Quality-Aware Argument Re-Ranking for Comparative Questions

Notebook for the Touché Lab on Argument Retrieval at CLEF 2022

Niclas Arnhold, Philipp Rösner and Tobias Xylander

Martin-Luther-Universität Halle-Wittenberg

Abstract

In this paper, we describe the team's Asuna participation in the Touché shared task on Argument Retrieval for Comparative Questions. We submit one run to the task. Apart from the BM25F retrieval algorithm we use concepts like tokenization, lemmatization, summarization, query expansion and machine learning approaches such as DistilBERT and SVM in our approach. At the core of our approach is re-ranking documents based on their argument quality.

Keywords

Touché 2022, Comparative queries, Argument retrieval, Argument quality

1. Introduction

The majority of comparative search engine question-like queries (e.g. "Should I major in Philosophy or Psychology?") require retrieved web documents to include relevant and high-quality arguments for and against the to-be-compared options [1]. This requires a retrieval system to account not only for relevance but also for argument quality and stance [2].

In this paper, we describe our team's participation in the Touché shared task on Argument Retrieval for Comparative Questions [3]. We propose an argument quality-aware re-ranking approach to address the aforementioned challenges in argument retrieval. Our approach consists of: (1) a preprocessing pipeline that builds the index, (2) a search pipeline that for each topic does an initial search followed by tokenization, lemmatization and query expansion, in order to construct expanded queries, and (3) a re-ranker which processes the final set of documents per topic.

Our proposed approach consists of two pipelines: a preprocessing pipeline and a search pipeline.
The preprocessing pipeline extracts document-specific characteristics and builds the index.
The search pipeline processes topics and uses a BM25F search approach which uses previously calculated document characteristics as BM25F fields.

CLEF'22: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
niclas.arnhold@student.uni-halle.de (N. Arnhold); philipp.roesner@student.uni-halle.de (P. Rösner); tobias.xylander@student.uni-halle.de (T. Xylander)
https://gitlab.informatik.uni-halle.de/amhek/ir-asuna (P. Rösner)

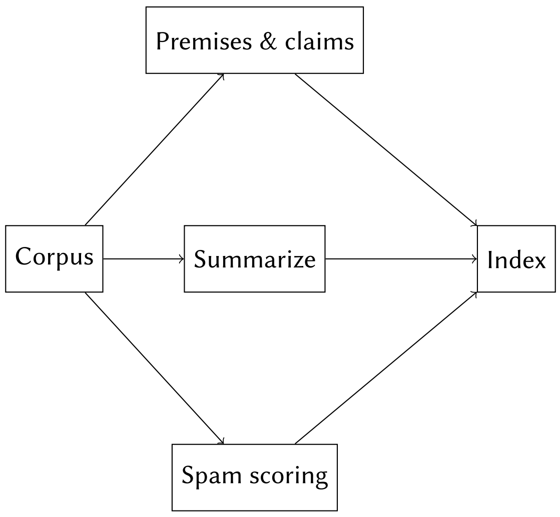

Figure 1: Structure of the preprocessing pipeline

2. Preprocessing pipeline

Our team's approach builds an extended index for our search engine in order to use the extended attributes in our re-ranker and machine learning models.

For each document of the corpus we calculate an extractive summary and a spam score as well as premises and claims.
These steps make it very easy for our search pipeline afterwards to retrieve these attributes from our index for arbitrary documents.

2.1. Premises & Claims

In our re-ranking process we take into consideration the count of premises and claims per document, as proposed by the group "Rayla" of Alhamzeh et. al [4] of the CLEF 2021 Touché Lab second shared task.

For that we use, like the group "Rayla" the TARGER project, a neural argument mining program [5]. For each document we sent a request to the TARGER-API and add the resulting premises and claims as new fields.

We do not consider main claims and main premises, as our re-ranker only considers the respective count of premises and claims and their argument quality.

2.2. Summarization

Summaries can be used in BM25F as fields, so as part of our preprocess pipeline we calculate extractive summaries for each document and store them in the index.

Extractive summaries rank the words in a document by relevance in the document and create one (or more) document-representing sentences which only use words from the given document.

In contrast, abstractive summaries attempt to guess the meaning of the document and create a novel description of the document via machine learning approaches. As abstractive summaries can be quite expensive to calculate for the whole corpus, we focus on extractive summaries in our team's approach.

For extractive summaries we used the LexRank graph-based method supplied by the `sumy` Python library.

