Exploring BERT Synonyms and Quality Prediction for Argument Retrieval

Tommaso Green[a], Luca Moroldo[a] and Alberto Valente[a]

[a]*University of Padua, Italy*

Abstract

This paper attests the participation of the Yeagerists team from University of Padua in Touchè @ CLEF 2021 challenge, specifically in the Argument Retrieval for Controversial Questions shared task. We show our retrieval pipeline architecture and discuss about our approach, which employs a DirichletLM retrieval model coupled with transformers-based models for both query expansion and argument quality re-ranking. For the first, after having explored several possibilities, we decided to deploy a BERT-based synonym substitution tecnique. For argument quality re-ranking, we built an approach based on previous work and explored how different models from the BERT family performed in predicting a quality score for a given argument.

Keywords

Argument Retrieval, Automatic Query Expansion, Information Retrieval, Argument Quality, Transformers, BERT

1. Introduction

Search engines are nowadays one of the most important means to retrieve information for our society, however, they are not specialised for tasks related to the retrieval of nuanced and complex information. As the influence of search engines in opinion formation increases, it is of paramount importance for them to be able to address citizens' enquries on both general matters ("Should the death penalty be allowed?") or personal decisions ("Should I invest in real estate?") with relevant and high-quality results. This type of task, called *argument retrieval*, requires for the system to respond to user queries with arguments, which could be defined as a set of premises with evidence that lead to a conclusion. Often arguments have a stance, i.e. the author's position (in favour or against) on the debated subject. Clearly, this scenario requires to find a proper trade-off between how relevant an argument is with respect to the user query and its intrisic quality.

This paper summarizes our submission to the Touché @ CLEF shared task on Argument Retrieval for Controversial Questions which was centred on the usage of transformer-based models for query expansion and argument quality re-ranking.

"Search Engines", course at the master degree in "Computer Engineering", Department of Information Engineering, University of Padua, Italy. Academic Year 2020/21

tommaso.green@studenti.unipd.it (T. Green); luca.moroldo@studenti.unipd.it (L. Moroldo); alberto.valente.3@studenti.unipd.it (A. Valente)

As a brief introduction, our approach consists of two phases: in the first we use a simple Lucene-based searcher with our query expansion subroutine on top of it. In the second phase the relevance scores of retrieved documents are combined with the document quality scores provided by our argument quality module. We show that combining these components we can achieve competitive performance by properly selecting similarity functions, ranking strategies and other model-specific parameters.

The paper is organized as follows: Section 2 introduces related works; Section 3 describes our approach; Section 4 discusses our main findings; finally, Section 5 draws some conclusions and outlooks for future work.

2. Related Work

2.1. Argument Search Engines

As described in Wachsmuth et al. [1] an argument comprises a statement, i.e. the author's position on the topic, and premises, which are usually supported by evidence. Several subtasks constitute this area of research: argument mining, i.e. the extraction of an argument from raw text, argument retrieval, with the related techniques to increase the recall of the system such as *Query Expansion (QE)*, and argument re-ranking, so as to consider other parameters in addition to the relevance score, for example argument quality.

Some recent approaches in developing self-contained argument search engines include the args.me search engine, introduced in Wachsmuth et al. [1] and revised in Ajjour et al. [2]. A similar approach was used by IBM's *Project Debater* [3], where a topic-classifier was used to mine arguments from recognized sources (e.g. Wikipedia) at index-time. Differently from the above mentioned, ArgumenText [4] is more similar to web search engines as it indexes entire documents and delays argument mining to query-time.

2.2. Query Expansion

Query Expansion (QE) is a tecnique which consists in expanding a user query to increase its effectiveness in the search process, mainly by reducing its ambiguity and inserting new related keywords. As reported in Azad and Deepak [5], one of the best candidates for implementing a successful query expansion routine is to make use of the recently developed transformer-based models [6], which proved impressive performance and ability to grasp semantic nuances. For this reason, our query expansion approach aligned to that of Victor [7], who experimented on how contextual embeddings produced by a *Masked Language Model (MLM)*, such as BERT [8], can be applied in generating query expansion terms. Furthermore we took inspiration from the idea [9] of using an end-to-end BERT-based terms substitution approach, which proposes and validates substitute term candidates based on their influence on the global contextualized representation of the query. A big difference from [9] is that we do not apply dropout to target word's embedding to partially mask it, but we completely mask the word and let BERT generate a large enough batch of candidates from which to choose the best ones.

