QuARk: A GUI for Quality-Aware Ranking of Arguments

Markus Nilles
s4manill@uni-trier.de
Trier University
Trier, Germany

Lorik Dumani
dumani@uni-trier.de
Trier University
Trier, Germany

Ralf Schenkel
schenkel@uni-trier.de
Trier University
Trier, Germany

ABSTRACT

With the Web augmenting every day and computers increasingly getting more powerful, research in the field of computational argumentation becomes more and more important. One of its research branches is argument retrieval, which aims at finding and presenting users the best arguments for their queries. Several systems already exist for this purpose, all having the same goal but reaching it in different ways. In line with existing work, an argument consists of a claim supported or attacked by a premise. Now that argument retrieval has become a separate task in the CLEF lab Touché, displaying the ranking is becoming increasingly important.

In this paper we present QuARk, a GUI that allows users to retrieve arguments from a focused debate collection for their queries. Since we strictly distinguished between frontend and backend and kept the communication between them simple, QuARk can be extended to integrate various argument retrieval systems, assuming some modifications are made. In order to demonstrate the GUI, we show the integration of a complex retrieval algorithm that we also presented in the CLEF lab Touché. Our retrieval process consists of two parts. In the first step, it finds the most similar claims to the query. Therefore, the user can select between different standard IR similarity methods. The second step ranks the premises directly related to the claims. Therefore, the user can choose to rank the arguments either by quantitative, qualitative, or a combined measure.

CCS CONCEPTS

• **Information systems** → Document representation; **Information retrieval query processing**; **Probabilistic retrieval models**; **Similarity measures**.

KEYWORDS

GUI, argument retrieval system, argument search, ranking arguments, argument clustering.

ACM Reference Format:

Markus Nilles, Lorik Dumani, and Ralf Schenkel. 2021. QuARk: A GUI for Quality-Aware Ranking of Arguments. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3404835.3462795

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462795

1 INTRODUCTION

Argumentation has always been important and exists at least as long as mankind. We need reasoning to form well-founded opinions for ourselves, or to convince others to certain standpoints. This applies to trivial matters such as to decide which smartphone to buy as well as to important ones, such as in politics. Arguments can even lead to actions that influence and change complete lifestyles. And since arguments are crucial in political debates because they are the major factors for voting parties, we can state that even democracies base on arguments [14].

In the literature and especially in the field of computational argumentation an *argument* is defined to be a *claim* supported by or opposed to a *premise* [9]. Normally, the claim is a controversial point of view, which is supposed to be made either more plausible or illogical by means of evidence in form of premises [1]. An example for a claim is "*nuclear energy is dangerous*". A supporting premise to this claim is "*radiation is harmful to health*". An opposing premise to this claim is "*experts know the risks of nuclear energy and reduced them to a minimum*".

Due to the information age and the associated growing Web, as well as the computing power of modern computers, the research branch of computational argumentation emerged. A subfield of this is argument retrieval, which involves listing the best arguments for user queries so that the user can form an opinion. In order to accomplish this, it is necessary to have argument retrieval systems. The idea of an argument search engine for end users is not new. Wachsmuth et al. [15] and Stab et al. [13] have already introduced such search engines (more to argument retrieval engines in Section 2).

At least through the CLEF lab Touché, argument retrieval increased enormously in importance but one shortcoming of these GUIs is that they can only use their own argument retrieval systems. Therefore, the community needs a system that is able to integrate and display rankings of arguments of a system. In this paper we present QuARk, a GUI that users can apply for their own systems. During implementation, we paid particular attention to keeping the frontend and backend as strictly separate as possible to simplify the integration of other frameworks. Still, it naturally needs some modifications to integrate other systems there. We demonstrate the functionality and visualization of the tool by incorporating a system [6, 7] we presented in the CLEF lab Touché [3–5].[1] [2]

This demo addresses people interested in argument retrieval and argument search. After forming a query, the user can choose between different similarity and configuration methods. The program will then display the clusters of arguments for the corresponding query. These clustered arguments originate from debate portals. The

[1] The GUI is available at the URL http://argumentsearcher.uni-trier.de:8080/.
[2] The source code as well as short video of the usage of the GUI is available at the URL https://basilika.uni-trier.de/nextcloud/s/XpQ8RGi3Ry1YREQ.

user can examine all semantically equal arguments in the clusters as well as variables that help to understand the ranking.

