SEUPD@CLEF: Team Gamora on Argument Retrieval for Controversial Questions

Notebook for the Touché Lab on Argument Retrieval at CLEF 2022

Alessandro Benetti[1], Michele De Togni[1], Giovanni Foti[1], Ralton Lacini[1], Andrea Matteazzi[1], Enrico Sgarbossa[1] and Nicola Ferro[1]

[1]*University of Padua, Italy*

Abstract

This paper is the report of the work we have done for Argument Retrieval CLEF 2022 Touché Task 1 as Gamora team. Argument Retrieval CLEF 2022 Touché Task 1 focuses on the problem of retrieving and ranking relevant pairs of sentences from a collection of arguments for a given controversial questions. After an analysis of the structure and the possibilities of manipulating document data, we concentrated our work on query management, applying methods of query expansion, reduction, query boost to conclude with satisfactory results with the last two solutions listed. We came up with two systems, one based on a two-stage operation, double index and double search, and one based on sentence pair ranking based on its argumentative quality, assessed by a machine learning model.

Keywords

Information retrieval, Controversial questions, Argument retrieval, Query reduction, Query boost, Argument quality

1. Introduction

In this paper we describe our solution, as team Gamora, for the CLEF 2022 Touché [1] Task 1 [2]. According to Touché Task 1, our goal is to realize an information retrieval system that can support people searching arguments to be used in conversations. We developed a Java based system that takes as input a list of more than 360.000 documents from args.me corpus dataset [3] and it returns a ranked list with the best pairs of sentences. As required, the two sentences in this pair may come from two different arguments and they are both related to the right topic in order to have a global vision of pros and cons or some information about a certain discussion point. We decided to develop simultaneously two different solutions, in order to achieve different results and compare them. In the following we refer to these two as *S1* and *S2*. All the differences between them are described in Section 3.

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

alessandro.benetti.1@studenti.unipd.it (A. Benetti); michele.detogni@studenti.unipd.it (M. De Togni); giovanni.foti@studenti.unipd.it (G. Foti); ralton.lacini@studenti.unipd.it (R. Lacini); andrea.matteazzi.2@studenti.unipd.it (A. Matteazzi); enrico.sgarbossa.1@studenti.unipd.it (E. Sgarbossa); ferro@dei.unipd.it (N. Ferro)
http://www.dei.unipd.it/~ferro/ (N. Ferro)
0000-0001-9219-6239 (N. Ferro)

The paper is organized as follows: Section 3 describes our approach; Section 4 explains our experimental setup; Section 5 discusses our main findings; finally, Section 7 draws some conclusions and outlooks for future work.

2. Related Work

The task of retrieving arguments for controversial questions involves three main stages: the retrieval of suitable arguments, the augmentation of these results via query expansion and a reranking step of the retrieved arguments.
For the retrieval of the arguments existing frameworks employ statistical language models such as Dirichlet LM or ranking functions like BM25 [4]. For the augmentation of the retrieved results, query expansion components in past experiments employ WordNet based systems [5] and neural models such as BERT [6] alike.
For the ranking and reranking of the arguments it has been proven effective to leverage both the relevance and the quality of an argument with respect to the query [4]. The Webis-ArgQuality-20 Corpus [7] contains a collection of arguments from the argsme corpus, each with scores for its rhetorical, logical and dialectical quality along with a combined score for these three components. Gienapp [8] used a Support Vector Regressor trained on this dataset's combined quality score to assess the quality of an argument and used this measure along with topical clustering to rerank the documents. In a similar fashion, Green et al.[6] used BERT-based models trained on the Webis-ArgQuality-20 Corpus.

3. Methodology

Our system is composed by 3 sections: the first one takes care of doing the parse of all the documents, the second section create the index file and there is the third part, that is the search part. Both the solutions S1 and S2 have that structure but they differ in the methods that have been used. For each section, we are going to discuss the differences between the two approaches.

3.1. Parsing

Considering the fact that we have a single file .csv that collects all documents, we want to avoid to load in ram the entire file to parse it. The solution was to use a iterator to read line by line and parse it.