2.3. Spam scores

For calculating spam scores for documents we used the Webis corpus-waterloo-spam-cw12 data set. By merging this data set and the given corpus with Pandas[6], we receive spam scores for most of the passages of the corpus.

Later, in the re-ranking process, we use these scores to re-rank documents partially based on their respective spam score.

2.4. Building the index

Via the Pyserini[7] indexing mechanism we build an index which includes the original documents and all previously calculated document characteristics.

This enables us to query the index with BM25F and fields for extractive summaries, premises, claims and cleaned documents.

Under the hood of Pyserini the Lucene document generator and indexing process is used.

3. Search pipeline

At the beginning of the search pipeline we retrieve an initial ranking of 40 documents given the query. We then continue with tokenizing and lemmatizing the contents of those documents. With these processed contents we then perform *Latent Dirichlet Allocation* to extract the topics that belong to those documents.

Taking those topics and their related word lists as well as the initial query we then create a number of extended queries that will have some tokens replaced with synonyms after. These new queries we afterwards combine into a single new query again with the query builder.

After creating this new query it is used to perform another BM25F search. A pre-trained *DistilBERT* is then used to judge the Argument Quality of every document. Stance Detection of the documents through a *DistilBERT* model is the second to last step in the pipeline before a re-ranking is performed on the documents according to a number of factors including high argument quality, low spam likelihood (that was gained from the initial search) and so on.

3.1. Initial search for related documents

After the user has entered their query we first retrieve the top 40 relevant documents via the *pyserini Simplesearcher.* As fields for the Simplesearcher we use the summary weights as well

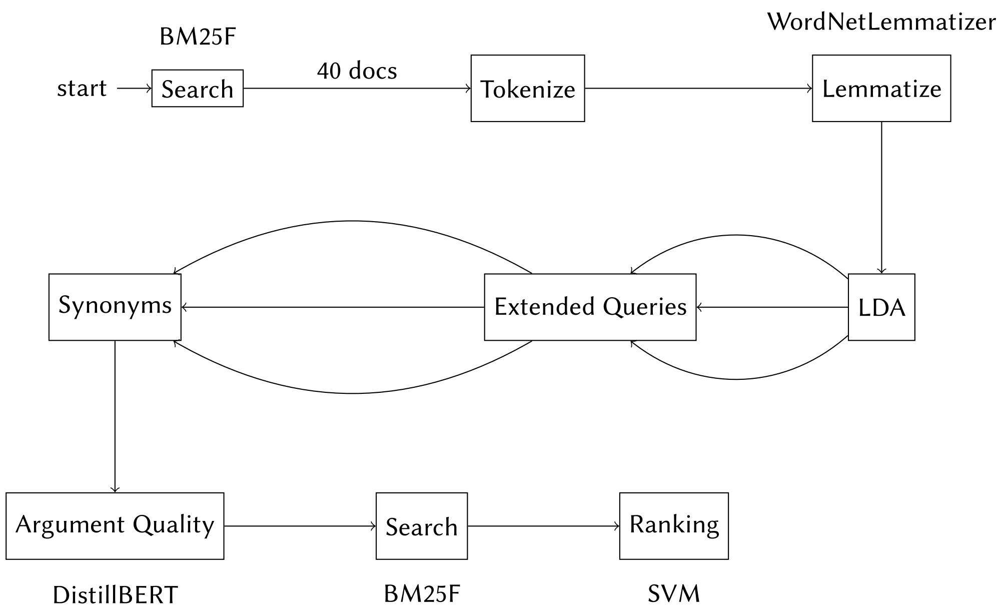

Figure 2: Our search pipeline for answering comparative questions

as the weights of the premises and claims of the documents.

All the following steps aim to improve on the initially retrieved ranking.

3.2. Tokenizing & Lemmatizing

From the retrieved documents the contents are tokenized and the resulting number of tokens in each document's contents is saved. To achieve this we use 2 different tokenization functions: One function that first removes any punctuation then uses the gensim.utils library to tokenize the individual words followed by a function that uses the nltk.stem.WordNetLemmatizer aswell as the nltk.corpus.stopwords list to turn the tokens into their basic form and remove any common english stopwords. As a result a list of all word tokens of the document is returned.

Our other tokenization function only removes punctuation and uses *nltk.sent_tokenize* to return a list of individual sentences of the given document (nltk and all nltk methods that were used in this project are described in [8]).

Also the number of sentences together with the amount of premises and claims in a retrieved document are added to a *docs_list* array.

3.3. Using LDA to build extended queries

At times the user might enter a word in their query that is not the most commonly used word of the overall topic they want a search result about to fulfil their information need. On such occasions it is possible to achieve an improvement of the query result by also taking into account the most common terms the entered query most likely belongs to.

