

Exploring Argument Retrieval for Controversial Questions Using Retrieve and Re-rank Pipelines

Notebook for the Touché Lab on Argument Retrieval at CLEF 2021

Raunak Agarwal¹, Andrei Koniaeav¹ and Robin Schaefer¹

¹*Department of Linguistics, University of Potsdam, 14476 Potsdam, Germany*

Abstract

This notebook documents Team Macbeth's contribution to the CLEF 2021 shared task *Touché: Argument Retrieval for Controversial Questions*. Our approach consists of different configurations of a two-step retrieve and re-rank pipeline. We experimented with sparse and dense approaches for argument retrieval and trained query-document cross-encoders for argument re-ranking. Our findings suggest that a sparse retriever combined with a custom re-ranker performed the best out of all our approaches.

Keywords

Argument Retrieval, Sentence Embeddings, Semantic Search

1. Introduction

In this notebook, we present our approaches for *Touché 2021 Task 1: Argument Retrieval for Controversial Questions* [1]. The task entails retrieval of arguments on a focused document collection crawled from debate portals [2]. The aim here is to assist users with finding "strong" arguments that support or oppose their position on a given controversial topic.

We discuss some of the recent work on information retrieval (IR) methods (Section 2), after which we present a high-level overview of our experiments (Section 3), as well as results (Section 4). For better reproducibility of our experiments, we also make available the source code¹ and the trained models².

2. Related Work

While standard information retrieval systems have focused largely on sparse bag-of-words-based approaches such as BM25, recent trends in IR indicate the performant nature of a two-step *retrieval and re-ranking* pipeline, where a sizeable number of candidate documents are first retrieved using the aforementioned sparse representations, and then re-ranked using (trainable) neural models [3].

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

✉ ragarwal@uni-potsdam.de (R. Agarwal); koniaeav@uni-potsdam.de (A. Koniaeav); robin.schaefer@uni-potsdam.de (R. Schaefer)

 © 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://github.com/raunak-agarwal/touche-2021-task-1>

²<https://huggingface.co/ragarwal>

Attempts are also being made to get rid of sparse representations altogether through the use of dense retrieval systems [4]. A standard dense retrieval architecture comprises a transformer-based encoder, which is fine-tuned on a given training corpus with queries and relevant documents. The encoded documents are usually added into an inverted index based on approximate nearest neighbours. There is also work which shows that combining sparse and dense representations can further enhance the performance of these IR systems [5].

Our submissions for the Touché shared task center around the above methods.

3. Experiments

3.1. Experimental Setup

All our experiments were computed on a setup comprising of an Intel Xeon E5-2650 CPU (24 cores, 256 GB RAM) and 2 NVIDIA GTX 1080Ti GPU's (24 GB VRAM). We also used *Weights and Biases*³ to track our experiments.

3.2. Pre-training

We pre-trained the entire args.me corpus on a Masked Language Modeling (MLM) task introduced first by BERT [6] and later modified by Liu et al. in RoBERTa [7]. RoBERTa demonstrated an improvement on BERT's performance with a small adaptation to the pre-training task, hence we chose to follow their approach.

Our motivation for pre-training was to make sure that our model first learns from the domain-invariant representations present in RoBERTa-base, and then enhances these representations through (continued) pre-training on our custom domain. This kind of domain-adaptive pre-training has been known to offer gains in task performance [8].

We used the hyper-parameters presented in the RoBERTa-base model and trained it for 10 epochs⁴, generating a domain-specific RoBERTa-base model with perplexity ≈ 4.1 .

3.3. Re-annotation

The organisers of Touché 2021 provide the participants with 2298 relevance judgements to allow training/evaluation of their systems. These relevance judgments are the result of crowd-sourcing efforts of Mechanical Turk⁵ workers - a practice which has been criticised for its questionable data quality [9], leaving aside major ethical considerations concerning labour exploitation [10].

Our initial plan was to use these annotations to train a sentence-pair classifier. After a closer look, however, we found that these annotations were riddled with errors and therefore, not suitable as a training set.

Instead of eliminating their use altogether, we decided to re-annotate all of the 2298 relevance judgements.⁶ We went through two rounds of annotation for each query-document pair, and achieved the following metric for inter-annotator agreement: Krippendorff's alpha = 0.39

³<https://wandb.ai/>

⁴<https://wandb.ai/ragabet/roberta-base>

⁵<https://www.mturk.com/>

⁶The annotations are available on our git repository.

