

# Argument Retrieval for Comparative Questions based on independent features

Notebook for the Touché Lab on Argument Retrieval at CLEF 2021

Thi Kim Hanh Luu<sup>1</sup>, Jan-Niklas Weder<sup>1</sup>

<sup>1</sup>Martin Luther University of Halle-Wittenberg, Universitätsplatz 10, 06108 Halle, Germany

## Abstract

In this paper, we present our submission to a shared task on argument retrieval for comparative questions at CLEF 2021. For the given comparative topics, we retrieved relevant documents from the ClueWeb12 corpus using BM25-based search engine ChatNoir and rerank them with our approach. Our approach combines multiple natural language processing techniques such as part-of-speech (POS), word embeddings, language models such as BERT, argument mining, and other machine learning methods. Using the TARGER Tool, BERT or PageRank we generated scores which are then used for the re-ranking. A Support Vector Machine is used to learn weights for the final ranking of documents on basis of those scores. The presented results on the given topics from the shared task last year evaluated by nDCG@5 measures showed, that one configuration of our approach can improve the nDCG@5 by approx 0.07 if we only consider the evaluated documents. Furthermore, one of our approaches reached the fourth place in the Argument Retrieval for Comparative Questions task.

## Keywords

information retrieval, comparative search engine, argument retrieval, natural language processing, machine learning

## 1. Introduction

Every person has to make decisions several times a day. Some of these decisions hardly have any influence on the person, some decisions can have a decisive impact on the person's life. An example of such an important decision might be at which university one should study or for which job one should apply. There is often a point where you have to weigh several options against each other and are forced to choose one of them. So it could be that you have to choose one of the universities. For this decision, you need the information that supports each side. Here it would be useful if there was some kind of search engine that would provide the pros and cons of a given comparative question. This is exactly the kind of problem that Touché Task 2 is concerned with. The goal here is to generate a ranking for documents from the ClueWeb12 corpus using search engine ChatNoir so that documents that provide great value for a comparative question are ranked as high as possible [1]. We present in the following our approach with which we aim at solving this task and providing people with the best possible arguments so that they can weigh up for themselves and finally make an informed decision.

---


CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

✉ Thi.Luu@student.uni-halle.de (T. K. H. Luu); Jan-Niklas.Weder@student.uni-halle.de (J. Weder)

🌐 <https://github.com/hanhluukim> (T. K. H. Luu); <https://github.com/JanNiklasWeder> (J. Weder)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

As an objective, we can formulate that we aim to improve with respect to last year's baseline and ideally also reach most effective approach of the last year. It should be mentioned here that there was only one submission last year that exceeded the baseline and that had an improvement for the nDCG@5 of 0.012 [2].

The structure of the paper is as follows. First, we review some related approaches that describe the basis of our software in more detail. This part is followed by our query expansion steps that extend the original query to retrieve additional documents that also match the searched topic. After that, we deal with the scores used for improving the ranking. The scores are followed by an overview of our entire pipeline along with the options provided by the architecture we have chosen. The next topic will be the results we got from applying our pipeline to last year's dataset. The paper is then closed by the conclusion, which reviews the most important parts of the pipeline and suggests some possible improvements.

## 2. Related Work

We have used some areas in our approaches such as argument mining and synonyms extraction.

### 2.1. Argument Mining

Argument mining is the process of automatically extracting arguments from unstructured texts. This process is becoming increasingly important with regard to the automatic processing of texts from the web [3]. For a comparative question, a search engine can use argument mining to find relevant documents on the web that contain arguments regarding a concept. There are several developments for argument mining by applying natural language processing (NLP) methods. We chose TARGER because it allows us to use the software through an API and was designed explicitly for web documents in mind. TARGER is an open-source neural argument mining tool for tagging argument units, like premises and claims [4].

### 2.2. Synonyms extraction

Synonyms extraction is a research problem, which is helpful to text mining and information retrieval [5]. For example, an automatic query expansion using synonyms is capable of improving the retrieval effectiveness [6]. The WordNet database can be used to obtain synonyms of an English word [7]. This method can lead to a problem of semantic gap [8], for example, *i go to the bank* and *i sit on the bank*. To find the right synonyms for this word *bank*, we need to understand the context of these sentences. The use of this dictionary is also sometimes expensive because of the manual building and maintaining of database [9]. An improved approach is to incorporate word embeddings to deal with a semantic problem [10]. There are several methods to calculate the similarity between words using word embeddings, such as Manhattan distance [11]. In our approach, we use cosine similarity for synonyms extraction.

