Argument Retrieval Using Deep Neural Ranking Models

Saeed Entezari and Michael Völske

Bauhaus Universität Weimar
saeed.entezari@uni-weimar.de, michael.voelske@uni-weimar.de

Abstract Conversational argument retrieval is the problem of ranking argumentative texts in a collection of focused arguments in order of their relevance to a textual query on different topics. In this notebook-paper for Touché by taking a distant supervision approach for constructing the query relevance information, we investigate seven different deep neural ranking models proposed in the literature with respect to their suitability to this task. In order to incorporate the insights from multiple models into an argument ranking, we further investigate a simple linear aggregation strategy. By retrieving relevant arguments using deep neural ranking models, it will be inspected to what extent the systems whose main concentration is on relevant documents, would be able to retrieve arguments which meet various quality dimensions of the arguments. Our test results suggest that the interaction-focused networks provide better performance compared to the representation-focused networks.

1 Introduction

Arguments may have existed since humans first started communicating [4]. People use arguments in order to prove or contradict an opinion, in particular on controversial topics where opinions diverge widely. Rieke et al. define an argument as a unit composed of a claim (conclusion) and its supporting premises [10]. Generally, premises can support or attack a claim: the premises of one claim can be used to support or attack other claims. A conclusion could be a word, phrase or even a sentence. Typically the premises are texts composed of multiple sentences or paragraphs.

Due to the variety of opinion towards controversial topics, a corresponding query typically does not have a single correct answer, and getting an exhaustive overview can take considerable time [14]. In this situation, a ranking model which can neutrally retrieve the arguments on all sides of a controversial topic can provide users with a reasonable approach toward difficult questions. Such argument retrieval systems can benefit debate support and writing assistance systems, as well as automated decision making and opinion summarization.

This paper describes our contribution to the Touché 2020 shared task on conversational argument retrieval. By taking a distant supervision approach, our primary focus of investigation is on a variety of neural ranking models that have been proposed in the literature in recent years [3,8,15,16], and how they can apply to the conversational argument retrieval setting. In our experiments, a basis retrieval model such as BM25 produces an initial ranking which is then re-ranked by the deep neural model (except in the case of end-to-end models, which operate without an initial retrieval). We compare seven different neural ranking models overall. In addition to tackling this problem with individual neural rankers, we also explore a simple rank aggregation scheme based on a linear combination of the models' scores. Based on the test results, interaction-focused networks outperform significantly the representation-focused networks. Using the contextualized embedding representation, the convergence in the training phase happens faster and a certain level of performance could be achieved.

In what follows, we first review a selection of relevant related works on argumentation and argument retrieval, as well as the shared task setting. In Section 3, we briefly introduce the ranking models that comprise our study; we include recurrent siamese networks, kernel-based neural ranking models, different variants of contextualized embedding-based models, as well as stand-alone neural rankers. Section 4 explains our experimental setting including data preprocessing, training the various ranking models, as well as our aggregation setup, and Section 5 showcases our results. We conclude with a summary and discussion of our results in Section 6. . . .

2 Background and Related Work

Args.me, one of the first prototypes of an argument search engine [14] ranks arguments crawled from debate websites using the classical BM25F retrieval model. ARGUMENTEX retrieves topic-related arguments from a large collection of web documents [11] in a three-stage approach: (1) retrieving relevant documents using BM25, (2) identifying arguments in those documents, and (3) classifying the arguments into pro and con. To evaluate an argument's convincingness, Habernal and Gurevych proposed the use neural networks [6]: using on annotator judgments on how convincing the arguments are, a bidirectional LSTM is trained to predict which of a given pair of arguments is more convincing.

Dumani proposed a two-stage system for argument retrieval [4], which first retrieves the conclusions related to a given query, and then returns the premises associated with those conclusions. He suggested different similarity measures to semantically match conclusions to the query, such as plain language models with additional smoothing, and taking the textual context of the claim into account; these would be used to search through clusters of premises in the second stage.

The criteria for ranking arguments can be categorized into three main groups related to different argument quality aspects [13]: **Logical** aspects focus on the *soundness* of the arguments; logical arguments will have acceptable premises relevant to their conclusions. **Rhetorical** aspects pertain to the ability to persuade [7], and evaluate how successful an argument is in persuading its target audience [1]. **Dialectical** aspects assess the degree to which an argument helps its recipients formulate their own stance on the topic—this may also be considered as the *utility* of the argument [13]. Our study focuses especially on retrieving arguments relevant to a given query, and as such we are mainly concerned with retrieving *logical* arguments.