In order to determine the most likely topic the query belongs to and all the words belonging to each topic existing in the 30 previously returned documents we use *Latent Dirichlet Allocation*[9] (short: LDA; as described by Blei et al [9]). As a basis for LDA we chose the dictionary of *Gensim Corpus* and input the tokens of the query. As the number of words belonging to a topic we decided to choose 5 and as the number of most likely topics LDA should consider we have chosen 3.

From this we gain a number of topics. Out of this list of topics we extract a list of words without scores belonging to the same topic as any word in the query to form the expanded queries. Finally we collect all the word tokens occurring in each of these extended queries.

3.4. Refining results with synonyms

The redundant words found in the previous step are then replaced by some of their synonyms found via *src.search.synonyms*. For each of those synonym replaced queries we search for the top 30 documents including their score. To keep the importance of a document occurring in different extended queries we add a third of the score from all of the same duplicate documents together. After all these scores are computed the documents are reranked by their new score.

3.5. Argument Quality

To formulate justified answers for a comparative question it is substantial to find good arguments that support or oppose the objects of interest. For the CLEF 2021 Touché Lab second shared task the group "Rayla" of Alhamzeh et al. [4] used a network architecture, namely *DistilBERT*, proposed by Sanh et al. [10] to extract argumentative sentences on every document. They used the ratio of argumentative sentences to all sentences of the document as one of many features for reranking the results. In their work they experimented with different BERT-based architectures and found out that DistilBERT has a comparable effectiveness to other models for this task which is in accordance with the original work by Sanh et al. [10]. Team Rayla decided to use DistilBERT because of the better efficiency in terms of model size and running time. Overall they achieved good results for the Touché 2021 task.

Because of these insights we decided to use *DistilBERT* in our work too but instead of argument extraction the model is applied to predict argument quality of the premises and claims extracted by the TARGER-API [5]. For that task we use a pre-trained DistilBERT model with an additional linear layer and trained it on a regression task. We trained the model on the *Webis-ArgQuality-20* data set proposed by Gienapp et al. [11] which was split into 80% train set, 10% validation set and 10% test set. Among other data the data set contains premises for 20 controversial topics and different quality scores for topic-premise-pairs. Furthermore the data set contains query formulations for the topics. As input the "long query" together with a corresponding "premise" was passed to the model. The model was fine-tuned to predict "combined quality" scores of the data set which were normalised to the interval $[-1, 1]$ where a higher score means better quality of the argument. For implementation we used *DistilBertForSequenceClassification* and the *Trainer API* of *Hugging Face* (huggingface.co).

After the model was trained we use it in our pipeline to predict argument quality scores for re-ranking. Therefore a quality score for every premise and every claim extracted from a document by TARGER [5] is predicted by passing the query and premise or claim to the model. Then the scores for one document are summed up so that a document with a lot of arguments and good arguments has a higher score than a model with few arguments or arguments with minor score. If no claim and no premise are found in the document the score is set to -2.0. We suppose that this strategy extends the idea of team Rayla [4] because not only the frequency of the arguments in the documents is payed attention to but also the quality of these arguments.

For evaluation the model achieved the scores shown in table 1 which were compared to a baseline model which always predicts the mean score of the training set.

Table 1
Results for Argument Quality prediction; score: Mean-Squared-Error

Set	Training	Validation	Test
Baseline	0.2982	0.2857	0.2640
DistilBERT	0.0614	0.06463	0.1272

3.6. Stance Detection

Besides the quality of the arguments per se one can be interested in the stance of them. Regarding comparative questions a text can support the first object or the second object, is neutral towards the objects or has no stance. In our opinion a text having no stance could be less relevant for answering a comparative question because it does not need to respond to the question. On the other side a text having a stance towards one of the objects or both objects has to be related to the objects and therefore to the question.

Because of that we decided to train a model for stance detection and use the predicted stance as a feature for reranking. Regarding to section 3.5 we decided to use *DistilBERT* [10] for stance detection. For training we used the *Webis-Stance-Dataset* from Bondarenko et al. [2] which contains 956 questions, answers and corresponding stances of the answers. There are 4 different stance labels:

0	No stance
1	Neutral
2	Pro first object
3	Pro second object

We split the data stratified into 80% train set, 10% validation set and %10 test set. For training we used a approach similiar to the one Bondarenko et al. [2] achieved good results with. First we finetuned the pretrained masked language model by masking the objects in the answers. For masking we used the list of position of both object mentions in the answer provided by the dataset. As input the model received the question and masked answer. After finetuning the masked language model a linear layer was added to the end of the model and than the model was finetuned for stance detection using the same data and input. For implementation we used *DistilBertForSequenceClassification* and the *Trainer API* of *Hugging Face* (huggingface.co).

