Using BERT to retrieve relevant and argumentative sentence pairs

Nils Wenzlitschke[1], Pia Sülzle[1]

[1]*Leipzig University, 04109 Leipzig, Germany*

Abstract

In the age of the Internet, it should not be a problem to find enough information on a controversial topic within a short time to be able to form a well-founded opinion. However, finding arguments from different points of view on a particular topic is still difficult. Therefore, the Touché Shared Tasks of this and the last years are concerned with argument retrieval, that is to find relevant and qualitative arguments. In this paper, we present our results for this year's Touché Task 1. We test and combine various state-of-the-art argument retrieval methods and natural language processing to retrieve relevant sentence pairs on controversial topics. Evaluating the various combinations of our methods, we compare the different approaches. We find that using DirichletLM as retrieval model yields longer sentences than using BM25 and that pairing single sentences with the Next Sentence Prediction of BERT works better than with the sentence similarity of SBERT. We evaluate the possible combinations using precision@10 and nDCG@10. Our best retrieval system achieves a precision@10 of 0.67 and a nDCG@10 of 0.73 using sentence classification in preprocessing, DirichletLM, querying the arguments with Boolean queries using noun chunking, pairing the sentences via Next Sentence Prediction and re-ranking using a BERT-base method.

Keywords

Argument Retrieval, Natural Language Processing, BERT

1. Introduction

While facts can be looked up in seconds in modern search engines like Google[1], Bing[2] and DuckDuckGo[3], these search engines often fail to provide arguments on either side of a controversial issue. This makes it difficult to form a well-founded opinion on complex ambiguous issues. Thus, to learn about arguments from both sides of a controversial issue, one must resort to something other than the typical search engines mentioned above. Following discussions on such topics can give one the opportunity to get an overview of arguments from both sides of an issue. Such discussions often take place on social media. However, social media often do not provide a good basis for an opinion-forming process, as people quickly get caught up in a bubble that exclusively reproduces opinions that already exist in their heads [1]. With controversial topics, however, it is usually essential to look at all sides of the issue in order

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
wenzlitschke@studserv.uni-leipzig.de (N. Wenzlitschke); suelzle@studserv.uni-leipzig.de (P. Sülzle)

[1]https://www.google.com
[2]https://www.bing.com
[3]https://duckduckgo.com

to form an well-founded opinion. This is why argument search engines which are presenting different views on a controversial topic are becoming increasingly important.

The *Touché @ CLEF: Argument Retrieval 2022* [2] Shared Task 1: "Argument Retrieval for Controversial Questions" calls scientists for new approaches on argument search. The goal of this task is to implement an information retrieval system that returns summarized arguments on controversial topics. The task specifies the search results to be pairs of sentence IDs taken from a preprocessed version of the *args.me* corpus [3]. The three dimensions of quality which should be focused on are: “(1) each sentence in the pair must be argumentative (…), (2) the sentence pair must form a coherent text (…), and (3) the sentence pair constitutes a summary of a single argument (…).” [2]. We focus our efforts on using different state-of-the-art argument retrieval and natural language processing methods. The retrieval system consists of different modules that together form the holistic pipeline, which are combined in different ways for the evaluations to find the retrieval system that yields the best results in the end.

In this paper we will discuss related work, focusing on works from which we drew the inspiration and approaches we used for our retrieval system in Section 2. In Section 3 we will go into more detail about our methodological approach and present and explain the individual optional components of our retrieval system. After that, we will discuss the evaluation and elaborate on our results in Section 4. In Section 5, we conclude to what extent the individual modules of our retrieval system impact the relevance and present the combination of our methods that achieved the best results in the evaluation. After that, in Section 5.1, we will also point out future directions in which our retrieval system could be improved.

2. Related Work

This work draws on existing information retrieval methods, argument mining and natural language processing. The following sections introduce the args.me corpus, methods of argument classification, ranking, and sentence pairing.

2.1. args.me Corpus

Our retrieval system is based on an index from a preprocessed version of the args.me corpus provided by the shared task organizers. The original version of this dataset was introduced in 2019 by Ajjour et al. [3]. It was created by crawling four selected online debate portals, namely (1) debatewise.org, (2) idebate.org, (3) debatepedia.org and (4) debate.org. The dataset consists of 387,606 arguments.

All of the crawled portals are structured similarly. They consist of user created forum-threads; each thread discusses a controversial topic. In all of the four portals, when contributing their point of view to a thread, users must choose a stance for it. Arguments in the preprocessed dataset are modeled as a triple of (1) a conclusion which mostly is the title of the discussion in the portal, (2) a premise which is a comment of a user on the related conclusion and (3) a pro or con stance towards the conclusion which is chosen by the respective user [4]. The stance of a premise is assumed to correspond to the stance the replying user took towards the topic. Additionally, each argument is given a unique ID. The arguments are organized in rows of topics with a unique topic ID, a conclusion and a list of premises with stance annotations.

