SEUPD@CLEF: Team hextech on Argument Retrieval for Comparative Questions. The importance of adjectives in documents quality evaluation

Notebook for the Touché Lab on Argument Retrieval at CLEF 2022

Alessandro Chimetto[1], Davide Peressoni[1], Enrico Sabbatini[1], Giovanni Tommasin[1], Marco Varotto[1], Alessio Zanardelli[1] and Nicola Ferro[1]

[1]*University of Padua, Italy*

Abstract

This report explains our approach to solve the Task 2 challenge about Argument Retrieval for Comparative Questions proposed by the third Touché lab on argument retrieval at CLEF 2022. Given a comparative topic, the task is to retrieve relevant argumentative passages from a collection of documents for either compared object or for both.

Our approach follows the Information Retrieval pipeline that is: the parsing of the document collection, the indexing of the parsed documents by using an analyzer with commons filters, and the query matching using a retrieval model. In addition, we implemented an index field that catches the overall quality of each passage for our specific task.

From an analysis of our results, the implementation of the quality field slightly improves the ranking of the retrieved documents.

Keywords

Argument Retrival, Lucene, Comparative questions, CLEF 2022 Task 2

1. Introduction

The Touché Lab is organized by the CLEF, which is a large-scale evaluation initiative for the IR (Information Retrieval) task for the European community. As a team from the Search Engines course of the master degree in Computer Engineering at the University of Padua, we are participating, with the team name of Captain Tempesta, to the 2022 Touché challenge Task 2 [1, 2], which is focused on comparative questions. The scope of this task is to retrieve useful information from a document collection that answer the aforementioned comparative question which are stored in the topic file. The corpora we have used was provided by the Touché Lab organizers, and it is composed by 0.9 million text passages taken from a ClueWeb12 [3]

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

alessandro.chimetto.1@studenti.unipd.it (A. Chimetto); davide.peressoni@studenti.unipd.it (D. Peressoni); enrico.sabbatini@studenti.unipd.it (E. Sabbatini); giovanni.tommasin@studenti.unipd.it (G. Tommasin); marco.varotto.3@studenti.unipd.it (M. Varotto); alessio.zanardelli@studenti.unipd.it (A. Zanardelli); ferro@dei.unipd.it (N. Ferro)

http://www.dei.unipd.it/~ferro/ (N. Ferro)

0000-0001-9219-6239 (N. Ferro)

collection.
Our IR System is based on various types of word frequencies that capture the quality of the English phrase syntax. As a consequence we have introduced an attribute describing the overall syntax quality for each document. The types of word frequencies we used are:

- **Symbol frequency:** the frequency of the symbols with respect to the total number of words in the passage;
- **Words length frequency:** difference between the frequency of short words and the frequency of long words with respect to the total number of words in the passage;
- **Comparative adjective frequency:** the frequency of the comparative adjectives with respect to the total number of words in the passage;
- **Adjective frequency:** the frequency of the total number of adjective, both comparative and descriptive, with respect to the total number of words in the passage.

The main purpose of the introduction of these frequencies during the indexing phase is to re-rank the retrieved documents. Moreover, according to the different calculated frequencies, the passage should be placed lower in the ranking if there are plenty of symbols, or the words length does not follow the least effort principle. At the same time, the passage should appear at top ranks if it contains many comparative and descriptive adjectives.
The paper is organized as follows:

- Section 2 describes our methodology and our work flow to solve the task;
- Section 3 explains our experimental setup;
- Section 4 discusses our experimental results;
- Section 5 reports the conclusions.

2. Related Work

We started our project from the basic Retrieval System in the Search Engines course repository: inside it there are various project examples useful to learn about Information Retrieval.

The project skeleton is based on the Hello-IR example that provides the basics of the Lucene library, e.g. the standard analyzer and a query searcher. Furthermore, we improved our analyzer by taking suggestions from the Hello-Analyzer example, in particular, we adopted the POS recognizer and the Lovins Stemmer filter [4]. Finally, we used the BM25 similarity score as suggested by the literature [5, 6].

To improve the basic Retrieval System, we used the query expansion technique by adding synonyms to the query[7, 8, 9]. Moreover, we take inspiration from [9] and we tried to capture the importance of the different types of adjectives for each document.