2.1 Touché task and Dataset

The Touché @ CLEF shared task on Conversational Argument Retrieval (Task 1) targets a retrieval scenario in a focused argument collection to support argumentative conversations [2]. The focused argument collection in this case is the args.me corpus,[1] which forms the setting for our study in combination with a collection of argumentative queries. While the arguments in this dataset are annotated with a stance, our models do not consider this for the purpose of evaluating their relevance to the given queries.

3 Models

Four categories of deep neural ranking models have been used in this study. Each category may include one or multiple network variations. For the all networks the hinge loss function (a pairwise loss function) which is typical for ranking tasks is used to train the models. Optimizing this loss function will contribute the models to put related documents over the unrelated ones. Note that except SNRM which is trained using TensorFlow 1.3, the rest of networks have been trained and validated in PyTorch 1.2. The models were trained on 7 different GPUs in parallel and took a day to get all models trained. The inference phase of all models can reproduced in the TIRA platform [9] and takes half an hour and 4 to 5 hours for the case of classical and contextualized embedding respectively. Due to the lack of GPU, reproducing the training results in TIRA would take a long time.

3.1 Recurrent Based Siamese Model

For the purpose of investigating the representation-based networks in the task of argument retrieval we have used Siamese network which are typically used for producing similarity score. Gated Recurrent Units (GRU) are used to produce representation of query and documents. The concatenation of the query and the document representations are then fed to a linear layer to produce a similarity score [12]. Bidirectional units with a hidden size of 512 have been used for GRU units and the linear layer is a fully connected network with an input size of 4×512 to 1 (the concatenation of two bidirectional units produces an output with the dimensionality of 4 times of the hidden state).

3.2 Kernel Based Neural Ranking Models

The Kernel based Neural Ranking Models (KNRM) aims to produce a similarity score for a given query and document pair by focusing on modeling the interaction that they have using RBF kernels. This model is composed of three important parts: translation model, kernel pooling, and learning to rank model [15]. The similarity score is produced by a fully connected learning-to-rank layer. The input of this layer is the result of applying RBF kernels to each row of a translation matrix whose elements are the cosine

[1] https://webis.de/data/args-me.html

similarities of the query and the document terms. The original implementation of the kernel based models and Siamese network is available [2].

Another variation of the kernel based neural ranking model used in this study is convolutional KNRM (Conv-KNRM). The most important difference between this network and KNRM is the use of a set of convolutional filters to form different n-gram embeddings. In the cross-matching layer, the similarity of the query and the document n-grams is calculated using cosine similarity [3]. Kernel pooling, the learning-to-rank layer, and the cost function are the same as for the previous network.

3.3 Contextualized Embedding for Ranking

The Contextualized Embeddings for Document Ranking (CEDR) model aims to improve ranking performance with the help of a deeper understanding of text semantics [8]. Unlike the traditional word embeddings such as word2vec or GloVe, contextualized language models consider the contexts of each word occurrence in order to assign it an embedding. For instance, the word *bank* may have different representations in different sentences depending on the context it occurs in.

Among the contextualized embedding techniques, BERT has proven to be one of the best performing in different NLP tasks. Through its ability to encode multiple text segments, BERT allows us to make informed judgments about the similarity of text pairs [8]. In this study we have used the BERT-base uncased model which produces a vector of 768 dimensions for the tokens. The original implementation of the networks which have used contextualized embedding can be found in GitHub [3].

Vanilla BERT Compared to the other deep neural ranking models using contextualized embedding, a relatively simple ranking model is obtained by the fine-tuning of the BERT model with a linear layer stacked at top [8]. During training, this linear layer requires a relatively larger learning rate than the pretrained BERT weights, which we only want to adjust slightly.

BERT and DRMM The language model knowledge encoded in the contextualized embeddings can be combined with any existing neural ranking model simply by stacking it on top of the BERT model [8]. One of the deep ranking models that we have used in this role is the DRMM model to see how the performance will change [5]. As the DRMM on its own did not represent a convincing performance on the validation set, we have excluded its result from reporting.