In our pipeline we used the trained model for predicting the stances using the query and the document passage content.

In our experiments the model achieved the results in table 2.

Table 2
Results for Stance Detection prediction

Set	Training	Validation	Test
Perplexity	4.04	4.21	4.16
Accuracy	0.4437	0.4583	0.2917
Micro-F_1	0.4437	0.4583	0.2917
Macro-F_1	0.3093	0.30369	0.1763

3.7. Re-ranking

In their conclusion Team Rayla [4] mentioned that for future work they plan to use a machine learning model to learn the re-ranking of the retrieved documents based on extracted features. In our work we want to try this approach. For that we use a *Random Forest*. As input the Random Forest receives the following features:

- BM25f-score
- number of times the document was retrieved
- number of tokens in document
- number of sentences in document
- number of premises in document
- number of claims in document
- waterloo spam-scores [12]
- predicted argument quality
- predicted stance

For training we used the 100 topics and relevance labels from *Touché 2020*[13] and *Touché 2021*[14]. The relevance labels are 0, 1 and 2 where 0 means not relevant and 3 means 2 highly relevant.

First the query was processed by our retrieval pipeline which outputs the retrieved documents and corresponding features for every document. Then the retrieved documents were merged with the annotations from the relevance labels data set to get annotated features which were used to train the Random Forest. Every passage was given the *doc_id* of the whole document for training. From a practical point of view giving every passage the same relevance is incorrect but for our approach it is essential to be able to merge the relevance scores with the retrieved passages to get annotated data.

The Random Forest predicts the relevance "class" of the document. The predicted relevance class is multiplied with the prediction probability to get scores with which the documents are re-ranked from highest to lowest score. We suppose that this approach helps to have truly relevant documents at high ranks because the documents with a high probability or confidence for label 2 are ranked to the first places.

The re-ranker was implemented by using the *RandomForestClassifier* provided by *scikit-learn* [15]. For our experiments we had 1747 samples which were split into 80% train set, 10% validation set and 10% test set. The Random Forest achieved the results in table 3.

Table 3
Results for Random Forest relevance classification

Set	Training	Validation	Test
Micro-F_1	0.5877	0.5429	0.4857
Macro-F_1	0.4886	0.4334	0.3657

4. Results

In the following section we present and discuss our results for 3 different topics.

In 4 you can see the content, the argument quality score, the stance classification and the final score for the second topic of the CLEF 2022 Touché Lab second shared task. In table 5 and table 6 you can see the results for topic 3 and topic 9, respectively.

Table 4
Results for topic Nr. 2: Which is better, a laptop or a desktop?

content	arg_qual	stance	final_score
Also should mention both laptop.. & desktop	-3.05	PRO FIRST	1.34706
(The "desktop" category includes..	-0.56	PRO FIRST	1.33970
Another method, which is optimised for..	-6.11	PRO FIRST	1.26536
Which Is Best for School: Laptop or Desktop?..	-4.07	PRO FIRST	1.2526
Well, wonder no more. It really comes down..	-1.02	PRO FIRST	1.24925

The document contents seem to represent, what the comparative question is about. Though, the argument quality score is quite different per document. It's interesting that most of the retrieved documents classify with a PRO FIRST stance.
Judging the final scores, we can deduce that our retrieval mechanism can find many documents with similar scores, therefore it finds many documents answering the question.
For topic 3, most of the retrieved documents classify as PRO SECOND stance. By the final score

Table 5
Results for topic Nr. 3: Which is better, Canon or Nikon?

content	arg_qual	stance	final_score
If this sounds bad, it really isn't because..	-6.28	PRO SECOND	1.5444
Then, in the 1950s, Nikon became the 35mm..	-2.018151	PRO SECOND	0.6433
The great DSLR shootout noise reduction test..	-5.23	PRO SECOND	0.6279
Review Canon PowerShot G1 X Review Fujifilm..	-2.85	PRO SECOND	0.615
Nikon D3S vs Canon EOS 1D Mark IV: overview..	-4.18	PRO SECOND	0.6134

we can see that the first document is seen much more relevant than all other documents, which means that our retrieval mechanism has issues finding relevant documents for the question.
Similar to topic 3, we can see for topic 9 that the first document has a much higher score than all other documents. The stances strongly tend to PRO SECOND and the argument quality scores are mixed.

