Development of an IR System for Argument Search

Marco Alecci[a], Tommaso Baldo[a], Luca Martinelli[a] and Elia Ziroldo[a]

[a]*University of Padua, Italy*

Abstract

Search engines are the easiest way to find the information that we need in our daily life, and they have became more and more powerful in the last years. Anyway, they are still far from perfection, and some problems afflict also the more advanced search engines. In this paper we discuss our approach to the problem of argument retrieval documenting our participation to the CLEF 2021 Touché Task 1. In particular, we present our IR system for the args.me corpus, a collection of documents extracted from web debate portals. After a pre-processing phase of the documents, we tried to use different methods like query expansion and re-ranking based on sentiment analysis. In the final part we report the results of our experiments and discuss about them and about other possible strategies that can be applied in the future.

Keywords

Information Retrieval, Search Engine, Argument Retrieval

1. Introduction

In the last decade, our everyday life has became more and more strictly connected to the web and the use of search engines is one of the most common tasks in our daily routine. Indeed they are the easiest and most reliable way to get information about anything we need, but unfortunately they are still far from perfection. One of the problems that afflict search engines concerns the retrieval of arguments, that according to previous existing works, could be defined as a single conclusion supported by one or more premises [1].

To give our contribution to the resolution of this problem, we decided to participate to the Touché 2021 Lab on argument retrieval [1] proposed by CLEF[2] because we believe that argument retrieval is a crucial feature , especially in these days, when the web sources such as social media community and blogs are growing faster and faster. Among two different Tasks proposed from Touchè Lab we decided to take part to Task 1 that regards argument retrieval from debates on controversial topics. The dataset is the one used by the argument search engine args.me [2] and we chose to use the downloadable corpus.[3]

"Search Engines", course at the master degree in "Computer Engineering", Department of Information Engineering, University of Padua, Italy. Academic Year 2020/21

marco.alecci@studenti.unipd.it (M. Alecci); tommaso.baldo@studenti.unipd.it (T. Baldo); luca.martinelli.1@studenti.unipd.it (L. Martinelli); elia.ziroldo@studenti.unipd.it (E. Ziroldo)

[1]https://webis.de/events/touche-21/
[2]http://www.clef-initiative.eu/
[3]https://zenodo.org/record/3734893

The paper is organized as follows: in section 2 we describe some related works for what concern argument retrieval. Instead, in section 3 we examine our approach to solve the task. After the pre-processing of the documents we tried to implement three different strategies : different weights for the fields of a document, query expansion with synonyms from WordNet[4] and re-ranking with sentiment analysis. To select the parameters and the weights used by our methods, we relied on the scores obtained from our system using the topics from Touché 2020 Lab. Going further, in section 4 we describe the experimental setup we used during our works, meanwhile section 5 is for result analysis. Finally section 6 is about our final considerations and discussions for possible future works.

2. Related Work

Different previous studies has been carried out to try to resolve the problem of argument search, but our starting point was the overview of the last year edition of the Touché Lab [3]. The common approach followed by the participating teams was constituted by three main parts: (1) a retrieval strategy; (2) an augmentation component like query expansion (3) a re-ranking component which modifies the score of the initially retrieved documents.

The most two used model were BM25 and LMDirichlet, while other few teams used DPH or TFIDF. The argument search engine args.me [4], from which the corpus was extracted, is based on the retrieval model BM25. Anyway, previous studies compared different retrieval models and demonstrated how LMDirichlet and DPH are better suited for argument retrieval [5].

2.1. Pre-processing

A fundamental step for argument retrieval is the pre-processing of the documents. One possible approach presented by Dumani et al. [6] is to group premises that support the same conclusion. After this is possible to calculate a score that indicate how much a premise is convincing in comparison to other premises of the same claim. The solution provided by Bundesmann et al. [7] uses a machine learning approach to process the initial documents and assign them a score indicating their argumentative quality. According to Wachsmuth et al. [8] they computed a score for each one of these three aspects: Logical quality, Rhetorical quality and Dialectical quality. The approach followed by Staudte et al. [9] regards primarily the pre-processing of the words instead of the whole documents. They started with basic things such as removing punctuation, URL and square brackets, but then they also introduced more specific rules such replacing a repetition (>2) of the same letter with a single one. Indeed, in blogs and social media users frequently write in colloquial language repeating the same letter more than once. They also deleted arguments smaller than 26 words since users make short arguments to express agreement or disagreement with a previous argument rather than to express their own reasons.

[4] https://wordnet.princeton.edu/

2.2. Query expansion

A query represent the lack of information by a user, but usually they are a bit too short to find the most relevant documents. For example, due to a vocabulary mismatch the *Information Retrieval (IR)* system can discard a document that in reality was relevant. To avoid this problem query are often expanded with more terms to reduce the gap between the query and concepts that users wanted to express. The approach was followed by Akiki et al. [10] make use of GPT-2 model [11] to add argumentative text to the original query. Then a new set of queries is built from the generated sentences. Another possible solution is the use of lexical properties to add new terms to the original query. Bundesmann et al. [7] implement this strategy by adding synonyms taken from WordNet database.

2.3. Re-ranking

After the retrieval of the most relevant documents, an IR system can re-rank the candidates to consider additional criteria that can involve different features of the documents. In the paper provided by Shahshahani et al. [12], they described how their final ranking is produced using the learning-to-rank library RankLib[5] to incorporate argument quality and Named Entity Recognition. Their assumption is that recognize entities means that the premises are more persuasive and effective. Another possible strategy to follow is the use of sentiment analysis to determine how much a user is emotionally involved. Indeed to deal with argument retrieval, it is crucial to be able to understand the emotions and the writer's frame of mind. Since several studies [2] underline that an emotional argument is more powerful than a neutral or impassive one, Staude et al. [9] decided to encourage the emotional documents combining their DPH score with the one calculated with sentiment analysis. By contrast, another team from the previous edition of Touché, decided to assign an higher score to the neutral arguments, assuming that a neutral sentiment coincides with higher relevance of a document.

3. Methodology

As a starting point, we pre-processed all the documents contained in the args.me corpus, removing stop words and applying different filters. To create the index and to perform the search we relied on Apache Lucene.[6] Since BM25 and LMDirichlet were the most used models in the previous edition of Touché Lab, and since also args.me search engine relies on BM25 we decided to use the Lucene's implementation of these two models. Then we tried three different methods to improve the performance of our IR system:

- Assigning different weights to different fields of the documents.
- Query expansion using synonyms extracted from WordNet.
- Re-ranking using the score obtained by performing sentiment analysis on the documents.

[5]https://sourceforge.net/p/lemur/wiki/RankLib
[6]https://lucene.apache.org/

First, we followed each one of these strategies separately to find the best parameters/weights to use with them. After, we tried to combine all the three techniques at the same time to see the effects with respect to the base implementation.

3.1. Pre Processing

Our approach in creating Lucene Documents[7] was to store different information in independent fields, in order to assign distinct weights to each field. Additionally to the field that store the identification number of the document and the field that store the stance of the document, we decided to create three other fields for the premises, the conclusion and the body. The body field, in particular, contains both premises and conclusion, and extra information about the document. These information are, in order, acquisition time, source URL, topic, author, author role and author organization, source domain and discussion title. We decided to not keep the source text because we noticed that it contains too much useless terms, such as copyright information, navigation menus, site map etc.…

We decided to adopt the `ClassicTokenizer`[8] provided by Apache Lucene. This is a simple grammar-based tokenizer constructed with the lexical analyzer generator JFlex. It's designed to be a good tokenizer for most European-language documents: it splits words at punctuation characters, removing them. However, a dot that's not followed by a whitespace is considered part of a token. As a result it splits words at hyphens, unless there's a number in the token, in which case the whole token is interpreted as a product number and is not split. It recognizes email addresses and internet hostnames as one token.