BERT and KNRM As an alternative to DRMM, we also combine the aforementioned KNRM model with the contextualized embedding. In our study, we use KNRM with static embedding, i.e. the BERT weights are not adjusted at all during training in this case. As we have already trained KNRM with static embedding, this setting will give us a good illustration of how the pretrained contextualized embedding will effect the performance of the model.

[2] https://github.com/thunlp/Kernel-Based-Neural-Ranking-Models/tree/master/src
[3] https://github.com/Georgetown-IR-Lab/cedr

3.4 Stand Alone Neural Ranking Models

All the networks that have been discussed up to now require a small set of candidate documents for re-ranking, which must be provided by a traditional retrieval model. As such, the performance of the model is limited by what the first-stage ranker (in our case BM25) can provide. By contrast, the stand-alone neural ranking model (SNRM) builds an inverted index from a latent sparse representation of the input document collection, which is searched directly with a corresponding representation of the query. This representation is achieved by an hour-glass shaped fully-connected network, and captures the semantic relationships between the query and documents [16]. During retrieval, SNRM finds those documents whose representations have non-zero in the same positions as the query; hence, the sparser the query representation, the faster the retrieval will be [16]. For this reason, the SNRM training procedure optimizes a traditional hinge loss term in combination with a sparsity objective. The original implementation of the network in TensorFlow can be found in GitHub [4].

4 Experiments

This section discusses the experiments of this study. In order to do an ad-hoc retrieval task we require the relevance information of the query and document pairs known as qrel file which can be derived from the click-through or query log information. In the provided dataset in Touché task however, we have just the annotation of the argument components. Thanks to the distant supervision that we have taken, we consider the annotated premise of each argument as a related document to the conclusion of the argument, which is considered as a query in the collection. For a typical ranking task, we still require unrelated documents to the queries. By using fuzzy similarity between the queries (conclusions), we assign the corresponding premise of the unrelated conclusions (conclusions with less fuzzy similarity score) to each argument. This way we form a binary version of qrel information for the dataset and prepare it to train ranking models for the task of ad-hoc retrieval argument task on it.

4.1 Training and Validation Data

We believe that the arguments whose premise lengths are less than 15 tokens could not be considered as convincing and good arguments. As a result we set aside such arguments. We have split the dataset into training and validations set. After the preprocessing step we are left with 312248 training and 4885 validation arguments. We tried to keep the validation set small in order to incorporate more information in the training phase while still allowing a meaningful assessment of model performance during validation. Note that we have selected the arguments with exactly 5 premises to be in validation set. According to the distant super vision approach, these premises would be the related documents to the conclusion of the argument. For each argument we assigned 100 unrelated premises.

[4] https://github.com/hamed-zamani/snrm

As the preprocessing phase of the contextualized embedding networks is a bit different (in contrast to the static embedding, in contextualized embedding the punctuation do not require to be tokenized) we formed two separate training and validation set for these networks. Note that the training and validation arguments are the same for these sets so that the results could be comparable.

4.2 Model Training

We keep the batch size to 32 for different networks. For all the networks, in order to have 8 evaluations per epoch, after 1239 training batches we run the validation to evaluate the performance of the network and if the MAP@20 measure was better than the best result obtained so far, the saved model is replaced correspondingly. As the query relevance information that we have formed for the dataset is in a binary format, we believe that MAP@20 would be a better evaluation measure compared to nDCG@20 as it is designed mostly for the soft similarity score of relevance. We run the different networks for 10 epochs. For the models with contextualized embedding, as the curves suggest, there is no need to train for this many epochs. We have trained them for 5 epochs. This saves the time and avoids complex computations out of which we do not get noticeable improvement. The average error and validation curves for different networks are displayed for every evaluation that we have done on the validation set. Note that the validation points are displayed in percentage and the coordinates of the best MAP@20 achieved in the corresponding run (the step number and the MAP@20 value) have been written displayed on the MAP curve with a blue dot.

Recurrent Network We keep the dimensionality for the input tokens to 100 and the learning rate to 0.001. The hidden size for the GRUs have been selected to be 512. For the linear layer we have the dropout layer with the rate of 0.5.

KNRM We decided to have 21 bins for this network as it was suggested by Xiong et al. [15]. Learning rate and word embedding dimensionality are as the same as recurrent network.