Next, Section 2 introduces to related work. Then, Section 3 explains the backend and the retrieval process that works in the background of our system [6, 7]. After that, we describe the frontend and the default user interactivity in Section 4. Finally, Section 5 concludes the paper.

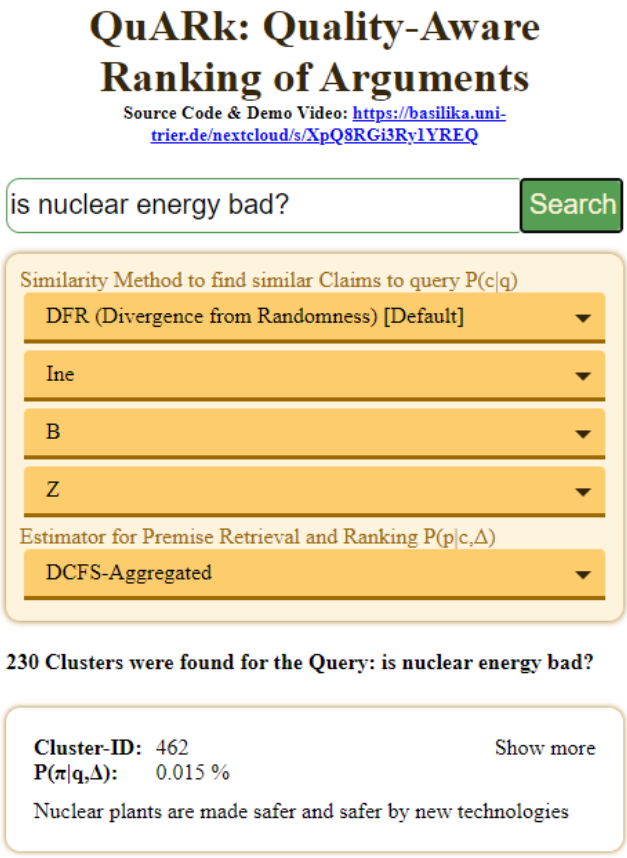

Figure 1: Visualization of the GUI showing the start page containing the elements query, ranking options and results. The similarity methods are for claim retrieval. The option for premise retrieval accomplish a re-ranking of the premises. Below is an example for a cluster result showing a representative.

2 RELATED WORK

In this section, we briefly review the most popular argument search engines and explain both similarities as well as differences to our GUI. For work in the area of argument retrieval, we refer to the CLEF lab Touché [3–5].

Wachsmuth et al. [15] present Args, one of the first argument search engine prototypes.[3] Args runs on the dataset from Ajjour et al. [2] which is now also the official dataset of the CLEF lab Touché [3–5]. The dataset draws its arguments from five debate portals indexed by the Java framework Apache Lucene. [4][5]For the ranking they use the scoring model BM25F [12]. More precisely, they index the premises together with their associated claims, in order to give the claims more importance. Given a user's keyword query, the system retrieves, ranks, and presents premises supporting and attacking the query by taking similarity of the query to the premises, their corresponding claim, as well as the context information from the whole discussion into account.

Stab et al. [13] present ArgumenText, an argument search engine for topic-relevant argument search in heterogeneous texts.[6] The arguments origin from mining large amounts of Web-scraped texts. The system outputs a ranked list of with supporting and attacking arguments for a user-given query. This system first finds relevant documents before it identifies relevant premises on the sentence level there. For ranking, they utilize ElasticSearch together with the scoring model Okapi BM25 [11].[7]

In our work, we also use the corpus from Ajjour et al. [2] as it is the official dataset of Touché. In contrast to the previous mentioned models, ours works more strictly with a two-step retrieval and does not take into account textual similarity between query and premise because convincing premises do not need to have much textual overlap to the query. For example, the premise "*radiation is harmful to health*" might be more convincing to the claim "*nuclear energy is dangerous*" than the premise "*nuclear energy sounds dangerous*" although the textual overlap is higher with the latter premise. Among others, our system allows more user interaction, i.e., the user can choose between several methods for claim retrieval and different ranking options for premises. Since the CLEF lab Touché does not require the distinction between pro and con in its argument retrieval task and this was consequently not taken into account in the underlying backend, it was also neglected in the frontend. However, since we make the complete GUI publicly available and the frontend can be modified with little effort, it is left to future work to include stances there.

