Applying "PageRank for Argument Relevance" to the *args.me* Corpus

Stefan Rosenlund, Louis Fahrenholz, Phillip Klein, Merlin Flach and Adrian Steppat

University of Leipzig

Abstract. The original PageRank algorithm is widely known as an efficient method to rank web pages in search engine results. But for argument retrieval the PageRank algorithm wasn't tested yet with an argument dataset of web scale. That's why in this paper we adapt the original PageRank algorithm for argument retrieval to use it in our new created argument search engine. The main questions of our paper are how efficient the PageRank approach is when working with a large argument dataset like the *args.me* corpus and which graph building method performs best when argument quality and argument topics are very diverse. For that we first briefly explain which criteria an argument of high quality must satisfy. After that we are presenting our approach to the task given by *Touché @ CLEF* and explain the PageRank based retrieval model our search engine uses to calculate the argument relevance. Especially this includes our approaches for building the argument graph (e.g. matcher types) and how our search engine combines the query related similarity scores with the relevance scores computed by using PageRank. Afterwards we use our search engine to compare the graphs of the different graph building methods and to measure their efficiency. In general we found out that with the right graph building method the use of PageRank improved the efficiency of our search engine. Furthermore our data shows that it is favorable to have a higher recall when matching premises with conclusions than to have 100% precision during matching. Another observation we made is that also weighted argument graphs, in which the matching precisions serve as edge weights, perform very well. At the end we also point out some important requirements to make PageRank perform well. The most important one is good quality of the provided argument corpus.

1 Introduction

How long has it been, since you have had an argument with someone about something even as simple as why wearing a mask in public is beneficial for your health? Probably not that long ago. Discussions have and will always be an important aspect of social interaction between humans. An argument usually has two purposes: it is used to change people's points of view or to persuade them to accept new points of view or you use it to persuade people to a new behavior or a certain action. Usually the goal of a discussion is to show others the right or reasonable side of a topic, an action et cetera. But what's right is often in the eye of the beholder and has to be chosen by each person individually; a task riddled with subjectivity. To assist in these choices, arguments are often used to evaluate one's point of view. In the past, those arguments usually came from the experiences and observations of people and corresponding literature. While this is also true today, the most widely and most often used source of information nowadays is undoubtedly the world wide web. Running out of arguments in an online discussion, quickly pulling up a search engine and looking for data that further assists your point of view is a common thing to do. While a search engine can easily answer factual queries directly, it struggles as soon as there is not a single correct answer, but rather many controversial opinions. As current search engines offer little support, getting a comprehensive overview of popular but also controversial pros and cons often takes long. To further enhance the difficulty, the trend of alternative facts and even fake news makes assessing the credibility of facts and their sources a requirement. While the tasks of mining arguments from the web [8] and assessing their relevance [17] have already been tackled and a prototype argument search engine *args.me* [18] was already created, there is still need to continue researching in those fields as well as creating a search engine, that can expand its index for large web crawls. And given the greater input from that scale, the methods used to match arguments as well as ranking them need to be adapted as well.

In this paper we aim to tackle the task, given by *Touché @ CLEF*, to create a search engine that retrieves strong arguments for and against a given topic. While tending to our objective, we ultimately studied some of the main pillars of argument search, namely indexing, retrieval and ranking. The result of that task is our take on such an application. Running a query on it, it will retrieve arguments from an index consisting of the *args.me* corpus [1] and find suitable candidates that are ranked by a combination of the query depending similarity score and the PageRank score. We chose the PageRank approach, because it is, to this day, successfully used by big search engines to rank web pages. Finally, we use *TIRA* Integrated Research Architecture [12] to evaluate our search engine's retrieval models by submitting runs on *TIRA*. In respect to the evaluation, *Touché @ CLEF* uses crowdsourcing to evaluate the submitted runs based on the relevance but also on the quality of arguments [4]. As the application was created for a dataset with less than 400.000 arguments, our solution can merely be called a prototype and future tweaks are likely necessary, to adapt to a much greater amount of input, making it a base to build upon, to shape it into a reliable web argument search engine.