Beyond this we implemented the `LowerCaseFilter`,[9] in order to normalize all tokens to lower case. This also allows terms of the query to match with terms in the documents written, for example in upper case. The next filter we used is the `LengthFilter`.[10] This filter keeps tokens with a length between 3 and 20 characters, removing the others. A significantly improvement on the score has been noted, due to the exclusion of many words not informative, such as I, be, me, a etc.… The last filter we applied is a custom filter that excludes equal consecutive letters if they're more than three. This filter is useful to remove typos or words emphasized e.g. *helllo* or *yesssss* becames relatively *hello* and *yess*.

3.1.1. StopLists

Stopword filtering is a common step in preprocessing text because it removes lots of not informative words. We realized that stoplists have a considerable impact on the nDCG@5 score. So we tried different lists, as reported in Tab. 1 and Tab. 2. The max score were obtanied with the stoplist EBSCOhost:[11] it is a list of 24 words used in EBSCOhost medical databases

[7] https://lucene.apache.org/core/8_8_1/core/org/apache/lucene/document/Document.html

[8] https://lucene.apache.org/core/8_8_1/analyzers-common/org/apache/lucene/analysis/standard/ClassicTokenizer.html

[9] https://lucene.apache.org/core/8_8_1/analyzers-common/org/apache/lucene/analysis/core/LowerCaseFilter.html

[10] https://lucene.apache.org/core/8_8_1/analyzers-common/org/apache/lucene/analysis/miscellaneous/LengthFilter.html

[11] https://connect.ebsco.com/s/article/What-are-the-stop-words-used-in-EBSCOhost-medical-databases-MEDLINE-and-CINAHL

MEDLINE and CINAHL. In general we saw that lists with more words generally decrease the score. In fact using an empty stoplist the score was overall on the average. After that we tried to create a stoplist with the 150 most frequent terms in the index (150 custom). We found that it has an average score, too. Then we have integrated EBSCOhost with the ten, twenty and thirty most frequent terms not yet present in the stoplist. The score with the first two attempts was just a little lower than stock EBSCOhost, and it decreases significantly adding more words (e.g. EBSCOhost+30), confirming that in this situation a small stoplist is the best solution.

Stock stoplists	Number of words	nDCG@5
tent1	400	0.5599
Air3z4	1298	0.5757
zettair	469	0.5790
smart	571	0.5895
terrier	733	0.5919
cook1988	221	0.6043
taporwave	485	0.6068
postgre	127	0.6078
nltk	153	0.6078
lexisnexis	100	0.6131
NO STOPLIST	0	0.6189
corenlp	28	0.6211
okapi	108	0.6224
ranksnl	32	0.6249
lucene_elastic	33	0.6256
ovid	39	0.6259
lingpipe	76	0.6260
EBSCOhost	24	**0.6265**

Table 1: nDCG@5 scores obtained with different stock stoplists.

Custom stoplists	Number of words	nDCG@5
150_custom	150	0.6066
ebsco+10	34	0.6258
ebsco+20	44	0.6258
ebsco+30	54	0.6123

Table 2: nDCG@5 scores obtained with custom stoplists.

3.1.2. Stemmers

Stemming is the reduction of a word into its base form, called stem. In particular we tried in four different ways, synthesized in Tab. 3. First of all we didn't use any form of stemming.

After that we tried to implement three different stemmers included in Lucene package. We started with the `EnglishMinimalStemFilter`,[12] that simply stems plural English words to their singular form. Then, in the second way we used the `KStemFilter`.[13] This filter implements the Krovetz stemmer, an hybrid algorithmic-dictionary that produces words. For the last, we tried the most used filter in IR, the Porter stemmer, implemented in Lucene as `PorterStemFilter`,[14] that eliminates the longest suffix possibile, working by steps and trying to delete each suffix every time, until it reaches the base form for generate stems. As already seen in section 3.1.1, adding complexity to the system, the score obtained decreases, this probably due to limitations of stemmers used [13].

Stem Filter	nDCG@5
No Stem	**0.6265**
English Minimal Stem	0.6184
Krovetz Stem	0.5747
Porter Stem	0.5401

Table 3: nDCG@5 scores obtained using different stemmers.

3.2. Different fields' weights

Since documents have more than one field to search in, at query time it is possible to assign different weights to each field. In this way, a term found in a field with an higher weight, will also have an higher impact on the final score of the document. As already explained in section 3.1, we decided to have three different fields containing respectively the body, the premises and the conclusion. We noticed that premises are the most informative field, instead the conclusions are often composed by one single term, and very rarely this is relevant. According to these considerations, the best score would be obtained assigning an higher weight to the premises, and a lower one to the body and the conclusions.

To choose the best values, we wrote a Python program that automatically calculates, using trec_eval, the nDCG@5 among all possibilities of weights (to each field) from 0 to 1, with a step of 0.25. The best five combinations of weights are in the Tab. 4, using BM25 as similarity, and in the Tab. 5, using LMDirichlet similarity. In the Tab. 17 in section 7 are listed all the tried combinations for both similarities. These results are in agreement with the previous considerations and they confirm our theories.

3.3. Query Expansion

Query expansion is a technique used to match more relevant documents, by expanding or reformulating the basic search query. To improve the retrieval performances of our model we

[12] https://lucene.apache.org/core/8_8_1/analyzers-common/org/apache/lucene/analysis/en/EnglishMinimalStemFilter.html
[13] https://lucene.apache.org/core/8_8_1/analyzers-common/org/apache/lucene/analysis/en/KStemFilter.html
[14] https://lucene.apache.org/core/8_8_1/analyzers-common/org/apache/lucene/analysis/en/PorterStemFilter.html

Body	Premises	Conclusions	nDCG@5
0.0	1.0	0.25	**0.4150**
0.25	1.0	0.25	0.4143
0.5	1.0	0.25	0.4032
0.5	0.75	0.25	0.4029
0.25	0.75	0.25	0.4023

Table 4: nDCG@5 scores obtained with different fieds' weights and BM25.

Body	Premises	Conclusions	nDCG@5
0.25	1	0	**0.7379**
0	1	0	0.7345
0.25	0.75	0	0.7331
0.5	1	0	0.7239
0.5	0.75	0	0.7123

Table 5: nDCG@5 scores obtained with different fields' weights and LMDirichlet.

tried to integrate query expansion in our IR system by adding to a query all the synonyms of the terms that are left after the pro-processing phase. In particular, we decided to use WordNet: a lexical database of semantic relations between words. In fact the `SynonymMap`[15] object of the WordNet package allows to load the file downloaded from WordNet[16] into an hash map that can be used for fast high-frequency lookups of synonyms. We decided to assign a different weight to the synonyms added at query time to give them more or less importance in the search. We tried different values and the results are reported in Tab. 6. As can be noticed, using BM25 similarity with a weight of 0.4 to the synonyms there is an increase of the evaluated score. On the contrary, using LMDirichlet similarity adding synonyms brings no improvement. This probably is caused by an increment of noise that causes matches with non relevant documents, decreasing the final score.

[15]https://lucene.apache.org/core/8_8_1/api/contrib-wordnet/org/apache/lucene/wordnet/SynonymMap.html
[16]https://wordnet.princeton.edu/download

	nDCG@5	
Synonyms Weight	**BM25**	**LMDirichlet**
No synoynms	0.3938	**0.7345**
0.1	0.4113	0.6986
0.2	**0.4159**	0.6483
0.3	0.3973	0.5913
0.4	0.3898	0.5267
0.5	0.3764	0.4731
0.6	0.3596	0.4273
0.7	0.3304	0.3847
0.8	0.2931	0.3406
0.9	0.2584	0.2892
1.0	0.2253	0.2564

Table 6: nDCG@5 scores obtained with different weight to synonyms in query expansion.

