Exploring BERT Synonyms and Quality Prediction for Argument Retrieval

Tommaso Green[a], Luca Moroldo[a] and Alberto Valente[a]

[a]*University of Padua, Italy*

Abstract

This paper attests the University of Padua's Yeagerists team participation in Touchè @ CLEF 2021 challenge, specifically in the Argument Retrieval for Controversial Questions shared task. We show our retrieval pipeline architecture and discuss about our approach, which employs a DirichletLM retrieval model coupled with transformers-based models for both query expansion and argument quality re-ranking. For the first, after having explored several possibilities, we decided to deploy a BERT-based synonym substitution of nouns, adjectives and past participles. For argument quality re-ranking, we built an approach based on previous work and explored how different models from the BERT family performed in predicting a quality score for a given argument.

Keywords

Argument Retrieval, Automatic Query Expansion, Information Retrieval, Argument Quality, Transformers, BERT

1. Introduction

Search engines are nowadays one of the most important means to retrieve information for our society, however, they are not specialised for tasks related to the retrieval of nuanced and complex information. As the influence of search engines in opinion formation increases, it is of paramount importance for them to be able to address citizens' enquries on both general matters ("Should the death penalty be allowed?") or personal decisions ("Should I invest in real estate?") with relevant and high-quality results. This type of task, called *argument retrieval*, requires for the system to respond to user queries with arguments, which could be defined as a set of premises with evidence that lead to a conclusion. Often arguments have a stance, i.e. the author's position (in favour or against) on the debated subject. Clearly, this scenario requires to find a proper trade-off between how relevant an argument is with respect to the user query and its intrisic quality.

This paper summarizes our submission to the Touché @ CLEF shared task on Argument Retrieval for Controversial Questions which was centred on the usage of transformer-based models for query expansion and argument quality re-ranking.

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

tommaso.green@studenti.unipd.it (T. Green); luca.moroldo@studenti.unipd.it (L. Moroldo); alberto.valente.3@studenti.unipd.it (A. Valente)

CEUR Workshop Proceedings (CEUR-WS.org)

As a brief introduction, our approach consists of two phases: in the first we use a simple Lucene-based searcher with our query expansion subroutine on top of it. In the second phase the relevance scores of retrieved documents are combined with the document quality scores provided by our argument quality module. We show that combining these components we can achieve competitive performance by properly selecting similarity functions, ranking strategies and other model-specific parameters.

The paper is organized as follows: Section 2 introduces related works; Section 3 describes our approach; Section 4 explains our experimental setup; Section 5 discusses our main findings; finally, Section 6 draws some conclusions and outlooks for future work.

2. Related Work

2.1. Argument Search Engines

As described in Wachsmuth et al. [1] an argument comprises a statement, i.e. the author's position on the topic, and premises, which are usually supported by evidence. Several sub-tasks constitute this area of research: argument mining, i.e. the extraction of an argument from raw text, argument retrieval, with the related techniques to increase the recall of the system such as *Query Expansion (QE)*, and argument re-ranking, so as to consider other parameters in addition to the relevance score, for example argument quality. Some recent approaches in developing self-contained argument search engines include the args.me search engine, introduced in Wachsmuth et al. [1] and revised in Ajjour et al. [2]. It mines and indexes arguments in the offline-phase of development by harvesting them using distant supervision, exploiting the explicit debate structure: the effect is to improve precision and to make the corpus topic-agnostic. For retrieval, they use BM25F [3] and give conclusions more weight than premises.

A similar approach was used by IBM's *Project Debater* [4], where a topic-classifier was used to mine arguments from recognized sources (e.g. Wikipedia) at index-time. For retrieval, simple topic filtering and ranking were used. Overall, the selection of high-quality sources improves the recall, but this also depends on the quality of classifiers. Nevertheless, IBM's system was able to successfully participate in a live-streamed debate against a world champion.[1]

Differently from the above mentioned, ArgumenText [5] is more similar to web search engines as it indexes entire documents and delays argument mining to query-time. This approach increases recall and makes it possible to make a decision on whether a span of text is an argument conditioned on the particular query at hand.