3. Methodology

The methodology we adopted to build our IR System is based on basic English sentence analysis. We started by making two assumptions: the first one is related to the correct syntax and the

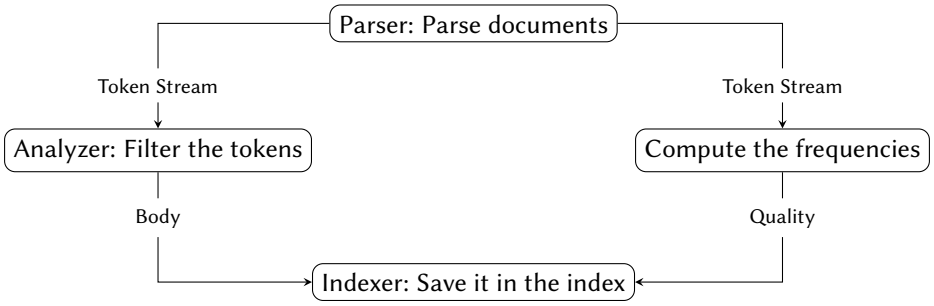


Figure 1: A simple view of the offline phase of the system architecture.

correct structure of an English sentence [10]. The second assumption is strictly related to the Touché task we are facing, the argument retrieval for comparative questions.
According to the aforementioned assumptions, a document acquires importance if it contains an appropriate number of adjectives, in particular the comparative ones. Indeed, to properly describe or compare different subjects, the sentence has to contain one or more adjectives. Moreover, the same document is not informative if it contains an elevate number of symbols with respect to the number of words. Instead the document is supposed to be an informative and readable one if it follows the Zipf's least effort principle [11, 12], which states that words of short length are more common than long words.

From the second assumption, being the adjective important in this task, we removed all of them from the stoplist, in order to not lose significative and informative tokens. Moreover we added to the query the synonyms of each adjective and other possible key words.

For the remaining part of the IR System we adopted the usual pipeline offered by the Lucene framework, as can be seen in Figure 1. Now we will describe this architecture in details.

3.1. System architecture

We used the Java programming language and the Lucene library [13] to implement and to develop our IR System. The Parser, the Indexer, and the Searcher were build upon examples seen during the lectures.

The system architecture can be divided in the following stages:

Parser

To parse the topics file we used the `org.w3c.dom` package [14] of the Java standard library, which has a function to parse the XML file in which the list of topics is stored.

To parse the passages file we used the java.util.zip package [15] from the Java standard library to decompress the gzipped file in real time. It is chained to the Json parsing method offered by the Gson library [16].

3.1.1. Analyzer

The source of our Analyzer is the standard Tokenizer. Then it maps the returned token stream applying filters in the following order:

1. **Stop filter**: it removes the most common English words that are not informative [17]. This filter works in a case insensitive mode to remove also capitalized and case mistyped words. Moreover it does not remove the adjectives, as stated in section 3.
2. **Lowercase or brand filter**: for what concerns letter case, we convert all the tokens to lowercase in order to match the same token in all possible writing combinations. In fact, normally, brand and product names can be converted to lowercase without losing information. However, there is an exception for all the words which refer to famous brands [18], and at the same time to common English words [19] (e.g. Apple the brand and apple the fruit). In this exceptional case we do not perform any token modification. It is important for this task to preserve brand names information, since a lot of queries involves comparison between brands or products.
3. **Lowercase copy filter**: it integrates the previous filter taking care of initial capitals and possible writing errors, which could be misled with product names. To overcome these situations, for each non-lowercase token,[1] we duplicate the non-lowercase tokens: one copy will remain as the original, the other will become lowercase. Finally we obtain a token for the name and a token for the word.
   The last two filters were written in a separated way to possibly allow the insertion of grammatical analysis filter. This is required since the token duplication alters the structure of the sentences.
4. **Lovins stem filter**: this is a famous stem filter, designed by Julie B. Lovins, which produces words stripped by their suffixes [4].

3.1.2. Indexer

We used the standard Lucene Indexer with the BM25 similarity [5, 6]. The standard Lucene Indexer starts by creating an inverted index. This type of index is called inverted index because it inverts a page-centric data structure (page->words) to a keyword-centric data structure (word->pages). In the next phase, the Indexer will populate the inverted index by analyzing the passages one by one using an analyzer. In practice we extended the basic analyzer, as described in section 3.1.1.

During the indexing phase, for each document the quality score is computed (as said in section 1). During the search phase, the quality score is multiplied to the query score, with the objective of re-ranking the retrieved documents. The aim is to penalize bad written documents and to promote comparative passages with respect to descriptive ones.