Collectively, our contributions in this paper are:

1. a prototype argument search engine
2. a PageRank based argument retrieval model
3. several approaches for building the argument graph

2 Related Works

In order to retrieve strong arguments for and against a given issue, one must first determine the quality of an argument. In the theoretical assessment of argumentation quality, Wachsmuth et al. [16] comprehensively review existing theories to assess arguments on three quality aspects: strength, persuasiveness and reasonableness. On the practial side of quality assessment, in [11] manual annotations - labels - are used to determine the strength of a student essay's argument. Habernal and Gurevych [7] assess the convincingness of argument pairs to annotate a large dataset using crowdsourced labels to build an argument graph. Similar to the convincingness of argument pairs, in [15] arguments are labeled sufficient if an arguments premises provide enough evidence for accepting the conclusion/claim, or else they are labeled insufficient to assess argument quality. However, for getting a holistic overview of pros and cons of a given topic or issue, labels remain subjective and previous works on assessing argument quality don't focus on retrieving pro and con arguments in regards to a controversial topic. Following Wachsmuth et al. [17], the already established argument search engine *args.me* [18] uses the PageRank algorithm, as originally proposed by Page et al. [10], and BM25F, a variant of BM25, for ranking arguments. For our search engine we also use the proposed adaptation of the PageRank algorithm [2], where the relevance of an argument depends on the reuse of conclusions. Furthermore we evaluate several approaches to that very PageRank algorithm to build our graph using the [1] *args.me* corpus and *TIRA* Integrated Research Architecture [12] to discuss the effectiveness of our search engine. Looking further into the field of assessing argument relevance and its future, Dumani et al. [6] propose a framework similar to Wachsmuth et al. [17] but they [...] systematically compared 196 methods for identification of similar claims by textual similarity, using a comparable large corpus of (claim, premise) pairs crawled from several debate portals. Building on the foundations of Wachsmuth et al. [16], Lauscher et al. [9] further analyse the three introduced quality aspects in the domains of Q&A forums, debate forums and review forums, conduct an annotation study [...] resulting in the first large-scale English corpus annotated with theory-based argument quality scores and propose the first computational approaches of theory-based argument quality assessment.

3 Argument Retrieval Models

As we aim to further explore the "PageRank for Argument Relevance" approach proposed by Wachsmuth et al.[17] our retrieval model is heavily dependant on an argument graph. We chose not to experiment with many different actual retrieval models or algorithms since our focus lies on investigating different methods of building this graph and comparing them against each other. Therefore we chose well established retrieval models to reduce complexity and increase comparability with other papers. We resort to Apache's Lucene library for implementing those retrieval models.

3.1 Data

At this point it is worth noting how our dataset is structured. We use the *args.me* corpus[1]. Every argument in this dataset consists of one conclusion and a list of premises that either support or contradict this conclusion. We can make use of this structure in the process of graph building.

3.2 Argument Graph

Definition of the Argument Graph The argument graph $G = (A, E)$ is a directed graph of arguments from the *args.me* corpus. A is the set of arguments from this corpus so each argument $a \in A$ from the corpus is a node in the argument graph. An edge $e = (a_1, a_2) \in E \subseteq A \times A$ will be added to G when a premise of a_1 reuses the conclusion of a_2. Some more advanced graph building implementations may also assign an initial weight to the edges.

This definition of the argument graph contrasts the graph definition proposed by Wachsmuth et al.[17] In their definition $(a_1, a_2) \in E$ implies that a_2 uses the conclusion of a_1 as one of its premises. This means in our graph's definition the edges are inverted compared to the edges in the graph definition proposed by Wachsmuth et al.

The reason as to why we chose to deviate from Wachsmuth et al.'s proposal for defining the argument graph is that this allows us to use the standard implementation of the original PageRank algorithm[10] provided by JGraphT which we use to model the argument graph. Details of this are described in the corresponding section below.

Graph Building An inherent problem we faced was that the structure of arguments from the *args.me* corpus doesn't intuitively induce an argument graph. We aim to tackle this by applying a relevance function to a tuple consisting of arguments: $\mu : A \times A \mapsto \mathbb{R}$.