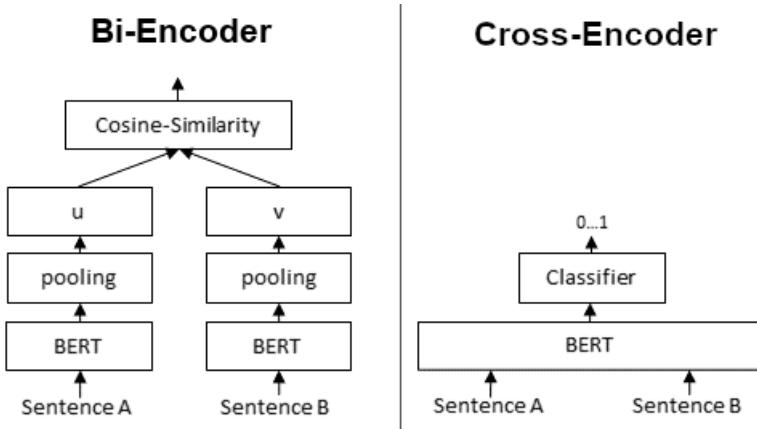


Figure 1: Pair-wise scoring architectures (Figure taken from the SBERT website [11])

Due to time constraints, the runs we submitted were trained only on the first round of annotations. The relatively low inter-annotator score suggests our runs would've turned out slightly different had we trained our models on an average of the two annotation rounds.

3.4. Sentence Embeddings

When it was first introduced, BERT set new state-of-the-art results on various NLP tasks, including question answering, sentence classification, and sentence-pair regression. A big disadvantage of the BERT’s network structure, however, was its inability to generate sentence embeddings based on single-input sequences.

To overcome the above issue, we used UKP Lab’s Sentence-BERT (or SBERT) [12] which is a modification of the standard BERT architecture. SBERT adds a mean pooling operation on top of the contextualized word vectors generated by BERT/RoBERTa. This enables the generation of semantically meaningful sentence/document embeddings which can be used for downstream tasks. We made use of the regression objective function described in their paper. A pair-wise regressor was trained using cosine-similarity between the two embeddings u and v (where u is the query embedding and v is the document embedding). The objective function was optimized using mean-squared-error loss.

The terminology used in SBERT is further refined by Humeau et al. [13] where the following approaches for pair-wise sentence scoring are defined: Bi-Encoders and Cross-Encoders. (See Figure 1).

3.4.1. Bi-Encoder

The architecture introduced in SBERT is what is now known as a bi-encoder. Using a bi-encoder, each sentence can be encoded into an independent sentence embedding. The creation of these vector representations enables efficient document retrieval through the use of standard similarity measures (such as Euclidean distance/cosine-similarity) in the embedding space.

After the pre-training step (3.2), we trained a bi-encoder using the query-document annotations described in 3.3. This bi-encoder was used to generate document embeddings for the entire corpus, giving us an embedding space of size $m * n$, where m is the embedding size and n is the total number of documents. This embedding space was then indexed by a dense retriever as described in 3.5.2.

Note: Each document in the corpus consists of premises and a conclusion. To generate document embeddings, we ignore the conclusion and use only the premises.

3.4.2. Cross-Encoder

A cross-encoder is analogous to the standard BERT design where full-attention is applied across tokens over an input sentence pair. While a bi-encoder takes two inputs and returns two representations (or embeddings), cross-encoders take two inputs and return a single decision directly. They outperform bi-encoders on pair-wise sentence scoring tasks at the cost of speed.

Since cross-encoders are slow and do not produce independent embeddings, they cannot be used for retrieval tasks. We used them in the second step of our pipeline to re-rank documents where a cross-encoder was trained (after MLM pretraining 3.2) on the annotations as described in 3.3. As a baseline, we also made use of a cross-encoder pretrained on the MSMARCO dataset. [14]

3.5. Retrieval Models

3.5.1. Sparse: BM25 (Elasticsearch)

BM25 is a traditional bag-of-words-based retrieval function which scores the relevancy of documents for a given query using the frequencies of common terms between the query and document. As a variation of the TF-IDF function, it is sensitive to the token frequencies as well as their inverse document frequencies.

Due to its simplicity, computational efficiency, and performance, BM25 serves as a critical component of large-scale search applications and serves as the de facto industrial standard in IR tasks. To index our id-document pairs, we used the implementation available in Elasticsearch⁷ with the default settings enabled.