The quality is internally represented by a convex combination of the following frequencies:

Symbol frequency A document with plenty of not informative symbols (e.g. #, emoji, ...) is penalized because it could be a bad written passage, and it usually refers to click bait,

[1] Thanks to the previous filter, they refer to both words and names.

scam or promotional pages.
Some symbols brings no penalty if they are in an appropriate quantity (e.g. !, ?, ...): the penalty of each of those symbols is computed as $1 - 1/n$ where n is the n^{th} occurrence of the symbol.

All characters, except ASCII intervals `,-;`, `A-Z`, `a-z` and `' '`, are considered symbols. At the following symbols we assigned the increasing penalty $1 - 1/n$, instead of the fixed one (1): `?`, `%`, `$`, `&`, `*`, `+`, `/`, `<`, `=`, `>`, `@`, `_`, `"`, `'`, `(`, `)`, `[`, `]`.

Words length frequency To assure that the document follows the Zipf's least effort principle, we compute the difference between the short words frequency and the long words one. We consider as short tokens the words of length less or equal than four. Finally we rescale the aforementioned difference between 0 and 1.

Adjective frequency A document, in order to be descriptive or comparative, it must contain adjectives. To capture this property we compute the frequency of adjectives with respect to the total number of words. Intuitively, the higher the frequency, the more descriptive is the document.

Comparative adjective frequency As the same reasoning of the previous frequency, the higher the frequency, the more descriptive is the document. To achieve this we compute the ratio between the number of comparative adjective divided by the total number of adjectives.

The last two contributions have a lighter weights in the convex combination, in fact these frequencies distinguish between two types of good documents, preferring comparative ones. To classify different adjectives we prepared two different lists, the first list contains the comparative adjectives and the second contains the descriptive ones. The adjectives were taken from an online dictionary.[2]

Eventually, we add a bias to the convex combination as form of smoothing, to avoid bringing final scores to zero.

3.1.3. Searcher

A part from the aforementioned re-ranking (see section 1) by document quality, we used the standard Lucene IndexSearcher with the same configuration of the Indexer, that is we used the BM25 similarity measure related to the Vector Space Model (VSM). Lucene scoring uses a combination of the VSM of Information Retrieval and the Boolean model to determine how relevant a given Document is to a User's query. We used a basic and common similarity measure to calculate the relevance score because we focused more on the quality score of the passages.

For what concerns relevance evaluation, we used the boolean model to build a single query composed by the following sub-queries, each appropriately boosted and using the boolean clause *should* (logical or):

[2] https://www.dictionary.com

1. The first sub-query contains all the terms returned by the same Analyzer used in the Indexer (section 3.1.1). We assigned the highest boost to this query because it represents the user information need.
2. In the second sub-query we expand the previous one adding the synonyms of the terms, taken from a list [20].
3. The last sub-query contains only the N-grams detected by the POS analyzer. In practice we perform a part of speech analysis by using the openNLP libraries [21] on the title of the topic to detect sequence of multiple nouns that together form an N-gram, N stand for the number of nouns in the sequence.

Even if for each topic we return 1 000 ranked documents, we decided to retrieve 10 times that number, because when we multiply the query score by the quality one, there is the possibility that some documents ranked at a position greater than 1 000 would climb the ranking. Without this expedient, those documents would not be considered at all.

4. Experimental Setup

In this section we describe the experimental setup of our system, starting with the data provided by the CLEF lab for the specific Task 2. After that we describe how we measured our IR System effectiveness by assessing a sample of the retrieved documents. Finally, a description of our repository and of the hardware we used.

4.1. Data Description

The CLEF organization provided us:

- The **corpus**: about 0.9 million passages taken from ChatNoir dataset [22]. Each passage is organised in a JSON object containing the identifier, the body, and the link to the ChatNoir collection.
- The **topics**: each of the 50 topics is an XML entry composed by several tags, namely the number, the title, the description, and the narrative.

4.2. Evaluation measures

To overcame the lack of the qrels file, we assessed a sample formed by the first 5 ranked documents of 10 randomly chosen topics. We gave a score of 0 for the documents not containing any useful information, 1 for descriptive documents and 2 for well written comparative passages. From this we computed the nDCG score [23], which is used to evaluate multi-graded rankings.

4.3. Repository Organization

The Git repository of our project is available at the following url: https://bitbucket.org/upd-dei-stud-prj/seupd2122-hextech. The repository is located on bitbucket.org and contains the source code, the experiments and the results. The directory code contains all the necessary classes for build and test our project. In the root there is a file named `run.sh` which allows an easy run on TIRA [24].