Graph building is the process of applying μ to to tuples of conclusion and premise. For the sake of efficiency we create an index of the terms in the premises of the arguments[1] and use techniques of information retrieval build the argument graph. We implement three different classes of argument matchers each of which serves as an implementation of the relevance function μ:

1. Identity Matcher
2. Similarity Matcher without Weighted Edges
3. Similarity Matcher with Weighted Edges

We hope that matches will correlate fairly well with paraphrases of the same statements.

[1] These terms are treated just like query terms, see below

Matchers

Identity Matcher For a given tuple $(a_1, a_2) \in A \times A$ this matcher assigns the relevance 1 to the resulting edge if a premise of a_1 is exactly identical to the conclusion of a_2 (be aware that this excludes stop words and that the terms are stemmed). Otherwise $(a_1, a_2) \notin E$.

$$\mu(a_1, a_2) = \begin{cases} 1 & \text{if } a_1 = a_2 \\ 0 & \text{else} \end{cases} \tag{1}$$

As this is a very restrictive approach we don't expect G to have a lot of edges.

Similarity Matcher without Weighted Edges The similarity matcher uses a tf-idf[14] approach to match conclusions and premises of arguments. It assigns a relevance of 1 to the resulting edge if the score that is calculated by the tf-idf algorithm is greater than a certain threshold[2] ϵ.

$$\mu(a_1, a_2) = \begin{cases} 1 & \text{if tf-idf}(a_1, a_2) > \epsilon \\ 0 & \text{else} \end{cases} \tag{2}$$

Otherwise the relevance 0 will be assigned i.e. $(a_1, a_2) \notin E$.

Similarity Matcher with Weighted Edges This matcher is very similar to the similarity matcher without weighted edges as it also relies upon tf-idf[14] algorithms. The difference is that this matcher assigns the score calculated by the tf-idf algorithm as the weight of an edge.

$$\mu(a_1, a_2) = \text{tf-idf}(a_1, a_2) \tag{3}$$

3.3 PageRank

The original PageRank proposed by Page et al.[10] is calculated recursively with the function:

$$p(d) = (1 - \alpha) \cdot \frac{1}{|D|} + \alpha \cdot \sum_i \frac{p(d_i)}{|D_i|} \tag{4}$$

for $d \in D$ where D is the set of documents i.e. the set of web pages, $d_i \in D : (d_i, d) \in E$ and $D_i \subseteq D : d' \in D_i \implies (d_i, d') \in E$, in two ways:

In "PageRank for Argument Relevance" Wachsmuth et al.[17] use an adapted version $\hat{p}$ of the original PageRank function p compute an argument's PageRank:

$$\hat{p}(a) = (1 - \alpha) \cdot \frac{p(d) \cdot |D|}{|A|} + \alpha \cdot \sum_i \frac{\hat{p}(a_i)}{|P_i|} \tag{5}$$

for $a \in A$ where $a_i \in A : (a, a_i) \in E$ and P_i is the set of premises a_i has.

[2] Different values will be compared in the Evaluation section

As stated above we use a standard implementation of the original PageRank algorithm which deviates slightly from that of Wachsmuth et al. To counteract this deviation the edges in our graph are inverted compared to Wachsmuth et al. Also we can't use Wachsmuth et al.'s proposed formula for ground relevance as we don't have access to the original PageRanks of the web pages the arguments were mined from. The remaining difference $|P_i|$, the denominator in the right sum, which normalizes the impact of a_i's relevance. We decided to change this to $|A_i|$, the in-degree of a_i. We expect that this change will grant slightly better results than the normalization proposed by Wachsmuth et al. [17] as it is ineffective for our dataset as the overwhelming majority of arguments in the *args.me* corpus only have one premise anyway.

3.4 Relevance Calculation

Wachsmuth et al.[17] compared four different methods to deduce an argument's relevance r_a from its PageRank. The best results were achieved by summing up the PageRanks of each argument that is linked by an argument a. $r_a = \sum_i p(a_i)$ where $p(a_i)$ is the PageRank of each premise of a. We chose to implement this method. In order to mitigate the influence of conclusions with very high PageRank we limit the argument relevance to ten.