### 3. Query expansion

User queries sometimes do not accurately reflect the user's information needs. For example, the queries are too short or the user uses other semantically similar words, that are not available in the retrieval system. To improve the retrieval process, the users' queries can be expanded through different query expansion methods [12]. In our approach, we generated several new queries based on the given query using natural language processing, part-of-speech method, word embeddings and combinations of them. The following explains how the query expansion was implemented.

#### 3.1. Expansion methods

In our first method Preprocessing, a new query is generated by lemmatization of a single word of the original query, for example, *Which four wheel truck is better: Ford or Toyota?* to *Which four wheel truck be good: Ford or Toyota?*. In this method, we are not interested in a concrete comparison between two objects *Ford* and *Toyota*, but it is expected that this transformation to lemmas will return more results about these objects from the retrieval system. To note here is that preprocessing belongs to the query expansion step but is separate from other expansion methods in our implementation.

To focus on the comparison part of a query sent by the user, we assume in the second expansion method that the search engine should identify the terms from the query that stay on a comparative relationship, for example: *four wheel truck, better, Ford* and *Toyota* are comparative terms. In the retrieval process, the search engine should find documents, which contain these comparative terms. In this technique, the part-of-speech (POS) tagger from the spaCy language model [13] is used to select these terms and generate a new query from them (see 1). We assume that the comparative terms of a query can have the following POS tags: *number, verbs, adjectives, adverbs, nouns, proper nouns*. Numbers, adjectives, nouns, and proper nouns can represent the objects to be compared in the query. When comparing two actions, such as *should I buy or rent?*, verbs and adverbs can play the role of the comparative terms. The newly created query in this method with POS contains only these comparative terms and other not comparative terms will be removed.

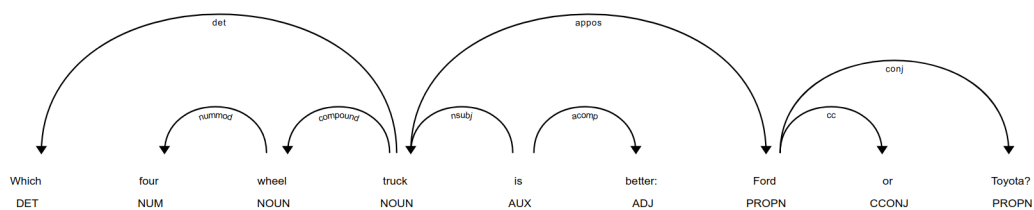


Figure 1: Identification of Part-Of-Speech-Tags visualized by <https://spacy.io/usage/visualizers>

To deal with the problem of semantically similar words, for example, the user uses *notebook* in his query and a factually relevant text contains only the word *laptop*, three methods are implemented using WordNet [7], word embeddings [10], and word sense embeddings [14]. The WordNet method expands the original query by adding synonyms of comparative terms, which have NOUN-tag, to the end of the original query. The two other embedding based methods replace the comparative terms by found synonyms to create a new query. The comparative terms used here have already been identified in the second method above.

In the first variant, we use WordNet dictionary, a lexical database for the English language to find the synonyms. The more specific a query is, the higher the probability that no document contains all words of the query. It can lead to a problem that no results from ChatNoir will be returned. For this reason, only the first five synonyms returned by WordNet for each word of NOUN-comparative terms are used.

For specific names such as *Google*, *Yahoo*, and *Canon* in a query, no synonyms can be found by the WordNet dictionary, however similar terms such as *search engine*, *camera* or other similar organizations such as *Bing*, *Nikon* are sometimes helpful in the retrieval process. For example, the user does not want to compare the financial factors of the organizations with the query *which is better, Google or Yahoo?*, but is interested in the functionality of the two search engines. The similar word *search engine* can help to describe the information needs of the user clearer in this example. Moreover, specific names of organizations may be changed after time and only a few documents with the old names are found. In this case, the documents with the new or similar names may be relevant to the user's query. Therefore, semantic similar words by using wordembeddings are searched in our approach and used for the two next query expansion embedding methods. For each comparative term, the top two similar words will be selected by the two highest cosine similarities.