When parsing documents, what we realised is that the *premises* field contains a number of sentences taken from the *source-text* field belonging to the *context*. These, in turn, are then subdivided by the full stop to create a vector of sentences and the sentence contained within the *conclusion* field is also added to the latter.
Moreover, compared to last year, this year we have included in the dataset the previously mentioned *context* in which there are a series of data such as the title of the topic, all the source code, the publication date, the source, the author and more.
Some of these data could be used to refine the search, for example based on the goodness of the

source/author or comparing the date of publication and the date on which the query is made, but it is a deepening that for complexity and time we have not pursued.

We manually inspected the corpus to gather information about its structure. Documents are sourced from four different debate discussion portals: “idebate”, “debate.org”, “debatepedia”, “canadian-parliament”. The structure of the documents from these sources is the same for the id, conclusion, premises, sentences fields but differ in the subfields of the field context, where, among other differences, canadian-parliament sourced documents don’t provide a “discussionTitle” subfield but provide a “topic” subfield instead. Also, out of the 365’308 documents provided in the corpus, 998 ids of these documents are shared with other documents, with some ids getting repeated in up to 8 documents. This results in a total of 2’214 documents with non-unique ids. Since our task is to retrieve a pair of sentences, we also analyzed the sentences of the corpus. In total, the corpus contains 6’123’792 sentences across all the documents. Since as noted above, some of the documents are non-unique, we counted the sentences with unique document id and unique sentence text bringing the total number of sentences to 5’337’409, meaning that around 13% of the sentences in the corpus are either duplicates or from a duplicate document..

3.2. Indexing

After parsing the documents, we have created a class *ArgsParsed* in which we have inserted a series of functions that return values in string form for the various main fields that we have decided to index.
To the Lucene document we decided to add:

- the id of the document
- the *discussionTitle*, if present, otherwise the *topic*
- the text of the *sentence* for each sentences / premises text
- the *conclusion*
- the *stance*

In the S1 branch we decided to index the text of sentences concatenated because it is already filtered of some non-useful words and in the S2 directly the premises field.

In S1 we used a custom Analyzer to which we have added the standard tokenizer, a token filter of the KStem type and another lower case. On the other hand, in S2 we experimented with different combinations of filters, thanks to the fact that we created a system that allows us to run it several times and update the parameters at each repetition. In the end, the best results in terms of *nDCG* were achieved using stopwords filter, length filter, kstemmer and the standard tokenizer.

We provide two tables with the results of trec_eval. The Table 1 refer to the relevance results, while the Table 2 refers to the quality results.

3.3. Searching

At this stage, as with document indexing, we parsed the file containing the queries. The document contained 50 questions, each with four tags: num, topic, description and narrative. We only considered the identifier and the title (topic) to carry out the search.

At this point we added the queries to Lucene's BooleanQueryBuilder, carried out the search and obtained a first run but since it was the first it became the yardstick for our subsequent attempts, listed later in a dedicated section Results and Discussion, Section 5.

The file that Lucene returns from the run, however, does not conform to the task requested by Touché because we have a file with the results of the best documents associated with the queries ordered by score, but what we need to obtain is a file to which, for each query, the two best argumentative sentences are associated and these two can also belong to different documents, as long as they do not contradict each other. In order to better understand the scenario of argumentative sentences, we consulted the article about *Identifying Argumentative Questions in Web Search Logs* [9].

Here, we again divide the two approaches: in S1 we decided to create a new index from the sentences alone based on the results of the first run and then perform a new search on the latter, while in S2 we applied the second idea, i.e., again based on the results of the first run, create all possible combinations of pairs of sentences and then perform a relevance analysis and return the highest scoring pairs in the correct format, thus avoiding a new index and a subsequent search.

3.3.1. S1 solution

Limiting the maximum number of documents to be retrieved for each topic, we considered a sample of sentences from all these documents to create the second index, discarding sentences that are too short assuming that a sentence, in order to be argumentative, must contain at least a minimum number of words. We also decided to discard those sentences that are too long because we noticed that very long sentences, i.e. that are composed of more than 10000 characters, always contains strange characters such as base64 images or binary strings. As a first attempt, this second search was carried out by considering the topic title without any modification in order to verify its correct functioning and obtain a starting point to compare future alternative solutions, the first of which was based on query expansion made by WordNet. By manually analyzing some sentences obtained for the various topics, it didn't work well neither adding 1-2 synonyms to the topic title nor replacing original words with synonyms, probably because the quality of the synonyms found by WordNet was often poor as it caused the topic to drift. We then also tried a trained online model (explained in a separate section 3.4) to get more relevant synonyms but again no sign of major improvement.