The difference between the provided preprocessed version of the args.me corpus and the original version is that the preprocessed version contains the premises split up into sentences in an additional column. Since the arguments for the task should consist of two single sentences each, the preprocessed version of the args.me corpus thus provides a better basis for our work. Each sentence in the preprocessed corpus has a unique sentence ID it can be referenced by.

2.2. Ranking

Potthast et al. [5] compared the four standard retrieval models BM25, TF-IDF, DPH, and DirichletLM. The conclusion of their work is that DPH and DirichletLM are relatively similar in terms of relevance. However, TF-IDF and BM25 performed worse in the comparison. Even though Potthast et al.[5] found that BM25 is not as good as Dirichlet, we use BM25 for comparison.
Gienapp [6] also uses DirichletLM as a retrieval model for textual relevance in *Quality-aware Argument Retrieval with Topical Clustering*, referring to the frequent use of this model in the context of Touché Shared Task 2020. The most frequently used retrieval model was DirichletLM followed by BM25 [7] in the submissions for last year's Touché Task. For comparison we use BM25 in addition to DirichletLM.

2.3. Argument Classification

As described in Section 1, one of the quality dimensions of the task is that each sentence of a retrieved sentence pair must be argumentative. To approach the quality dimension of argumentativeness, we found two promising methods in the literature [8, 9].
Gienapp et al. [8] used a *support vector machine* (SVM) to classify given texts passages into arguments and non-arguments. This SVM was trained on the *Webis-ArgQuality-20* corpus [8]. Regardless of the premise's topic, a binary decision is made whether this text span is an argument or not. Gienapp et al. [8] also trains a support vector regression model, which determines the quality of a text span classified as an argument. Because the text quality is not one of the quality dimensions in this year's Touché task, we decide to use only the SVM for classifying text passages into arguments for our retrieval model.

Another method of classifying arguments was presented by Reimers et al. [9]. Contrary to the methods above, they worked on sentence level argument classification. Also, their approach takes a topic into account, to which the sentence can either be an argument for, an argument against or not argumentative. For this, they fine-tuned multiple language models such as ELMo and BERT to the task of argument classification.

The best performing models they present are BERT-base$_{topic}$ and BERT-large$_{topic}$. They are fine-tuned versions of the BERT-base and BERT-large models, respectively. The input is a topic and a sentence. The model classifies the sentence either as an argument for, an argument against or not argumentative. The model was trained on the *UKP Sentential Argument Mining* Corpus, which annotated 25,492 sentences over eight controversial topics. We used their pretrained BERT-base$_{topic}$, which we refer to as ACL$_{BERT}$ in our experiments.

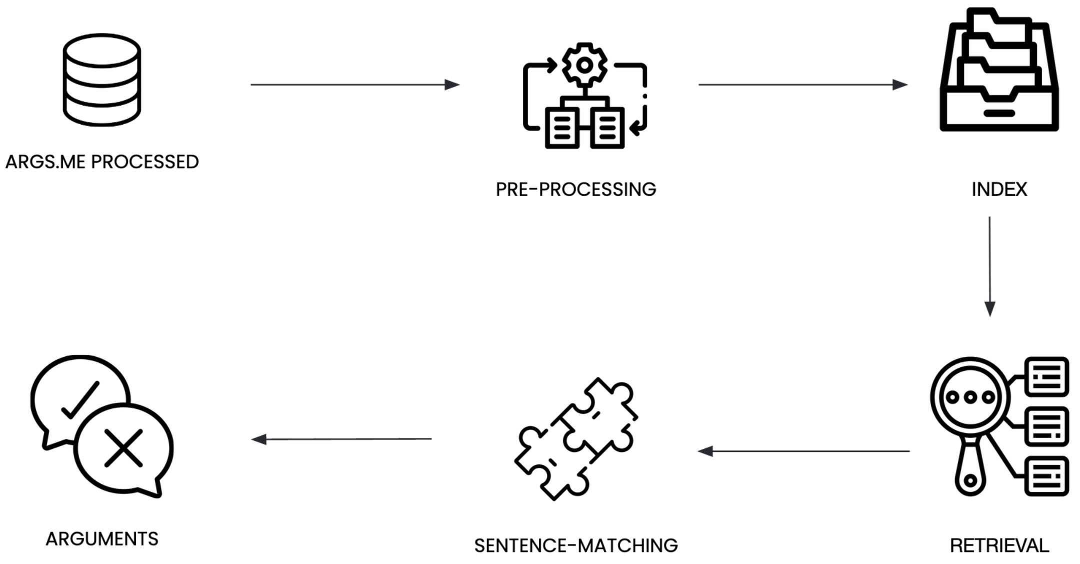

Figure 1: Abstract overview of our retrieval system. It consists of the preprocessed args.me corpus (Section 3.1), preprocessing (Section 3.2), the indexing (Section 3.3), the retrieval (Section 3.4), and the sentence-matching (Section 3.5.2), which yield the argument pairs.