2.3. Argument Quality

The application of transformer-based pre-trained language models to argument quality prediction was explored in Gretz et al. [10], where they compared the performance of a Bi-LSTM with GloVe embeddings and SVR with 3 different variants of BERT and provided an extensive dataset on argument quality called *IBM-Rank-30k*.

3. Methodology

Our solution is composed of 3 parts: a Lucene-based indexer and searcher, a query expansion module and a quality measurement module. The procedure consists in reading the topics (i.e. queries) from a file, performing query expansion for each topic, running the searchers to retrieve a set of arguments, measuring the quality of a portion (of size n_{rerank}) of the retrieved arguments, and (re)ranking the arguments using the quality score.

3.1. Indexing

We developed a Lucene-based Java program to index the Args.me corpus. The main steps performed to create the index are:

- Arguments parsing: the result of this step is a set of parsed arguments containing an ID, a body, a stance, and a title.
- Lucene document creation: each parsed argument is transformed into a Lucene document containing the argument ID, body, and stance. Furthermore, if the parsed argument has a title, the title is included too. These fields were all indexed and their original content stored. We also stored the frequencies for the tokens in the body and title.
- Index writing: a single Lucene segment (using a compound file) containing all the previously created documents was constructed.

The Analyzer is responsible for the tokenization of the body and title of an argument. We used Lucene StandardTokenFilter and LowerCaseFilter: we did not remove stopwords. Finally, we used the LMDirichletSimilarity retrieval model.

3.1.1. The Args.me corpus

The Args.me corpus consists of 5 collections of arguments obtained from different sources. Each collection is a list of arguments containing a variety of fields: the body, which is the text written by the user to support a claim; A stance, that can be "PRO" or "CON" w.r.t. the parent discussion; A title, which summarizes the discussion that the argument belongs to. The title can be written by the author of the argument or inherited from the discussion, and it is sometimes empty. We chose to discard both duplicated arguments having the same ID as well as arguments having an empty body.

3.2. Searching

Given the text of a query, e.g. any Touché topic, we parsed it using Lucene's MultiFieldQueryParser. This parser allowed us to search for any match in both the body and the title (when present) of an argument, assigning different boosts depending on the field where the match occured.

3.3. Query Expansion

The Query Expansion subroutine generates a new set of queries for each query of a list, which in our case is the list of Touchè topics. It works as follows:

1. The query is tokenized and *Part of Speech (PoS)* tagged; The query tokens which are recognized as nouns, adjectives or past participles are masked, replacing them with BERT's [MASK] special token, then use BERT to generate the best 10 tokens that, according to its bidirectional attention mechanism, fit in place of each [MASK] token;
2. Compute the BERT embeddings of these 10 tokens and compare them, using cosine similarity, to the embedding of the original token; We considered as embeddings the 3072-dimensional vectors obtained concatenating the weights of the last 4 layers of the BERT fully connected neural network. This is the best performing embedding configuration according to the BERT authors themselves [8].
3. Perform a two-phase screening, where we keep only the best tokens among the 10 that have good similarity score. If no token is good enough, we use BERT again to generate new batches of candidates, keeping also less similar tokens. Provided the lists of new tokens for each position of [MASK], compute their cartesian product to compose the list of new queries and take from it a set of *max_n_query* queries at random.

3.4. Argument Quality

3.4.1. The Argument Quality Dataset

To build our argument quality predictor, we decided to use the Argument Quality Dataset of Gienapp et al. [11], as it contains 1271 arguments extracted from the args.me corpus, each having detailed quality scores as well as topical relevance. Quality scores were obtained by almost 42k pairwise judgements. We made our model predict only the overall quality, which can be obtained as the vector with rhetorical, logical and dialectical quality as components.