4.4. Hardware used

To test our system, we used a personal machine, with an AMD Ryzen 9 3950X 16-Core CPU @ 3.49 GHz and 16 GIB of RAM running Windows 10 x64. For the submission we uploaded (through SSH) our retrieval models into a dedicated TIRA virtual machine.

4.5. Script used

To divide comparative from descriptive adjectives we created a script. Such script connects to www.dictionary.com through an HTTP call; given the response from the website, the script inserts the adjective into the corresponding file.

5. Results and Discussion

In this section we provide a summary of the performance and results of our system, starting with the quality of the documents, and then the comparison between the ranking of retrieved documents. Finally we will examine the relevant issues we encountered.

5.1. Quality

To validate our hypothesis, we compared three types of passages with different quality.

The first passage has a low quality (0.302), and in facts it does not bring useful information:

> ~> sup_eq_neg_inf meet_eq_neg_join ~> inf_eq_neg_sup add_eq_meet_join ~> add_eq_inf_sup meet_0_imp_0 ~> inf_0_imp_0 join_0_imp_0 ~> sup_0_imp_0 meet_0_eq_0 ~> inf_0_eq_0 join_0_eq_0 ~> sup_0_eq_0 neg_meet_eq_join ~> neg_inf_eq_sup neg_join_eq_meet ~> neg_sup_eq_inf join_eq_if ~> sup_eq_if mono_meet ~> mono_inf mono_join ~> mono_sup meet_bool_eq ~> inf_bool_eq join_bool_eq ~> sup_bool_eq meet_fun_eq ~> inf_fun_eq join_fun_eq ~> sup_fun_eq meet_set_eq ~> inf_set_eq join_set_eq ~> sup_set_eq meet1_iff ~> inf1_iff meet2_iff ~> inf2_iff meet1I ~> inf1I meet2I ~> inf2I meet1D1 ~> inf1D1 meet2D1 ~> inf2D1 meet1D2 ~> inf1D2 meet2D2 ~> inf2D2 meet1E ~> inf1E meet2E ~> inf2E join1_iff ~> sup1_iff join2_iff ~> sup2_iff join1I1 ~> sup1I1 join2I1 ~> sup2I1 join1I1 ~> sup1I1 join2I2 ~> sup1I2 join1CI ~> sup1CI join2CI ~> sup2CI join1E ~> sup1E join2E ~> sup2E is_meet_Meet ~> is_meet_Inf Meet_bool_def ~> Inf_bool_def Meet_fun_def ~> Inf_fun_def Meet_greatest ~> Inf_greatest Meet_lower ~> Inf_lower Meet_set_def ~> Inf_set_def Sup_def ~> Sup_Inf Sup_bool_eq ~> Sup_bool_def Sup_fun_eq ~> Sup_fun_def Sup_set_eq ~> Sup_set_def listsp_meetI ~> listsp_infI listsp_meet_eq ~> listsp_inf_eq meet_min ~> inf_min join_max ~> sup_max

As we expected, the reported document has very long terms, a plenty of bad symbols and no adjectives.

Next we report an higher quality (0.635) document:

> Without the breeder, there would be no dogs. Without the dogs, there would be no kennel clubs, no dog shows, no judges, no handlers, no trainers, no dog food

companies, no dog publications. Despite their importance, breeders represent a very small segment of the dog world, which in turn, creates the dog business. Furthermore, they are the ones who seldom, if ever, make a profit, even in the most popular breeds; and since they cannot take a livelihood from their breeding activities, they must be able to rely on some other source of income. Why then, do people ever become Breeders?? A breeder has, in her mind, a perfect dog that she someday hopes to create.

Compared with the low quality document, it follows the Zipf's principle, it contains a properly number of symbols and an adeguate number of adjectives. Indeed, it is obvious that this passage is readable and offers much more information than the former.

5.2. Rank comparison

The 6 runs we tested[3] are presented in Table 1. The mean nDCG of our assessment are computed by considering only the pool of assessed passages instead of the whole corpus. Moreover we reported the Touché mean nDCG computed on the relevance judgments performed by the organizers.