3.5.2. Dense: Approximate Nearest Neighbours (hnswlib)

Despite its robustness, BM25 has several shortcomings. It suffers from the *lexical gap* problem [15], a common occurrence in systems built on sparse representations; empirical results have also shown that it overly penalizes very long documents [16].

To overcome the above problems, we deployed BM25's sparse retriever alongside a dense retriever. Experimental results demonstrate that the contextual text representations from BERT are more effective than BM25 on retrieval tasks [4].

Constructing a dense retriever was a two step process: first, we encoded the entire corpus into a dense vector space using the bi-encoder described in 3.4.1. Second, the representations

⁷https://lucene.apache.org/core/7_0_1/core/org/apache/lucene/search/similarities/BM25Similarity.html

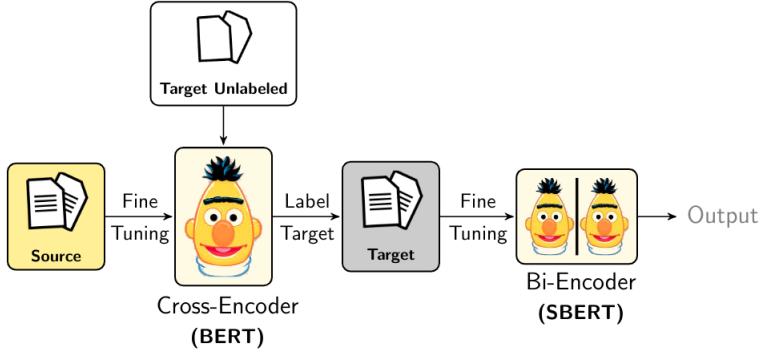


Figure 2: Pipeline for weakly-supervised learning (Figure taken from the Augmented SBERT paper [18])

created by the bi-encoder were indexed using a library that implements approximate nearest neighbours search (hnswlib).⁸

Approximate nearest neighbour search is an important step in efficiently generating similar document vectors for a given query vector. The alternative is to attempt cosine-similarity of the query vector with every single document vector i.e. brute force. We chose hnswlib since systems based on hierarchical navigable small world graphs (HNSW) [17] represent the current state-of-the-art in approximate nearest neighbour search.⁹

3.6. Data Augmentation

For our data augmentation approach, we utilized the methodology described in the Augmented SBERT paper [18] where a pre-trained cross-encoder was used to weakly label a sample of unlabeled query-document pairs. The query-document pairs were sampled using BM25, fed into a cross-encoder trained on MSMARCO to generate *silver* labels, which were then appended to the *gold* training set to train an augmented bi-encoder. (Figure 2)

3.7. Retrieve and Re-rank

The two-step pipeline of retrieve and re-rank has been known to work well on IR tasks. Given a search query, the first step is to retrieve a large list of candidate documents which are potentially relevant for the query. For the retrieval stage, we experimented with a sparse retriever (3.5.1), a dense retriever (3.5.2), and a combination of both (by simply appending the outputs of the two retrievers).

In the second step, we used a re-ranker based on a cross-encoder (3.4.2) that scores the relevancy of all the retrieved candidates (Figure 3). We experimented with a custom cross-encoder trained on our annotations and a pre-trained cross-encoder trained on the MSMARCO dataset. For each query, 100 candidate documents were retrieved and sent to the cross-encoder

⁸<https://github.com/nmslib/hnswlib>

⁹<http://ann-benchmarks.com/>

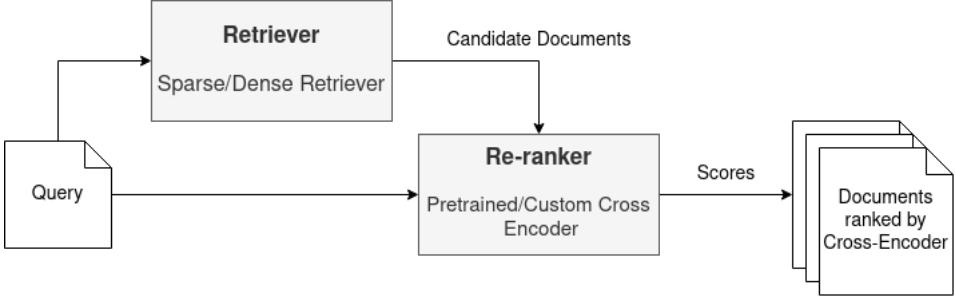


Figure 3: Overview of the retrieve and re-rank pipeline: candidate documents are retrieved using a sparse/dense retriever and then re-ranked using a pretrained/custom cross-encoder

Table 1

Summary of evaluation (Measure used: **nDCG@5**)

Run	Retriever	Augmenter	Reranker	Relevance	Quality
1	Sparse	No	Pretrained Cross Encoder	0.456	0.525
2	Sparse	No	Custom Cross Encoder	0.608	0.803
3	Sparse + Dense	No	Custom Cross Encoder	0.607	0.783
4	Dense	No	Custom Cross Encoder	0.507	0.75
5	Sparse + Dense	Yes	Custom Cross Encoder	0.554	0.752

for the purposes of re-ranking. After re-ranking, only the top 50 documents were included in the final submission file.