Heuristics

Query reduction So, we tried another solution, based on a manual query reduction in order to detect words that we wanted to exclude from the query (e.g. articles, adverbs), keeping only those 2-6 main terms of the query title. This version was born out of an attempt to give weight to certain terms in the query in order to make them more important.

Query boosting This feature consists in splitting the original topic in the list of its words and for each word create a *TermQuery*. This is later wrapped in a *BoostQuery* object in which we apply a boost, consisting in a numerical value. Every *BoostQuery* object is then added to a *BooleanQuery* in order to build the final query. This technique is used in combination with *Query decorators*, explained below, in order to give more weight to original topic words, rather than the *decorators*. The results seemed satisfactory, we read a tens of sentences for each topic in order to be confident we were heading in the right direction.

Query decorators This feature consists in attaching a number of keywords, called *decorators*, to the original query. An example of such used keywords is: *consequence, conclusion, reason, proof, fact, argument, therefore, show, result etc.* These, when attached to the original query, will increase the chance that the retrieved sentence will be more argumentative.

An example of how a query changed in these steps is given below:

1. Should teachers get tenure?
2. teachers tenure
3. teachers*4 tenure*4 consequence conclusion reason proof evidence therefore show

Once again, we improved on the previous results, obtaining sentences of a reasonable length, inherent to the topic and in agreement with each other.

For this kind of task attempts were made with both LMDirichlet similarity and BM25 similarity and only with the last one we obtain logically correlated and relevant sentences. With LMDirichlet many of the sentences with high scores were completely out of context.

3.3.2. S2 solution

In this approach we decided to choose the best sentence pairs using an argument quality regression model that evaluates the argumentative quality of each sentence obtained as concatenation of the sentences in the pair.
The sentence pairs to be evaluated are gathered starting from the best documents retrieved in the run file produced during a Lucene-based search on for a list of queries (or topics as in our case). For each topic, the ids of the best documents are gathered, and for each document the sentences with the best quality are kept in a list shared between documents for the same topic. Then all possible combinations of rank two between sentences in this list are computed and the concatenation of the two sentences is evaluated according to the quality regression model. The final score of a pair is obtained as weighted average between the predicted quality of the composed sentence and the mean of the score of the two documents the original sentences belong to.
To evaluate the quality of the sentences we trained a Support Vector Regressor model on the "Webis Argument Quality Corpus 2020 (Webis-ArgQuality-20)" dataset [7] in a similar way to Gienapp[8] and Bundesmann et al.[10]. We created two models that differ between them only in the vectorizer model. One uses a TFIDF Vectorizer which considers only frequency of tokens, the second one is an encoder based on the SBERT architecture [11] which encodes also relevant semantic information.

3.4. Model for query expansion

In order to improve the search, we have tried to enhance the queries through the expansion of important keywords with synonyms. In particular, in order to find the more important terms in the queries, we have exploited *Part of Speech (PoS)* [12] for selecting just the keywords corresponding to meaningful tags. Then, once the more meaningful terms have been extracted, for each of them we have found the top two closest words to add to the query, in such a way to avoid the addition of derived words with respect to the corresponding word on the query. The model used to perform such a task is GloVE [13], which is an unsupervised learning algorithm for obtaining vector representations for words. Its training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. In particular, we found a family of pretrained models from [14] and after some experiments, it turned out that the best model for our documents collection was "glove-twitter-200", which is based on 2B tweets, 27B tokens, 1.2M vocab, all uncased. The interesting feature of this family of pretrained model is the implementation of the method "most_similar", which returns, given a word, the top ten closest words in vector space sense with a score of closeness for each of them.

4. Experimental Setup

In order to ensure compatibility and repeatability of the experiments, the used paradigm was the Cranfield one. This approach is based on three main components:

- Documents collection: which contains all the documents to index (the information to retrieve);
- Topics: which are the surrogates for the user query;
- Relevance Judgment: this is written in the qres.txt file and contains the ground-truth of the ranked lists obtained querying for the topics on the documents collection;

Documents collection
The documents collection was provided by CLEF and contains 365408 documents about various topics. The collection is an 9 GB file in csv format.

Topics
Also the topic file was provided by CLEF and it contains 50 different topics.