2.4. Sentence Pairing

An essential step for accomplishing the Touché task is matching the retrieved sentences since quality dimensions (2) and (3) of the task (Section 1) refer to pairing sentences. We use two methods in order to pair sentences. The first method we use pairs sentences by similarity. First, we transform the sentences into sentence embeddings. For that, we use *Sentence-BERT* (SBERT) by Reimers and Gurevych [10]. SBERT is an extension of BERT, which, by utilizing Siamese and triplet network structures, reduces the processing time for the transformation of sentences into meaningful sentence embeddings. Second, we match the sentences calculating the cosine similarity between these sentence embeddings. This forms our initial pairing approach. Our second method also uses BERT, a transformer that is intended to capture the relationship between sentences. For this purpose, BERT was trained with a *Next Sentence Prediction* (NSP) task. This means that BERT was given labeled training data consisting of two sentences, where consecutive sentence pairs got the label *IsNext* and random sentence pairs got the label *NotNext* [11]. Since BERT was trained with exactly such a task, it was apparent to us to use BERT for exactly this purpose. Therefore, our second approach utilizes BERT to find the best match for a given sentence using NSP.

3. Methodological Approach

In this Section, we describe all methods we tested to retrieve topic relevant and qualitative sentence pairs from arguments. These methods compose optional components in our retrieval system. First, we describe the necessary preprocessing and indexing steps of our retrieval system. Then, we explain our approaches to retrieve topic relevant sentences using different query structures, noun chunking, and re-ranking via Query-Based Argument Classification. Lastly we detail on out approach to pair sentences using sentence similarity and NSP.

The retrieval system consists of optional preprocessing steps as described in Section 3.2, indexing of the sentences with Elasticsearch[4] (Section 3.3), retrieval of sentences with one or more methods described in Section 3.4 and one of the two sentence pairing methods described in Section 3.5. An abstract overview of the individual modules of the retrieval system can be seen in Figure 1.

3.1. Corpus

From the provided preprocessed corpus args.me described in Section 2.1, the single sentences were used for our retrieval system. We use single sentences, assuming that the sentence that best completes or matches a retrieved sentence does not necessarily have to come from the same source argument. Therefore, our retrieval steps are performed on these single sentences. Consequently, we do not further include the arguments from which the sentences originate in the search, except for argument classification based on the Webis-ArgQuality-20 corpus described in Section 3.2.2.

3.2. Preprocessing

Initially, we remove duplicates of single sentences on an exact text match to ensure the variety of search results and to avoid identical arguments. In an exploratory analysis of the corpus, we found many sentences that we were confident would never produce a satisfying result, because (1) they were not proper, that is well-formed, sentences at all (e.g. a URL, a reference, grammatically incorrect) or (2) they were not argumentative in any way (e.g. off-topic, spam, filler sentences, sentences just not making a point towards the topic), and thus not fulfilling the quality dimension, that every retrieved sentence should be argumentative. We experimented with ways to automatically detect and remove such content from the corpus through natural language sentence classification, and two argument classification approaches, which we explain in the following.

3.2.1. Natural language sentence classification

To remove the not proper language sentences, we established a preprocessing step to categorize our given premises and conclusions into proper sentences and not proper sentences. We started from the simple heuristic that a natural language sentence always contains a verb. Thus, we check the premises and conclusions for the presence of a verb. To do so, we tagged each of these premises and conclusions using NLTK's POS tagger [12]. Then, we checked the list of POS tags of a sentence for the presence of a verb tag by creating a list of all verb tags of the POS tagger and compared the tag list of the respective sentence with this list. We declared

[4]https://www.elastic.co

premises or conclusions for which no word was tagged with a verb tag from the reference list as non-sentences and not processed further in the retrieval system.