We use models of word2vec embeddings and sense2vec from the spaCy library to generate word vectors for the comparative terms [13][15]. The word2vec model for the English language of spaCy was trained on the English dataset of OntoNote 5.0, which contains various genres of text, like news and web data [16]. The sense2vec model was trained on the 2015 portion of the Reddit comments corpus [17]. Both pre-trained models are suitable for our task with web documents. In the word2vec method for each word from the comparative terms, the top two similar words are determined using cosine similarity. If a candidate word is not present in the vocabulary of word2vec, no similar words are extracted. The sense2vec model is an extension of word2vec, where separate embeddings are learned for each sense of word, based on its POS tag. For example: in the query *Which is better, Canon or Nikon?*, *Canon* is recognized as a Proper Noun by POS Tagger or as a PRODUCT by Named Entity Recognition. In this case, similar words should be searched with the consideration of the entity PRODUCT. To make it possible, we use the sense2vec model to find similar words also by cosine similarity.

After the top two similar words have been found for each comparative term using word2vec and sense2vec, we replace these terms with their associated similar words. This replacement process can generate several new queries, for example *which is better, laptop or desktop?* will be extended to *which is better, notebook or desktop?* and *which is better laptop or PC?*, because *notebook* is the similar word of *laptop* and *PC* is the similar word of *desktop*. Suppose we have  $n$  comparative terms in our query. If two similar words are found for each term, there are  $2 * n$  new queries created by word2vec and also  $2 * n$  by sense2vec.

If all above expansion methods are used, we have then a maximum of  $3 + 2 * (2n)$  new queries from the given original topic. Since we need a final ranking without duplicates in the end search results, all retrieval documents from this part will be merged in the next step.

### 3.2. Merging

After query expansion, the original query and all other expanded queries are sent separately to search engine ChatNoir [1]. We keep all punctuation throughout the queries. For each query, 100 documents are by default retrieved. This means that for each original topic, a maximum of  $100 * (3 + 4 * n)$  documents are called. Since the submitted queries are similar, the set of retrieved documents usually contains multiple duplicates but with different scores. This problem motivates us to filter the search results so that duplicate documents are not shown to the user.

In our approach, the identical documents returned for each query have different relevance scores and at the same time have different importance concerning the user's expectation. We assume that the documents for the original queries are more likely to be relevant to the user than the documents from the extended queries, i.e. the documents retrieved by the original queries meet the user's information need better. In advance, each query expansion method is assigned a different importance weight. In our default approach, we use these values for weights:  $w_{\text{original}} = 2, w_{\text{pos}} = 1.5$ . Other expanded queries will be assigned with the same importance weight value:  $w_{\text{wordnet}} = w_{\text{word2vec}} = w_{\text{sense2vec}} = 1$ .

Given  $T$  is tag names of all queries:  $T = \{\text{original, pos, wordnet, word2vec, sense2vec}\}$ . If a document  $d$  occurs with different relevance scores  $r_{d_t}$  in the different expansion methods  $t \in T$ , these relevance scores are recalculated by multiplication with predefined importance weights. Using these weighted scores, a merge process is performed. We tested two merge functions. In the first variant, the maximum weighted score is selected from all weighted scores and assigned to the document:  $r_d = \max(r_{d_t} * w_t)$ . We assume here that the document is rated as relevant to the user query if it is highly ranked in at least one expansion method. In the second variant, the average of all weighted scores is assigned to the document:  $r_d = \text{average}(r_{d_t} * w_t)$ . In this case, the document is relevant if it has a high relevance score in all query expansion methods.

## 4. Scores

### 4.1. Argumentative scores

The motivation for our argumentative score approach is based on the fact that an answer suitable for a comparative question should be supported by multiple statements and premises. To implement it, the argument mining system TARGER with the model *classifyWD\_dep* was used, because documents from the ClueWeb12 dataset were collected from English websites and the model *classifyWD\_dep* was trained on the similar dataset web discourse [4] [18]. The TARGER system identifies argumentative units, premises and claims, on the token level in the input document. It returns for each token a label and related probability. An argumentative score is computed for the document by using the average of valid returned probabilities. In our

approach, a returned probability is valid, if its token is at the beginning or inside of premises or claims in the document. The outside labels from TARGER are not relevant.

If  $\mathcal{L}$  is the set of all labels of valid argumentative units for premise and statements. For each word  $w$  from the document  $d$ , TARGER outputs a tuple  $(l_w, p_w)$  of the most likely predicted argumentative unit label and corresponding prediction probability. We use only  $(l_w, p_w)$  if  $l_w \in \mathcal{L}$  is a valid argumentative unit.  $d_{\mathcal{L}}$  consists of only words  $w$  with  $l_w \in \mathcal{L}$ . The argumentative score for the document  $d$  is then computed as follows:

$$p = \frac{1}{|d_{\mathcal{L}}|} \sum_{w \in d_{\mathcal{L}}, l_w \in \mathcal{L}} p_w$$

$$\text{arg\_score}(d) = \begin{cases} p & p \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

In our approach, the threshold  $\theta$  defines the minimum argumentativeness a document needs to have. A document is considered an argumentative document if its argumentative score is greater than 0.55. If the threshold is higher than 0.55, we hardly get argumentative documents. Our experiments with topics from the shared task last year have also shown, that the threshold value 0.55 had provided the best result.