Relevance Judgment
In order to test the different analyzers performance, the qrels were used, so the output ranked lists were computed also from the 50 topics. Then for the submission the 2021 topics list has been used.

For both solutions *S1* and *S2*, our work is based on the following experimental setups:

- Used collections: *Apache Maven, Lucene*;
- Java JDK version 18;

- Version control system: *git*;
- Repository: https://bitbucket.org/upd-dei-stud-prj/seupd2122-javacafe/src/master
- During the develop and the experimentation we have used our own computer and in the end we have run our code using Tira;

We provide three different results for S1 solution by manually enabling/disabling certain features, resulting in three different runs.

The first one, called *"StandardDoubleIndex"* consists in a standard Java/Lucene pipeline carried out by first searching the topic title in the first index and then on the second index, as described here. 3.3.1

The second pipeline, called *"HeuristicsDoubleIndex"* follows the same path as the first one, but the following heuristics are applied:

- *Query reduction*;
- *Query decoration*;
- *Query boosting*;

As a third attempt, called *"HeuristicsOnlyQueryReductionDoubleIndex"* only the query reduction technique is applied.

In our second solution *S2* we created a class used for creation and searching on the index. The input parameter for the indexer and searcher objects are given as command line arguments.

5. Results and Discussion

In this section we provide graphical and numerical results about the experiments we conducted during the project development.
The two different solutions *S1* and *S2* had in common the first run, which is the one on the documents indexing, so the following graph is very similar for both.

In the following graph 1 we report the values of the interpolated precision against recall analysis of the three different method we have used which differ by their similarity: *kstemstopengpos-bm25*, *kstemstopengpos-lmdir* and *kstemstopengpos-multi*. The graph shows that Lucene's MultiSimilarity gave us better result, in particular comparing to BM25Similarity. Indeed, it is possible to see a bigger Area Under the Curve (AUC) for the curve corresponding to MultiSimilarity and this translates in a better trade-off between precision and recall. Such trade-off can be better seen in Table 1 since the F-score is the harmonic mean between precision and recall.

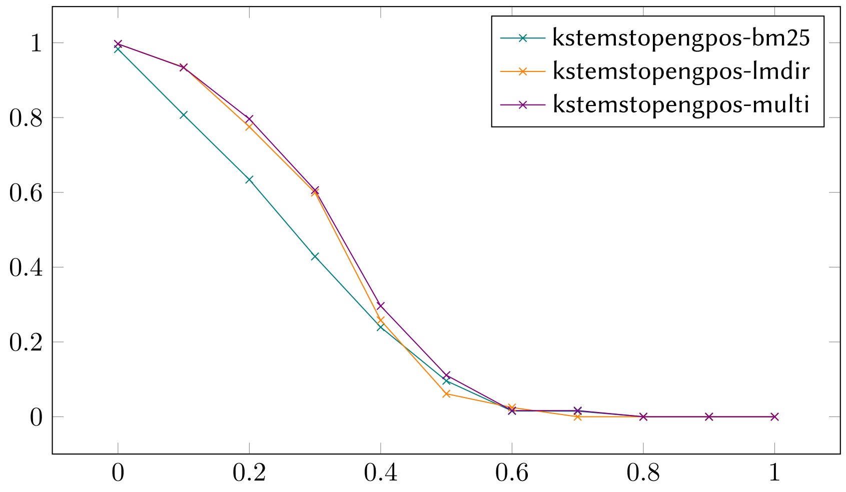


Figure 1: Plot of the Interpolated precision against recall of the three methods.

5.1. Comparison about set measures and rank based

In the following configuration, we used the stopword filter in the analyzer. The following tables report the results of trec_eval. The Table 1 refer to the relevance results, while the Table 2 refers to the quality results. Note that *Multi* value under *Similarity* column is the Lucene's MultiSimilarity, that we built by combining BM25 and LMDirichlet similarities.