3.2.2. Webis-ArgQuality-20 Argument Classifier

In order to remove sentences from the corpus that are not argumentative, we trained an SVM to classify arguments. This is based on the approach by Gienapp et al. [8] described in 2.3. We trained the SVM using the Webis-ArgQuality-20 corpus [8]. One problem in implementing the SVM is that the underlying Webis-ArgQuality-20 corpus consists of whole argument phrases. These are coherent sentences that together make up an argument, rather than single sentences. Therefore, we used the argument phrases from the preprocessed args.me corpus. Thus, the SVM classifies the argument phrases whether they are arguments or not. Then, the complete passages classified as no argument are removed from the corpus and no longer considered in further processing. Thus, it is assumed that passages that the SVM has classified as no arguments do not contain relevant or argumentative sentences.

3.2.3. Classifying Arguments with BERT

The second method of removing non-argumentative sentences is based on ACL_{BERT}, which classifies sentences as argumentative or not argumentative with respect to a given input topic, as described in Section 2.3. The defining principle is that the sentences are not classified in isolation, but in the context of a topic.

We labelled all sentences of our dataset with respect to it's corresponding conclusion. As topic information, we provided the model with the corresponding conclusion of each sentence. Sentences classified as *noArgument* were removed from the corpus. All sentences classified as *Argument_for* or *Argument_against* were kept. The reasoning is that sentences, that are not argumentative with respect to their corresponding conclusions, have little chance to be strong arguments in any context. The stance information was not used for our retrieval system because the classified stance in this case refers only to the underlying conclusion of the respective sentence.

3.3. Indexing

We use Elasticsearch to index the premises and conclusions in the preprocessed args.me corpus. Each document in our index consists of the sentence ID associated with the args.me record, the text of the sentence, and its corresponding conclusion. In addition, using Elasticsearch's inherent analyzer pipeline, different filters are applied to each document. These filters are stemming, stop word removal, and text lower casing. The filters are applied on both the sentence and conclusion text fields.
Depending on the retrieval system composition, these documents are then ranked using either DirichletLM or BM25. We determine the parameter μ for DirichletLM retrieval by determining the average length of the individual sentences in the corpus. μ is 116 after the preprocessing step of sentence classification.

3.4. Retrieval

After indexing the documents using Elasticsearch, the retrieval is done using the Elasticsearch Search API. For this purpose, we call the Elasticsearch Search API with different query requests. The query requests consist of different query components depending on the retrieval system composition. We distinguish between a simple, naive baseline approach that understands the entered search term as a composition of individual terms and a more refined approach considering Boolean queries. The refined approach considers terms in the text field of the document sentence and terms in the conclusion field of the index, as described in 3.4.1. Thus, sentences with one of the search terms in the sentence and in the corresponding conclusion field are given particular significance. For this second approach, we have developed an additional feature as will be described in Section 3.4.2. This feature uses natural language processing to extract specific terms from the search query and give them particular weight. In Section 3.4.3, we describe our re-ranking approach. Thus, we use the ACL$_{BERT}$. After the single sentences have been retrieved using one of the two query methods, they are evaluated with ACL$_{BERT}$ for their argumentativeness concerning the query and, if so, ranked further up.

3.4.1. Query

The queries are full text queries, which we direct to the endpoint of the Elasticsearch index in Elasticsearch's query Domain Specific Language (DSL). These full text queries allow searching the text of a specific document field (sentence or conclusion) in our index. We process the query string with the same analyzer pipeline used to index the Elasticsearch fields, described in Section 3.3. For ranking the sentences, we use either BM25 or DirichletLM.

We use different full text queries of the query DSL. As a first approach we use a simple match query. The match query searches by default on one document field and separates the query string into individual terms, then combined with a logical `or` operator. The second - more elaborate - approach uses the Boolean query *match_bool_prefix* of the Query DSL Language. This allows for queries that consist of several subqueries. With each subquery that matches a document, the calculated score for this document increases concerning the root-query. Different fields can also be taken into account using this query type. For example, we use this query type to search the sentence and conclusion fields simultaneously. Thus, documents in the index that match query terms in both, the sentence field and the conclusion field, are ranked higher. In addition to viewing the query string as a series of individual terms linked with an `or` operator, we add another subquery that connects the query terms with an *and* operator. Thus, documents in the index with the exact wording are additionally boosted. Since not all parts of a search query have the same significance for the search results and only in very few cases the entire query string occurs in the text we have refined the second query approach. We extract certain groups of terms of the query string to search for them using the *match_phrase_query*. The *match_phrase_query* scores exact matches much higher when the search query occurs in the same way in the text of the searched field.