3.5 Indexing

After the relevance has been calculated for every single argument a search index is built. This search index consists of the terms found in an arguments conclusion (those terms are treated just like query terms, see below). Also for each argument the calculated argument relevance is saved as a static feature to the search index.

3.6 Retrieval Process

Queries The query strings are processed in standard manner. Query strings get converted to lowercase and punctuation marks are removed. Afterwards they get tokenized into separate terms. All stop words are removed from this set of terms and every term is stemmed.

Retrieval The arguments are retrieved by finding exact matches of query terms in the index. All arguments that have at least one of the query terms in their conclusion are retrieved.

Ranking The retrieved arguments are then ranked based on two features: a query dependant baseline score r_q and the static feature argument relevance r_a.

For the baseline score we will be using the BM25 algorithm introduced by Robertson et al.[13] as it should result in a reasonably good result even without taking into account the argument relevance.

To combine the query dependant baseline score with the static argument relevance to a score S that can be used for ranking we use a linear combination of both values. As the question of how to weigh a static feature against a baseline score like BM25 has already been thoroughly discussed by Craswell et al.[5], we chose the naive approach of calculating S as

$$S = r_q + \omega \cdot \log(r_a) \tag{6}$$

where the scalar $\omega \in \mathbb{R}_0^+$ is the weight that we will tune in an exploratory manner in the Evaluation section. This is a natural approach as PageRanks are generally distributed by the law of power[3]. This should enable us to investigate if taking the argument relevance into account for ranking has an impact and how this impact changes depending on different graph building methods.

4 Evaluation

4.1 Purpose and Metrics

In this chapter we want to analyse how well our retrieval model performs overall and in dependency of the proposed matcher types. Furthermore we want to analyse the correlation between our matching types and the graph they produce.

As the effectiveness measure we employ the "normalized Discounted Cumulative Gain" at rank 5 (nDCG@5) and rank 10 (nDCG@10). For that we obtain the ideal rankings for 50 topics from a crowd-sourcing project. Their human assessors labelled the arguments in regards of topical relevance and argument quality. Argument quality dimension used were logical cogency, rhetoric and utility for "the users stance building-process" [4].

To analyse the correlation between matching types and graph structure, we are mostly interested in the number of edges (overall, average, median) and the edge distribution. At second we want to measure which impact the choice of the matcher type has on the correlation between argument length (i.e. conclusion length) and average matching score (i.e. edge weight).

4.2 Dataset

We use a dataset consisting of over 385.000 arguments provided by Wachsmuth et al [18]. That dataset was crawled from five different debate portals, by far the largest proportion comes from debate.org. [18]. Conclusions have an average length of 7.4 words and premises have an average length of 202.9 words [18]. That's why we don't expect our graph to have many edges when using identity matching. From each argument of the dataset we use its id, conclusion and its premises.

[3] It should be noted that this will most likely lead to our results being not as good as they could be in theory, if this was the objective of this paper.

4.3 Implementations of Matcher Types

All implementations of the matcher types remove stop words from conclusions and premises, set all letters to lowercase and stem each word using the Porter stemming algorithm. Each matcher implementation utilizes an index consisting of all premises.

Identity The matcher first looks into the premise index using a phrase query. The results are then compared to the specific conclusion to filter premises out that aren't a sub phrase of the conclusion.

TFIDF_[4.5, 6.0, 7.5] This matcher is an implementation of the matching approach "Similarity without weighted edges". It uses the vector space model with tf · idf for term weighing and cosine-similarity for computing the matching score:

$$s(q,d) = \text{cosine\_similarity}(q,d) = \frac{V(q) * V(d)}{|V(q)| * |V(d)|} \tag{7}$$

There are two reasons for this selection. Firstly we can use the provided implementation of Lucene (the euclidean norm $|V(d)|$ is replaced by a different length normalization factor, which normalizes the document vector "to a vector equal to or larger than the unit vector" [3]). Secondly and more importantly the scores of the cosine similarity are normalized to a certain degree which is why it provides comparability of two or more queries.