3.4. Re-ranking

In the last step, we re-ranked the top 30 documents retrieved from the previous phase performing a sentiment analysis on the arguments. To perform the analysis we used the VADER tool [14] and in particular the Java port provided by Animesh Pandey on Github [17]. This tool allows to compute a value between -1 and 1 for each argument. Values greater than 0 represent a positive sentiment from the author, while values lower than 0 indicate negativity. The values that are closer to 0 express neutral sentiment.

We decided to try two different approaches to do the re-ranking:

1. Promote emotional documents combining the score from the previous phase with the sentiment analysis score, using the Eq. 1

$$\frac{1}{3} * Score + \frac{2}{3} * |Sentiment| * Score \tag{1}$$

2. Promote neutral documents instead of emotional ones, using the Eq. 2

$$\frac{1}{3} * Score - \frac{2}{3} * |Sentiment| * Score \tag{2}$$

We tried to re-rank both with sentiment score computed on premises and on conclusions to see which strategy could be the right one. The results are provided in Tab. 7.

[17] https://github.com/apanimesh061/VaderSentimentJava

As we can see, with the sentiment scores computed on the conclusion the scores decrease drastically with both approaches and with both models. This is probably due to the fact that conclusions are often composed by few words (sometimes only one) and so the sentiment score doesn't perfectly express the true sentiment of the author. Using the sentiment scores computed on the premises, the scores drops almost to zero when we give more importance to neutral documents, while we can see a little improvement of the score (BM25) and almost the same value (LMDirichlet) when we promote emotional documents. Hence we can affirm that an higher absolute value of the sentiment score leads to better argumentation and so to a general high relevance of the retrieved documents.

	nDCG@5			
	Sentiment on premises		**Sentiment on conclusions**	
	BM25	**LMDirichlet**	**BM25**	**LMDirichlet**
No sentiment	0.3938	**0.7345**	**0.3938**	**0.7345**
Neutral is better	0.0811	0.0569	0.0811	0.0569
Emotional is better	**0.4362**	0.6952	0.1423	0.1414

Table 7: nDCG@5 scores obtained with BM25 and LMDirichlet similarities and different configurations of sentiment analysis.

4. Experimental Setup

Touchè Task 1 offers us the possibilities to access the args.me corpus via the API of args.me search engine or downloading the file containing all the documents. We decided to download the entire corpus and in the particular the version updated to 2020-04-01.

4.1. Data Description

The updated version of the args.me corpus contains 387,740 arguments crawled from four debate portals (debatewise.org, idebate.org, debatepedia.org, and debate.org), and 48 arguments from Canadian parliament discussions. The arguments were extracted using heuristics that are designed for each debate portal. Each argument is identified by an ID an it is constituted by a conclusion and one or more premises. For each document there are also present some information about the context like the source URL, the title of the discussion and many others.

For what concern the topics, Touché Lab provided us 50 controversial topics (the query potentially issued by a user), Each topic has both pro and con relevant arguments present in the document collection.

4.2. Evaluation measures

We used the *Normalized Discounted Cumulated Gain (nDCG)* score with an evaluation depth of 5 since this is the same evaluation measure used by Touché Lab to evaluate our runs. In particular, we used the implementation provided by the trec_eval library,[18] to measure the performance of our IR system. The nDCG is the result of Eq. 3. Parameter b indicates the patience of the user in scanning the result lists, and usually it is a value of 2 for an impatient user, or 10 for a patient user. Since the result is not bounded in [0,1], it is necessary normalize the score dividing nDCG by the *Ideal Discounted Cumulated Gain (iDCG)*, provided by Touché Lab, as can be seen in Eq. 4.

$$DCG@5 = \sum_{n=1}^{5} \frac{relevance_n}{\max(1, \log_b(i+1))} \tag{3}$$

$$nDCG@5 = \frac{DCG@5}{iDCG@5} \tag{4}$$

4.3. Repository Organization

The source code is available on our BitBucket repository [19]. In the repository we can find the `runs` folder that contain our best runs respectively with BM25 and LMDirichlet for both the topics of the 2021 and of the 2020 Touché Lab edition. Here we can also find the `calcWeight.py` python script that we used to find the best weights to assign to the different fields of the documents. Inside the `src/main` folder there is the Java folder containing all the Java source code, and the `resources` folder containing the stoplists and the WordNet database file.

```
- /runs/
    - Best runs for both 2020 and 2021 topics
- /src/main/
    - /java/it/unipd/dei/jpp/
        - analyze/
        - fields/
        - filter/
        - index/
        - parse/
        - search/
        - utils/
        - ToucheIR.java
    - /resources/
        - Stoplists, WordNet database file and OpenNLP files
- pom.xml
- JPPTouche.sh
- calcWeight.py
```

[18] https://trec.nist.gov/trec_eval/
[19] https://bitbucket.org/upd-dei-stud-prj/seupd2021-hg/src/master/

4.4. Hardware components

As follows, the devices that were used for experiments:

- ASUS VIVOBOOK PRO N580GD-E4087T (Processor: Intel® Core™ i7 8750H; Memory: 16GB DDR4 2400 MHz SDRAM; Video card: NVIDIA® GeForce® GTX 1050 4GB DDR5; Storage: SSD M.2 512 GB SATA 3.0 + HDD 2.5" 1 TB 5400 RPM
- ASUS VIVOBOOK PRO N580GD-E4085T (Processor: Intel® Core™ i7 8750H; Memory: 16GB DDR4 2400 MHz SDRAM; Video card: NVIDIA® GeForce® GTX 1050 4GB DDR5; Storage: SSD M.2 128 GB SATA 3.0 + HDD 2.5" 1 TB 5400 RPM

5. Results and Discussion

5.1. Results

As previously mentioned, we first tried all the different techniques separately to choose the best parameters/weights for each method, then we merged all together using the three strategies at the same time. In Tab. 8 we reported the best score achieved for each method and in the last line we show the score obtained by using all the techniques.

	nCDG@5	
	BM25	**LMDirchlet**
Base	0.3938	0.7345
Best different fields weights	**0.4698**	**0.8026**
Best query expansion with synonyms	0.4159	0.6986
Best re-ranking with sentiment analysis	0.4362	0.6952
Merging all three strategies	0.4521	0.6661

Table 8: nDCG@5 final scores obtained with the different presented strategies.

Looking at Tab. 8 we can notice that with the BM25 similarity all the three methods worked well increasing the score of the base case, but the score achieved with the combination of all the three techniques is not the best one. For what concerns the LMDirichlet model we can see that the final score is lower than the base one, while the best is the one that doesn't use query expansion and sentiment analysis. Hence, these two techniques have not worked well with the LMDirichlet model and so they leads to lower performance when merging all the techniques.

The score achieved with the query expansion is very low and it's almost half of the base one. One possible explanation to this phenomenon is the fact that there are too many synonyms for each word and this introduces noise that degrades the performance of the search. In fact according to previous studies [15] there is no way using only WordNet to select an appropriate subset of synonyms.

The sentiment analysis leads to a very small improvement for BM25 but for LMDirichlet the score is almost the same. Looking manually at the documents retrieved after the first phase, we discovered that almost all the documents in the top positions have an high sentiment value. According to this, the re-ranking probably doesn't work very well because the documents with an higher sentiment value are already marked as the most relevant ones. Hence it isn't possible to improve the nDCG@5 score because the top ranked documents are also the ones with the higher sentiment score.

Finally we can affirm that the LMDirichlet model is better than BM25 for argument retrieval confirming the results obtained by the teams of the previous edition of Touché Argument Retrieval Lab [3].

5.2. Statistical Analysis

We decided to perform a statistical analysis on the results discussed in previous sections. In particular, in section 5.2.1 we analyze the performance of the system when using query expansion with synonyms taken from Wordnet, because this technique was the less useful one. While, in section 5.2.2 we analyze the best results obtained for each method (Tab.8). Finally, in section 5.2.3 we perform an analysis between the best run obtained with BM25 and the best run obtained using LMDirichlet.