CKNRM The parameters for the network are the same as for KNRM model. Convolutional layers are 2D filters whose input is of dimension 1 and the output has the dimensionality of 128. The window sizes of the convolution layers are 1, 2, and 3 as suggested by Dai et al. [3]. The ReLu activation function has been applied on the output of the convolutional layers.

Vanilla BERT The learning rate for the BERT layers are much smaller than for the linear layer as we do not intend to make large changes to the pretrained contextualized embedding. We keep the learning rate of the BERT layers to be $2 * 10^{-5}$ and for the linear layer the learning rate is 10^{-3}. For the purpose of generalization we add a dropout layer with the probability of 0.1. The linear layer has the input size of 768 to 1. 768 is the embedding dimensionality for a token in BERT model.

Table 1: Best achieved evaluation scores of the models

Metrics @20			
Model	MRR	MAP	nDCG
GRU	28.4	24.1	38.05
KNRM	84.35	72.64	80.24
Conv-KNRM	86.72	73.32	82.08
SNRM	82.41	70.14	78.97
Vanilla BERT	95.12	88.5	91.00
KNRM BERT	94.57	90.18	89.80
DRMM BERT	95.97	88.09	91.34

BERT and DRMM The learning rates for the BERT and non-BERT layers are the same as the Vanilla BERT. The number of bins is 11. For the feed-forward network we exploited 2 hidden layers of 256 and 5 units.

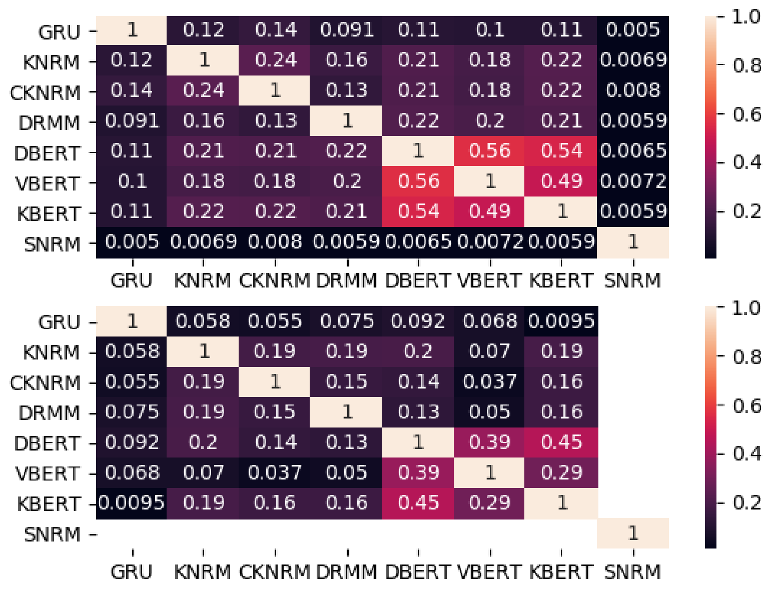
BERT and KNRM The Learning rate for the fine tuning of the BERT layers and training the KNRM layers are kept the same as for the Vanilla BERT model. The number of bins is 11 and the parameters for RBF functions are kept as what was suggested by the authors as the results on the sample data were acceptable.

SNRM For this model we did not use any hidden layer and it showed reasonable decrease of cost function on the training set. Learning rate is selected to be 10^{-4} and no drop out was used.

We have trained all the models in parallel on 8 GPUs. Table 1 shows the best evaluation scores achieved by different models.

Aggregation Now that we have the retrieved documents from each model, we can aggregate the results by producing a score which is the result of linear aggregation of the model scores. As the first step of aggregation, we analyze how diverse the result of the networks are. This would give a hint how reliable the network results are. Figure 1 illustrates two measures of ranking diversity namely Jaccard and Spearman. Considering the network results for the retrieved documents as vectors with the dimensionality of the retrieved documents and values of ranking score, we took the mean of the Jaccard and Searman measures over the 50 test queries for illustrating how diverse the result of the networks from each other are. We decided to exclude SNRM in the aggregation as its results are diverse from the rest of the models.

The linear regression is trained on the model results for the validation set. The trained model is then applied on the document scores for the test queries achieved from different models. All the model scores have been normalized to be in the same range.