2.2. Query Expansion

Query Expansion (QE) is a tecnique which consists in expanding a user query to increase its effectiveness in the search process, mainly by reducing its ambiguity and inserting new related keywords. As mentioned before, this kind of manipulation of user queries is intended to improve the recall of the system, by including some relevant but unusual documents in the list of retrieved ones. To get deeper knowledge about query expansion topic, the reader is suggested to take a glance at Azad and Deepak's [6] related survey.

[1]The video recording of the debate is available here

As reported in this paper [6], one of the best candidates for implementing a successful query expansion routine is to make use of the recently developed transformer-based models [7], which proved impressive performance and ability to grasp semantic nuances, especially in the IR field called *global analysis* [6], which includes all approaches that rely solely on knowledge that can be extracted from the query itself or its context, in contrast to *local analysis* approaches, relying on both direct or implicit user feedback.

Specifically, word embeddings have been largely explored as a potential method for query expansion: Kuzi et al. [8] trained a word2vec [9] *Continuous Bag of Words (CBOW)* model on their search corpora and used embeddings to generate expansion terms that are semantically related to the queries as a whole or to some of its terms.

As a continuation on that work, our query expansion approach aligned to that of Victor [10], who proposed an extension of what Kuzi et al. [8] achieved, experimenting on how contextual embeddings produced by a *Masked Language Model (MLM)*, such as BERT [11], can be applied in generating query expansion terms.

Furthermore, as we discuss in detail in Section 3, we took inspiration from the idea [12] of using an end-to-end BERT-based terms substitution approach, which proposes and then validates substitute term candidates based on their substitution's influence on the global contextualized representation of the query, all without using any labeled data or external resources.

As stated in the paper [12], the latter choice was justified by the fact that not only lexical resources (e.g. WordNet for synonyms) are likely to overlook good candidates that are not synonyms of the target words, but also they are not able to take into account the substitution's influence on the global context of the query.

Apart from this, a big difference from [12] is that we are not applying dropout to target word's embedding to partially mask the word, thus providing BERT with a hint about what terms it is required to propose, but instead we completely mask the word and let BERT generate a large enough batch of candidates from which to choose the best ones.

2.3. Argument Quality

Wachsmuth et al. [13] provide a categorization of the key components that contribute to the overall quality of an argument: *logical quality* considers how sound an argument is, i.e. how the conclusion follows from the premises, *rethorical quality* measures how persuasive an argument is for the audience and *dialectical quality* represents the effectiveness of the argument in making the audience create their own stance. The authors stress that dialectical quality builds on rhetorical, and rhetorical builds on logical. The application of transformer-based pre-trained language models to argument quality prediction was explored in Gretz et al. [14], where they provide an extensive dataset on argument quality called *IBM-Rank-30k*. They compared the performance of a Bi-LSTM with GloVe embeddings and SVR with 3 different variants of BERT: BERT-Vanilla, where the [CLS] token of the last 4 encoder layers were concatenated and fed to a feed-forward neural network with ReLU and sigmoid activation to produce a score in $[0, 1]$, BERT-FineTune where they fine-tuned the entire model (BERT encoder and linear layer on top), and finally BERT-FineTuneTOPIC, identical to BERT-FineTune but with the input enriched with the topic as in [CLS]<topic>[SEP]<query>[SEP].

2.4. Overview of Touché 2020

CLEF 2020 Summary [15] describes the Touché 2020 task and the proposed approaches in the year before the writing of this paper.

The task consists on retrieving a set of arguments from a collection that could support or challenge a stance with respect to a topic. To evaluate the significance of the retrieved arguments, the Touché 2020 Task 1 uses the nDCG@5 score on a set of manually labeled arguments.

The solutions submitted for the Touché 2020 Task 1 have a similar general strategy that makes use of a retrieval strategy, an augmentation component, and a re-ranking component. The CLEF 2020 Summary [15] reports that the most effective approaches use query expansion as augmentation component and a quality re-ranker as re-ranking component, but only a few submissions improved the score obtained with relatively simple baselines. The top-performing approaches use different ways of query augmentation, either generating synonyms or generating entirely new queries with language models. Other approaches use cluster-based result augmentation, but it appears to be less effective when compared to query augmentation.
Two main re-ranking solutions were proposed: one based on argument quality and another one based on sentiment analysis. It seems that the approaches relying on trained models, e.g. quality prediction, perform generally worse, maybe due to the limited amount of training data.