3.4.2. Adapting the BERT Family to the Argument Quality Dataset

Building on Gretz et al. [10], we decided to explore the possibility of using pre-trained language models for predicting the overall quality of an Argument. We make a distinction in the training process of models made of a transformer-based encoder and a traditional feedforward neural network:

- Adaptation: while keeping the weights of the encoder freezed, only the dense layer on top is trained on the new task;

- Fine-tuning: the whole model is fine-tuned on the new task.

Both approaches have advantages and disadvantages, and in the end we decided to go with the first approach as it reduces model complexity (lowering the possibility of overfitting) and requires less computational resources. In addition to this, as reported in [12] fine-tuning seems to be more appropriate when pre-training task and transfer tasks are closely related. This was not the case as BERT [8] for example is pre-trained on MLM and *Next Sentence Prediction (NSP)* (both can be framed as classification tasks) and the target task required to produce a real-valued score (regression task), so we decided to proceed with adaption.

The model works as follows: given an argument (truncated at 512 tokens), an encoded representation is computed using the [CLS] embedding of one of the models mentioned below. Finally, this representation is passed to a feedforward neural network of depth 3 which computes the MSE loss w.r.t. the original target value.

For the dataset, we used a train-validation-test split of 80%-10%-10%. Both hyperparameter selection and training were performed in a deterministic way by setting a specific seed.

3.4.3. Model Selection

Differently from Gretz et al. [10], we decided to explore 4 different models of the BERT family: BERT [8], DistilBERT [13], RoBERTa [14] and ALBERT [15]. BERT is by far the most famous one using the transformer encoder to create bi-directional contextualized word representations, and several variants have been published due to its omnipresence in state-of-the-art NLP. Specifically, we used the HuggingFace [16] implementations of the aforementioned models: bert-base-uncased, distilbert-base-uncased, albert-base-v2 and roberta-base.

For hyperparameter selection, an exhaustive grid search was performed over the following set of parameters: learning rate, Adam weight decay [17], batch size and dropout probability. If the latter is set to 0, a simple feedforward neural network with ReLU activations was used, otherwise AlphaDropout [18] was applied to the [CLS] embedding and to the hidden activation of the feedforward neural network. In that case ReLUs were substituted with SeLUs [18].

For all configurations we tracked the R^2 score on both training set and validation set and picked the best combination for each of the 4 models according to R^2 on validation set. The 4 selected models were then trained for 30 epochs, using early stopping and saving the best model according to validation R^2. Finally, the model performance was assessed on the remaining test set. Every experiment was logged using the online WandB logger [1] [19].

3.5. Ranking Functions

We decided to apply the argument quality re-ranking only to $D^q_{n\_rerank}$ i.e. the top *n_rerank* arguments according to the relevance score for each query q. For the final list of results, we had to combine relevance and quality scores. We tried three different ranking functions, using a parameter $\alpha \in [0, 1]$ that represented the importance of the quality score for a document d with relevance score $r(d)$ and a quality score $q(d)$:

[1] To see further details and logs about the training process, see here

- Normalization function: w.r.t. a query q, we re-ranked the list by normalizing relevance and quality scores:

$$R(q,d) = (1-\alpha)\, r_{norm} + \alpha\, q_{norm} = (1-\alpha)\, \frac{r(d)}{\max_{d' \in D^q_{n\_rerank}} r(d')} + \alpha\, \frac{q(d)}{\max_{d' \in D^q_{n\_rerank}} q(d')} \quad (1)$$

- Sigmoid Function: in order to compress relevance and quality scores, we decided to use a re-scaled sigmoid function with a scale parameter β, $\sigma(\beta x) := \frac{1}{1+e^{-\beta x}}$. The function used was:

$$R(q,d) = (1-\alpha)\, \sigma(\beta r(d)) + \alpha\, \sigma(\beta q(d)) \quad (2)$$

  The parameter β was used to counter the typical "squishing" of the sigmoid, which maps large positive or negative values into close output values (closer to 1 or 0 respectively), while values near the origin can assume a wider spectrum of values. As β tends to 0, the steepness of the sigmoid decreases and allows for a wider neighbourhood of values centred in the origin to get more diverse values.

