A Framework for Argument Retrieval

Ranking Argument Clusters by Frequency and Specificity

Lorik Dumani(✉), Patrick J. Neumann, and Ralf Schenkel(✉)

Trier University, 54286 Trier, Germany
{dumani,s4paneum,schenkel}@uni-trier.de

Abstract. Computational argumentation has recently become a fast growing field of research. An argument consists of a claim, such as *"We should abandon fossil fuels"*, which is supported or attacked by at least one *premise*, for example *"Burning fossil fuels is one cause for global warming"*. From an information retrieval perspective, an interesting task within this setting is finding the best supporting and attacking premises for a given query claim from a large corpus of arguments. Since the same logical premise can be formulated differently, the system needs to avoid retrieving duplicate results and thus needs to use some form of clustering. In this paper we propose a principled probabilistic ranking framework for premises based on the idea of TF-IDF that, given a query claim, first identifies highly similar claims in the corpus, and then clusters and ranks their premises, taking clusters of claims as well as the stances of query and premises into account. We compare our approach to a baseline system that uses BM25F which we outperform even with a primitive implementation of our framework utilising BERT.

Keywords: Argumentation retrieval · Argument clustering · Argument ranking · Argument search

1 Introduction

Computational argumentation is an emerging research area that has recently received increasing interest. It deals with representing and analysing arguments for controversial topics, which includes mining argument structures from large text corpora [8]. A widely accepted definition for an argument is that it consists of a *claim* or a standpoint, for instance *"We should abandon fossil fuels"*, which is supported or attacked by at least one *premise*, for example *"Burning fossil fuels is one cause for global warming"* or *"Poor people cannot afford alternative fuels"* [21]. The claim is the central and usually also a controversial component, which should not be accepted by the reader without further support (by premises) [28].

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG) within the project ReCAP, Grant Number 375342983 - 2018–2020, as part of the Priority Program "Robust Argumentation Machines (RATIO)" (SPP-1999).

J. M. Jose et al. (Eds.): ECIR 2020, LNCS 12035, pp. 431–445, 2020.
https://doi.org/10.1007/978-3-030-45439-5_29

From an information retrieval perspective, an interesting task within this setting is finding the best supporting (pro) and attacking (con) premises for a given query claim [31]. This has applications in many domains, including journalism and politics, and in general is relevant for making informed decisions. By now, existing (Web) search engines like Google only provide the most relevant documents to the user, but cannot structure their results in terms of claims and premises. There is a relatively large body of work on how arguments can be mined from text (see [8] for a recent survey). In this paper, we build upon established research on argument search engines and focus on effectively retrieving premises for a query claim from a large corpus of already mined arguments. Here, a query can be either a controversial topic (e.g. *"fossil fuels"*) or statement (e.g. *"we should abandon fossil fuels"*), and the task of the system is to retrieve a ranked list of pro and con premises for the query. Since the same logical premise can be formulated semantically similar, an argument retrieval system has to avoid retrieving duplicate results and thus needs to use some form of clustering.

Previous approaches in this area have focused on estimating the relevance of premises in combination with the corresponding claims, using BM25F [30] for example. The novel contribution of this paper is a principled probabilistic ranking framework for premises that, given a query claim, first determines highly similar claims in the corpus, and then clusters and ranks their premises, taking clusters of claims as well as the stances of query and premises into account.

The remainder of this paper is structured as follows: Sect. 2 discusses related work. Section 3 introduces necessary notation and Sect. 4 presents our probabilistic ranking framework. Section 5 describes details of the implementation of our framework in which we use BERT [11] to capture the vectors of premises and applied hierarchical clustering. In Sect. 6 we evaluate our approach with a large corpus [12] consisting of 63,250 claims and about 695,000 premises and compare it to a baseline system that uses BM25F. Section 7 concludes the paper and discusses ideas for future work.

2 Related Work

Stab et al. [27] present ArgumenText [4]. Their argument retrieval system first retrieves relevant documents, then it identifies relevant arguments. We do not address the argument mining task. Our work is more similar to the work of Wachsmuth et al. [30] who present Args [3], one of the first prototypes of an argument search engine. Args operates on arguments crawled from five debate portals (such as debate.org and idebate.org). Given a user's keyword query, the system retrieves, ranks, and presents premises supporting and attacking the query, taking similarity of the query with the premise, its corresponding claim, and other contextual information into account. They apply a standard BM25F ranking model implemented on top of Lucene. In our prior work [12], we build on the work of Wachsmuth et al. and 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. The results

imply that matching similar claims to a query claim with Divergence from Randomness (DFR) [2] yields slightly better results than BM25 [24]. Thus, we will make use of DFR to find the most similar claims to a query claim.