5.2.1. Query expansion with synonyms from Wordnet

In Fig. 1, we have the boxplots of the runs obtained by using BM25 and query expansions with synonyms taken from Wordnet (Tab. 6). By looking at the plot we can notice that most runs seem to have a normal distribution, while the last run presents an outlier. Furthermore, we can observe that the first seven runs look similar among them, while the last four can be considered as a part of a second group.

We perform ANOVA Test (Tab. 9) on the runs and we obtained a p-value of 0.000405. Since the p-value is lower than 0.05 we reject H0 (the null hypothesis that all the means are equal) meaning that there is at least one pair of runs that are significantly different. In fact, performing a multiple pairwise comparison (Tukey's HSD Test) (Tab. 10), we can notice that runs such as *BM25 Syn01* and *BM25 Syn10* are significantly different due to a p-value (0.015635) below 0.05. Since the use of query expansion with synonyms, does not leads to an highly significant difference (only 4 pairs are significantly different), it is better to avoid overhead by not including query expansion. Hence, we can confirm that synonyms do not work well and do not increase the performance as we stated in section 5.1.

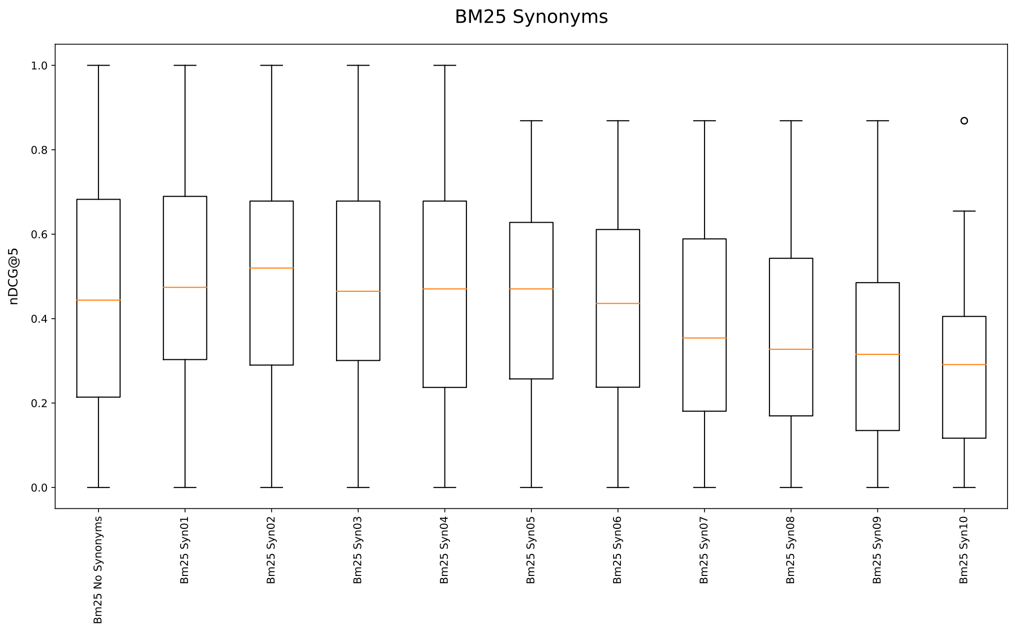

Figure 1: Boxplots of runs using BM25 and synonyms.

	df	**sum_sq**	**mean_sq**	**F**	**PR (>F)**
Runs	10	2.084216	0.208422	3.268949	0.000405
Residual	539	34.365564	0.063758	/	/

Table 9: ANOVA of runs using BM25 and synonyms.

group1	**group2**	**p-value**
Bm25 Syn01	Bm25 Syn10	0.015635
Bm25 Syn02	Bm25 Syn10	0.009431
Bm25 Syn03	Bm25 Syn10	0.020016
Bm25 Syn04	Bm25 Syn10	0.032616

Table 10: Pairwise T Test of runs using BM25 and synonyms and p-value lower than 0.05.

In Fig.2, we have the boxplots of the runs obtained by using LMDirichlet and query expansions with synonyms taken from Wordnet (Tab. 6). By looking at the plot we can notice that the runs do not seem to have a normal distribution. Indeed, the first three runs present outliers and the last three runs are not balanced due to IQR located in the lower portion of the graph.

Even if the boxplots do not seem distributed as a normal distribution, since we have 50 topics for all the runs, for the Central Limit Theorem we can consider ANOVA robust to violations of normality. Hence, we perform ANOVA Test (Tab. 11) on the runs and we obtained a p-value of $1.100593e-32$. Since the p-value is much lower than 0.05 we reject H0 (the null hypothesis that all the means are equal) meaning that there is at least one pair of runs that are significantly different. Without looking at the p-value, we can affirm this, also by looking at the boxplots, since the first three are clearly different from the last three. In fact, performing a multiple pairwise comparison (Tukey's HSD Test) (Tab. 12), we can notice that there are a lot of pairs that have a very low p-value. Due to this, we can say that using query expansion with synonyms we obtain a significant difference, but since the median of the run without synonyms is the highest one, it is better to not use query expansion as we already stated in section 5.1.

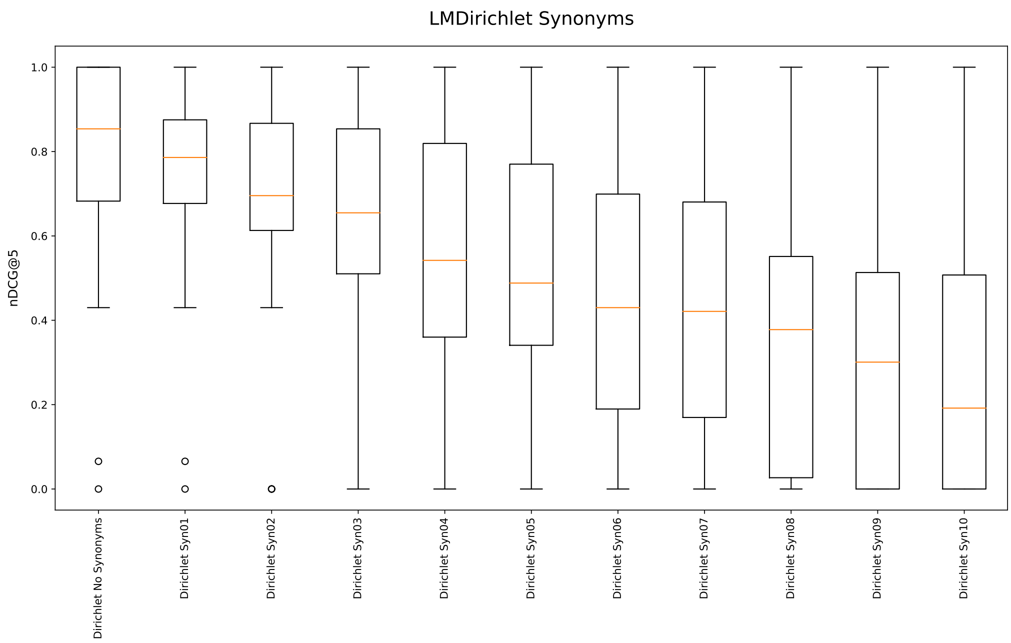

Figure 2: Boxplots of runs using LMDirichlet and synonyms.

	df	sum_sq	mean_sq	F	PR (>F)
Runs	10	15.28627	1.528627	20.743936	1.10e-32
Residual	539	39.719076	0.07369	/	/

Table 11: ANOVA of runs using LMDirichlet and synonyms.