Table 6
Results for topic Nr. 9: Why is Linux better than Windows?

content	arg_qual	stance	final_score
Many of these companies offer both Windows..	-4.61	PRO SECOND	1.6109
(...) Extracto del documento de windows linux..	-1.85	PRO SECOND	0.6560
Check our hosting FAQs SIGNUP..	-1.54	PRO SECOND	0.6234
Free Pascal is really quite nice, but..	-3.08	PRO SECOND	0.6173
Consider: For Linux software developers..	-4.93	PRO SECOND	0.6165

Judging by the 4th result, even documents with seemingly no relation to the actual question are retrieved.

In general, we observe that quite a few spam-documents managed to get into the results. Overall though, the retrieved documents mostly relate to the question.

References

[1] A. Bondarenko, P. Braslavski, M. Völske, R. Aly, M. Fröbe, A. Panchenko, C. Biemann, B. Stein, M. Hagen, Comparative web search questions, in: J. Caverlee, X. B. Hu, M. Lalmas, W. Wang (Eds.), 13th ACM International Conference on Web Search and Data Mining (WSDM 2020), ACM, 2020, pp. 52–60. URL: https://dl.acm.org/doi/abs/10.1145/3336191.3371848.

[2] A. Bondarenko, Y. Ajjour, V. Dittmar, N. Homann, P. Braslavski, M. Hagen, Towards understanding and answering comparative questions, in: K. S. Candan, H. Liu, L. Akoglu, X. L. Dong, J. Tang (Eds.), 15th ACM International Conference on Web Search and Data Mining (WSDM 2022), ACM, 2022, pp. 66–74. URL: https://dl.acm.org/doi/10.1145/3488560.3498534. doi:10.1145/3488560.3498534.

[3] A. Bondarenko, M. Fröbe, J. Kiesel, S. Syed, T. Gurcke, M. Beloucif, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of touché 2022: Argument retrieval, in: M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg, V. Setty (Eds.), Advances in Information Retrieval. 44th European Conference on IR Research (ECIR 2022), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2022.

[4] A. Alhamzeh, M. Bouhaouel, E. Egyed-Zsigmond, J. Mitrović, Distilbert-based argumentation retrieval for answering comparative questions, Working Notes of CLEF (2021).

[5] A. Chernodub, O. Oliynyk, P. Heidenreich, A. Bondarenko, M. Hagen, C. Biemann, A. Panchenko, TARGER: Neural argument mining at your fingertips, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Florence, Italy, 2019, pp. 195–200. URL: https://aclanthology.org/P19-3031. doi:10.18653/v1/P19-3031.

[6] Wes McKinney, Data Structures for Statistical Computing in Python, in: Stéfan van der Walt, Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 56 – 61. doi:10.25080/Majora-92bf1922-00a.

[7] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, R. Nogueira, Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations,

in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 2356–2362. URL: https://doi.org/10.1145/3404835.3463238. doi:10.1145/3404835.3463238.

[8] E. L. Bird, Steven, E. Klein, Natural Language Processing with Python, O'Reilly Media Inc., 2009.

[9] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.

[10] 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).

[11] L. Gienapp, B. Stein, M. Hagen, M. Potthast, Efficient pairwise annotation of argument quality, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 5772–5781.

[12] G. V. Cormack, M. D. Smucker, C. L. Clarke, Efficient and effective spam filtering and re-ranking for large web datasets, Information retrieval 14 (2011) 441–465.

[13] A. Bondarenko, M. Fröbe, M. Beloucif, L. Gienapp, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2020: Argument Retrieval, in: A. Arampatzis, E. Kanoulas, T. Tsikrika, S. Vrochidis, H. Joho, C. Lioma, C. Eickhoff, A. Névéol, L. Cappellato, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. 11th International Conference of the CLEF Association (CLEF 2020), volume 12260 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg New York, 2020, pp. 384–395. URL: https://link.springer.com/chapter/10.1007/978-3-030-58219-7_26. doi:10.1007/978-3-030-58219-7_26.

[14] 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, in: K. Candan, B. Ionescu, L. Goeuriot, H. Müller, A. Joly, M. Maistro, F. Piroi, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. 12th International Conference of the CLEF Association (CLEF 2021), volume 12880 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg New York, 2021, pp. 450–467. URL: https://link.springer.com/chapter/10.1007/978-3-030-85251-1_28. doi:10.1007/978-3-030-85251-1_28.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.