Run ID	F-Score	Precision	Recall	nDCG@5	Similarity	Len. Filter	Eng. Filter	Stemmer
porterstop	**0.4152**	**0.3704**	**0.5502**	0.6542	Multi			Porter
kstemstopengpos	0.4094	0.3656	0.5426	0.6737	Multi		O	KStem
kstemstop	0.4087	0.3648	0.5421	0.6733	Multi			KStem
kstemlenstopengpos	0.4065	0.3624	0.5397	**0.6744**	Multi	O	O	KStem
kstemstoplen	0.4061	0.3620	0.5394	0.6702	Multi	O		KStem
stop	0.4009	0.3604	0.5294	0.6403	Multi			
stopengpos	0.3982	0.3580	0.5257	0.6444	Multi		O	
stoplen	0.3969	0.3572	0.5234	0.6398	Multi	O		
kstemstopengpos	0.3832	0.3400	0.5136	0.5898	BM25		O	KStem
kstemlenstopengpos	0.3827	0.3396	0.5130	0.5900	BM25	O	O	KStem
kstemstop	0.3827	0.3396	0.5130	0.5864	BM25			KStem
kstemstopengpos	0.3819	0.3424	0.5076	0.6439	LMDir		O	KStem
kstemstop	0.3815	0.3420	0.5072	0.6446	LMDir			KStem
kstemstoplen	0.3806	0.3376	0.5108	0.5925	BM25	O		KStem
porterstop	0.3804	0.3368	0.5077	0.5699	BM25			Porter
porterstop	0.3782	0.3400	0.4960	0.6105	LMDir			Porter
kstemstoplen	0.3771	0.3384	0.5004	0.6406	LMDir	O		KStem
stop	0.3758	0.3384	0.4995	0.6146	LMDir			
stop	0.3752	0.3344	0.5006	0.5877	BM25			
stopengpos	0.3751	0.3380	0.4983	0.6146	LMDir		O	
stopengpos	0.3751	0.3344	0.5004	0.5903	BM25		O	
stoplen	0.3724	0.3364	0.4909	0.6123	LMDir	O		
stoplen	0.3649	0.3248	0.4894	0.5911	BM25	O		

Table 1: Relevance table for *S2* ordered by F-Score.

Run ID	F-Score	Precision	Recall	nDCG@5	Similarity	Len. Filter	Eng. Filter	Stemmer
porterstop	**0.4315**	**0.4856**	**0.3992**	**0.8092**	Multi			Porter
kstemstopengpos	0.4277	0.4804	0.3962	0.7986	Multi		O	KStem
kstemlenstopengpos	0.4272	0.4792	0.3962	0.7957	Multi	O	O	KStem
kstemstop	0.427	0.4796	0.3956	0.8006	Multi			KStem
kstemstoplen	0.4268	0.4788	0.3957	0.7915	Multi	O		KStem
stop	0.4198	0.4728	0.3881	0.8052	Multi			
stopengpos	0.4178	0.4704	0.3864	0.8043	Multi		O	
stoplen	0.4175	0.4704	0.3859	0.8028	Multi	O		
kstemstop	0.4098	0.4612	0.3791	0.7963	LMDir			KStem
kstemstopengpos	0.4094	0.4608	0.3787	0.7979	LMDir		O	KStem
kstemstoplen	0.4026	0.4528	0.3727	0.7869	LMDir	O		KStem
stop	0.4025	0.4544	0.3717	0.7942	LMDir			
stopengpos	0.4022	0.454	0.3713	0.794	LMDir		O	
porterstop	0.4017	0.4528	0.3712	0.8056	LMDir			Porter
stoplen	0.4003	0.452	0.3696	0.7889	LMDir	O		
stop	0.3876	0.4336	0.3605	0.7164	BM25			
stopengpos	0.3874	0.4332	0.3603	0.7189	BM25		O	
kstemstop	0.3871	0.4316	0.3607	0.7125	BM25			KStem
kstemstopengpos	0.387	0.4316	0.3605	0.7153	BM25		O	KStem
kstemlenstopengpos	0.3848	0.4288	0.3586	0.7012	BM25	O	O	KStem
porterstop	0.3847	0.43	0.3578	0.7141	BM25			Porter
kstemstoplen	0.3826	0.4264	0.3566	0.7016	BM25	O		KStem
stoplen	0.379	0.4232	0.353	0.7117	BM25	O		

Table 2: Quality table for *S2* ordered by F-Score.