group1	group2	p-value
Dirichlet No Synonyms	Dirichlet Syn04	0.001
Dirichlet No Synonyms	Dirichlet Syn05	0.001
Dirichlet No Synonyms	Dirichlet Syn06	0.001
Dirichlet No Synonyms	Dirichlet Syn07	0.001
Dirichlet No Synonyms	Dirichlet Syn08	0.001
Dirichlet No Synonyms	Dirichlet Syn09	0.001
Dirichlet No Synonyms	Dirichlet Syn10	0.001
Dirichlet Syn01	Dirichlet Syn04	0.012452
Dirichlet Syn01	Dirichlet Syn05	0.001
Dirichlet Syn01	Dirichlet Syn06	0.001
Dirichlet Syn01	Dirichlet Syn07	0.001
Dirichlet Syn01	Dirichlet Syn08	0.001
Dirichlet Syn01	Dirichlet Syn09	0.001
Dirichlet Syn01	Dirichlet Syn10	0.001
Dirichlet Syn02	Dirichlet Syn05	0.048064
Dirichlet Syn02	Dirichlet Syn06	0.001632
Dirichlet Syn02	Dirichlet Syn07	0.001
Dirichlet Syn02	Dirichlet Syn08	0.001
Dirichlet Syn02	Dirichlet Syn09	0.001
Dirichlet Syn02	Dirichlet Syn10	0.001
Dirichlet Syn03	Dirichlet Syn07	0.007336
Dirichlet Syn03	Dirichlet Syn08	0.001
Dirichlet Syn03	Dirichlet Syn09	0.001
Dirichlet Syn03	Dirichlet Syn10	0.001
Dirichlet Syn04	Dirichlet Syn08	0.027259
Dirichlet Syn04	Dirichlet Syn09	0.001
Dirichlet Syn04	Dirichlet Syn10	0.001
Dirichlet Syn05	Dirichlet Syn09	0.011902
Dirichlet Syn05	Dirichlet Syn10	0.001127
Dirichlet Syn06	Dirichlet Syn10	0.036596

Table 12: Pairwise T Test of runs using LMDirichlet and synonyms.

5.2.2. Best runs

In Fig. 3, we have the boxplots of the runs obtained by using BM25 with all the different techniques and with all of them merged together (Tab. 8). By looking at the plot we can notice that almost all the runs seem to have a normal distribution, also IQR are similar among them and the medians are almost aligned.

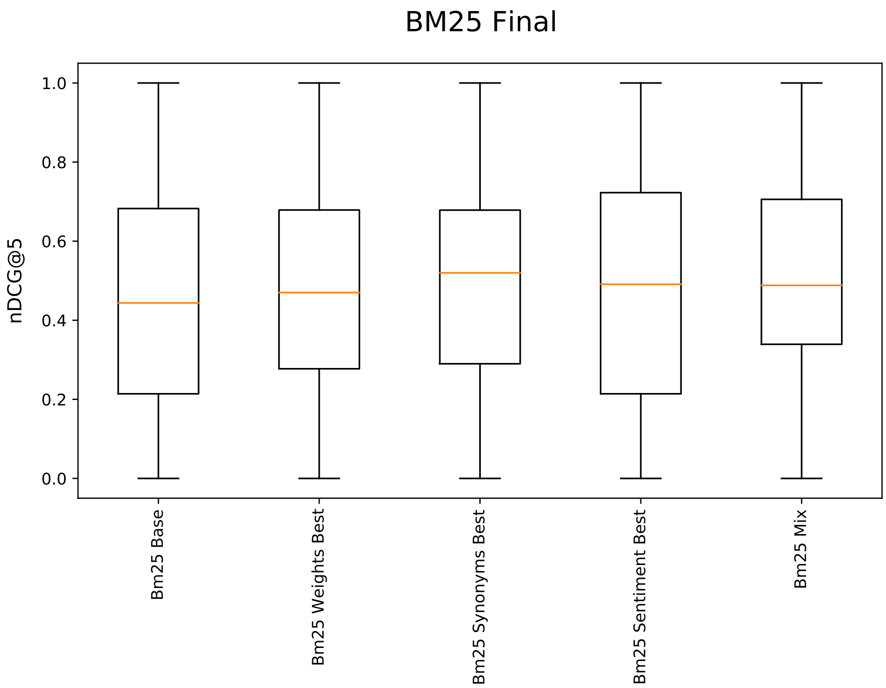

Figure 3: Boxplots of best runs with BM25.

We perform ANOVA Test on the runs and we obtained a p-value of 0.846705. Since the p-value is greater than 0.05 we cannot reject H0 (the null hypothesis that all the means are equal) and so we can not affirm that the runs are significantly different, confirming what observed in the plot. Furthermore, performing a multiple pairwise comparison (Tukey's HSD Test), we can notice that almost all the pairs have a very high p-value. Results are shown in Tab. 13 and Tab. 14.

	df	sum_sq	mean_sq	F	PR (>F)
Runs	4	0.104653	0.026163	0.345953	0.846705
Residual	245	18.528528	0.075627	/	/

Table 13: ANOVA of best runs with BM25.

group1	group2	p-value
Bm25 Base	Bm25 Weights Best	0.9
Bm25 Base	Bm25 Synonyms Best	0.9
Bm25 Base	Bm25 Sentiment Best	0.9
Bm25 Base	Bm25 Mix	0.749016
Bm25 Weights Best	Bm25 Synonyms Best	0.9
Bm25 Weights Best	Bm25 Sentiment Best	0.9
Bm25 Weights Best	Bm25 Mix	0.9
Bm25 Synonyms Best	Bm25 Sentiment Best	0.9
Bm25 Synonyms Best	Bm25 Mix	0.9
Bm25 Sentiment Best	Bm25 Mix	0.9

Table 14: Pairwise T Test of best runs with BM25.

In Fig. 4, we have the boxplots of the runs obtained by using LMDIRICHLET with all the different techniques and and with all of them merged together (Tab. 8). By looking at the plot we can notice that the runs do not seem to have a normal distribution because of the presence of outliers. Moreover, the first two runs are not symmetric, indeed their IQRs are located in the higher portion of the graph.

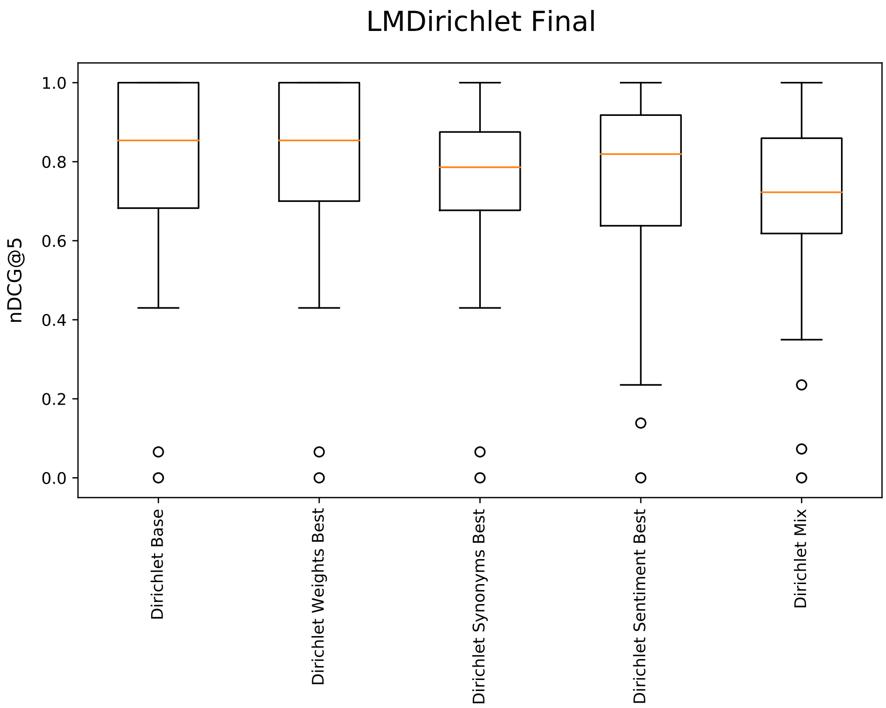

Figure 4: Boxplots of best runs with LMDIRICHLET.