3.4.2. Noun Chunking

Compound terms, phrases composed of more than one noun, can have a different meaning than the individual words that compose them, such as *sex education* [13]. Therefore, these compound terms should be given special consideration in the query. In evaluating our initial experiments, we found that too little value was placed on compound terms. This is because each word of the query was evaluated and searched as a single term in our basic query process. To address this problem, we tried to find a way to consider the compound terms and the nouns in the query and boost them. For this task, we deploy a method that we call *noun chunking*. For this, we use the Noun Chunker from spaCy[5]. Noun chunks can be considered as "base noun phrases" and consist of a noun and descriptive word that relates to the noun. These descriptive word can be any word specifying the base noun. We pass the query to spaCy which returns a list of noun chunks. These noun chunks are then appended to the Boolean query in *match_phrase_query* and thus considered separately again when retrieving results in the query process. It is also possible to give the *match_phrase_query* an extra boost to highlight these noun phrases even more.

3.4.3. Re-Ranking: Query-Based Argument Classification

We want our retrieval system to place strongly argumentative sentences on high ranks. When doing the first evaluations, we noticed that the most argumentative sentences would not always be the highest ranking. To combat this, we once again utilized ACL_{BERT} described in Section 2.3. The classifier takes a sentence and a topic as input, which we hand over from the list of sentences retrieved by Elasticsearch. In contrast to how we use ACL_{BERT} in preprocessing, where we use the conclusion as a topic, we now use the query. This approach considers the properties of a sentence (argumentative or not) depending on the particular input and does not assume that a sentence is inherently argumentative. Sentences classified as an argument for or an argument against were boosted, so they would rank first before the sentences classified as no argument.

3.5. Sentence Pairing

Sentence pairing describes the task of matching two relevant sentences for a specific query. We tried different approaches to decide whether two sentences match. The first approach pairs the sentences based on their cosine similarity. In the second approach, the probability that a second sentence follows the first sentence is calculated. The determined value is maximized in both cases, and the corresponding sentence is assigned its optimal match.
We use the number of sentence pairs that should appear in the output to determine the number of top sentences. *Top sentences* are the single sentences relevant to the query which form the first sentence of each retrieved sentence pair. The remaining sentences to compare are the retrieved sentences without the *top sentences.* Pairing is performed between these *top sentences* and the pool of the remaining sentences. The number of sentences to be compared with each top sentence is many times larger than the desired number of sentence pairs in the output. Once

[5]https://spacy.io

a sentence is matched with a top sentence, it is removed from the pool of the match candidates, so it cannot appear for twice. Matching is performed in descending order so that the entire pool of match candidates is available for the first sentence, and this pool becomes successively smaller.

3.5.1. Sentence Similarity

We match sentences by following the intuition that if a retrieved sentence is relevant and argumentative to the associated query and similar to a second sentence, the latter will complement the first sentence and strengthen the argument. Based on the approach described in Section 2.4, we first transform the sentences into sentence embeddings. These sentence embeddings of the top sentences are now matched against the sentence embeddings of the remaining match candidates in terms of cosine similarity. Each top sentence is then assigned the sentence for which the cosine similarity is the highest.

3.5.2. Next Sentence Prediction

As an alternative sentence pairing method to sentence similarity, we experimented with Next Sentence Prediction (NSP) which we explained in more detail in Section 2.4. Thus, for each sentence that our retrieval system returns, we compute the probability that a specific other sentence follows it. This approach follows the idea that a sentence that is relevant and argumentative by our retrieval system concerning the query is likely to be followed by a sentence that (1) syntactically goes along with the structure of the previous sentence and (2) strengthens and completes the mentioned argument by the first sentence. In descending order, we calculate for each sentence the probability computed with every other sentence that has not been used in any combination so far. Then, we assign each sentence to the sentence that is most likely to follow it.
To implement this, we use the pre-trained *bert-base-uncased model* from Hugging Face's Transformer Library [14], which is based on the work of Devlin et al. [11]. First, the model transforms both sentences into sentence embeddings using the pre-trained BERT tokenizer. Then the model predicts how likely the second sentence is to follow the first sentence. The pair of sentences that achieves the highest score among all possible combinations of first and second sentences is the best match. Using the NSP model, the desired number of sentence pairs is formed from the *top sentences* and the less highly ranked sentences.