The work on argument quality and ranking is also a subarea addressed in the community. Habernal and Gurevych address the relevance of premises [15]. They confronted users in a crowdsourced task with pairs of premises to decide which premise is more convincing. Then, they used a bidirectional LSTM to predict which of two given arguments is better. In a follow-up work [14], they also investigate in the constitution of convincing arguments. Wachsmuth et al. [32] consider the problem of judging the relevance of arguments. An overview of the work on computational argumentation quality in natural language, including theories and approaches is provided by them. Their work can be used to determine the quality of arguments and thus also for the ranking.

Reimers et al. [23] deal with clustering premises. ELMo [22] and BERT [11] were used to classify and cluster topic-dependent arguments. They improve the baseline for both tasks but also recognise that arguments can address multiple aspects and therefore belong to multiple clusters. We build upon this work by using BERT to cluster claims as well as premises. As they do, we use a hard clustering algorithm and leave soft clustering algorithms for future work since this paper intends to set up the foundation and show the potential of the framework.

3 Problem Definition and Notations

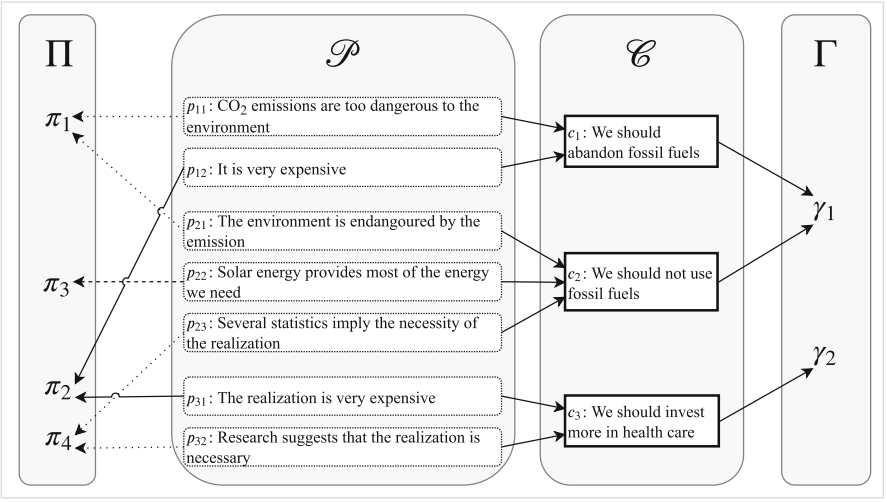
We assume that we work with a large corpus of argumentative text, for example collections of political speeches or forum discussions, that has already been mined and transferred into claims with the corresponding premises and stances.

We consider the following problem: Given a controversial claim or topic, for example "*We should abandon fossil fuels*", a user searches for the most important premises from the corpus supporting or attacking it. It is important to take into account that even if different claims or premises are semantically equivalent, they will usually be formulated in different ways, so we will consider clusters of claims (and clusters of premises) with the same meaning instead of isolated claims and premises. Finding this clustering of premises and claims as well as choosing a good representative of each result cluster to show to the user are additional tasks of the system.

We will now introduce some notations used in the remainder of the paper. Let $\mathcal{C}$ be the set of all claims in our corpus. A *claim cluster* $\gamma_j \subseteq \mathcal{C}$ is a subset of claims with the same meaning, and a *claim clustering* $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$ is a disjoint partitioning of $\mathcal{C}$ into claim clusters. The function $\gamma : \mathcal{C} \to \Gamma$ assigns to a claim $c_i \in \mathcal{C}$ its corresponding cluster γ_j (which exists and is unique).

Let $\mathcal{P}$ be the set of all premises in the corpus. We write $p \to c$ if $p \in \mathcal{P}$ appears as a premise for $c \in \mathcal{C}$ in the corpus, and $p^+ \to c$ if p supports c. Similar to claim clusters, we consider *p*remise clusters $\pi_j \subseteq \mathcal{P}$ of premises with the same meaning and the corresponding *premise clustering* $\Pi = \{\pi_1, \pi_2, \ldots\}$ as a disjoint partitioning of $\mathcal{P}$ into premise clusters. The function $\pi : \mathcal{P} \to \Pi$ assigns to a premise $p_i \in \mathcal{P}$ its corresponding premise cluster π_j.

For a premise cluster π_j, $C(\pi_j) \subseteq \mathcal{C}$ denotes the set of claims attacked or supported by premises in π_j. Note that two subsets $C(\pi_j), C(\pi_l)$ with $j \neq l$ may overlap for different premise clusters because the same premise or premises from the same cluster (e.g. '*it is very expensive*') can support or attack very different claims (e.g. '*nuclear energy*' and '*health care*'). Figure 1 gives an example of a corpus with similar claims and premises.



