons should be done within the same year. Among all embedding and similarity configurations, GPT2 embeddings outperformed all embeddings and dominated the top 261 best performing configurations with the 2019 dataset, and was among the leading models with the 2018 dataset (although XLNet did perform close to the GPT2 models).

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

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

Table 2: Top 5 performing models with the 2019 (top sub-table) and 2018 (bottom sub-table) validation datasets and their performance

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

A further comparison of the proximity measures is shown in Figure 2. The figure shows the maximum nDCG@5 reached by a distance measure regardless of the embedding method used. As the figure shows, the proximity measures seem to perform relatively similarly compared to one another on both the 2018 and 2019 datasets, but the top performing proximity measures are different.

Recall from Section 3.2, that three pooling methods were experimented with to create the document embeddings: min, max and mean pooling. As shown in Table 3, all top models use mean pooling. In addition, except for a few PEGASUS models, all 47,430 configurations show significantly higher results across all proximity measures when mean pooling is applied.

Finally, we analysed the influence of the chunk size when creating the document embeddings. As indicated in Section 3.2, articles were split into chunks to fit the models' requirements. We expected smaller chunks to decrease perfor-

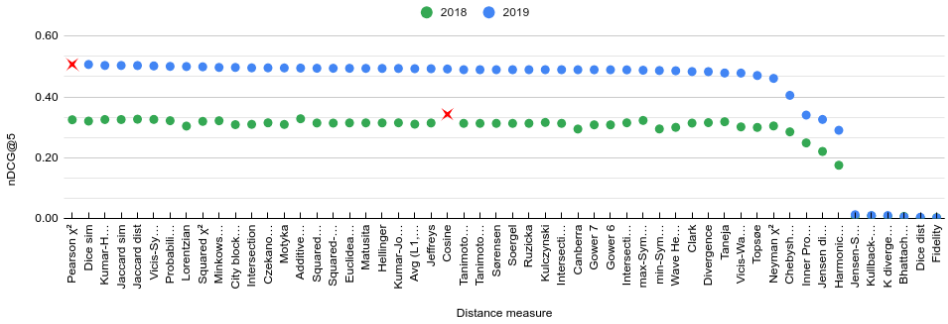


Fig. 2: Maximum nDCG@5 reached by each distance measure with the 2018 and 2019 data, independent of the embedding method. The top performing measure is marked with the a red "X".

mance, because each chunk contains less information, but to our surprise all but two models from Table 2 used the smallest chunk size (250 tokens).

4 Results and Analysis

Notwithstanding the results presented in Section 3.5, the classic BM25 model outperformed all embedding models by reaching nDCG@5 measures of 0.7418 (for 2019) and 0.5289 (for 2018). Based on this, at the recent 2020 shared task, we submitted both an embedding method along with the classic BM25.

4.1 TREC Runs

The validation of all 47,430 models of Section 3.5 was not ready in time for the 2020 TREC News task, therefore we used the best embedding model found from a smaller sub-set of experiments. We submitted 4 runs:

`gpt2_norm` is based only on GPT2 amplitude normalised embeddings with 250 token chunk size with mean pooling and Minkowski L_3 proximity measure. This configuration was chosen as it was the best one discovered among the embedding methods and proximity measures we had explored at the time of submission.

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

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

`d2v2019` is based on Doc2Vec embeddings computed from News TREC 2019 and cosine similarity as a proximity measure. This model was used because we used this approach last year in our participation to the track ([11]), and, for this year, we wished to compare last year's method to novel ones.

4.2 TREC Results

For each run and each topic, NIST provided us with our official score as well as the collective minimum, maximum and median scores. Table 3 shows the official overall scores. As shown in Table 3, two of our runs outperformed the collective median nDCG@5 of 0.5250. Among our submissions `es_bm25` achieved the highest score with an nDCG@5 of 0.5924, `combined` performed slightly below with 0.5873, while `gpt2_norm` and `d2v2019` performed below the collective median with nDCG@5 of 0.4541 and 0.4481 respectively.

Table 3: Overall results of our runs at TREC 2020

Run	**nDCG@5**
`es_bm25`	0.5924
`combined`	0.5873
`gpt2_norm`	0.4541
`d2v2019`	0.4481
TREC max	0.7914
TREC median	0.5250
TREC min	0.0660

4.3 Post-TREC Results

Once the validation of all models of Section 3.5 was complete, we simulated the official 2020 TREC News shared task to evaluate the top performing embedding configurations (see Table 4). We applied the methods to the entire 2020 document collection (669,890 articles, see Table 1) and used the TREC official scorer. Table 4 shows the final scores of these methods with the 2018–2020 queries.