- Hybrid Function: in this case, we applied the sigmoid to the quality scores while normalizing relevance scores:

$$R(q,d) = (1-\alpha)\, r_{norm} + \alpha\, \sigma(\beta q(d)) = (1-\alpha)\, \frac{r(d)}{\max_{d' \in D^q_{n\_rerank}} r(d')} + \alpha\, \sigma(\beta q(d)) \quad (3)$$

4. Results and Discussion

5. Conclusions and Future Work

References

[1] H. Wachsmuth, M. Potthast, K. Al-Khatib, Y. Ajjour, J. Puschmann, J. Qu, J. Dorsch, V. Morari, J. Bevendorff, B. Stein, Building an argument search engine for the web, in: Proceedings of the 4th Workshop on Argument Mining, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 49–59. URL: https://www.aclweb.org/anthology/W17-5106. doi:10.18653/v1/W17-5106.

[2] Y. Ajjour, H. Wachsmuth, J. Kiesel, M. Potthast, M. Hagen, B. Stein, Data Acquisition for Argument Search: The args.me corpus, in: C. Benzmüller, H. Stuckenschmidt (Eds.), 42nd German Conference on Artificial Intelligence (KI 2019), Springer, Berlin Heidelberg New York, 2019, pp. 48–59. doi:10.1007/978-3-030-30179-8_4.

[3] R. Levy, B. Bogin, S. Gretz, R. Aharonov, N. Slonim, Towards an argumentative content search engine using weak supervision, in: Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 2066–2081. URL: https://www.aclweb.org/anthology/C18-1176.

[4] C. Stab, J. Daxenberger, C. Stahlhut, T. Miller, B. Schiller, C. Tauchmann, S. Eger, I. Gurevych, ArgumenText: Searching for arguments in heterogeneous sources, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 21–25. URL: https://www.aclweb.org/anthology/N18-5005. doi:10.18653/v1/N18-5005.

[5] H. K. Azad, A. Deepak, Query expansion techniques for information retrieval: A survey, Information Processing Management 56 (2019) 1698–1735. URL: https://www.sciencedirect.com/science/article/pii/S0306457318305466. doi:https://doi.org/10.1016/j.ipm.2019.05.009.

[6] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Huggingface's transformers: State-of-the-art natural language processing, 2020. arXiv:1910.03771.

[7] D. Victor, Neuralqa: A usable library for question answering (contextual query expansion + bert) on large datasets, 2020.

[8] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.

[9] W. Zhou, T. Ge, K. Xu, F. Wei, M. Zhou, BERT-based lexical substitution, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 3368–3373. URL: https://www.aclweb.org/anthology/P19-1328. doi:10.18653/v1/P19-1328.

[10] S. Gretz, R. Friedman, E. Cohen-Karlik, A. Toledo, D. Lahav, R. Aharonov, N. Slonim, A large-scale dataset for argument quality ranking: Construction and analysis., in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 7805–7813.

[11] L. Gienapp, B. Stein, M. Hagen, M. Potthast, Webis Argument Quality Corpus 2020 (Webis-ArgQuality-20), 2020. URL: https://doi.org/10.5281/zenodo.3780049. doi:10.5281/zenodo.3780049.

[12] M. E. Peters, S. Ruder, N. A. Smith, To tune or not to tune? adapting pretrained representations to diverse tasks, in: Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), 2019, pp. 7–14.

[13] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, arXiv preprint arXiv:1910.01108 (2019).

[14] 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, arXiv preprint arXiv:1907.11692 (2019).

[15] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, in: ICLR 2020 : Eighth International Conference on Learning Representations, 2020.

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computa-

tional Linguistics, Online, 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.
[17] D. P. Kingma, J. L. Ba, Adam: A Method for Stochastic Optimization, in: ICLR 2015 : International Conference on Learning Representations 2015, 2015.
[18] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: Advances in Neural Information Processing Systems, volume 30, 2017, pp. 971–980.
[19] L. Biewald, Experiment tracking with weights and biases, 2020. URL: https://www.wandb.com/, software available from wandb.com.