As the threshold value we will test three values, 4.5, 6 and 7.5. We tested these values before with smaller datasets where they provided reasonable results.

TFIDF_Weighted This matcher is an implementation of the matching approach "Similarity with weighted edges". As the previous implementation is also computes the scores using the cosine-similarity. But this time we take the computed scores and save them as edge weights.

BM25_Weighted This matcher is our second implementation of the matching approach "Similarity with weighted edges". It is based on the Okapi BM25 retrieval model. We use the Lucene provided implementation for that. That means our matching scores are computed by this formula:

$$s(q,d) = \frac{\text{tf}(t,d)}{k_1 * ((1-b) + b * \frac{|d|}{d_{avg}}) + \text{tf}(t,d)} * \log(1 + \frac{|D| - \text{df}(t,D) + 0.5}{\text{df}(t,D) + 0.5}) \tag{8}$$

For the variables b and k_1 we use the recommended values $b = 0.75$ and $k_1 = 1.2$.

4.4 Constraints

There are two important constraints to our evaluation data we have to mention. The first constraint is that the matching scores of BM25 aren't normalized. We tried different approaches to do this normalization. But all of these had some serious theoretical errors. That's why we decided to test BM25 without normalization of scores. We will take this into account when analysing the efficiency of BM25_Weighted The second constraint affects all matcher implementations except the identity matcher. As already mentioned we would have to store a massive amount of edges into our graph. This would lead to a very long duration of our graph building process. That's the reason we decided to only add edges for the 10 best ranked premises for each conclusion to our graph. Because of that PageRanks and relevances will be smaller than usual. But it will not have big impacts because the scores at lower ranks are mostly relatively small.

Table 1. Efficiency Measurement

Matching Type	Δ nDCG@5	Δ nDCG@10
None (Baseline)	0	0
Identity	-0.005	-0.004
TFIDF_7.5	-0.002	-0.002
TFIDF_6.0	-0.002	-0.003
TFIDF_4.5	-0.001	+0.006
TFIDF_Weighted	+0.003	+0.003
BM25_Weighted	+0.015	-0.001

Table 2. Graph Analysis

Matching Type	Absolute Number of Edges	Average Incoming Degree	Median Incoming Degree	Highest Incoming degree	Arguments with Incoming Degree 0	Average Matching Score (Phrases < 5)	Average Matching Score (Phrases >= 10)
Identity	72924	0.189	0	704	343854	0.177	0.009
TFIDF_7.5	189195	0.49	0	10	305118	0.11	0.445
TFIDF_6.0	501414	1.299	0	10	241702	0.236	0.673
TFIDF_4.5	1436371	3.722	2	10	104056	0.594	0.951
TFIDF_Weighted	3854276	9.988	10	10	221	3.891	5.559
BM25_Weighted	3854276	9.988	10	10	221	7.64	15.847

4.5 Interpretation

To compute a baseline efficiency for our search engine we set ω from the scoring function 6 to zero (i.e. retrieved arguments without the usage of PageRank).

That way we measured a baseline efficiency of 0.152 at nDCG@5 and 0.144 at nDCG@10 for our search engine. Maybe this baseline efficiency would have been higher if we would have included premises into our similarity score computation.

In table 1 we compare the performance of our matcher type implementations to that baseline. As a result TDIDF_Weighted improves the performance of our search engine as it has better scores in both efficiency measures nDCG@5 and nDCG@10. BM25_Weighted has the highest score at nDCG@5 and TDIDF with threshold 4.5 the highest score at nDCG@10. In contrast our search engine did not produce better results when using the identity matcher. This means that our matching approaches "Similarity with weighted edges" and "Similarity without weighted edges" outperform the matching approach "Identity" (see chapter 3).

In table 2 we analyse the resulting argument graph of the different matching type implementations. As expected the identity matcher produces a graph with very few edges. 89.1% of all conclusions have zero matches. One conclusion has 704 matches because this number of premises equals the phrase "Vote con". As already mentioned we limit the influence of such arguments by setting the maximum relevance to 10. By far the most edges produce TFIDF_Weighted and BM25_Weighted. Arguments have an average incoming degree of 9.988. This value would certainly be higher but as explained we set the maximum degree to 10 due to performance issues of our search engine.