Even if the boxplots do not seem distributed like a normal distribution, since we have 50 topics for all the runs, for the Central Limit Theorem we can consider ANOVA robust to violations of normality. We performed ANOVA Test on the runs and we obtained a p-value of 0.268872. Since the p-value is greater than 0.05 we cannot reject H0 (the null hypothesis that all the means are equal) and so we can not affirm that the runs are significantly different. Moreover, we can confirm this with a multiple pairwise comparison (Tukey's HSD Test), observing that all the pairs have a p-value higher than 0.05. Results are shown in Tab. 15 and Tab. 16.

	df	sum_sq	mean_sq	F	PR (>F)
Runs	4	0.265698	0.066425	1.30461	0.268872
Residual	245	12.474236	0.050915	/	/

Table 15: ANOVA of best runs with LMDirichlet.

group1	group2	p-value
Dirichlet Base	Dirichlet Weights Best	0.9
Dirichlet Base	Dirichlet Synonyms Best	0.9
Dirichlet Base	Dirichlet Sentiment Best	0.81592
Dirichlet Base	Dirichlet Mix	0.372886
Dirichlet Weights Best	Dirichlet Synonyms Best	0.829768
Dirichlet Weights Best	Dirichlet Sentiment Best	0.718983
Dirichlet Weights Best	Dirichlet Mix	0.279774
Dirichlet Synonyms Best	Dirichlet Sentiment Best	0.9
Dirichlet Synonyms Best	Dirichlet Mix	0.857612
Dirichlet Sentiment Best	Dirichlet Mix	0.9

Table 16: Pairwise T Test of best runs with LMDirichlet.

5.2.3. BM25 vs LMDirichlet

We compared the best run obtained using BM25 with the best run obtained using LMDirichlet. By looking at the boxplots, the runs look very different and the median of the LMDirichlet Run is higher than the BM25 one.

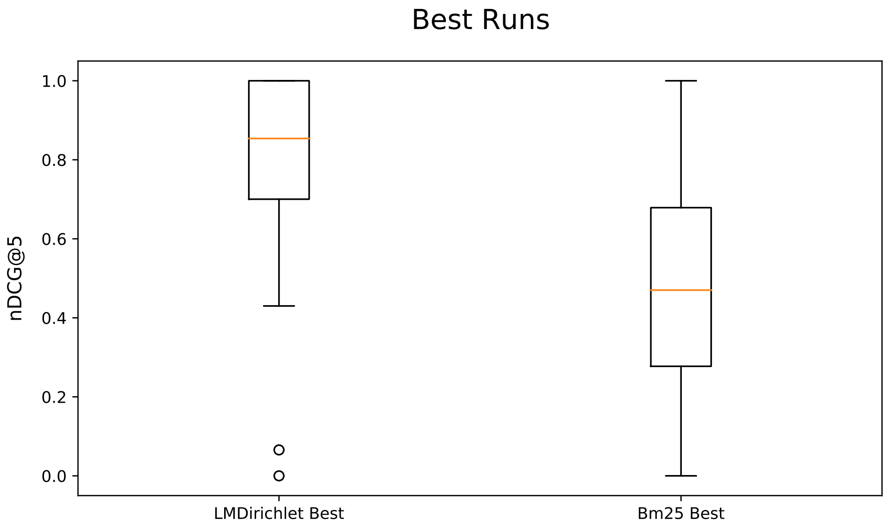

Figure 5: Boxplots of best runs with LMDirichlet and BM25

We then performed a T Student Test to compare the two runs and since the p-value obtained is very low ($6.168497e - 09$) and is below 0.05, we reject H0 (the null hypothesis that the two means are equal), so the run can be considered significantly different. Moreover, since the median of the run with LMDirichlet is higher, we can confirm that LMDirichlet model is better than BM25 one for argument retrieval as we stated in section 5.1.

5.3. Failure Analysis

We perform a failure analysis on the results discussed in previous sections to discover which topics not worked well for our system. First, we analyze the runs obtained using synonyms, since query expansion was the less useful technique that we used. Then, we look at the best runs that our system produced both with BM25 and LMDirichlet.

As a starting point, we analyze the best run with BM25 and synonyms (Fig. 6) and generally speaking we can affirm that BM25 performances are worse than the ones based on LMDirichlet. As we can see, this run registered a very low score in some topics. If we set aside topic 44 and 12 (where all the runs under-performed), we can focus our analysis on topic 8 and 40. In the first case, after reading the top 5 documents retrieved by this run, we can notice the trend to retrieve documents with short and not logical premises. Indeed, the phrases are short and it is not present a train of thought that make the argument a strong one. Moreover, if we compare these results with the top 5 of runs with LMDirichlet (which scored 1) the difference of argument quality is sparkly clear. The trend to retrieve document with short premises is even more apparent if we look at topic 40, where 4 out of 5 results are documents that consist only of two or three lines. Furthermore, they are basically responses to opponents and so they have short and straightforward premises, very poor from the argumentative and dialectical sides. As we will see later, these statements are also valid for the best run with BM25.

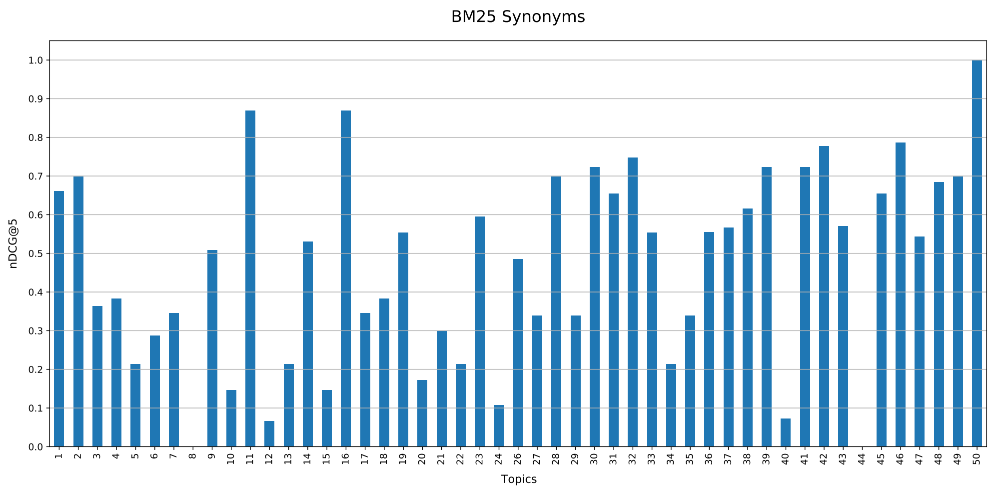

Figure 6: Failure analysis on best run with BM25 and synonyms

Then, analyzing the best run with LMDirichlet and synonyms (Fig.7), we can notice that these results are quite similar to the best run obtained with LMDirichlet (Fig. 9). In particular, this run scores better result than the other with LMDirichlet in the following topics: 23, 32, 34, 35, 42 and 47; but on the other hand, scores a slightly lower result in more other topics. For example, in topic 40 there is a relevant gap between this run and the best one with LMDirichlet because this run retrieved a document not present in the *qrels*. The same is also valid for topic 8: two documents retrieved using and synonyms are not present in the *qrels*.