3 BACKEND

In this section, we examine the process of finding arguments that runs in the background when a query comes in. For this GUI we implemented the probabilistic framework described in detail in our groundworks [6, 7] and presented in the CLEF lab Touché. Below, we revise, due to space limitations, only the most important points.

3.1 Preprocessing Steps

To process queries, the system first needs a collection of arguments. The underlying dataset is that of Ajjour et al. [2], which is the official dataset for the CLEF lab Touché and consists of about 391k arguments in form of (claim, premise) pairs. We initially processed this dataset by producing (claim, premise lists) pairs. This involved merging multiple premises into the same list if their corresponding claim is textually exactly the same.

[3]www.args.me

[4]The scraped debate portals are debate.org, debatepedia.org, debatewise.org, forandagainst.com, and idebate.org.
[5]https://lucene.apache.org/
[6]www.argumentsearch.de
[7]https://www.elastic.co/

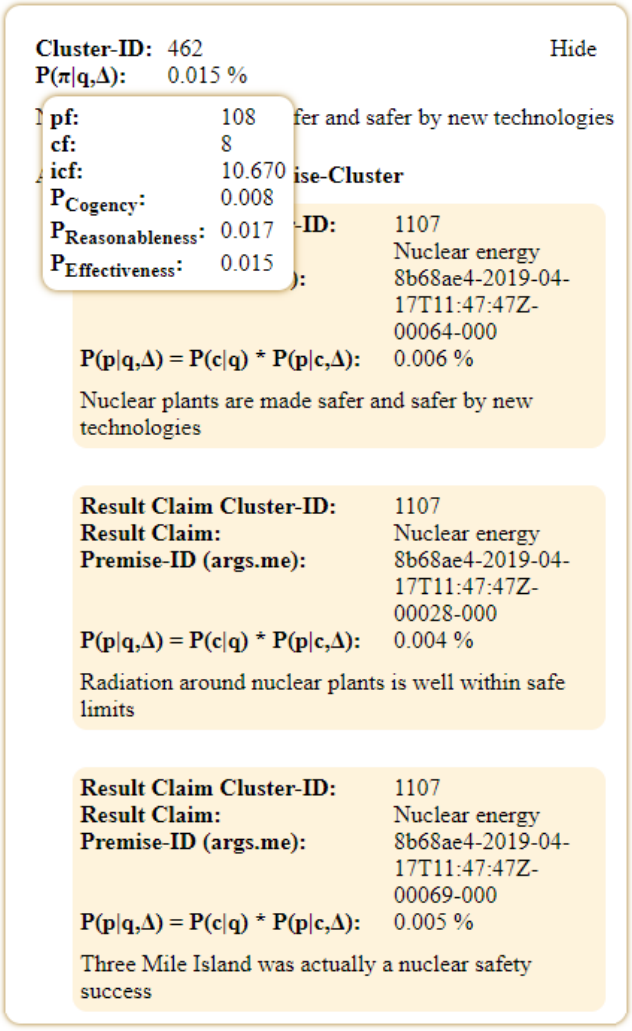

Figure 2: Visualization of the GUI showing the content of a cluster after clicking the button "*Show more*" in Figure 1. The hover effect additionally shows the components that produce the final score. The premise ID is the same as for Args.

Since several premises can be assigned to a claim, we can now form also pairs of premises for a claim. For these we can then train classifiers, which decide for each pair, which of the two is more convincing with respect to the claim for a certain argument quality dimension, e.g., whether the argument is logically conclusive or just provokes emotions. To train these classifiers, we used the dataset of Wachsmuth et al. [16], which consists of 320 arguments, for which three experts assessed a total of 15 different argument quality dimensions. We concentrated on the three main dimensions (1) logical quality in terms of the *cogency* or strength of an argument, (2) rhetorical quality in terms of the persuasive effect of an argument or argumentation (called *effectiveness*), and (3) dialectical quality in terms of the *reasonableness* of argumentation for resolving issues. By observing their mean values per dimension we derived which argument is more convincing. Then, for the dataset we work with, we counted for each premise of a claim how often it was more convincing for the particular dimensions compared to the other premises of the same claim. We refer to this as *dimension convincing frequency* (DCF).