	GRU	KNRM	CKNRM	DRMM	DBERT	VBERT	KBERT	SNRM
GRU	1	0.12	0.14	0.091	0.11	0.1	0.11	0.005
KNRM	0.12	1	0.24	0.16	0.21	0.18	0.22	0.0069
CKNRM	0.14	0.24	1	0.13	0.21	0.18	0.22	0.008
DRMM	0.091	0.16	0.13	1	0.22	0.2	0.21	0.0059
DBERT	0.11	0.21	0.21	0.22	1	0.56	0.54	0.0065
VBERT	0.1	0.18	0.18	0.2	0.56	1	0.49	0.0072
KBERT	0.11	0.22	0.22	0.21	0.54	0.49	1	0.0059
SNRM	0.005	0.0069	0.008	0.0059	0.0065	0.0072	0.0059	1

	GRU	KNRM	CKNRM	DRMM	DBERT	VBERT	KBERT	SNRM
GRU	1	0.058	0.055	0.075	0.092	0.068	0.0095	
KNRM	0.058	1	0.19	0.19	0.2	0.07	0.19	
CKNRM	0.055	0.19	1	0.15	0.14	0.037	0.16	
DRMM	0.075	0.19	0.15	1	0.13	0.05	0.16	
DBERT	0.092	0.2	0.14	0.13	1	0.39	0.45	
VBERT	0.068	0.07	0.037	0.05	0.39	1	0.29	
KBERT	0.0095	0.19	0.16	0.16	0.45	0.29	1	
SNRM								1

Figure 1: The heat map of the Jaccard (upper) and Spearman (lower) correlation coefficient for the 50 test queries

4.3 Test Queries

After training the models and getting the best one from the validation phase, it is time to give the models the test queries and see what documents would be ranked top. Except the SNRM model which has generated inverted index and can retrieve the documents on its own, other networks require to be provided with candidate documents (premises). To this end we make use of BM25.

We first group all the arguments based on the normalized conclusion column. Using BM25 we retrieve the most relevant normalized conclusions. We select the top 100 normalized conclusions. The premises corresponding to retrieved normalized conclusions are the candidate documents to be ranked by the neural networks. Note that each of the normalized conclusion may have a different number of premises. Consequently, the number of documents to be ranked may vary for different test queries. Figure 2 shows how we provide the trained networks with the document-query pairs to rank in the test phase. After getting the document scores, we sort them based on the score in a descending way. We introduce the top 100 premises as the retrieved arguments for each test query. There are 50 test queries which results in 5000 retrieved arguments by each model.

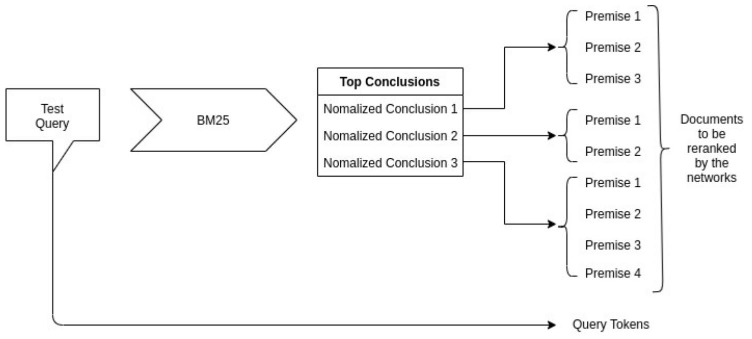
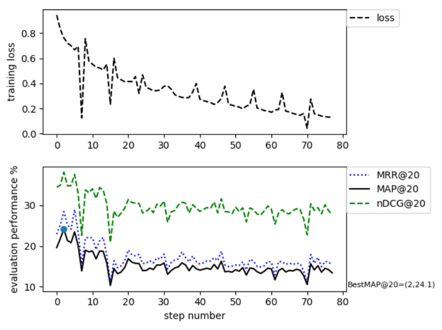