4. Evaluation

We performed evaluation using the relevance judgements¹⁰ and quality judgements¹¹ provided by the organisers of the shared task. The metric used was nDCG@5. The results are in Table 1.

5. Conclusion

In this paper, we outlined Team Macbeth’s contribution to the CLEF lab Touché. Our central approach consisted of using tried-and-tested methods for information retrieval and re-ranking. We pre-trained the args.me corpus on a masked language modeling task, re-annotated the relevance arguments from Touché 2020, and attempted neural methods for both retrieval and re-ranking. The combination of a sparse retriever and a custom neural re-ranker stands out as the best method in terms of both argument relevance as well as argument quality.

¹⁰<https://webis.de/events/touche-21/touche-task1-51-100-relevance.qrels>

¹¹<https://webis.de/events/touche-21/touche-task1-51-100-quality.qrels>

References

- [1] A. Bondarenko, L. Gienapp, M. Fröbe, M. Beloucif, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of touché 2021: Argument retrieval, 2021.
- [2] Y. Ajjour, H. Wachsmuth, J. Kiesel, M. Potthast, M. Hagen, B. Stein, Data acquisition for argument search: The args.me corpus, in: KI, 2019.
- [3] R. Nogueira, K. Cho, Passage re-ranking with bert, 2020. [arXiv:1901.04085](https://arxiv.org/abs/1901.04085).
- [4] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Y. Wu, S. Edunov, D. Chen, W. tau Yih, Dense passage retrieval for open-domain question answering, in: EMNLP, 2020.
- [5] Y. Luan, J. Eisenstein, K. Toutanova, M. Collins, Sparse, dense, and attentional representations for text retrieval, 2021. [arXiv:2005.00181](https://arxiv.org/abs/2005.00181).
- [6] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. [arXiv:1810.04805](https://arxiv.org/abs/1810.04805).
- [7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. [arXiv:1907.11692](https://arxiv.org/abs/1907.11692).
- [8] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, N. A. Smith, Don't stop pretraining: Adapt language models to domains and tasks, 2020. [arXiv:2004.10964](https://arxiv.org/abs/2004.10964).
- [9] D. Hauser, G. Paolacci, J. J. Chandler, Common concerns with mturk as a participant pool: Evidence and solutions, 2018. URL: psyarxiv.com/uq45c. doi:[10.31234/osf.io/uq45c](https://doi.org/10.31234/osf.io/uq45c).
- [10] D. Schlagwein, D. Cecez-Kecmanovic, B. Hanckel, Ethical norms and issues in crowdsourcing practices: A habermasian analysis, Information Systems Journal (2018). doi:[10.1111/isj.12227](https://doi.org/10.1111/isj.12227).
- [11] N. Reimers, I. Gurevych, Sentence-transformers documentation, <https://www.sbert.net/>, 2019. (Accessed on 05/28/2021).
- [12] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. [arXiv:1908.10084](https://arxiv.org/abs/1908.10084).
- [13] S. Humeau, K. Shuster, M.-A. Lachaux, J. Weston, Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring, in: ICLR, 2020.
- [14] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, T. Wang, Ms marco: A human generated machine reading comprehension dataset, 2018. [arXiv:1611.09268](https://arxiv.org/abs/1611.09268).
- [15] A. Berger, R. Caruana, D. A. Cohn, D. Freitag, V. Mittal, Bridging the lexical chasm: statistical approaches to answer-finding, in: SIGIR '00, 2000.
- [16] Y. Lv, C. Zhai, When documents are very long, bm25 fails!, Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval (2011).
- [17] Y. A. Malkov, D. A. Yashunin, Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2018. [arXiv:1603.09320](https://arxiv.org/abs/1603.09320).
- [18] N. Thakur, N. Reimers, J. Daxenberger, I. Gurevych, Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks, 2021. [arXiv:2010.08240](https://arxiv.org/abs/2010.08240).