By looking at both tables 1 and 2 we observe that a large amount of edges doesn't lead to efficiency losses. For example TFIDF with threshold 4.5 produces 20 times the number of edges that identity matching produces but still has higher efficiency scores. That's why we can conclude that a lower matching precision (to a certain degree) doesn't negatively impact the retrieval efficiency.

Also in table 2 we compare the average matching scores/ edge weights (of incoming edges) by conclusion length. As described in chapter 3.3 we set the edge weights to 1 for the matching approaches 'Identity' and 'Similarity without weights'. Regarding identity matching it shows that longer phrases have significantly fewer edges than short phrases. That's because the more words two phrases contain the higher is the probability that one of these words is different between both phrases. Furthermore the table shows that the scores of BM25 deviate much more than the ones from TFIDF. We expected this because BM25 doesn't have query length normalization at all. The data from table 1 is unclear whether this has bad impacts on retrieval efficiency because BM25_Weighted has a higher score than TFIDF_Weighted at nDCG@5 but a lower score at nDCG@10.

Table 3. Tuning of ω for the *TFIDF_Weighted* Matcher

ω	*Baseline*	…	1.0	1.1	1.2	1.3	1.4	…
nDCG@5	*0.152*	…	0.150	0.152	0.155	0.153	0.148	…

4.6 Tuning of ω

We ran a number of training runs to find the ideal value for ω, the weight in the scoring function (6). Exemplary results can be seen in table 3. Those ideal values differ substantially depending on the matcher used to build the argument graph. Based on whether optimizing for nDCG@5 or nDCG@10 the ideal ω also differs. The ideal values we found for ω span from 0.0[4] to 1.2 depending on those characteristics.

5 Discussion and Conclusion

We developed the first ever argument search engine that utilizes a PageRank based retrieval model and can work on argument datasets of web scale. With that we evaluated several approaches of building the argument graph from a given argument dataset. This way we can provide a solid theoretical and technical ground for future works that will do further research on argument graphs. Furthermore we not just employ the proposed PageRank based retrieval model from Wachsmuth et. al [17] . We extend it by combining its relevance scores with the similarity scores from BM25 and provide a formula for that process.

As mentioned at the beginning we mostly were interested in a PageRank based retrieval model because the original PageRank did perform very well. One of our questions was whether this approach can also improve the efficiency of an argument search engine that works with datasets of web scale. Our evaluation data show that this question can be answered with "yes". But there are some very important points you have to take into consideration when using PageRank for argument retrieval. The biggest one is the quality of your argument graph. The argument graph is influenced by the type of matching you choose. After some preprocessing (stop words, stemming) we employed empirical models for that task. But we are convinced that there are other good approaches from computer linguistic for that. Especially concepts that focus more on semantics (like algebraic retrieval models) need to be further evaluated for the task of building an argument graph.

But still, the success of these matching types depend heavily on the argument dataset you are working on. The better the quality of your argument dataset the better the results of your PageRank based retrieval model. One thing you need to ensure is that the arguments in the dataset are suitable for the implicit assumptions of your argument model. An important assumption could be that the arguments are logical cogent to a certain degree. Otherwise there can be a lot of arguments that have "fake" premises or a "fake" conclusion. For example a lot of arguments in our dataset had the premise "Vote con", which isn't a real premise. We tried to tackle this problem by setting a maximum for the computed relevance score. But we think that it would be worth the effort to filter such arguments out of the dataset even before the graph building. Models

[4] Meaning sometimes the baseline score is better on its own

for evaluating logical cogency of arguments already exist and need to be tested at web scale and in the context of building an argument graph.

As you see even for PageRank based retrieval models there are a lot more concepts that need to be evaluated. But this isn't a surprise given that research on argument retrieval is only in its beginning. But because this research area is getting more and more important in today's world we are confident that there will be major progress in the near future, including for PageRank based retrieval models.