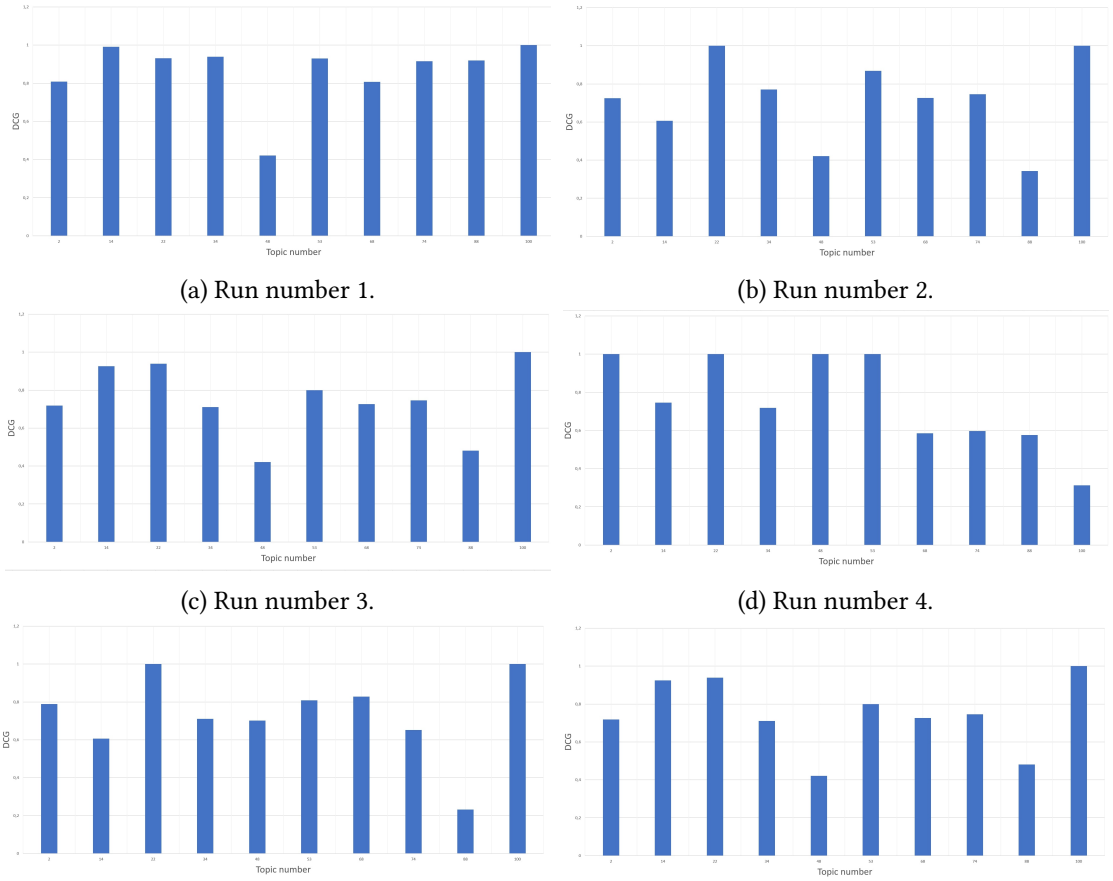
Run	Lovins	Quality weights					Query boost			Mean nDCG	
	stemmer	symbols	adj.	comparative	Zipf's	bias	q1	q2	q3	our	Touché
1	no	1	0.7	0.4	0	1	1.3	1.2	1.25	0.866	0.589
2	no	1	0.7	0.4	0.6	1	1.3	1.2	1.25	0.720	0.593
3	no	1	0.8	0.6	0.6	1	1.35	1.25	1.2	0.747	0.584
4	yes	1	0.7	0.4	0.6	1	1.3	1.2	1.25	0.753	0.566
5	no	-	-	-	-	-	1.3	1.2	1.25	0.733	0.597
6	no	-	-	-	-	-	1.35	1.25	1.2	0.747	

Table 1
Parameters used in the 6 runs. The last two do not use the document quality.

We compared the mean of nDCG measures results (visible in Figure 2) and extrapolated the following considerations:

- By comparing the first vs the fifth run, we can observe the importance of the quality: the latter has a lower mean nDCG caused by not taking into consideration documents quality.
- The Lovins stem filter brings some improvements. This can be seen from the mean nDCG scores of the second and the fourth run.
- By adding the Zipf's weight, the effectiveness decreases: the mean nDCG of run 2 is lower than run 1. This unexpected result could be caused by a wrong calculation of this metric. Moreover the introduction of a new score component made the others less important.
- Modifying the query boosts, the nDCG could improve, as can be seen in the third run with respect to the second.

[3]We had to remove the last since Touché accepts up to 5 runs.



(a) Run number 1.

(b) Run number 2.

(c) Run number 3.

(d) Run number 4.

(e) Run number 5.

(f) Run number 6.

Figure 2: nDCG with patience 2 for several topics.

- In the last case the quality does not have such an impact: in fact in the sixth run, which differs from the third only for not using the quality, the results are the same. It could be that in this example the quality weights are not as influential as the query boosts.