$\mathcal{P} = \{p_{11}, p_{12}, p_{21}, p_{22}, p_{23}, p_{31}, p_{32}\}$

$\pi_1 = \{p_{11}, p_{21}\}$
$\pi_2 = \{p_{12}, p_{31}\}$
$\pi_3 = \{p_{22}\}$
$\pi_4 = \{p_{23}, p_{32}\}$

$\Pi = \{\pi_1, \pi_2, \pi_3, \pi_4\}$

$\pi(p_{11}) = \pi(p_{21}) = \pi_1$
$\pi(p_{12}) = \pi(p_{31}) = \pi_2$
$\pi(p_{22}) = \pi_3$
$\pi(p_{23}) = \pi(p_{32}) = \pi_4$

$\mathcal{C} = \{c_1, c_2, c_3\}$

$\gamma_1 = \{c_1, c_2\}$
$\gamma_2 = \{c_3\}$

$\Gamma = \{\gamma_1, \gamma_2\}$

$\gamma(c_1) = \gamma(c_2) = \gamma_1$
$\gamma(c_3) = \gamma_2$

$C(\pi_1) = \{c_1, c_2\}$
$C(\pi_2) = \{c_1, c_3\}$
$C(\pi_3) = \{c_2\}$
$C(\pi_4) = \{c_2, c_3\}$

Fig. 1. Example for a corpus with clusters of similar claims $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$ and clusters of similar premises $\Pi = \{\pi_1, \pi_2, \ldots\}$.

A claim may come with a stance, and different claims may have different stances, even though they deal with the same topic. To see why this is important, consider the following example claims and their stances: c_1 = *"We should use fossil fuels"* (positive stance), c_2 = *"We should abandon fossil fuels" (negative stance)*, c_3 = *"Fossil fuels"* (neutral stance), and c_4 = *"Should fossil fuels be used?"* (neutral stance). We treat claims with neutral stances as if they had a positive stance. For a query asking for *"increase usage of fossil fuels"*, supporting premises would be premises that support c_1, c_3, c_4, but also premises that attack c_2. Similarly, attacking premises would be those attacking c_1, c_3, c_4 or supporting c_2. Let q and c be query and claim on the same topic, then if q and c have the same stance, a premise supporting c will also support q. Also, if q and c have opposite stance, a premise supporting c will attack q. We write $q \uparrow\uparrow c$ if the

stances of q and c are aligned and $q \uparrow\downarrow c$ otherwise. We further assume that all claims within the same cluster have the same stance.

4 Probabilistic Ranking Framework

4.1 Probability of Premise Clusters

Given a query claim q, the goal is to find the best clusters of supporting and attacking premises π^+, π^- for q in the corpus. Here, $P(\pi^+|q)$ defines the probability that a user would pick π as the supporting cluster of premises for q amongst all premise clusters in the corpus. Furthermore, $P(\pi^-|q)$ is defined analogously for attacking clusters.

To compute these probabilities, we first consider single premises and claims and extend this to clusters afterwards; we then will discuss how stances can be taken into account. We will restrict the examination to supporting premises, attacking premises are computed analogously.

First we estimate the probability $P(p^+|q)$ that the user picks the supporting premise p for query claim q. We assume the following user model: To pick a supporting premise, the user initially selects a matching claim c for q amongst all claims in the corpus with probability $P(c|q)$, and then picks a premise p with probability $P(p^+|c,q)$ amongst all supporting premises of this claim. Considering that p may support multiple claims, $P(p^+|q)$ can thus be written as

$$P(p^+|q) = \sum_{c \in \mathcal{C}} P(c|q) \cdot P(p^+|c,q) \tag{1}$$

where $\sum_{c \in \mathcal{C}} P(c|q) = 1$. Since $P(p^+|c,q) = 0$ if p is not a premise of c as the user picks only premises of c, we can restrict the summation to claims for which p appears as premise. In addition, we assume that $P(p^+|c,q) = P(p^+|c)$, i.e. p is picked as support for c independently from q.

To include the stances of query and claims, we must consider that an attacking premise of a claim with opposite stance to the query can also be picked as a supporting premise of the query. This results in the following updated expression:

$$P(p^+|q) = \sum_{c:p \to c} P(c|q) \cdot \big(P(q \uparrow\uparrow c) \cdot P(p^+|c) + P(q \uparrow\downarrow c) \cdot P(p^-|c)\big) \tag{2}$$

with $P(p^-|c)$ describing the probability that p is picked as an attacking premise of claim c, $P(q \uparrow\uparrow c)$ being the probability that q and c have the same stance, and $P(q \uparrow\downarrow c)$ being the probability that q and c have opposite stance.

Finally, to compute the probability of picking a premise cluster instead of a single premise, we additionally need to aggregate over all premises in the cluster; this works since premise clusters are disjoint by construction:

$$P(\pi_j^+|q) = \sum_{p \in \pi_j} P(p^+|q) \tag{3}$$

Note that if the user does not make a distinction between supporting and attacking clusters, but instead just wants good premise clusters, we can extend the experiment such that the user first throws a fair coin to decide if he will pick a supporting or attacking premise cluster. This leads to the following probability for picking premise cluster π_j:

$$P(\pi_j|q) = \frac{P(\pi_j^+|q) + P(\pi_j^-|q)}{2} \tag{4}$$

4.2 Estimating the Probabilities

We now present possible estimators for each of the probabilities used in our ranking framework. While we think that these estimators are reasonable, there are clearly many other ways for their estimation, for example taking argument quality [32] into account; this is left for future work.