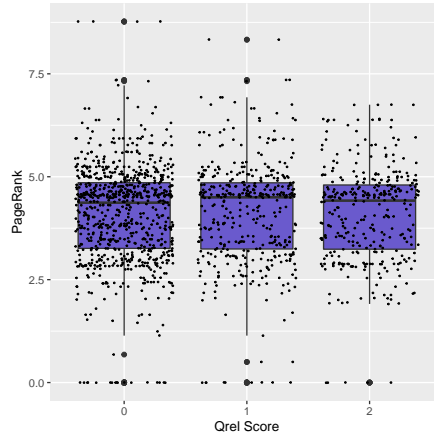
## 4.2. Trustworthiness

Another important characteristic of a good argument is its truthfulness, or if this information cannot be derived directly, at least the reliability of the source. We concluded that another score that encodes exactly this information may be beneficial for our problem. Unfortunately, determining the trustworthiness of a website is not a simple task and may not even be clearly definable. Utilizing PageRank we try to extend our pipeline with a trust component. This should allow us to weigh the different answers against each other and thus create another score on which we can rely [19].

In a rough summary, Pagerank is a way to define the relevance of a website without analytically processing its content. PageRank accomplishes this by taking advantage of the link structure of the Web. The assumption here is that there links to reliable websites are more frequent than links to unreliable ones and that reliable websites in turn mainly link to reliable websites. Thus, the link structure of the web is interpreted as a self-assessment of trustworthiness. PageRank now only reflects this structure in a single value per website. So this score can be seen as some kind of peer review [20].

Here we use PageRank to obtain a pseudo trustworthiness. For this, we will use a reimplementaion of PageRank, OpenPageRank [21]. In the following, we will briefly look at the correlation between PageRank and the ratings we know from last year’s evaluation to make a statement about the information content of this score alone [2].

If we now look at Figure 2, we can see that there is no clear relation between PageRank and the ratings of the documents. This assumption is supported by a small Spearman correlation of about -0.003. Since one would expect strictly monotonic relationships if PageRank could make a statement about the Qrel score, these values speak against this assumption. Therefore, it can be assumed that the PageRank score will not provide much additional value.



**Figure 2:** The PageRank obtained depending on the relevance score of the documents. Zero means not relevant, one denotes relevant and two indicates very relevant. The PageRank values range from 0 to 10 with 0 being the lowest and 10 being the best. Each dot represents a document.

### 4.3. Relevance Prediction

Due to the availability of assessed documents based on their relevance for a specific topic from last year, we decided to try a supervised learning approach. Especially since the results from last year utilized BERT to embed documents and topics and calculated a similarity score based on those embeddings, worked comparable well [2]. We decided to build on this idea and use BERT to predict whether a particular document is relevant or not. BERT stands for Bidirectional Encoder Representations and is an open-source machine learning framework for natural language processing [22]. So the problem defined here was interpreted as a two-sentence classification problem, as it is called in the SimpleTransformers library [23]. This means that we use two sentences as input and receive a class as the result. The two input sentences consist on the one hand of the query and on the other hand of the complete document for which we want to determine the relevance. It should be mentioned that the implementation used here results in the structure of the input for BERT being such that the query and the document are separated by the separator token. Thus the input consists of the classifier token followed by our query which is in turn followed by the first separator token then the corresponding document is added and finally the second separator token is appended. Furthermore, it should be mentioned here that we predict the importance labels with the help of BERT. Additionally, it is worth mentioning that we predict the importance labels with the help of BERT. So we obtain 0, 1, or 2, whereby these values represent the classes irrelevant, relevant, and very relevant.

Our goal was to train BERT so that we obtain the appropriate relevance scores as a result. As base model, we use bert-base-cased. SimpleTransformers uses Hugging Face in the background to provide the corresponding pre-trained models [24]. It would be therefore straightforward to use other models instead of BERT or bert-base-cased. The decision to use specifically this model is mainly due to the hardware we have available. The chosen parameters for the Fine-tuning process were determined based on the cross-entropy, and we used the parameters that provided the best cross-entropy.