The previous analysis could be affected by a subjective classification of the document relevance, and according to us, especially for the first run which has a very deviated nDCG. This is clearly visible by comparing our mean nDCG with the Touché nDCG, in fact our metrics are always higher especially for the first run. Moreover, the official results are more flattered, stating that the runs are very similar as can be seen in section 6.

5.3. Relevant issues

During the implementation of the quality score, we tested the following metric: the frequency difference of the two most frequent words. The idea was that in comparative documents this difference has to be small since the two compared objects would be nominated many times.

Unfortunately this was not the case: many documents are affected by some noise and the most frequent words are not the subjects of the comparison.

Another issue is that the quality not always represents what we expect, for example this bad written document achieve the highest quality (0.81):

> J Ul ’M r-, .-i I CO ("") () c Cf) ’"CJ :>-’0) .j. J ’M A .-i 0 0 ..e () A p~ ;:l OM 0 P UO X OJ CO != i ::>. : .-i .-i 0 0 l-l P IN! f;t:l F’l m Lf) CO Lf)\O N \0 NO\o r- .. CO \0 e-i r-.. N CO ..;t Or-.. 00r-.. .P Cf) .j. J .-i 0 0 ..e o Cf) ~ ~ ’U ’"CJ OJ OJ ’M ’M ~ ~ OM ’M PO r-, \0 ~ P 0 ~ .j. J .j. J P Q) P Q)’"CJ ’"CJ P OJ P .j. J Q) .j.

In this case there are only short words, therefore our component referred to the Zipf’s principle incorrectly boosts the quality of the document. A better computation of this metric could solve the problem.

6. Statistical Analysis

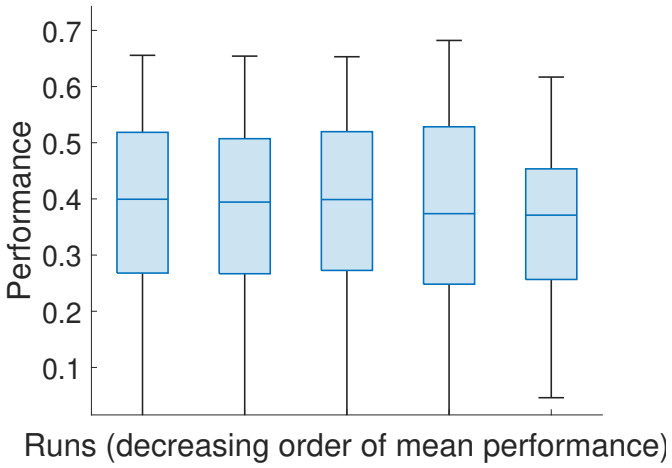


Figure 3: Box-plots of the average precision of each run.

In the following we report the statistical analysis of the runs reported in the previous section. In Figure 3 we report the MAP (Mean Average Precision) of each run in a box-plot. We observe that the is no meaningful difference between the runs.

Moreover, in Figure 4 we report the MAP calculated separately for each topic in a box-plot. From the last we observe that the majority part of the boxes have small performance variance, therefore, the runs have similar performances, given the topic, as can be further seen in Figure 3.

One curious observation is that, the performance decrease as a linear function and this is problematic in the case we want to improve the search of low performance topics because there is not a marked division between high and low performance topic. One possibility to analyze our system is to check if the outliers for each topic belongs to the same run. We suppose that this is not the case since, as we said before, there is not a significant difference among the runs.