6. Statistical Analysis

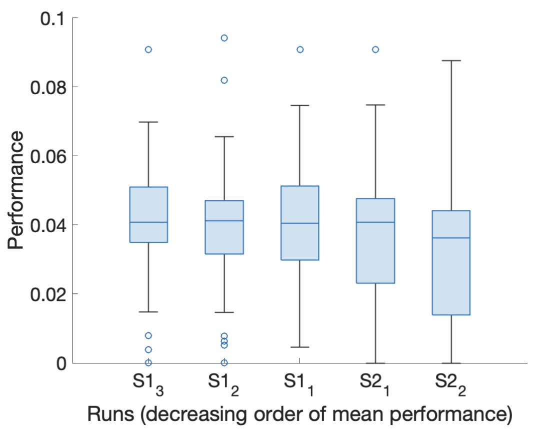
In this section we conduct a study of the performances of our systems based on the 5 runs we submitted to CLEF [1] (Table 3). The specific settings of each run are indicated in Table 4. Furthermore each of these runs also use a lowercase filter and the Krovetz Stemmer (KStem) in the Analyzer and also skip repeated sentences and sentences too long or too short. Considering the boxplots of the runs based on Average Precision (AP) Figure 2a and Normalized Discounted Cumulative Gain at 10 (nDCG@10) Figure 2b, it is possible to see how the medians are similar among all the runs. To study this phenomenon an analysis of variance (ANOVA) test with a significance of 0.05 was conducted on the nDCG@10 and Average Precision performances of the runs. Table 3a and Table 3b show that the test fails to reject the null hypothesis, hence the runs are not statistically different under the significance level we considered. Figure 3a and Figure 3b highlight how runs from the two different solutions S1 and S2 compare between them, with runs from the same solutions being similar between them and runs from different solutions being more dissimilar between them, particularly when looking at nDCG@10 performance. To confirm this analysis and also whether the runs are really different in terms of mean performance, so if the null hypothesis $H_0 : \mu_x = \mu_y$ for runs x and y holds, we conducted Student's t test for some runs groups in Table 5. From the p-values it turns out that H_0 is always failed to reject since the p-values are always higher than 0.05, hence the means of these runs are not statistically different.
Another interesting results visible from the boxplots is that the order of run's performance is approximately the same, with higher mean performance in S1 compared to S2. Performance per topic of our system is highlighted in Figure 4 which shows the nDCG@10 for each one of the 50 topics.

Run Tag	Run ID
$S1_1$	seupd2122-javacafe-gamoraHeuristicsDoubleIndex
$S1_2$	seupd2122-javacafe-gamoraStandardDoubleIndex
$S1_3$	seupd2122-javacafe-gamoraHeuristicsOnlyQueryReductionDoubleIndex
$S2_1$	seupd2122-javacafe-gamora_tfidf_kstemstopengpos_multi_YYY
$S2_2$	seupd2122-javacafe-gamora_sbert_kstemstopengpos_multi_YYY

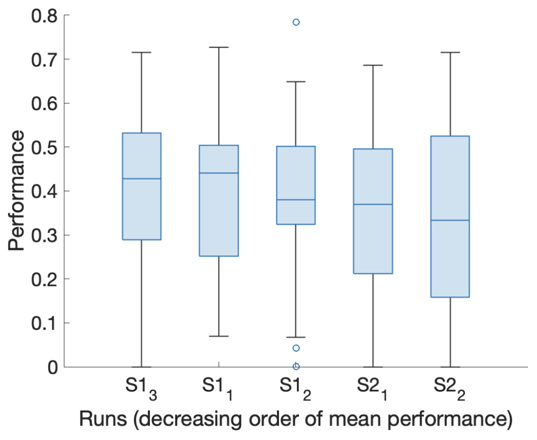
Table 3: Run Tags.

Run tag	Sol.	Sim.	Analyzer filters		Notes
$S1_1$	S1	BM25			All heuristics (query reduction decorators boosting)
$S1_2$	S1	BM25			S1 baseline (only java–lucene pipeline no heuristics)
$S1_3$	S1	BM25			Only query reduction heuristic
$S2_1$	S2	Multi	Stopword filter	English possessive filter	TF-IDF vectorizer for ML reranker
$S2_2$	S2	Multi	Stopword filter	English possessive filter	SBERT encoder for ML reranker

Table 4: Run Details.



(a) Average Precision



(b) nDCG@10

Figure 2: Boxplots of the 5 runs in decreasing order of mean performance.