Figure 2: Candidate documents to be re-ranked in the test phase

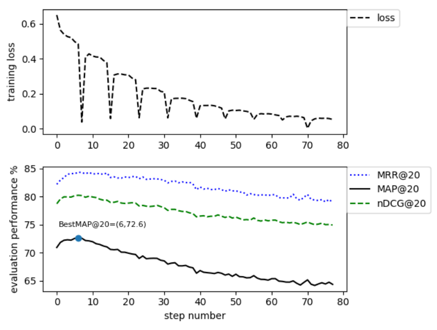
5 Results

Training and validation results Figure 3 showcases the training progress for a selection of the models in our study. Each subplot comprises a learning curve in the top half, which shows the development of the training loss over the epochs shown along the x-axis. Note that the mean of the error over 1239 batches are represented. The bottom half of each plot shows the development of the retrieval performance on the validation set performed for every 1239 training batch—as measured in terms of mean reciprocal rank (MRR), mean average precision (MAP), and normalized discounted cumulative gain (nDCG)—over the same time steps. The plots highlight that the contextual-embedding based retrieval models (Figures 3d, 3e, and 3f) converge faster, and achieve better validation performance than the other models (Figure 3a, 3b, and 3c) that don't incorporate contextual-embedding information.

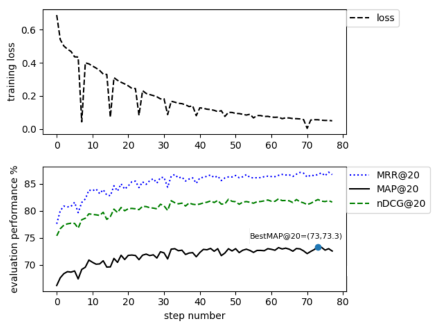
Test Results Table 2 shows the performance of the different models in the test phase provided by Touché committee. We assume that the models whose test results are not provided did not achieve better score than the displayed scores. The nDCG@5 has been reported as the test results. Evaluation of the retrieved arguments is done by human annotators based on the argument quality dimensions discussed by Wachsmus et.al in [13]. Devising the strategies for mapping the interaction of the input pairs may result in more promising models in the ad-hoc tasks. Represent-focused networks cannot have a good performance in retrieving relative arguments as they overlook the interaction of the input pairs. KNRM achieved the best score and ranked fourth among the competitors of the shared task. Exploiting the contextualized embedding contributes to achieve a certain level of test score which can be improved by devising more intuitive structures on the BERT weights. The best models with the best validation scores are not the best ones in the test phase. This may due to some facts: in the validation we focused on the top 20 retrieved documents while in the test phase, top 5 hits are targeted for each model. Furthermore, in the validation phase the models had to rank 105 premises. For the re-ranking in the test phase, however, this number is much larger ranging from 150 to 1200 arguments. Consequently, it is not surprising that the test scores would be of



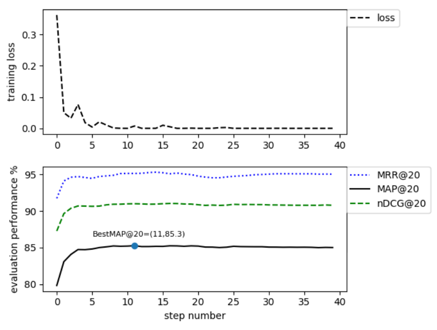
(a) Recurrent Based Siamese



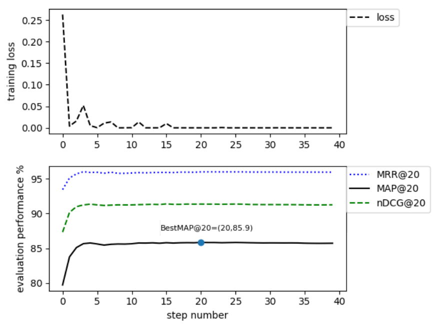
(b) KNRM



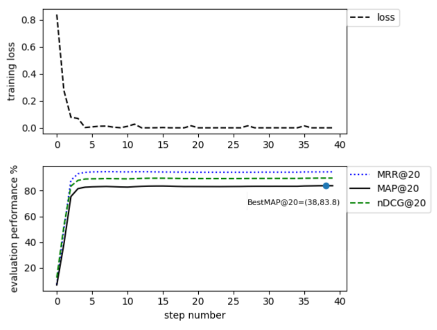
(c) CKNRM



(d) Vanilla BERT



(e) BERT and DRMM



(f) BERT and KNRM

Figure 3: Training and Validation curves

a lower grade. It can be interpreted that not necessarily related arguments would meet the other argument quality dimensions. Comparing the results of the validation and the test phase, highlights the importance of acquiring a dataset by which, developing the models for retrieving arguments meeting the other argument quality dimensions is possible. Considering the fact that only a few models of the competitors outperform the baseline method (Dirichlet LM with the score of 75.6%) reflects that retrieving the arguments meeting the quality dimensions of the arguments is not a trivial task.