References

1. Ajjour Y., Wachsmuth H., Kiesel J., Potthast M., Hagen M., Stein B.: Data Acquisition for Argument Search: The args.me corpus. In: Proceedings of the forty-second German Conference on Artificial Intelligence (KI 2019). pp. 48-59.
2. Al-Khatib K., Wachsmuth H., Hagen M., Köhler J., Stein B.: Cross-Domain Mining of Argumentative Text through Distant Supervision. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016). pp. 1395-1404.
3. Apache Lucene Documentation: `https://lucene.apache.org/core/8_5_1/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html`
4. Bondarenko A., Hagen M., Potthast M., Wachsmuth H., Beloucif M., Biemann C., Panchenko A., Stein B.: Touché: First Shared Task on Argument Retrieval. In: Pablo Castells et al, editors, Advances in Information Retrieval. 42nd European Conference on IR Research (ECIR 2020), volume 12036 of Lecture Notes in Computer Science, pages 517-523, April 2020. Springer.
5. Craswell N., Robertson S., Zaragoza H., Taylor M.: Relevance Weighting for Query Independent Evidence. In: Proceedings of the twenty-eighth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR 2005).
6. Dumani L., Neumann P., Schenkel R.: A Framework for Argument Retrieval - Ranking Argument Clusters by Frequency and Specificity. In: Advances in Information Retrieval - forty-second European Conference on IR Research (ECIR 2020). pp. 431-445.
7. Habernal I., Gurevych I.: Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM. In: Proceedings of the fifty-fourh Annual Meeting of the Association for Computational Linguistics (ACL 2016). pp. 1589-1599.
8. Habernal, I. and Gurevych I.: Argumentation Mining in User-Generated Web Discourse. In: Computational Linguistics 43 (2017). pp. 125-179.
9. Lauscher A., Ng L., Napoles C., Tetreault J.: Rhetoric, Logic, and Dialectic: Advancing Theory-based Argument Quality Assessment in Natural Language Processing. In: arXiv abs/2006.00843 (2020).
10. Page L., Brin S., Motwani R., Winograd T.: The PageRank Citation Ranking: Bringing Order to the Web. In: Technical Report. Stanford InfoLab (1999).
11. Persing I., Ng V.: Modeling Argument Strength in Student Essays. In: Proceedings of the fifty-third Annual Meeting of the Association for Computational Linguistics and the seventh International Joint Conference on Natural Language Processing (ACL 2015, IJCNLP 2015). pp. 543-552.

12. Potthast M., Gollub T., Wiegmann M., and Stein B.: TIRA Integrated Research Architecture. In: Nicola Ferro and Carol Peters, editors, Information Retrieval Evaluation in a Changing World, The Information Retrieval SeriesSpringer (2019).
13. Robertson S., Walker S., Jones S., Hancock-Beaulieu M., Gatford M.: Okapi at TREC-3. In: Proceedings of the third Text Retrieval Conference (TREC 1994).
14. Spärck Jones K.: A Statistical Interpretation of Term Specificity and its Application in Retrieval. In: Journal of Documentation, Vol. 28 No. 1 (1972). pp. 11-21.
15. Stab C., Gurevych I.: Recognizing Insufficiently Supported Arguments in Argumentative Essays. In: Proceedings of the fifteenth Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017). pp. 980-990.
16. Wachsmuth H., Naderi N., Hou Y., Bilu Y., Prabhakaran V., Thijm T., Hirst G., Stein B.: Computational Argumentation Quality Assessment in Natural Language. In: Proceedings of the fifteenth Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017). pp. 176-187.
17. Wachsmuth H., Stein B., Ajjour Y.: PageRank for Argument Relevance. In: Proceedings of the fifteenth Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017). pp. 1116-1126.
18. Wachsmuth H., Potthast M. Al-Khatib K., Ajjour Y., Puschmann J., Qu J., Dorsch J., Morari V., Bevendorff J., Stein B.: Building an Argument Search Engine for the Web. In: Proceedings of the fourth Workshop on Argument Mining (ArgMining 2017). pp. 49-59.