$P(c|q)$ denotes the probability that c is "relevant" for query q, which can be estimated using standard text retrieval approaches; in our experiments, we will use Divergence from Randomness [2]. Since most retrieval approaches are not probabilistic in nature, we need to recalibrate the computed scores such that their values correspond to probabilities.

$P(p^+|c)$ is the probability that p is chosen amongst all supporting premises of c. Here, we will not use textual similarity of p and c since good premises supporting or attacking a claim often have only small textual overlap with the claim. As an example, consider a user searching for premises supporting the claim "*we should abandon fossil fuels*". A good premise could be "*wind and solar energy can already provide most of the needed energy*", which does not overlap at all with the claim. Instead, we will estimate this based on two different frequency statistics: the *premise frequency* $pf(p^+, c)$, which describes the frequency with which premise p is used as support for claims within c's claim cluster, i.e. with the same meaning as c, and the *claim frequency* $cf(p^+)$, which is the number of claim clusters for which premise p is used as support. Intuitively, we prefer premises that appear frequently within a claim cluster, and we may want to give lower weight to premises that appear within most or even all claim clusters. This is exactly the same principle used in the TF-IDF term weight [25]. We therefore use the *inverse claim frequency* $icf(p^+)$ in a form similar to standard IDF. Since the same "semantic" premise can appear in different textual formulations, we will consider its premise cluster instead of the actual premise when computing $pf(p^+)$ and $icf(p^+)$. We can formalise this as follows:

(i) $pf(p^+, c) = |\{p'^+ \rightarrow c' : p' \in \pi(p+), c' \in \gamma(c)\}|$

(ii) $icf(p^+) = \log\left(\frac{|\Gamma|}{|\{\gamma \in \Gamma:\ \exists p'^+ \in \pi(p), \exists c' \in \gamma \text{ such that } p'^+ \rightarrow c'\}|}\right)$

We then estimate $P(p^+|c)$ as

$$P(p^+|c) = \frac{pf(p^+, c) \cdot icf(p^+)}{Z} \tag{5}$$

where Z is a normalisation term computed as the sum of the unnormalised $pf \cdot icf$ products over all candidate premises; this is not needed for ranking the premises.

Estimating the probability that two claims (or, more generally, two statements) have the same stance is a surprisingly hard problem that has not yet been solved, especially if two statements have different stances [16]. We therefore omit this part of the framework in this paper and instead focus on the evaluation of the other parts, which form the core of the framework.

5 Implementation

Now we describe the concrete implementation of the framework, i.e. the clustering of claims as well as the clustering of premises.

Clustering the Claims. We cluster the claims in an offline operation with hierarchical clustering. For each claim, we calculate its embeddings using BERT [11][1]. This allowed us to create an agglomerative clustering [17], i.e. a bottom-up approach[2]. Compared to k-means [20], hierarchical clustering has the advantage of not needing to provide the number of resulting clusters beforehand. In general, only few parameters are expected here, which leads to less overfitting. For example, it expects only a method to determine the distance between two vectors and a method to link clusters. For the former we have taken the often used Euclidean distance function, and for the latter the widely used average linkage method [26], which calculates the mean of two clusters for connecting both. In order to determine a cutoff value for the clustering, we took the implementation of Langfelder et al. [19], which produces a dynamic tree cut. Contrasting constant height cut, amongst others it is capable of identifying nested clusters.

Clustering the Premises and Computing Results. Since there are usually many more premises than claims, precomputing their clustering is not viable. Instead, we use an approximation that clusters relevant premises at query time. After a query claim q arrives in the system, the top K most similar claims $R = \{r_i | 1 \leq i \leq K\}$ are retrieved from the corpus using Divergence from Randomness [2]. At the same time, we obtain $P(c|q)$ (after normalisation). Then the corresponding claim clusters are determined and all their premises $M = \{p | \exists c \in R,\ \exists c' \in \gamma(c)$ such that $p \rightarrow c'\}$ are retrieved from the corpus. From the set M, an expanded set M' is then constructed by adding, for each premise in M, its N most similar premises from the corpus, according to the state-of-the-art standard retrieval method BM25[3]. This ensures that our premise set is large enough

[1] We use the python framework FLAIR which supports document embeddings [1] and choose the pretrained model "large-uncased" where the output vectors have 4,096 dimensions.

[2] We perform the clustering with the scripting language R and the packages STATS and FASTCLUSTER.

[3] Note that we use DFR to find similar claims, but not to find similar premises, because we only have a study supporting the former [12]. Also, claims and premises differ in length as well as details and information [13].

to compute claim frequencies. Using BERT embeddings again, this expanded premise set is first hierarchically clustered and then a dynamic tree cut is made.