The most important parameters are the learning rate, which is  $1 * e^{-8}$ . The learning rate that is applied only to the classification layer which is  $1 * e^{-5}$ , the number of epochs that we set to 10, and the batch size which is 4. From the two different learning rates you might already see that we make almost no changes to the main model of BERT. This turned out to produce a better cross-entropy than training the whole model with a higher learning rate. We decided not to use the option of stopping the training earlier since this led to a deterioration of the cross-entropy in our experiments. Because the classes differ significantly in the number of associated data, we weighted each class with  $1 - w_c$ , where  $w_c$  is the relative frequency of class  $c$ . For the training, we used the evaluated documents of last year's task [2].

#### 4.4. Combining the individual scores

To combine the scores, we use a support vector machine. This allows us to determine a weighting for the individual scores based on the documents already evaluated last year [2]. As input for the SVM, we use the previously calculated scores and the score provided by ChatNoir with the request. If necessary, individual scores can be included or left out from the calculation of the final score. However, this does not apply to the score provided by ChatNoir, which is always used.

Since this is not just a classification problem, but the task is to compute a score, we use support vector regression. This means we get real numbers from our SVM and not labels as it was the case for example with BERT in chapter 4.3. For the implementation, we utilize Scikit-learn and keep the default settings [25]. Important is that we normalize the scores used as inputs by using the z-score standardization. Furthermore, these normalized values are then mapped by a sigmoid function onto the value range between zero and one. The standardization described and used in the implementation is defined in more detail in equation (1).

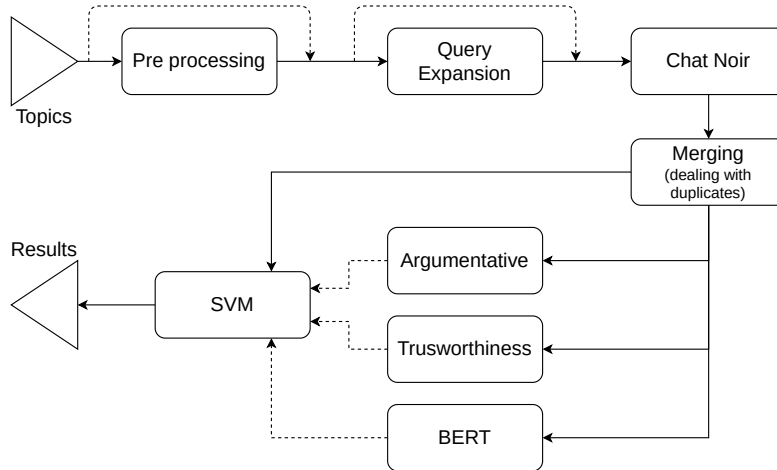
$$normalized\_score = sigmoid\left(\frac{input - \mu_i}{sd_i}\right) \quad (1)$$

$$sigmoid(x) := \begin{cases} \frac{1}{1+e^{-x}} & \text{if } x \geq 0 \\ \frac{e^x}{1+e^x} & \text{else} \end{cases} \quad (2)$$

where:  $input$  = a value belonging to one of the scores that is to be normalized  
 $\mu_i$  = the previously calculated mean for the score  
 $sd$  = the previously calculated standard deviation for the score

This is firstly to take advantage of the generally expected improved performance of an SVM on standardized data [26]. Second, the sigmoid function in particular is intended to catch outliers that could potentially overshadow the other scores, thus ensuring that all scores can exert their attributed influence on the final result. The result we receive from the SVM is used as our final score. If individual values are not available, these are set to the mean and then processed as usual.





**Figure 3:** Schematic illustration of the complete pipeline. The dashed lines show possible alternative routes or paths that can be disabled. Therefore, individual scores which are given to the SVM can be activated or deactivated. The only score that is always used is the one from ChatNoir. So the SVM can utilize at least one and at most four scores.

## 5. Complete layout of our approach

The pipeline presented here consists of parts that have been already discussed in sections 3 and 4. Those parts can be individually activated. On the one hand, this allows us to examine the different segments separately in the context of the complete pipeline, but it also makes it possible to add further scores to this basis with minimal effort because the individual parts function independently of each other.

Figure 3 provides a rough overview of how our pipeline is structured. There we can see that we start with the topics which are supplied as a list of queries. Those are manipulated accordingly in the preprocessing and in the query expansion step. With ChatNoir we get potential documents for the queries. At the same time, ChatNoir provides us with a first score. This score is the only non-optional in the pipeline and is therefore always used. ChatNoir is followed by the already described Merging step. After this, the selected additional scores are calculated and the intermediate results are mapped to a final score using the support vector machine. This is then stored in the standard Trec file format.