4. Evaluation

To evaluate the methods described above, we applied the retrieval system to ten test queries with varying components enabled, resulting in 8 different combinations. For each query, we retrieved ten results, which we then annotated rating each result with a score out of (-2, 0, 1, 2, 3). The heuristics we used to assign scores during the evaluation can be seen in Table 1. From the annotated results, we calculated the precision@10 and nDCG@10 for each evaluated combination of components. Precision@10 was calculated interpreting ratings of -2 and 0 as not relevant and 1, 2 and 3 as relevant results.

score	label
-2	spam
0	fulfills some criteria but not relevant to the query
1	relevant to the query and fulfills criteria (1)
2	relevant to the query and fulfills criteria (1) and (2)
3	relevant to the query and fulfills criteria (1), (2) and (3)

Table 1
For the score a pair of sentences could receive in our evaluation, there were certain criteria that we used as a guideline for assigning the scores.
Quality dimensions specified by the task were included in the scoring:
(1): each sentence in the pair is argumentative
(2): the sentence pair forms a coherent text
(3): the sentence pair constitutes a summary of a single argument

preprocessing	retrieval	query	re-ranking	pairing	precision@10	nDCG@10
	BM25	simple		sim	0.34	0.55
	BM25	simple		NSP	0.47	0.65
	BM25	simple	ACL$_{BERT}$	NSP	0.54	0.69
SCL	DLM	simple		NSP	0.36	0.62
SCL + ACL$_{SVM}$	DLM	simple		NSP	0.36	0.62
SCL + ACL$_{BERT}$	DLM	simple		NSP	0.39	0.57
SCL + ACL$_{BERT}$	DLM	simple	ACL$_{BERT}$	NSP	0.42	0.62
SCL	DLM	Boolean		NSP	0.48	0.59
SCL	DLM	Boolean + NC	ACL$_{BERT}$	NSP	**0.67**	**0.73**

Table 2
The evaluation results are measured for the individual retrieval system compositions in precision@10 and nDCG@10. SCL refers to the sentence classifier used in preprocessing (Section 3.2.1). ACL$_{BERT}$ refers to both, the preprocessing and the re-ranking step (Section 3.2.3, Section 3.4.3). ACL$_{SVM}$ is the argument classification approach described in (Section 3.2.2). DLM stands for DirichletLM, NC is noun chunking (Section 3.4.2), and sim refers to sentence similarity (Section 3.5.1). The best results are shown in bold.

The combinations to evaluate were chosen exploratively. First, we evaluated the retrieval system using BM25 as a ranking function, results shown in Table 2. Here, we compared the performance of our two sentence pairing methods. NSP outperformed sentence similarity in precision@10 and nDCG@10, which is why all following retrieval systens we evaluated utilize NSP.

The highest precision@10 and nDCG@10 we can report with BM25 is 0.54 and 0.69 respectively using NSP as the sentence-pairing method and ACL$_{BERT}$ for re-ranking.

Our second round of evaluation were all done with NSP because of the promising results in the first evaluation round as shown in Table 2. DirichletLM as a ranking function was chosen since it has proven to work well with argument retrieval in previous work, as described in Section 2.2. We found that the results were worse than our best performing retrieval system

with BM25. DirichletLM, NSP and the sentence classifier (Section 3.2.1) yield a precision@10 of only 0.36. Also, we found that the preprocessing with the SVM did not change the result at all. Results improved with ACL_{BERT} in preprocessing and improved again with the query based ACL_{BERT} enabled. Finally, our best performing retrieval system with DirichletLM and NSP, even beating our best retrieval system setup with BM25, utilized the sentence classifier in preprocessing, the Boolean query using noun chunking for querying, and the ACL_{BERT} for re-ranking, reaching a precision@10 of 0.67.
Lastly, we examined the sentence lengths of all approaches and sorted them according to whether they were performed with BM25 or DirichletLM. The average sentence length of the approaches shown in Table 2 performed with BM25 as the ranking method is 165.15, whereas the approaches using DirichletLM have a much longer average sentence length of 298.19.

5. Discussion

In the first part of this section, we will go through the individual optional methods of our retrieval system and take a closer look at their respective influence of each of them. In doing so, we will also discuss the shortcomings of the approaches. In the second part, we address possible further developments of our approaches that could improve our retrieval system in (Section 5.1).

Preprocessing can be divided into sentence classification and argument classification. We can say that our sentence classifier had an influence in the sense that non-sentences such as links appeared less frequent when going through the results checking a sentence for the presence of a verb. No significant change in the result set was observed after applying argument classification using the SVM. A possible explanation for this would be that the argument passages classified as non-arguments did not find their way into the final retrieved result set, so filtering them out beforehand did not further influence the result. The same reasoning applies to the ACL_{BERT}, which could not make any significant improvement in the preprocessing step, as can be seen in Table 2.