Unfortunately, BERT does not support more than 512 tokens, but some premises are longer. We have thus implemented the three variants BERT_{512}, BERT_{sw}, and BERT_{sent}. With BERT_{512} we simply truncate a premise after 512 tokens, i.e. the embeddings only refer to the first 512 tokens of a text. With BERT_{sw} we utilised a sliding window, i.e. for premises with more than 512 tokens we always considered only text spans with a maximum length of 512, but always shifted the window to the right by 256 until the end of the premise in order to keep as much context information as possible. Hence, for a text s that has more than 512 tokens, we get $\left\lceil \frac{|s|}{256} \right\rceil$ embeddings, of which the average is calculated pointwise at the end. With BERT_{sent} we determine embeddings for each sentence of a premise and finally form the average of all embeddings for a premise pointwise.

After the clustering, premise frequency and claim frequency are computed for each premise in the original set M as well as the final probabilities for each premise cluster. Lastly, the clusters have to be presented to the user in an adequate format. Therefore, a premise is chosen from each cluster as a representative. In our implementation, this is the premise p with the longest text.

6 Evaluation

Now we describe the evaluation of our approach which clusters and ranks premises with respect to given queries. First we explain the dataset and the baseline we used, then we describe the setup of the ground-truth of premise clusters and the evaluation metrics. Finally, we present the evaluation results.

6.1 Dataset and Baseline

We used the dataset of our prior work [12] which consists of 63,250 claims and about 695,000 premises extracted from four debate portals. After clustering, the 63,250 claims were distributed over a total of 10,611 clusters. The average cluster size is about 6.1, the median is 5.

The final evaluation corpus in this prior work consists of triples of the form (query claim, result claim, result premise) for a total of 232 query claims which are all related to the topic "energy". Result claims are these which were identified by pooling the top five similar claims for a query claim using standard IR methods. The result premises are associated with the corresponding result claims. Using this final evaluation corpus, we randomly selected 30 query claims and extracted 1,221 individual triples. As the premises later had to be clustered manually, we made sure that the union of the result premises of all result claims for each of the 30 query claims did not exceed the number 50.

The relevance of each premise for the corresponding query claim was assessed by two annotators on a three-fold relevance scale as "very relevant", "relevant", and "not relevant". Note that the actual result claims were not shown to the

assessors. The inter-annotator agreement, measured with Krippendorff's α [18], was 0.480 on a nominal scale and 0.689 on an interval scale, indicating that the annotation is robust. Disagreements between the annotations were discussed in order to achieve an agreement. After removing 26 triples because their premises were annotated as "spam" or "other", we obtained a final corpus $corp_{eval}$ of 1,195 triples consisting of 389 very relevant, 139 relevant, and 667 not relevant premises for the 30 queries.

As a baseline system, we implemented the approach proposed by Wachsmuth et al. [30] that indexes premises together with their claims and uses a BM25F scoring model [24], giving more importance to the claim than to the premise[4]. Since they gave no parameter settings, we use the default values 1.2 and 0.75 for k_1 and b, respectively [7]. As Wachsmuth et al. describe, the three fields *conclusion*, *full arguments*, and *discussion* were added to the BM25F method. In the field 'conclusion' we store the result claim, in the field 'full argument' the premise together with the result claim. The field 'discussion' reflects the context and contains the whole debate, i.e. the result claim and all its premises.

6.2 Ground-Truth and Evaluation Metrics

In order to setup a ground-truth for our experiments, we derive a ground-truth corpus $corp_{gt}$ by including only the 528 triples from $corp_{eval}$ where the premises were assessed either as relevant or as very relevant to the query claim.

For each of the 30 query claims, the premises of $corp_{gt}$ were clustered by two annotators. They were shown all result premises for a query claim, then they clustered them based on their subjectively perceived semantic similarity. One annotated, the other checked. Again, discordances were discussed in order to achieve an agreement. Please note, that the annotators were instructed to assign only premises with the same relevance level to the same (ground-truth) cluster, which also served as a pre-filter to reduce complexity.

Since we are searching for similar claims to a query claim in the first step, it is essential to know their stances in order to identify the stances of the premises to a query, so that the clustering of the premises can be divided into pros and cons. However, as it is (still) an unsolved problem to match the stance with a good probability [16], we will ignore the stance in this experiment and tackle this task in future work.

For each query, the ground-truth G then consists of clusters $G_1, \ldots, G_t$ such that each G_i contains premises with the same meaning and with the same relevance level assigned by the assessors. The relevance level assigned to premises in cluster G_i is denoted by $rel(G_i)$. We assume that the clusters are numbered such that $i \leq j$ implies $rel(G_i) \geq rel(G_j)$. Note that premises assessed as irrelevant are not included in any ground-truth cluster.

The user now asks for a summary of premises supporting and attacking the query claim. A good system will now retrieve, for a given query, a list of premises that (1) covers the different premises clusters in the ground-truth, (2) retrieves

[4] To implement a BM25F scoring model, we used the code described in [5,6].