Then, both claims and premises were clustered by their meaning. For this purpose, we transformed claims and premises into embeddings by applying Sentence BERT [10]. Then, both the sets of claims and premises were clustered agglomeratively using the average linkage method and the Euclidean distance. A dynamic tree-cut [8] served to determine the final clusters and thus the cluster ids. We constructed inverted indexes for claims and premises using Apache Lucene, where both cluster information and DCFs were included.

3.2 Query Processing

For a given query, our system works with a two-step retrieval. In the first step, it determines the most similar claims using a textual similarity method. In the framework this is expressed as probability $P(c|q)$, i.e., the probability that the user would pick claim c for query q. Now, it locates all claims that have the same claim cluster id as one of the returned claims. Then the premises directly tied to these claims are identified, as well as the premises with the same premise cluster ids. Whether a user would pick a premise p for c taking into account his preferred quality dimensions Δ is noted as $P(p|c, \Delta)$ in the framework. The probability that a user would select p for q is denoted by $P(p|q, \Delta) = P(c|q) \cdot P(p|c, \Delta)$. However, we are interested in clusters of premises rather than individual ones. Therefore, we work with the probability $P(\pi_j|q, \Delta)$ by aggregating the individual probabilities $P(p|q, \Delta)$ of all premises per cluster, i.e., $P(\pi_j|q, \Delta) = \sum_{p \in \pi_j} P(p|q, \Delta)$. As it is sufficient to show only one representative from a list of semantically similar premises, we select the longest one here, as it is intuitively the most precise.

In order to estimate $P(p|c, \Delta)$ to obtain a ranking we applied (1) quantitative and (2) qualitative goodness features. The quantitative goodness is noted with PF-ICF and can be determined exclusively by frequencies. It is based on TF-IDF and consists of the product of two components: the *premise frequency* $\text{PF}(p, c)$ and the *inverse claim frequency* $\text{ICF}(p)$. The first component describes the number of similar premises to p that are used to support (or refute) similar claims to c. Here, similarity refers to the meaning and hence to the clusters. The second component describes the inverse number of claim clusters for which p serves as support. For the qualitative goodness we make use of DCFs. Here we simply aggregate the three dimensions with Laplace-Smoothing.

4 FRONTEND

The frontend of QuARk is implemented by means of an Angular Web application.[8] Thus, it can be used from any browser.

Figure 1 visualizes the user interface. As we can see, it is simple to use and consists of a text field for the search query with a search button "*Search*" as well as an area for claim retrieval (step 1, see Section 3) ("*Similarity Method to find similar Claims*") and an area to choose between estimators for premise retrieval and ranking.

[8] https://angular.io/

After entering a query, it sends an HTTP-GET request to the backend and redirects the user to a result page. For the communication between backend and frontend, we committed ourselves to use JSON files as it allows researchers in future work to easily integrate their own systems. Encoding the query entered by the user into the URL with the options set makes it possible to bookmark the query and reopen it later without having to re-enter the search text and reset the options. The clusters are listed at the bottom in descending order of $P(\pi|q, \Delta)$, i.e. the probability that a user chooses the cluster π for the query q. In this ranking framework, this probability also reports the overall performance. Figure 2 shows the contents of a cluster when clicking on "*Show more*". More precisely, it shows all premises, of which only the representative was shown before in Figure 1. For each premise, besides the associated result claim and the premise id associated with args, there are also the individual factors introduced in the backend calculation in Section 3. To avoid overwhelming users with all the factors, we just let most of them display with a hover effect.

For claim retrieval, there are different standard IR methods provided by Apache Lucene together with their individual options that can be configured via drop-down menus. The GUI allows the user to choose between nine main similarity methods for finding similar claims to the query, which are shown in Figure 3. Additionally, the user can modify the parameters of the methods. The system presents the user only the options that are appropriate for the currently selected option. Some options require numbers as parameters. There, the user can adjust them in a text field. However, all parameters are always prefilled with default values. When the user changes a parameter, the frontend checks whether the entered value is valid. If not, it informs the user what she has to enter instead.

For estimating the premise quality, we utilized the methods described in Section 3. The user can choose between three ranking options: (1) PF-ICF (Default), (2) the aggregated DCFs values, and the average of the two.