Our approaches can be divided into two parts for indexing based on the underlying retrieval model. On the one hand, BM25 was used as the retrieval model that provided the best results until the last evaluation run of the DirichletLM retrieval model where we use the noun chunking approach. On the other hand, we obtained the best results using DirichletLM as the retrieval model, which will be described in more detail later. Moreover, we noted that DirichletLM produced nearly twice as long sentence pairs as BM25 for all the performed experiments. In some cases, we retrieved sentences that should have been divided into several single sentences, which they were not due to missing punctuation marks.

We also used a number of different approaches for retrieval. In our first experiments, we used a simple query where the words of the entered topic are understood as single terms. Using the Boolean query improved the resulting precision@10 value slightly. In the Boolean query, both the sentence and the conclusion field of the documents were included in the ranking. This led to additional results that did not include terms from the actual query in the sentence field, but still related to the query due to a matching conclusion. Thus, the field of match candidates for the pairing methods was expanded, resulting in more heterogeneous sentence-pair combinations. In

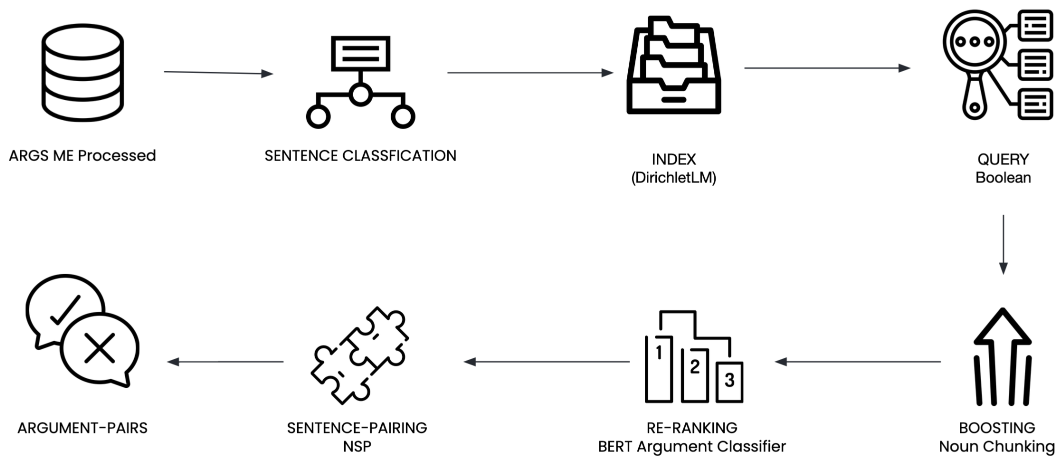

Figure 2: Final retrieval system, which yielded the results. It consists of the preprocessed args.me corpus (Section 3.1), the sentence classification (Section 3.2.1), the indexing by DirichletLM (Section 3.3), the retrieval by Boolean queries (Section 3.4.1) extended by the noun chunking (Section 3.4.2) and the subsequent re-ranking by ACL$_{BERT}$ (Section 3.4.3). Finally, the NSP method (Section 3.5.2) performs the pairing, which outputs the argument pairs.

addition, our noun chunking approach was also used in these Boolean queries, which increased the precision@10 value by boosting individual terms of the query. Besides the query, another crucial part of the retrieval is the ranking. ACL$_{BERT}$ is used again in the retrieval step to re-rank the queried results. The retrieved results classified by ACL$_{BERT}$ as argumentative concerning the associated query string were boosted and re-ranked further up the results. As a re-ranking component in the retrieval system, ACL$_{BERT}$ has a more significant impact on the results than the preprocessing step.

For sentence pairing, we tried two different approaches. The worse approach of the two was the sentence similarity approach. There were problems with the approach because the matched sentences were too similar. Too similar sentences often consist of the same words. The second sentence merely repeats the first and does not logically continue it. Therefore, we did not consider this approach further after our first round of evaluation and instead focused on the second approach. This approach yields better results regarding sentence pairing since NSP is not about finding similar sentences but about how likely it is that a sentence follows a given sentence.

Finally, after evaluating different combinations of the different modules of our retrieval system, we were able to identify the retrieval system that gave the best results. This retrieval system uses only the sentence classifier since this was the only classifier in the preprocessing step, producing noticeable evaluation differences. In addition, the retrieval step uses Boolean queries with noun chunking and ACL$_{BERT}$ for re-ranking. We used the retrieval model DirichletLM, and NSP was used as the sentence pairing method. As seen in the last row of Table 2, this combination obtained the best results. With a precision@10 value of 0.64 and an nDCG@10 of 0.74, our retrieval system does not yet function optimally, but offers room for improvement due to its modular structure. The application of certain modules led to an improvement of precision@k or nDCG@k. For example, noun chunking improved the precision@10 value, while the ACL$_{BERT}$ improved the nDCG@10 in the re-ranking step. Our final retrieval system can be seen in Figure 2.