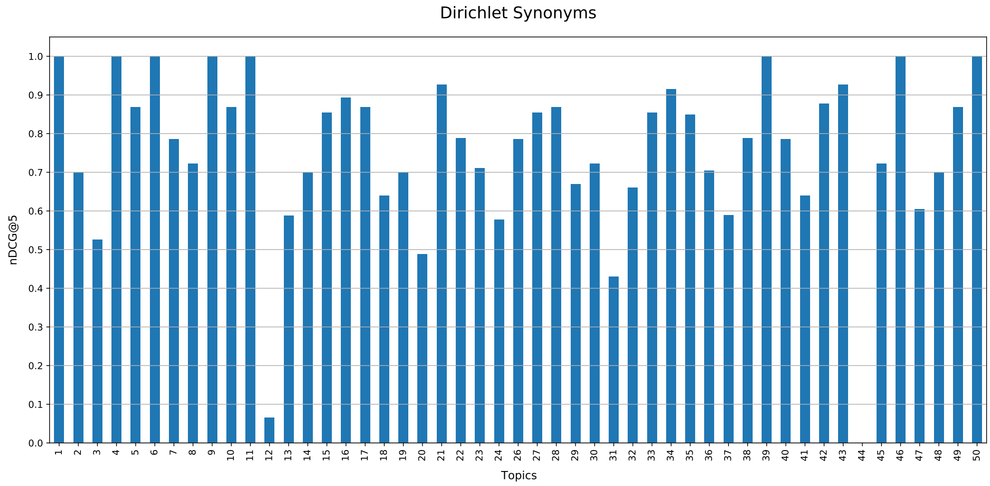

Figure 7: Failure analysis on best run with LMDirichlet and synonyms

In Fig. 8 we report the scores for the best run with BM25. The performances of this run are quite the same as the ones registered by BM25 using synonyms (Fig. 6). However, this run failed more topics than all the others, such as topic 19, for which this run scored 0 while the others scores are above the average of each run. Going deeper in the analysis, we can see that the top 5 documents retrieved by this run for topic 19 are related to it but totally lack of arguments. Indeed, three documents are responses to opponents and so they don't provide useful information, while the other two documents contain only some general considerations about the debate.

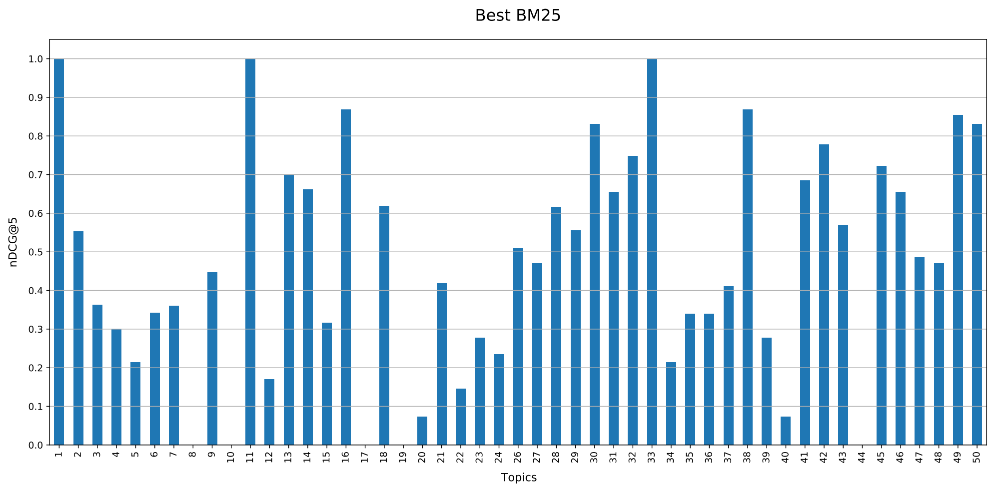

Figure 8: Failure analysis on best BM25 run

Finally, we analyze the best run with LMDirichlet (Fig. 9), which is the best among all the runs we produced. The two worst scores are for topic 44 and 12, as for the other runs. In the first case, topic 44 title is *"Should election day be a national holiday?"* and it is possible to notice that only two documents are judged as highly relevant in the *qrels* (one of these has a very short premise). We can also observe that in both documents the word *"election"* is often replaced by *"voting day"* or *"pooling day"*. All the top 5 document retrieved by the run are debates about a specific or "possible" holiday, such as *"Columbus Day Should Not be Celebrated"*, but they lack of the *"election part"*. Instead for what concerns topic 12, which title is *"Should birth control pills be available over the counter?"*, we can see that the top 5 documents are about the *"birth control"* but they are shallow debates and don't give useful information for the requested topic. Furthermore, in all of these documents, the words *"birth control"* are repeated very often.

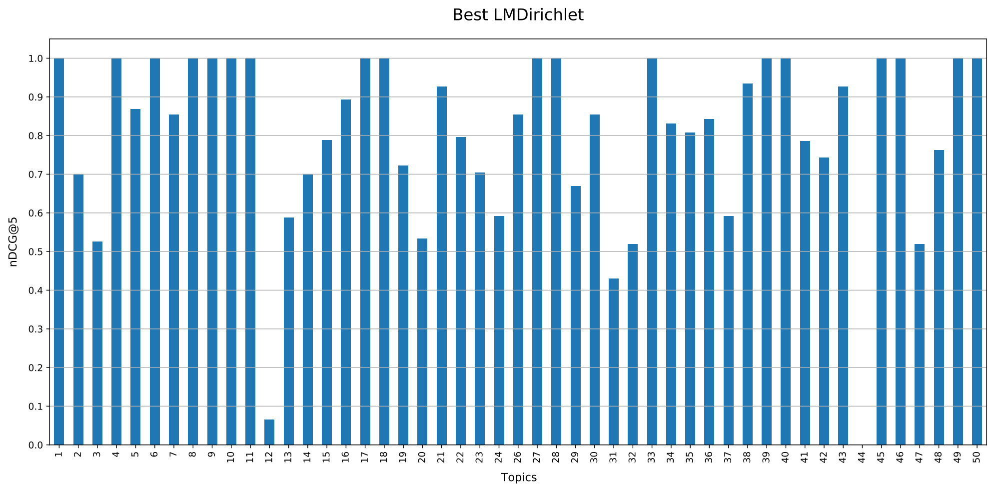

Figure 9: Failure analysis on best LMDirichlet run

6. Conclusions and Future Work

We implemented our IR system to retrieve the most relevant arguments to the given queries provided in the Touché shared task. We used both BM25 and LMDirichlet models, but we demonstrate that LMDirichlet is much better for what concern argument retrieval. We also show how much important is to give the right weight to the different parts of a document, since a lot of information can be useless during the search. Anyway there are some aspects that can be improved to reach better performances. For example instead of expanding the queries simply adding all the synonyms of a specific word, it would be better to associate a score to each synonym to indicate how much the two word are similar, and then use it to weight the different synonyms while performing the query. Another improvement can be done by using a better formula to re-rank the documents or maybe using a different score instead of the one retrieved with sentiment analysis. For example with a machine learning approach it would be possible to train a model to assign a quality score to each argument and then use this value to re-rank the top retrieved documents.

To conclude, we presented our approach to the problem of argument retrieval and we think that in the future always better solutions will be presented, especially with the help of machine learning.

References

[1] C. Lumer, Walton's argumentation schemes, OSSA Conference Archive (2016).

[2] H. Wachsmuth, N. Naderi, Y. Hou, Y. Bilu, V. Prabhakaran, T. A. Thijm, G. Hirst, B. Stein, Computational argumentation quality assessment in natural language, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, 2017, pp. 176–187.

[3] A. Bondarenko, M. Fröbe, M. Beloucif, L. Gienapp, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, et al., Overview of touché 2020: Argument retrieval, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2020, pp. 384–395.

[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: Proceedings of the 4th Workshop on Argument Mining, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 49–59. URL: https://www.aclweb.org/anthology/W17-5106. doi:10.18653/v1/W17-5106.

[5] M. Potthast, L. Gienapp, F. Euchner, N. Heilenkötter, N. Weidmann, H. Wachsmuth, B. Stein, M. Hagen, Argument search: Assessing argument relevance, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1117–1120. URL: https://doi.org/10.1145/3331184.3331327. doi:10.1145/3331184.3331327.

[6] L. Dumani, R. Schenkel, Ranking arguments by combining claim similarity and argument quality dimensions, Argument 2696 (2020). URL: http://ceur-ws.org/Vol-2696/paper_174.pdf.

[7] M. Bundesmann, L. Christ, M. Richter, Creating an argument search engine for online debates (2020).