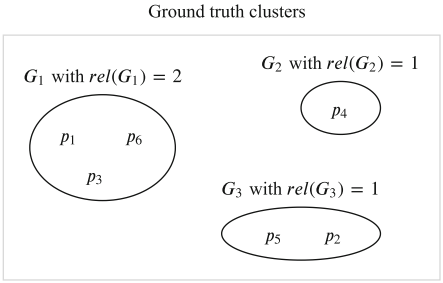
premises from highly relevant clusters before premises from "only" relevant clusters, and (3) does not retrieve multiple premises from the same cluster. Note that this setup is different from standard adhoc retrieval since the system must identify the various aspects of the results. It also differs from diversity-aware and novelty-aware approaches [9] since the user is interested in all aspects of the query, but asks for a single representative result per aspect only.

To evaluate the quality of the retrieved results, we use a simplified variant of α-nDCG [10], which we will later extend to work with clusters as results. We consider two sub-tasks here. In Task A, the system retrieves a list of premise clusters, whereas in Task B, the system needs to additionally decide for one representative premise from each cluster to show to the user. A system that would not at all consider premise clusters, for example by indexing and searching directly at the level of premises, can solve Task B only.

We will now first explain how to evaluate Task B with a simplified variant of α-nDCG [10] where we set $\alpha = 1.0$ and consider each ground-truth cluster as an information nugget. The system returns a sorted list of premises $R = (r_1, r_2, ..., r_k)$ where r_1 is the topmost result; we assume that there are no ties in the ranking (otherwise, ties will be broken arbitrarily). To compute the gain of the result at rank i, we first check if it appears in any ground-truth cluster; if not, its gain is 0. Otherwise, let g_j be the ground-truth cluster of r_i. If no result of this cluster has appeared up to rank $i-1$, the gain of r_i is $rel(g_j)$; otherwise, its gain is 0 since it does not contribute a novel aspect. As in standard nDCG, the discount for rank i is computed as $\frac{1}{\log_2 i}$ if $i \geq 2$ and 1 otherwise. In the ideal gain vector needed for computing nDCG, the component at position i is the relevance level $rel(G_i)$ of G_i, which is ideal since ground-truth clusters are ordered by descending relevance level.

To illustrate the principles of our metrics for Task B, consider the ground-truth shown in Fig. 2. The left visualises the ground-truth for a query with three clusters: G_1 which is highly relevant (score 2), and G_2 and G_3 which are relevant (score 1). On the right are the premises that the system has returned, sorted by their estimated relevance. The ideal gain vector for this ground-truth is $2, 1, 1$, corresponding to an ideal discounted cumulative gain of $2 + 1 + \frac{1}{\log_2(3)} \approx 3.63$. The gains for the result list retrieved by the system are $2, 1, 0, 0, 0, 0, 0, 1$ (since duplicate results from the same cluster are assigned a gain of 0), corresponding to a discounted cumulative gain of $2 + 1 + \frac{1}{\log_2(8)} = \frac{10}{3}$. The nDCG of this result list is thus (approximately) $\frac{10/3}{3.63} = 0.92$.

Task A is more difficult to evaluate since we do not have a list of premises, but of premise clusters (i.e. sets of premises); existing nDCG variants cannot be applied here since they operate on lists of documents, not clusters. To be able to apply the evaluation machinery introduced for Task B, we generate all possible result lists from the list of clusters, compute nDCG for each list, and aggregate the per-list values using either average, max, or min. If, e.g. our system returns two clusters $\pi_1 = \{p_1, p_2\}, \pi_2 = \{p_3, p_4\}$, then the result lists $(p_1, p_3), (p_1, p_4), (p_2, p_3), (p_2, p_4)$ are generated.



Premises retrieved by system	Assignment to cluster	Resulting gain (repetitions of the same cluster are penalised)
p_6	G_1	2
p_5	G_3	1
p_2	G_3	~~1~~ 0
p_1	G_1	~~2~~ 0
p_8	–	–
p_3	G_1	~~2~~ 0
p_7	–	–
p_4	G_2	1

Fig. 2. Example of premise clusters with graded relevance assessments.

6.3 Evaluation Results

The results of our evaluation can be found in Tables 1 and 2. Table 1 shows the evaluation of Task B, i.e. the mean nDCG@{5,10} values for all queries. Since this process requires the selection of a representative and is difficult to decide even for humans, we have simply taken the longest premise. The table reveals that the implementation BERT_{sw}, which calculates the premises' embeddings using the 'sliding window' method, performs best. For BERT_{sw} and BERT_{512}, the observed improvements over the baseline BM25F are statistically significant for nDCG@5 (tested with Welch's t-test [33] with $p = 0.05$).

Table 1. The evaluation results for Task B showing the mean nDCG values for the baseline and clustering methods BERT_m with premise preprocessing method m for the 30 queries. The p-values are related to the baseline.