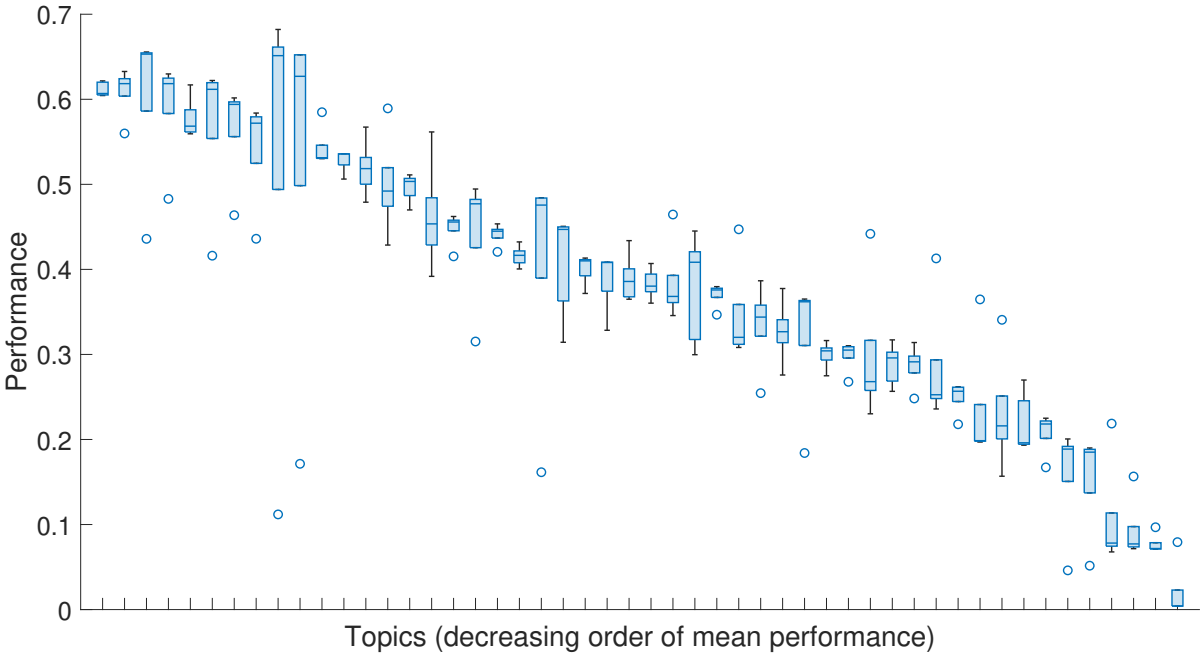


Figure 4: Box-plot of the average precision of each topic.

Source	SS	df	MS	F	Prob>F
Columns	0.0514	4	0.0128	0.4688	0.7586
Error	6.7148	245	0.0274		
Total	6.7662	249			

Table 2
Anova table.

Finally, our last statement is further supported by the ANOVA test, visible in Figure 5, where you can see that, even if the run number four is slightly different from the others, they are close with respect to performance. In the Table 2, among the other results, it is reported the F statistic value of the ANOVA test. This measure tell us that there are no relevant differences between the tested run group as the measure is close to 1. To more support the F statistic value, we performed a student's t-test on the runs 2 and 5 under the null hypothesis that the two runs are equals. By calculating the p-value, equal to 0.556 in the last test, we assure that there is not significant evidence against the null hypothesis.

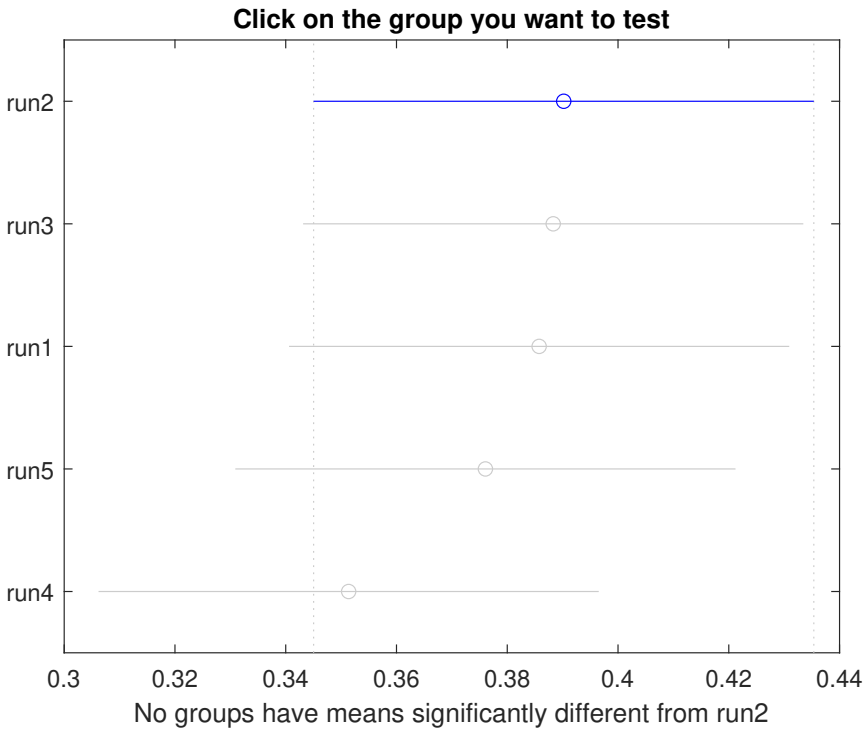


Figure 5: Anova test.