Important to note here is how we take care of ties. If we have a tie between multiple documents, all documents except the first one will have  $\epsilon_{100}$  subtracted from their final score.  $\epsilon_{100}$  is defined as  $100 * \epsilon$ , where  $\epsilon$  is the smallest possible float. We cannot use  $\epsilon$  directly, because we would introduce rounding errors that would cause ties to persist.

## 6. Results

We use the  $nDCG@5$  to evaluate our pipeline in its entirety. The  $nDCG$  is here calculated with the help of `trec_eval` using the measurement `ndcg_cut` [27]. Furthermore, the normal  $nDCG$  in the following refers to the one that considers all documents and the judged one will refer to

**Table 1**

The nDCG@5 values for some selected configurations of our pipeline. The nDCG@5 values shown here are first shown for all components alone, these values are each separated by lines from the next category in which the number of active components is increased by one. The selection as to which combination are shown here was made on the basis of observations made during the development. The 'nDCG@5' value describes the standard variant of the nDCG@5. The 'nDCG@5 (J)' value, on the other hand, only takes evaluated documents into account. The best nDCG value is marked in both cases. The combinations submitted for evaluation are marked accordingly.

Prepro.	Queryex.	Argumen.	Trust.	BERT	nDCG@5	nDCG@5 (J)	Submitted
					0.5187	0.5801	
×					0.5084	0.5842	
	×				0.4909	0.5870	
		×			0.4655	0.6270	
			×		0.4996	0.5924	
				×	<b>0.5122</b>	0.5791	
×	×				0.4879	0.5870	×
		×		×	0.4595	<b>0.6516</b>	×
×	×	×			0.4169	0.6221	×
×	×			×	0.5040	0.5819	×
×	×	×		×	0.4164	0.6462	×

the version that only considers evaluated documents. This corresponds to the '-J' flag when using trec\_eval. The decision to consider the normal nDCG as well as the judged one was made since it can happen that we rightly rank documents higher than the approaches of the last year, and thus we get a worse nDCG, although the order is better. The judged nDCG provides us with an impression of the extent to which the internal order among the evaluated documents has improved or worsened. Ideally, we could do without this approximation, but potential improvements might otherwise go unnoticed. In the first part, we will look at adding individual scores which then results in us utilizing a total of two scores. As a baseline, we use the two nDCG@5 values that ChatNoir achieves. These values can be found in the first row of the table 1. We will use these two values to determine the extent to which the ranking has improved or worsened.

The first thing to note here is that we always have a deterioration in the normal nDCG@5 compared to the baseline. This can be extended to all combinations of scores that we have examined in more detail and can therefore be seen in table 1. On the other hand, especially for the judged nDCG@5 adding only one score, it can be stated that we can observe an improvement for all combinations except for BERT. This noticeable difference will be further examined in one of the following passages. Here, the small change due to preprocessing or query expansion is to be expected, since by weighting the different origins of queries, the original ones are weighted with a factor of two and all others with a factor of 1.5 or 1. Due to this weighting, it is relatively unlikely that further documents slip upwards and the order in comparison to the baseline should remain mostly the same. Only strong outliers are likely to change the order, and in this case, ChatNoir itself would have classified them as a good fit.

The argumentative score worsens the normal nDCG@5 here but provides us with a significant

improvement in the judged one. This is also the largest judged nDCG@5 we have observed. Based on the observations already made during development, this fact is not a big surprise. The explicit value used for  $\theta$  was chosen so that we get an ideal nDCG@5. The nDCG@5 shown in Table 1 could thus vary quite a bit for unknown topics and might require a corresponding adjustment of the  $\theta$ .

The fact that the Trustworthiness Score has no or a relatively small influence was to be expected due to the very low correlation already described. Nevertheless, the improvement in the judged nDCG@5 should be emphasized here.

BERT falls somewhat out of the scheme here. On the one hand, adding the score computed with BERT gives the largest normal nDCG@5 among the combinations that added only one score, yet this value is still worse than the baseline. On the other hand, BERT has the worst score on the judged nDCG@5 within the same group. A possible explanation for this conspicuousness is that, compared to the other approaches, BERT might assign higher scores to documents that have already been evaluated, and assigns comparatively few non-evaluated documents higher scores. This is especially plausible if one takes into account that BERT's training was based on the evaluated documents of the last year and that BERT therefore should know the correct solutions. Taking this into account, the deterioration in both nDCG@5s compared to the baseline is unexpected.