Method	Representative	mean nDCG@5	p-value (nDCG@5)	mean nDCG@10	p-value (nDCG@10)
BM25F	–	.6383	–	.6087	–
BERT_{512}	longest premise	.6458	.008	.6097	.085
BERT_{sw}	longest premise	**.6782**	.002	**.6467**	.084
BERT_{sent}	longest premise	.5943	–	.5615	–

Since the baseline only returns a ranked list and not a ranked list of clusters, we interpret this list as clusters each with one entry in Table 2. We can infer from Table 2 that BERT_{sw} performs best. Using Welch's t-test with $p = 0.05$ once more, the observed improvement over the baseline is statistically significant for the mean average nDCG@5 but not for nDCG@10. Still, the results imply that BERT_{sw} is at least as good as the baseline for nDCG@10. Note that BERT_{sw} has not even been fine-tuned. Moreover, the results in Table 2 unambiguously underline the importance of clustering and even more the choice of the correct representative. If we always chose the best representative, then we always have

Table 2. The evaluation results for Task A showing the nDCG values for baseline BM25F as well as the mean average, minimum, and maximum nDCG values for clustering methods BERT_m with premise preprocessing method m. The p-values are related to the baseline.

Method	mean average nDCG	mean minimum DCG@5	mean maximum nDCG@5	p-value (mean average nDCG@5)	mean average nDCG@10	mean minimum nDCG@10	mean maximum nDCG@10	p-value (mean average nDCG@10)
BM25F	.6383	–	–	–	.6087	–	–	–
BERT_{512}	.6309	.4409	.7744	.439	.5987	.4292	.7381	.586
BERT_{sw}	**.6523**	.4561	.8053	.012	**.6187**	.4383	.7638	.203
BERT_{sent}	.5994	.4274	.7449	–	.5665	.4135	.7067	–

the maximum value and vice versa. Note that the premises used in our experiment are extracted from debate portals and thus are not always premises in the sense of argumentation theory, as they often consider more than one aspect.

7 Conclusion and Future Work

Clustering and ranking premises is a very difficult, but important task, since a user searching for premises wants them to be presented in a compact and complete format. In this paper, we made use of the idea of TF-IDF and presented a framework for clustering and ranking premises. We used premises from debate portals, which are partially from moderated websites, and of high quality but usually very long. We showed that ranking premises by their frequency and specificity has great potential since our implementation using BERT and a hard clustering algorithm outperforms the baseline BM25F although the model was not fine-tuned and the premises actually cover many aspects, so a premise could be assigned to several clusters.

In future work we will integrate soft clustering algorithms, for which we first have to break down the premises into their individual parts (e.g. *Argumentative Discourse Units* and *Elementary Discourse Units*) [29]. In addition, we will train different fine-tunings for different sentence embedding models in order to achieve better results. In our implementation, the clustering of the 695,000 premises was not precalculated, instead it was determined dynamically for a smaller subset, since this is a very computationally intensive task. Therefore, we will also precalculate the clusters of premises. To stay within the scope of this paper, we have assumed a flat hierarchy for argument graphs, where an argument consists of a claim and many premises, as they occur e.g. in debate portals. In the future we will extend our framework with more complex structures with more layers.

Acknowledgement. We would like to thank Manuel Biertz, Christin Katharina Kreutz, Alex Witry, and Tobias Zeimetz for their invaluable help in the annotations.

References

1. Akbik, A., Blythe, D., Vollgraf, R.: Contextual string embeddings for sequence labeling. In: Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, 20–26 August 2018, pp. 1638–1649 (2018). https://aclweb.org/anthology/C18-1139/
2. Amati, G., van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inf. Syst. **20**(4), 357–389 (2002). https://doi.org/10.1145/582415.582416
3. Args. https://www.args.me/index.html. Accessed 08 Jan 2020
4. ArgumenText. http://www.argumentsearch.com/. Accessed 08 Jan 2020
5. BM25F in lucene. github. https://github.com/o19s/lucene-bm25f/. Accessed 23 Sept 2019
6. Open source connections. BM25F in lucene. https://opensourceconnections.com/blog/2016/10/19/bm25f-in-lucene/. Accessed 23 Sept 2019
7. Standard values for k1 and b for BM25. https://www.elastic.co/guide/en/elasticsearch/guide/current/pluggable-similarites.html/. Accessed 23 Sept 2019
8. Cabrio, E., Villata, S.: Five years of argument mining: a data-driven analysis. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, 13–19 July 2018, pp. 5427–5433 (2018). https://doi.org/10.24963/ijcai.2018/766
9. Clarke, C.L.A., Craswell, N., Soboroff, I., Ashkan, A.: A comparative analysis of cascade measures for novelty and diversity. In: King, I., Nejdl, W., Li, H. (eds.) Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, 9–12 February 2011, pp. 75–84. ACM (2011). https://doi.org/10.1145/1935826.1935847
10. Clarke, C.L.A., et al.: Novelty and diversity in information retrieval evaluation. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, 20–24 July 2008, pp. 659–666. ACM (2008). https://doi.org/10.1145/1390334.1390446
11. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, 2–7 June 2019, Volume 1 (Long and Short Papers), pp. 4171–4186 (2019). https://aclweb.org/anthology/papers/N/N19/N19-1423/
12. Dumani, L., Schenkel, R.: A systematic comparison of methods for finding good premises for claims. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, 21–25 July 2019, pp. 957–960 (2019). https://doi.org/10.1145/3331184.3331282
13. Gleize, M., et al.: Are you convinced? Choosing the more convincing evidence with a Siamese network. In: Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, 28 July–2 August 2019, Volume 1: Long Papers, pp. 967–976 (2019). https://www.aclweb.org/anthology/P19-1093/
14. Habernal, I., Gurevych, I.: What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in web argumentation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, 1–4 November 2016, pp. 1214–1223 (2016). http://aclweb.org/anthology/D/D16/D16-1129.pdf