Runs	P-value
$S1_3$ vs $S1_1$	0.2608
$S1_3$ vs $S2_1$	0.0680
$S2_1$ vs $S2_2$	0.8952

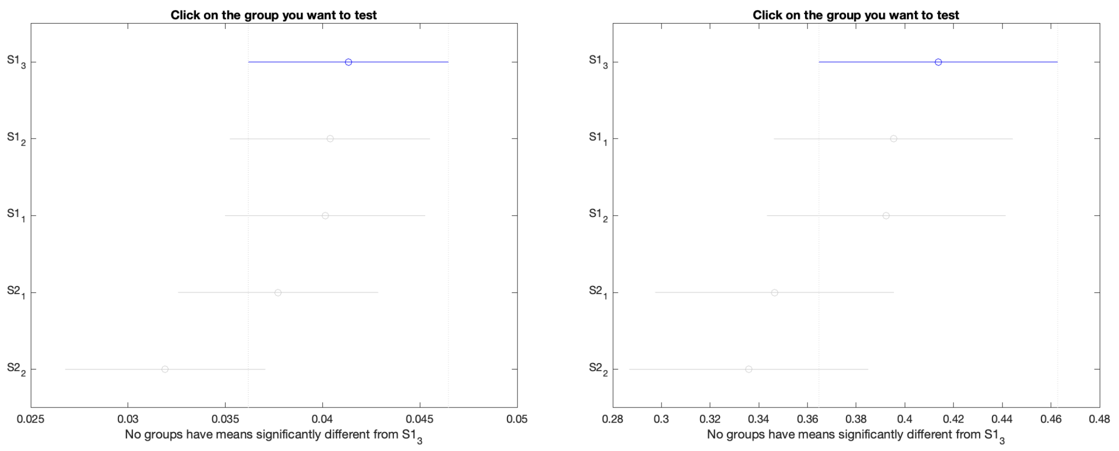
Table 5: Students t test under H_0 among pairs of runs x and y.
$H_0 : \mu_x = \mu_y, H_1 : \mu_x \neq \mu_y$

Source	SS	df	MS	F	Prob>F
Columns	0.0029	4	7.2455e-04	2.0385	0.0896
Error	0.0871	245	3.5544e-04		
Total	0.0900	249			

Table 6: Anova AP

Source	SS	df	MS	F	Prob>F
Columns	0.2271	4	0.0568	1.7565	0.1383
Error	7.9184	245	0.0323		
Total	8.1455	249			

Table 7: Anova nDCG@10



(a) Average Precision (b) nDCG@10

Figure 3: Analysis of Variance of the 5 runs.