We now proceed with the further combination based on the order of table 1. Therefore, the first thing that follows is preprocessing together with the query expansion. Again, these two preprocessing steps together produce a relatively small change due to their respective weights. Thus, the normal nDCG@5 lies between the two previous ones and the judged nDCG@5 is identical to the better one.

The combination of Argument Score and BERT is interesting at this point, as together they outperform the already well-performing judged nDCG@5 of only activating the Argument Score. In particular, this is noteworthy because based on the individually activated components, BERT does not appear to provide significant additional value. However, the improvement of the judged nDCG@5 shows that the information BERT can contribute benefits our pipeline. It is also important to note that this combination provides the best-judged nDCG@5 that we have observed in our experiment. Here it can be concluded that the information encoded by the argument score and BERT differ from each other. Furthermore, they complement each other in a way that benefits the ranking.

In the following, only combinations are considered in which the preprocessing component and the query expansion are activated. We start with the Argumentative Score. As expected from the observations already described, the Argumentative Score again performs well here and provides us with a judged nDCG@5 of over 0.6. However, this has worsened minimally compared to before by about 0.005. The normal nDCG@5 has worsened by  $\approx 0.05$ .

The combination of preprocessing, query expansion, and BERT show a degradation in normal nDCG@5 of  $\approx 0.008$  and an improvement in judged nDCG@5 of  $\approx 0.002$ .

A more significant change can be observed in the version that uses preprocessing, query expansion, argumentative score, and BERT. This has a significant degradation of  $\approx 0.07$  or  $\approx 0.04$  when considering the normal nDCG@5 compared to the two active components respectively. Interestingly, this combination also yields a worse judged nDCG@5. Comparing this combination with the versions that use preprocessing query expansion and the argument score

or BERT shows, as before, that BERT and argument score benefits from each other.

Based on the observations described, we have decided to submit the combinations marked in Table 1. The submission was done using TIRA. TIRA is a software solution that sandboxes the submitted software in a virtual machine and keeps a copy of it. Among other things, this would make it easier to use the submitted software again in the future [28].

## 7. Conclusion

As a first conclusion, our most promising approach is the combination of the argument score with BERT. This combination is followed very closely by these two scores together with preprocessing and query expansion. In particular, the argument score always prevails as the most important score in our approach and plays a role in all substantial improvements.

Furthermore, based on our observations, an order of utility can be established for the individual areas of our software. In this order, the argument score would undoubtedly be the most important. It would be followed by BERT, even if BERT in particular only provides an improvement as an extension in conjunction with the argument score. After BERT there would follow the Trusworthiness and last but not least would come Preprocessing and query expansion.

Another finding is that the pipeline presented here achieves an improvement in ranking among the evaluated documents. However, this result should be taken with a grain of salt, since some parts are designed to perform well in the problem studied here. In particular BERT as well as the SVM are addressed here. The problem that arises from this is that the parameters learned may not be applicable to new problems.

Nevertheless, we can state that we deliver several combinations through our pipeline that have the potential to outperform the baseline based on our observations.

For the future, our approach can serve as a foundation and can be extended relatively easily with new scores. This is particularly evident in the fact that many parts of the pipeline can act independently of each other and the exact number of scores is not fixed. If our formalities are complied to, the internal data structure would only have to be extended to include the new score and all other components would take this additional information into account and include it in their calculations.

## A. Code and data availability

Our complete code is available at <https://github.com/JanNiklasWeder/Touche-21-Task-2>. All necessary data will be downloaded automatically if not already locally available in the respective folder.

## References

- [1] M. Potthast, M. Hagen, B. Stein, J. Graßegger, M. Michel, M. Tippmann, C. Welsch, Chat-Noir: A Search Engine for the ClueWeb09 Corpus, in: B. Hersh, J. Callan, Y. Maarek, M. Sanderson (Eds.), 35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 2012), ACM, 2012, p. 1004. doi:10.1145/2348283.2348429.