Table 4: Performance of the top configurations with the validation set, the BM25 model and a new combined model on the entire document collection

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

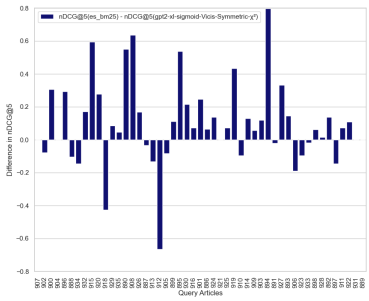
As the complete validation suggested, these top-performing configurations outperformed the embedding method submitted to the shared task, `gpt2_norm`

(gpt2 embedding with mean pooling, amplitude normalisation, chunk size of 250) which achieved an nDCG@5 of 0.4541 (see Table 3) but still performed below the overall TREC median of 0.5250 and es_bm25 (0.5924) and achieved an nDCG@5 of 0.4950. The new combined model (12), based on es_bm25 and model (4) (see Section 4.1), achieved an nDCG@5 of 0.5737; ranking lower than the es_bm25 model but higher than model (4) and the TREC median.

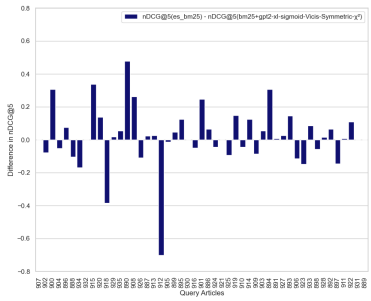
4.4 Analysis

As shown in Table 4, es_bm25 and model (12) are rather similar in terms of overall median nDCG@5, however when looking at individual topics, they return significantly different background links. Figure 3 shows this diversity graphically.

Figure 3(a) shows that es_bm25 performs better on most of the topics (see the upward bars) but significantly drops in performance on certain topics for which model (4) is very successful (see the long downward bars). For example model (4) returned the best result over all runs submitted for the topic #912 where es_bm25 missed the most relevant article. The query article and the most relevant backlink have less word overlap compared to the query article and the top backlink returned by es_bm25. Model (4), on the other hand, functioning on a different principle, was able to return the most relevant backlink in the first position.



(a) es_bm25 versus the model (4)



(b) es_bm25 versus the combined model (12)

Fig. 3: Difference in nDCG@5 scores for two pairs of models showing the diversity of background links for each topic

Figure 3(b) shows the per topic difference between es_bm25 and model (12). As the figure shows, model (12) improved on most of the topics for which es_bm25 outperformed model (4) without dropping in performance on most of the topics for which es_bm25 did not yield the top results.

5 Conclusions and Future Work

Through our experiments for the recent 2020 TREC News Track we have found that the best performing embedding methods are GPT2 and XLNet for the 2019

and 2018 validation sets respectively. In addition, regardless of the embedding, higher performances are reached when mean pooling, larger models and smaller token chunks are used. However, the best embedding configuration alone led to an nDCG@5 of 0.4950, which is significantly below the performance of the the classic Okapi BM25 method with an nDCG@5 of 0.5924.

This paper also showed that augmenting the BM25 model with GPT2 embeddings normalised with sigmoid funcion and using the Vicis-Symmetric χ^2 proximity measure, led to a more diverse sets of related articles with minimal decrease in performance (nDCG@5 of 0.5737 versus 0.5924).

This combination shows potential towards returning more diverse backlinks. By relying on different models with different implementations we can return topics potentially not visible to a single model system.