7. Conclusions and Future Work

In conclusion, by looking at the measures, we can say that the introduction of the document quality to re-rank the retrieved documents brings an improvement in terms of effectiveness. With respect of our previous considerations, the quality metric could be readjusted to be used also for other tasks.

Moreover our IR System has similar performances also without the quality metric, probably our Indexer and Searcher work fine on their own.

The normalized DCG is 72% for the worst run (2), and 86% for the best run (1). Thus, according to our relevance assessments, we can conclude our IR System seems to have good performances, but surely it can be improved.

We would have liked to try the following strategies:

Grammar analysis As said in section 3.1.1, we left the space in the Analyzer to add grammatical analysis filters. Thanks to the sentence analysis it is possible to improve the search.

Parameters tuning When the qrels will be available, it will be easier to tune the weights for the quality and the boosts for the query.

Repeated sentences detection During the document quality computation, it is possibile to detect repeated sentences in order to penalize bad structured or spam documents.

References

[1] A. Bondarenko, M. Hagen, M. Fröbe, M. Beloucif, C. Biemann, A. Panchenko, Touché task 2: Argument retrieval for comparative questions, 2022. URL: https://webis.de/events/touche-22/shared-task-2.html.

[2] A. Bondarenko, M. Fröbe, J. Kiesel, S. Syed, T. Gurcke, M. Beloucif, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2022: Argument Retrieval, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. 13th International Conference of the CLEF Association (CLEF 2022), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2022, p. to appear.

[3] J. Callan, The lemur project and its clueweb12 dataset, in: Invited talk at the SIGIR 2012 Workshop on Open-Source Information Retrieval, 2012. URL: https://www.lemurproject.org/clueweb12/.

[4] J. B. Lovins, Development of a stemming algorithm., Mech. Transl. Comput. Linguistics 11 (1968) 22–31.

[5] D. K. Harman, Overview of the third text retrieval conference (TREC-3), 500, DIANE Publishing, 1995.

[6] K. S. Jonesa, S. Walkerb, S. Robertsonb, A probabilistic model of information retrieval: development and comparative experiments part 2, Information Processing and Management 36 (2000) 840.

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

[8] A. Bondarenko, L. Gienapp, M. Fröbe, M. Beloucif, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2021: Argument Retrieval, in: K. Candan, B. Ionescu, L. Goeuriot, H. Müller, A. Joly, M. Maistro, F. Piroi, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. 12th International Conference of the CLEF Association (CLEF 2021), volume 12880 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg New York, 2021, pp. 450–467. URL: https://link.springer.com/chapter/10.1007/978-3-030-85251-1_28. doi:10.1007/978-3-030-85251-1_28.

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

[10] A. Radford, An introduction to English sentence structure, Cambridge university press, 2009.

[11] G. K. Zipf, Human behaviour and the principle of least effort, adisson, Wesley Press, Cambridge. https://doi. org/10.1002/1097-4679 (195007) 6 (1949) 306.

[12] J. K. Kanwal, Word length and the principle of least effort: language as an evolving, efficient code for information transfer (2018).
[13] T. A. S. Foundation, Apache lucene, 2022. URL: https://lucene.apache.org/core/9_1_0/index.html.
[14] Oracle, Package org.w3c.dom, 2022. URL: https://docs.oracle.com/en/java/javase/17/docs/api/java.xml/org/w3c/dom/package-summary.html.
[15] Oracle, Package java.util.zip, 2022. URL: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/zip/package-use.html.
[16] Google, Module com.google.gson, 2022. URL: https://javadoc.io/doc/com.google.code.gson/gson/latest/com.google.gson/module-summary.html.
[17] I. Brigadir, Default english stop words from different sources, 2019. URL: https://github.com/igorbrigadir/stopwords.
[18] A. Verma, M. Winkelmann, English words names brands place, 2016. URL: https://github.com/MatthiasWinkelmann/english-words-names-brands-places.
[19] K. Atkinson, Spell checking oriented word lists (scowl), 2020. URL: http://wordlist.aspell.net/.
[20] H. Robotics, hr-solr, 2019. URL: https://github.com/hansonrobotics/hr-solr.
[21] T. A. S. Foundation, Apache opennlp, 2022. URL: https://opennlp.apache.org.
[22] J. Bevendorff, B. Stein, M. Hagen, M. Potthast, Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl, in: L. Azzopardi, A. Hanbury, G. Pasi, B. Piwowarski (Eds.), Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2018.
[23] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Transactions on Information Systems (TOIS) 20 (2002) 422–446.
[24] 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.