(1) Divergence from Randomness • (2) Divergence from Independence • (3) Language model with Dirichlet-Smoothing • (4) Language model with Jelinek-Mercer smoothing • (5) Boolean Similarity • (6) TF-IDF • (7) Okapi BM25 • (8) Axiomatic approaches • (9) Information based approaches

Figure 3: Standard IR similarity methods for step 1, i.e., claim retrieval.

5 CONCLUSION AND FUTURE WORK

In this paper we have presented QuARk, an interactive GUI for argument retrieval, to meet the growing need in this community, as argument retrieval is becoming more and more important, not least because of the CLEF lab Touché. The GUI allows the user to influence the retrieval of arguments in several ways. Moreover, it is possible to evolve the backend as well as the frontend, to introduce new methods, or even to replace them with another backend.

In future work, we will include multiple systems and try to visualize the differences between them.

ACKNOWLEDGMENTS

We would like to thank Tobias Zeimetz for his invaluable help in setting up the server.

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG) within the project ReCAP, Grant Number 375342983 - 2018-2024, as part of the Priority Program "Robust Argumentation Machines (RATIO)" (SPP-1999).

REFERENCES

[1] 2014. Handbook of Argumentation Theory. Springer. https://doi.org/10.1007/978-90-481-9473-5

[2] Yamen Ajjour, Henning Wachsmuth, Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. 2019. Data Acquisition for Argument Search: The args.me Corpus. In KI (Lecture Notes in Computer Science, Vol. 11793). Springer. https://doi.org/10.1007/978-3-030-30179-8_4

[3] Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2020. Overview of Touché 2020: Argument Retrieval. In CLEF (CEUR Workshop Proceedings, Vol. 2696). CEUR-WS.org.

[4] Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2020. Overview of Touché 2020: Argument Retrieval - Extended Abstract. In CLEF (Lecture Notes in Computer Science, Vol. 12260). Springer. https://doi.org/10.1007/978-3-030-58219-7_26

[5] Alexander Bondarenko, Matthias Hagen, Martin Potthast, Henning Wachsmuth, Meriem Beloucif, Chris Biemann, Alexander Panchenko, and Benno Stein. 2020. Touché: First Shared Task on Argument Retrieval. In ECIR (Lecture Notes in Computer Science, Vol. 12036). Springer. https://doi.org/10.1007/978-3-030-45442-5_67

[6] Lorik Dumani, Patrick J. Neumann, and Ralf Schenkel. 2020. A Framework for Argument Retrieval - Ranking Argument Clusters by Frequency and Specificity. In ECIR (Lecture Notes in Computer Science, Vol. 12035). Springer. https://doi.org/10.1007/978-3-030-45439-5_29

[7] Lorik Dumani and Ralf Schenkel. 2020. Quality-Aware Ranking of Arguments. In CIKM. ACM. https://doi.org/10.1145/3340531.3411960

[8] Peter Langfelder, Bin Zhang, and Steve Horvath. 2009. Dynamic Tree Cut: In-depth description, tests and applications.

[9] Andreas Peldszus and Manfred Stede. 2013. From Argument Diagrams to Argumentation Mining in Texts: A Survey. International Journal of Cognitive Informatics and Natural Intelligence 7, 1 (2013).

[10] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In EMNLP-IJCNLP. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

[11] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In TREC, Vol. Special Publication 500-225. National Institute of Standards and Technology (NIST).

[12] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval 3, 4 (2009).

[13] Christian Stab, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna Gurevych. 2018. ArgumenText: Searching for Arguments in Heterogeneous Sources. In NAACL-HLT.

[14] Dietrich Trautmann, Johannes Daxenberger, Christian Stab, Hinrich Schütze, and Iryna Gurevych. 2020. Fine-Grained Argument Unit Recognition and Classification. In AAAI. AAAI Press.

[15] Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an Argument Search Engine for the Web. In ArgMining@EMNLP. https://doi.org/10.18653/v1/W17-5106

[16] Henning Wachsmuth, Benno Stein, Graeme Hirst, Vinodkumar Prabhakaran, Yonatan Bilu, Yufang Hou, Nona Naderi, and Tim Alberdingk Thijm. 2017. Computational Argumentation Quality Assessment in Natural Language. In EACL.