Table 2: Test scores of the models

Model	nDCG@5 (%)
GRU	x
DRMM	x
KNRM	68.4
CKNRM	x
SNRM	x
Vanilla BERT	40.4
KNRM BERT	31.9
DRMM BERT	37.1
Aggregation	37.2

6 Discussion

In this study, thanks to taking a distant supervision technique, we used the deep neural ranking models to retrieve the most relevant arguments to the given queries provided in the Touché shared task. Test results suggest that focusing on the interaction of the input-pairs would contribute to more promising results in the ad-hoc retrieval task. KNRM achieved the best test results and ranked fourth among the competitors. Exploiting the contextualized embedding will result in achieving a certain level of score, a more intuitive structure is still required for better results. A mathematical expression of the argument quality dimensions to be included in the cost function of the models seems to be a primary step that should be taken for the task of argument retrieval. As the relevant arguments are not necessarily the ones which meet the other argument quality measures, developing a dataset including the information regarding to the different argument quality dimensions along side the relevance information is mandatory for developing the models with good retrieved arguments. A long way for devising an end-to-end neural ranking model for retrieving acceptable arguments exists to get a reliable results for the task of argument retrieval.

References

1. Blair, J.A.: Groundwork in the theory of argumentation: Selected papers of J. Anthony Blair, vol. 21. Springer Science & Business Media (2011)
2. Bondarenko, A., Fröbe, M., Beloucif, M., Gienapp, L., Ajjour, Y., Panchenko, A., Biemann, C., Stein, B., Wachsmuth, H., Potthast, M., Hagen, M.: Overview of Touché 2020: Argument Retrieval. In: Working Notes Papers of the CLEF 2020 Evaluation Labs (Sep 2020)
3. Dai, Z., Xiong, C., Callan, J., Liu, Z.: 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. pp. 126–134 (2018)
4. Dumani, L.: Good premises retrieval via a two-stage argument retrieval model. In: Grundlagen von Datenbanken. pp. 3–8 (2019)
5. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. pp. 55–64 (2016)
6. Habernal, I., Gurevych, I.: Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1589–1599 (2016)
7. Kennedy, G.A.: Aristotle, on Rhetoric: A Theory of Civic Discourse, Translated with Introduction, Notes and Appendices. Oxford: Oxford University Press (2007)
8. MacAvaney, S., Yates, A., Cohan, A., Goharian, N.: Cedr: Contextualized embeddings for document ranking. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1101–1104 (2019)
9. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World. The Information Retrieval Series, Springer (Sep 2019)
10. Rieke, R.D., Sillars, M.O., Peterson, T.R.: Argumentation and critical decision making. Longman New York (1997)
11. Stab, C., Daxenberger, J., Stahlhut, C., Miller, T., Schiller, B., Tauchmann, C., Eger, S., Gurevych, I.: 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. pp. 21–25 (2018)
12. Varior, R.R., Shuai, B., Lu, J., Xu, D., Wang, G.: A siamese long short-term memory architecture for human re-identification. In: European conference on computer vision. pp. 135–153. Springer (2016)
13. Wachsmuth, H., Naderi, N., Hou, Y., Bilu, Y., Prabhakaran, V., Thijm, T.A., Hirst, G., Stein, B.: Computational argumentation quality assessment in natural language. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. pp. 176–187 (2017)
14. Wachsmuth, H., Potthast, M., Al Khatib, K., Ajjour, Y., Puschmann, J., Qu, J., Dorsch, J., Morari, V., Bevendorff, J., Stein, B.: Building an argument search engine for the web. In: Proceedings of the 4th Workshop on Argument Mining. pp. 49–59 (2017)
15. Xiong, C., Dai, Z., Callan, J., Liu, Z., Power, R.: End-to-end neural ad-hoc ranking with kernel pooling. In: Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval. pp. 55–64 (2017)
16. Zamani, H., Dehghani, M., Croft, W.B., Learned-Miller, E., Kamps, J.: From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 497–506 (2018)