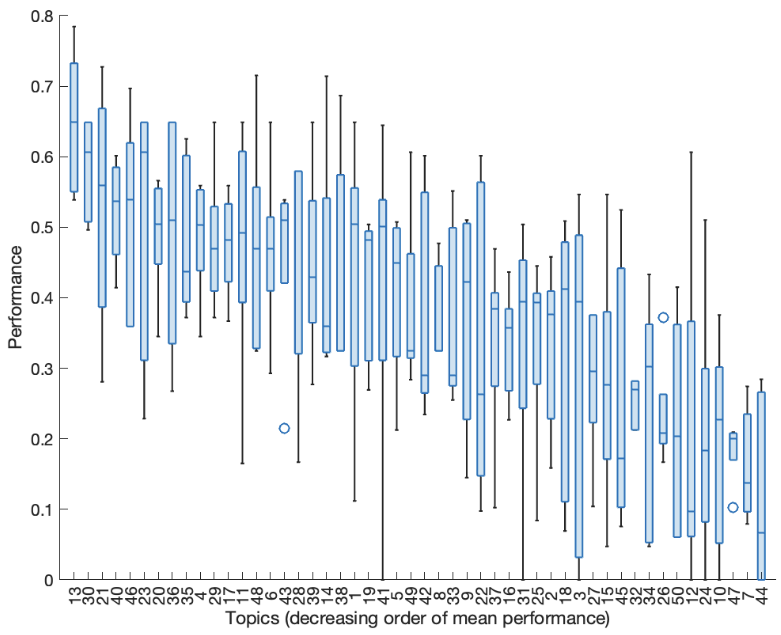


Figure 4: nDCG@10 per Topic of $S1_3$

7. Conclusions and Future Work

7.1. Discussion

Since the task of our system is, given a query, to retrieve a pair of sentences which have to be as much related as possible each other and with the query (e.g. they do not contradict each other), the goal is to find a way to evaluate the quality of two sentences as they would come from the same argumentation. Indeed, our idea is to integrate a machine learning model in order to be able to re-rank sentence pairs retrieved by a language model. To achieve this goal, datasets such as Webis-ArgQuality-20 [7] have been successfully used as training set for sentence quality evaluation models [10]. However datasets of this size limit both the performance of a model and the structure a model itself can have since more complicated models that could achieve better efficiency need more data. Thus, bigger and more diverse datasets on sentence argumentativeness and quality could lead to more expressive and precise models alike, along with the employment or creation of ad-hoc neural or natural language processing models for this task.

The diversity of the task led us not to focus too much on the results of previous years, so particular manipulations on the documents were not of interest to us, but instead we turned to operate on the queries, thus creating a kind of pseudo-relevance feedback system in the S1 approach. Staying with the latter, we notice that results are visibly improving during various attempt, but the system has one significant downside: speed. The fact that we have to create a second index immediately after performing the first search means that we cannot prepare it beforehand because we need the query, so creating the index and then performing a second search after the first one slows down the procedure considerably.

7.2. Conclusion and Future Work

In conclusion, we developed two different systems for this task, a system more focused on retrieving relevant sentences and a second system that searches for sentences pair of high argumentative quality. According to relevance judgements the first solution (S1) works better overall but the strategies employed by the second solution (S2) are close or better in performance when looking at quality and coherence judgements respectively. For this reason one step forward for our system could be merging the two strategies we propose. While the solution S1 provides a more refined set of sentences to choose from, it employs a naive strategy for the scoring of the couples it creates. In this context, the more thoughtful ranking of S2 that also tries to leverage the sentence pair quality as a sentence of its own could greatly boost the system's overall performance.

References

[1] 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: Experimental IR Meets Multilinguality, Multimodality, and Interaction. 13th International Conference of the CLEF Association (CLEF 2022), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2022, p. to appear.

[2] A. Bondarenko, S. Syed, M. Fröbe, T. Gurcke, B. Stein, J. Wachsmuth, M. Potthast, M. Hagen, Touché Task 1: Argument Retrieval for Controversial Questions, https://webis.de/events/touche-22/shared-task-1.html, 2022.

[3] 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.

[4] 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: [15], 2021, pp. 2258–2284. URL: http://ceur-ws.org/Vol-2936/#paper-205.

[5] E. Raimondi, M. Alessio, N. Levorato, A search engine system for touché argument retrieval task to answer controversial questions, in: [15], 2021, pp. 2423–2440. URL: http://ceur-ws.org/Vol-2936/#paper-217.

[6] T. Green, L. Moroldo, A. Valente, Exploring bert synonyms and quality prediction for argument retrieval, in: [15], 2021, pp. 2374–2388. URL: http://ceur-ws.org/Vol-2936/#paper-213.

[7] 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.

[8] L. Gienapp, Quality-aware argument retrieval with topical clustering, in: [15], 2021, pp. 2366–2373. URL: http://ceur-ws.org/Vol-2936/#paper-212.

[9] Y. Ajjour, P. Braslavski, A. Bondarenko, B. Stein, Identifying argumentative questions in web search logs (2022).

[10] M. Bundesmann, L. Christ, M. Richter, Creating an argument search engine for online debates., in: CLEF (Working Notes), 2020.

[11] N. Reimers, Sentence bert, 2022. URL: https://www.sbert.net.

[12] S. Bird, E. Klein, E. Loper, Natural language processing with python. o'reilly media inc., 2009. URL: https://www.nltk.org/book/ch05.html.

[13] C. D. M. Jeffrey Pennington, Richard Socher, Glove: Global vectors for word representation, 2014. URL: https://nlp.stanford.edu/pubs/glove.pdf.

[14] R. Technologies, Gensim-data, 2018. URL: https://github.com/RaRe-Technologies/gensim-data.

[15] G. Faggioli, N. Ferro, A. Joly, M. Maistro, F. Piroi (Eds.), Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum (CLEF 2021), number 2936 in CEUR Workshop Proceedings, Aachen, 2021. URL: http://ceur-ws.org/Vol-2936/.