5.1. Future Directions

In order to further develop our retrieval system, various modules could be refined, potentially yielding better results. For example, the use of the ACL$_{BERT}$ produced better results in the re-ranking process, but we did not use any fine-tuning. It would be possible to train the classifier on our evaluation data to improve the results and the re-ranking with the ACL$_{BERT}$. After we have done evaluation runs, there would be data with which the classifier could be trained on the topics given by Touché, in addition to the eight topics on which the classifier was originally trained.

A further improvement to re-rank the results would be possible based on ACL$_{BERT}$. In the current approach, the sentences classified as arguments by ACL$_{BERT}$ are pushed to the beginning of the retrieved sentences. To refine this re-ranking, it would be possible to count the classification towards the retrieval score by a boost and not just pushing the classified arguments to the beginning.

When evaluating the results, we noticed that the sentence pairs sometimes contradict each other despite their respective good argumentativeness or relevance. The given stances from the dataset cannot be used due to our approach, which considers the sentences independently of their respective original argument passage. One way to address this problem would be to use, for example, sentiment analysis or similar methods to ensure that the sentences are matched only with sentences that represent the same point of view.

6. Conclusion

The aim of the present research was to implement a retrieval system to obtain relevant and argumentative sentence pairs from the *args.me* corpus and evaluate its performance.
In the preprocessing step, we have removed duplicates and sentences that do not add value to the final result. The latter include non-proper language sentences, which were identified based on POS heuristics, and non-argumentative sentences, which were identified using two different argument classifiers ACL$_{BERT}$ and ACL$_{SVM}$ based on approaches by Gienapp [6] and Reimers et al. [9]. For indexing and retrieving the premises and conclusions, Elasticsearch with either DirichletLM or BM25 was used. In addition to a simple query where the search terms are understood as individual terms, we also used Boolean queries to retrieve the relevant arguments. Within the Boolean queries our noun chunking approach allowed certain terms to be weighted. The ACL$_{BERT}$ based on Reimers et al. [9] was used again after retrieval to re-rank the sentences based on their argumentativeness. In the last step, the obtained top sentences were matched with the remaining retrieved sentences based on two different approaches. In the first approach, the sentences were matched based on their cosine similarity using SBERT [10]. In the second approach, the sentences were matched using BERT and NSP [11].
Nine different combinations of these intermediate steps were evaluated using precision@10 and nDCG@10. The best combination with a precision@10 of 0.67 and an nDCG@10 of 0.73 used

our sentence classifier and ACL$_{BERT}$ in preprocessing, DirichletLM for indexing, the Boolean query with Noun Chunking, and ACL$_{BERT}$ again for re-ranking.

References

[1] W. Quattrociocchi, A. Scala, C. R. Sunstein, Echo chambers on facebook, SSRN (2016).

[2] 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.

[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] 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: K. Ashley, C. Cardie, N. Green, I. Gurevych, I. Habernal, D. Litman, G. Petasis, C. Reed, N. Slonim, V. Walker (Eds.), 4th Workshop on Argument Mining (ArgMining 2017) at EMNLP, Association for Computational Linguistics, 2017, pp. 49–59.

[5] M. Potthast, L. Gienapp, F. Euchner, N. Heilenkötter, N. Weidmann, H. Wachsmuth, B. Stein, M. Hagen, Argument search: Assessing argument relevance, 2019, pp. 1117–1120.

[6] L. Gienapp, Quality-aware argument retrieval with topical clustering, in: 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, Bucharest, Romania, September 21st - to - 24th, 2021, volume 2936 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 2366–2373.

[7] 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 2936 (2021) 2258–2284.

[8] L. Gienapp, B. Stein, M. Hagen, M. Potthast, Efficient Pairwise Annotation of Argument Quality, in: The 58th annual meeting of the Association for Computational Linguistics (ACL), ACL, 2020.

[9] N. Reimers, B. Schiller, T. Beck, J. Daxenberger, C. Stab, I. Gurevych, Classification and clustering of arguments with contextualized word embeddings, 2019.

[10] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019.

[11] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018).

[12] S. Bird, E. Loper, NLTK: The natural language toolkit, in: Proceedings of the ACL Interactive Poster and Demonstration Sessions, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 214–217. URL: https://aclanthology.org/P04-3031.

[13] A. Feuer, S. Savev, J. A. Aslam, Implementing and evaluating phrasal query suggestions for proximity search, Information Systems 34 (2009) 711–723.

[14] 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 Computational Linguistics, 2020, pp. 38–45.