[8] H. Wachsmuth, N. Naderi, Y. Hou, Y. Bilu, V. Prabhakaran, T. A. Thijm, G. Hirst, B. Stein, Computational argumentation quality assessment in natural language, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 176–187. URL: https://www.aclweb.org/anthology/E17-1017.

[9] C. Staudte, L. Lange, Sentarg: A hybrid doc2vec/dph model with sentiment analysis refinement, Methodology 1 (2020) 2.

[10] C. Akiki, M. Potthast, Exploring argument retrieval with transformers, Working Notes Papers of the CLEF (2020).

[11] A. Radford, K. Narasimhan, Improving language understanding by generative pre-training, 2018.

[12] M. S. Shahshahani, J. Kamps, University of amsterdam at clef 2020 (2020).

[13] A. Jivani, A comparative study of stemming algorithms, Int. J. Comp. Tech. Appl. 2 (2011) 1930–1938.

[14] C. Hutto, E. Gilbert, Vader: A parsimonious rule-based model for sentiment analysis of social media text, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 8, 2014.

[15] D. Parapar, A. Barreiro, D. E. Losada, Query expansion using wordnet with a logical model of information retrieval., IADIS AC 2005 (2005) 487–494.

7. APPENDIX A : Table with all combinations of field's weights with BM25 and LMDirichlet

Body	Premises	Conclusions	nDCG@5 BM25	nDCG@5 LMDirichlet
0	0	0	0.0024	0.0024
0	0	0.25	0.153	0.1618
0	0	0.5	0.153	0.1618
0	0	0.75	0.153	0.1618
0	0	1	0.153	0.1618
0	0.25	0	0.3938	0.7345
0	0.25	0.25	0.3191	0.6147
0	0.25	0.5	0.2954	0.5228
0	0.25	0.75	0.2835	0.4491
0	0.25	1	0.2631	0.414
0	0.5	0	0.3938	0.7345
0	0.5	0.25	0.3827	0.6606
0	0.5	0.5	0.3191	0.6147
0	0.5	0.75	0.3035	0.5709
0	0.5	1	0.2954	0.5228
0	0.75	0	0.3938	0.7345
0	0.75	0.25	0.3996	0.6829
0	0.75	0.5	0.3516	0.6524
0	0.75	0.75	0.3191	0.6147
0	0.75	1	0.3061	0.5947
0	1	0	0.3938	0.7345
0	1	0.25	**0.415**	0.6849
0	1	0.5	0.3827	0.6606
0	1	0.75	0.3438	0.6455
0	1	1	0.3191	0.6147
0.25	0	0	0.3309	0.6513
0.25	0	0.25	0.2562	0.5294
0.25	0	0.5	0.2397	0.4395
0.25	0	0.75	0.2326	0.4001
0.25	0	1	0.233	0.3596
0.25	0.25	0	0.3875	0.7095
0.25	0.25	0.25	0.351	0.6345
0.25	0.25	0.5	0.3036	0.585
0.25	0.25	0.75	0.2902	0.5395
0.25	0.25	1	0.2757	0.4881
0.25	0.5	0	0.3817	0.7239
0.25	0.5	0.25	0.3773	0.6605
0.25	0.5	0.5	0.3362	0.6278

Table continued from previous page

Body	Premises	Conclusions	nDCG@5 BM25	nDCG@5 LMDirichlet
0.25	0.5	0.75	0.3094	0.5938
0.25	0.5	1	0.2992	0.5648
0.25	0.75	0	0.3955	0.7331
0.25	0.75	0.25	0.4023	0.685
0.25	0.75	0.5	0.3672	0.6445
0.25	0.75	0.75	0.3363	0.6269
0.25	0.75	1	0.313	0.6002
0.25	1	0	0.3959	**0.7379**
0.25	1	0.25	0.4143	0.6903
0.25	1	0.5	0.3741	0.6603
0.25	1	0.75	0.3524	0.6411
0.25	1	1	0.3308	0.6271
0.5	0	0	0.3309	0.6513
0.5	0	0.25	0.2793	0.5823
0.5	0	0.5	0.2562	0.5294
0.5	0	0.75	0.2444	0.4877
0.5	0	1	0.2397	0.4395
0.5	0.25	0	0.3878	0.6962
0.5	0.25	0.25	0.3548	0.6423
0.5	0.25	0.5	0.3228	0.6058
0.5	0.25	0.75	0.2969	0.5631
0.5	0.25	1	0.2853	0.5292
0.5	0.5	0	0.3875	0.7095
0.5	0.5	0.25	0.3841	0.6624
0.5	0.5	0.5	0.351	0.6345
0.5	0.5	0.75	0.3266	0.6094
0.5	0.5	1	0.3036	0.585
0.5	0.75	0	0.3827	0.7123
0.5	0.75	0.25	0.4029	0.6698
0.5	0.75	0.5	0.3654	0.6462
0.5	0.75	0.75	0.34	0.6314
0.5	0.75	1	0.3239	0.608
0.5	1	0	0.3817	0.7239
0.5	1	0.25	0.4032	0.6896
0.5	1	0.5	0.3773	0.6605
0.5	1	0.75	0.3658	0.6384
0.5	1	1	0.3362	0.6278
0.75	0	0	0.3309	0.6513
0.75	0	0.25	0.2881	0.6014
0.75	0	0.5	0.2776	0.5608

Table continued from previous page

Body	Premises	Conclusions	nDCG@5 BM25	nDCG@5 LMDirichlet
0.75	0	0.75	0.2562	0.5294
0.75	0	1	0.2467	0.5082
0.75	0.25	0	0.3783	0.6887
0.75	0.25	0.25	0.3569	0.6451
0.75	0.25	0.5	0.3355	0.6095
0.75	0.25	0.75	0.3121	0.5785
0.75	0.25	1	0.2876	0.5584
0.75	0.5	0	0.3874	0.6989
0.75	0.5	0.25	0.384	0.6645
0.75	0.5	0.5	0.359	0.6273
0.75	0.5	0.75	0.3335	0.6187
0.75	0.5	1	0.3122	0.5941
0.75	0.75	0	0.3875	0.7095
0.75	0.75	0.25	0.3999	0.6766
0.75	0.75	0.5	0.3685	0.6531
0.75	0.75	0.75	0.351	0.6345
0.75	0.75	1	0.3302	0.6186
0.75	1	0	0.3833	0.7108
0.75	1	0.25	0.4022	0.6913
0.75	1	0.5	0.3796	0.6622
0.75	1	0.75	0.3698	0.637
0.75	1	1	0.3461	0.6316
1	0	0	0.3309	0.6513
1	0	0.25	0.2982	0.6131
1	0	0.5	0.2793	0.5823
1	0	0.75	0.2689	0.5492
1	0	1	0.2562	0.5294
1	0.25	0	0.3758	0.6804
1	0.25	0.25	0.3531	0.6529
1	0.25	0.5	0.3425	0.6171
1	0.25	0.75	0.312	0.6001
1	0.25	1	0.3046	0.5734
1	0.5	0	0.3878	0.6962
1	0.5	0.25	0.3812	0.6661
1	0.5	0.5	0.3548	0.6423
1	0.5	0.75	0.3465	0.6204
1	0.5	1	0.3228	0.6058
1	0.75	0	0.3894	0.7093
1	0.75	0.25	0.3986	0.673
1	0.75	0.5	0.3656	0.6548

Table continued from previous page

Body	Premises	Conclusions	nDCG@5 BM25	nDCG@5 LMDirichlet
1	0.75	0.75	0.3625	0.6281
1	0.75	1	0.3377	0.619
1	1	0	0.3875	0.7095
1	1	0.25	0.3995	0.687
1	1	0.5	0.3841	0.6624
1	1	0.75	0.3657	0.644
1	1	1	0.351	0.6345

Table 17: nDCG@5 scores obtained with all combinations of field's weights both for BM25 and LMDirichlet.