- [2] 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: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéal (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, number 2696 in CEUR Workshop Proceedings, Aachen, Germany, 2020. URL: [http://ceur-ws.org/Vol-2696/paper\\_261.pdf](http://ceur-ws.org/Vol-2696/paper_261.pdf).
- [3] M. Lippi, P. Torroni, Argumentation mining: State of the art and emerging trends, *ACM Trans. Internet Technol.* 16 (2016). URL: <https://doi.org/10.1145/2850417>. doi:10.1145/2850417.
- [4] A. Chernodub, O. Oliynyk, P. Heidenreich, A. Bondarenko, M. Hagen, C. Biemann, A. Panchenko, TARGER: Neural Argument Mining at Your Fingertips, in: M. Costajussà, E. Alfonseca (Eds.), 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), Association for Computational Linguistics, 2019, pp. 195–200. URL: <https://www.aclweb.org/anthology/P19-3031>.
- [5] L. Zhang, J. Li, C. Wang, Automatic synonym extraction using word2vec and spectral clustering, in: 2017 36th Chinese Control Conference (CCC), 2017, pp. 5629–5632. doi:10.23919/ChiCC.2017.8028251.
- [6] N. Kanhabua, K. Nørvåg, Quest: Query expansion using synonyms over time, in: J. L. Balcázar, F. Bonchi, A. Gionis, M. Sebag (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 595–598.
- [7] C. Fellbaum (Ed.), WordNet: An Electronic Lexical Database, Language, Speech, and Communication, MIT Press, Cambridge, MA, 1998.
- [8] F. Yan, Q. Fan, M. Lu, Improving semantic similarity retrieval with word embeddings, *Concurrency and Computation: Practice and Experience* 30 (2017). doi:10.1002/cpe.4489.
- [9] N. Mohammed, Extracting word synonyms from text using neural approaches, *The International Arab Journal of Information Technology* (2019) 45–51. doi:10.34028/iajit/17/1/6.
- [10] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, 2013. arXiv:1310.4546.
- [11] V. M K, K. K, A survey on similarity measures in text mining, *Machine Learning and Applications: An International Journal* 3 (2016) 19–28. doi:10.5121/mlaij.2016.3103.
- [12] C. Carpineto, G. Romano, A survey of automatic query expansion in information retrieval, *ACM Comput. Surv.* 44 (2012). URL: <https://doi.org/10.1145/2071389.2071390>. doi:10.1145/2071389.2071390.
- [13] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL: <https://doi.org/10.5281/zenodo.1212303>. doi:10.5281/zenodo.1212303.
- [14] A. Trask, P. Michalak, J. Liu, sense2vec - a fast and accurate method for word sense disambiguation in neural word embeddings, 2015. arXiv:1511.06388.
- [15] explosion, sense2vec: Contextually-keyed word vectors, <https://github.com/explosion/sense2vec>, 2018.
- [16] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, M. F. Jeff Kaufman, M. El-Bachouti, R. Belvin, A. Houston, Ontonotes release 5.0, <https://catalog.ldc.upenn.edu/LDC2013T19>, 2013.

- [17] Reddit comments corpus, <https://files.pushshift.io/reddit/comments/>, 2016.
- [18] I. Habernal, I. Gurevych, Argumentation mining in user-generated web discourse, *Computational Linguistics* 43 (2017) 125–179. URL: <https://www.aclweb.org/anthology/J17-1004>.
- [19] I. Zaihrayeu, P. P. da Silva, D. L. McGuinness, IWTrust: Improving user trust in answers from the web, in: *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2005, pp. 384–392. doi:10.1007/11429760\_27.
- [20] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web., Technical Report 1999-66, 1999. URL: <http://ilpubs.stanford.edu:8090/422/>, previous number = SIDL-WP-1999-0120.
- [21] A. T. P. Ltd, Open page rank, <https://www.domcop.com/openpagerank/>, ???? Accessed: 2021.
- [22] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, *CoRR* abs/1810.04805 (2018). URL: <http://arxiv.org/abs/1810.04805>. arXiv:1810.04805.
- [23] T. C. Rajapakse, Simple transformers, <https://github.com/ThilinaRajapakse/simpletransformers>, 2019.
- [24] 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, Online, 2020, pp. 38–45. URL: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.
- [25] 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, Édouard Duchesnay, Scikit-learn: Machine learning in python, *Journal of Machine Learning Research* 12 (2011) 2825–2830. URL: <http://jmlr.org/papers/v12/pedregosa11a.html>.
- [26] S. Ali, K. A. Smith-Miles, Improved support vector machine generalization using normalized input space, in: A. Sattar, B.-h. Kang (Eds.), *AI 2006: Advances in Artificial Intelligence*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 362–371.
- [27] C. Macdonald, I. Soboroff, B. Gamari, trec\_eval version 9.0.8, GitHub, 2020. URL: [https://github.com/usnistgov/trec\\_eval](https://github.com/usnistgov/trec_eval).
- [28] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA Integrated Research Architecture, in: N. Ferro, C. Peters (Eds.), *Information Retrieval Evaluation in a Changing World*, The Information Retrieval Series, Springer, Berlin Heidelberg New York, 2019. doi:10.1007/978-3-030-22948-1\_5.