15. Habernal, I., Gurevych, I.: Which argument is more convincing? Analyzing and predicting convincingness of web arguments using bidirectional LSTM. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, 7–12 August 2016, Volume 1: Long Papers (2016). https://www.aclweb.org/anthology/P16-1150/
16. Hanselowski, A., et al.: A retrospective analysis of the fake news challenge stance-detection task. In: Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, 20–26 August 2018, pp. 1859–1874 (2018). https://www.aclweb.org/anthology/C18-1158/
17. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Upper Saddle River (1988)
18. Krippendorff, K.: Estimating the reliability, systematic error and random error of interval data (1970)
19. Langfelder, P., Zhang, B., Horvath, S.: Dynamic tree cut: in-depth description, tests and applications (2009). https://horvath.genetics.ucla.edu/html/CoexpressionNetwork/BranchCutting/Supplement.pdf
20. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–136 (1982). https://doi.org/10.1109/TIT.1982.1056489
21. Peldszus, A., Stede, M.: From argument diagrams to argumentation mining in texts: a survey. Int. J. Cogn. Inform. Nat. Intell. **7**(1), 1–31 (2013). https://doi.org/10.4018/jcini.2013010101
22. Peters, M.E., et al.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, 1–6 June 2018, Volume 1 (Long Papers), pp. 2227–2237 (2018). https://www.aclweb.org/anthology/N18-1202/
23. Reimers, N., Schiller, B., Beck, T., Daxenberger, J., Stab, C., Gurevych, I.: Classification and clustering of arguments with contextualized word embeddings. In: Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, 28 July–2 August 2019, Volume 1: Long Papers, pp. 567–578 (2019). https://www.aclweb.org/anthology/P19-1054/
24. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retrieval **3**(4), 333–389 (2009). https://doi.org/10.1561/1500000019
25. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975). https://doi.org/10.1145/361219.361220
26. Sokal, R.R., Michener, C.D.: A statistical method for evaluating systematic relationships. Univ. Kansas Sci. Bull. **38**, 1409–1438 (1958)
27. Stab, C., et al.: ArgumenText: searching for arguments in heterogeneous sources. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2018, New Orleans, Louisiana, USA, 2–4 June 2018, Demonstrations, pp. 21–25 (2018). https://www.aclweb.org/anthology/N18-5005/
28. Stab, C., Gurevych, I.: Identifying argumentative discourse structures in persuasive essays. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, 25–29 October 2014, Doha, Qatar, A Meeting of SIGDAT, A Special Interest Group of the ACL, pp. 46–56 (2014). https://www.aclweb.org/anthology/D14-1006/

29. Stede, M., Afantenos, S.D., Peldszus, A., Asher, N., Perret, J.: Parallel discourse annotations on a corpus of short texts. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, 23–28 May 2016 (2016). http://www.lrec-conf.org/proceedings/lrec2016/summaries/477.html
30. Wachsmuth, H., et al.: Building an argument search engine for the web. In: Proceedings of the 4th Workshop on Argument Mining (ArgMining@EMNLP), pp. 49–59 (2017). https://doi.org/10.18653/v1/W17-5106. https://www.aclweb.org/anthology/W17-5106/
31. Wachsmuth, H., Stein, B., Ajjour, Y.: "PageRank" for argument relevance. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, 3–7 April 2017, Volume 1: Long Papers, pp. 1117–1127 (2017). https://aclweb.org/anthology/E17-1105/
32. Wachsmuth, H., et al.: Computational argumentation quality assessment in natural language. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, 3–7 April 2017, Volume 1: Long Papers. pp. 176–187 (2017), https://aclweb.org/anthology/E17-1017/
33. Welch, B.L.: The generalization of 'student's' problem when several different population variances are involved. Biometrika **34**(1–2), 28–35 (1947). https://doi.org/10.1093/biomet/34.1-2.28