Many avenues of research still need to be investigated. In particular, we used the embedding models off-the-shelf with no fine-tuning. Tuning the models for our specific dataset and task itself might improve the representation of the documents in vector-space, potentially providing a better ranking for backlinks. In addition, we would like to explore different ways of combining BM25 and embedding methods, in particular to better leverage the diversity of the results, instead of favoring common results.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments on an earlier version of this paper. This work was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Adomavicius, G., et al.: Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. ACM Trans. Inf. Syst. **23**(1), 103–145 (Jan 2005)
2. Cha, S.H.: Comprehensive survey on distance/similarity measures between probability density functions. Int. J. Math. Model. Meth. Appl. Sci. **1** (01 2007)
3. Day, N., Worley, D., Allison, T.: OSC at TREC 2020 - News track's Background Linking Task. In: TREC [30]
4. Deshmukh, A.A., Sethi, U.: IR-BERT: Leveraging BERT for Semantic Search in Background Linking for News Articles. arXiv (2020), 2007.12603
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv (2019), 1810.04805
6. Essam, M., Elsayed, T.: bigIR at TREC 2019: Graph-based Analysis for News Background Linking. In: TREC [29]
7. Fabbri, A., Li, I., She, T., Li, S., Radev, D.: Multi-News: A Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model. In: Proc. of the ACL. pp. 1074–1084. Florence, Italy (Jul 2019)
8. Grusky, M., Naaman, M., Artzi, Y.: Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies. In: Proc. of NAACL/HLT. pp. 708–719. New Orleans (Jun 2018)
9. Järvelin, K., Kekäläinen, J.: Cumulated Gain-based Evaluation of IR Techniques. ACM Transactions on Information Systems (TOIS) **20**(4), 422–446 (Oct 2002)

10. Kashyapi, S., Chatterjee, S., Ramsdell, J., Dietz, L.: TREMA-UNH at TREC 2018: Complex Answer Retrieval and News Track. In: TREC [28]
11. Khloponin, P., Kosseim, L.: The CLaC System at the TREC 2019 News Track. In: TREC [29]
12. Lavrenko, V., Croft, W.B.: Relevance based language models. In: Proc. of the 24th ACM SIGIR Conference. pp. 120—-127. New York, NY (2001)
13. Liu, Y., et al.: RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv (2019), 1907.11692
14. Lu, K., Fang, H.: Leveraging Entities in Background Document Retrieval for News Articles. In: TREC [29]
15. Lu, M., et al.: Scalable news recommendation using multi-dimensional similarity and Jaccard–Kmeans clustering. J. of Systems and Software **95**, 242–251 (2014)
16. Ma, Y., et al.: News2vec: News Network Embedding with Subnode Information. In: Proc. of EMNLP/IJCNLP. pp. 4843–4852. Hong Kong (Nov 2019)
17. MacAvaney, S., Yates, A., Cohan, A., Goharian, N.: CEDR. Proc. of the 42nd International ACM SIGIR Conference (Jul 2019)
18. Naseri, S., Foley, J., Allan, J.: UMass at TREC 2018: CAR, Common Core and News Tracks. In: TREC [28]
19. Okura, S., et al.: Embedding-Based News Recommendation for Millions of Users. In: Proc. of the 23rd ACM SIGKDD Conf. p. 1933–1942. New York (2017)
20. Qu, J., Wang, Y.: UNC SILS at TREC 2019 News Track. In: TREC [29]
21. Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training. Preprint (2018), https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
22. Radford, A., et al.: Language models are unsupervised multitask learners. Preprint (2019), https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
23. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: Proc. of EMNLP/IJCNLP. p. 3982–3992. Hong Kong (Nov 2019)
24. Soboroff, I., Huang, S., Harman, D.: 2018 News Track Overview. In: TREC [28]
25. Soboroff, I., Huang, S., Harman, D.: 2019 News Track Overview. In: TREC [29]
26. Soboroff, I., Huang, S., Harman, D.: 2020 News Track Overview. In: TREC [30]
27. Soboroff, I., Huang, S., Harman, D.: TREC 2020 News Track Guidelines v2.1 (May 2020), http://trec-news.org/guidelines-2020.pdf
28. TREC (ed.): NIST Special Publication: Proc. of the 27^{th} Text REtrieval Conference (TREC), https://trec.nist.gov/pubs/trec27/trec2018.html (2018)
29. TREC (ed.): NIST Special Publication: Proc. of the 28^{th} Text REtrieval Conference (TREC), https://trec.nist.gov/pubs/trec28/trec2019.html (2019)
30. TREC (ed.): NIST Special Publication: Proc. of the 29^{th} Text REtrieval Conference (TREC), https://trec.nist.gov/pubs/trec29/trec2020.html (2020)
31. Yang, P., Lin, J.: Anserini at TREC 2018: CENTRE, Common Core, and News Tracks. In: TREC [28]
32. Yang, Z., et al.: XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv 1906.08237 (2020)
33. Zhang, J., Zhao, Y., Saleh, M., Liu, P.J.: PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. arXiv 1912.08777 (2019)