3. Methodology

Our solution is composed of 3 parts:

- A Lucene-based indexer and searcher, developed in Java.
- A query expansion module, developed in Python.
- A quality measurement module, developed in Python.

These components are glued together in a Python executable file that reads the topics (i.e. queries) from a file, performs query expansion for each topic, runs the searchers to retrieve a set of arguments, measures the quality of a portion (of size n_{rerank}) of the retrieved arguments, and (re)ranks the arguments using the quality score.

The interaction between the main.py program and the searcher happens using files, as described in the Searching section 3.2.

3.1. Indexing

We developed a Lucene-based Java program to index the Args.me corpus. The main steps performed to create the index are:

- Arguments parsing: the result of this step is a set of parsed arguments containing an id, a body, a stance, and a title. Some information contained in the Args.me corpus was discarded and not included in the index. We also discarded arguments with an empty body, as we did not considered them as valid.

- Lucene document creation: each parsed argument is transformed into a Lucene document containing the argument ID, body, and stance. Furthermore, if the parsed argument has a title, the title is included too.

  These fields were all indexed and their original content stored. We also stored the frequencies for the tokens in the body and title.

- Index writing: a single Lucene segment (using a compound file) containing all the previously created documents was constructed.

The Analyzer is responsible for the tokenization of the body and title of an argument. We used Lucene's StandardTokenFilter and LowerCaseFilter: we did not remove stopwords.

Finally, we used the LMDirichletSimilarity retrieval model.

3.1.1. The Args.me corpus

The Args.me corpus consists of 5 collections of arguments obtained from different sources. Each collection is stored following the JSON format: a collection is a list of arguments (JSON objects) containing a variety of information.

The most important field of each argument is the body, which is the text written by the user to support a claim. All the arguments in the corpus contain a stance that can be "PRO" or "CON" w.r.t. the parent discussion. Eventually, an argument contains a title which summarizes the discussion that the argument belongs to. The title can be written by the author of the argument or inherited from the discussion, and it is sometimes empty. We found duplicated arguments having the same id, where the id of an argument was obtained by hashing the argument content and some other information. We discarded duplicated arguments. We also found arguments having an empty body: we discarded these arguments too.

3.2. Searching

Given the text of a query, e.g. any Touché topic, we parsed the query text using Lucene's MultiFieldQueryParser. This parser allowed us to search for any match in both the body and the title (when present) of an argument, assigning different boosts depending on the field where the match occured.

In order to integrate the Java-based searcher with the query expansion and argument quality re-ranking steps, which are developed using Python to exploit a wider set of available libraries, the searcher reads the input queries from a file (following the TREC format) and writes the arguments found in the index into another file with the JSON format. Each output argument contains the id, body, title, and stance of the associated argument stored in the index plus the score assigned by Lucene and the id of the query that retrieved that argument.

3.3. Query Expansion

The entire Query Expansion subroutine is defined inside the module *query_expansion_utils.py* which contains the function that has to be called to generate the new set of queries for a given

list of queries, which in our application are supposed to be the Touchè topics.

The algorithm implemented inside this function works as follows:

1. The BERT tokenizer and two pretrained models are loaded from the checkpoint *bert-base-uncased* stored in the project root folder;
2. The current query is tokenized and then *Part of Speech (PoS)* tagging is applied to it;
3. The query tokens which are recognized as nouns, adjectives or past participles are masked: to do this a number of queries with each of these tokens replaced with BERT's [MASK] special token are created;
4. Ask BERT to generate the best 10 tokens that, according to its bidirectional attention mechanism, fit in place of each position for the [MASK] token;
5. Compute the BERT embeddings of these 10 tokens and compare them, using cosine similarity, to the embedding of the token which was replaced with [MASK] at step 3.
   As an implementation choice, the embeddings considered are the 3072-dimensional vectors obtained concatenating the weights of the last 4 layers of the BERT fully connected neural network. This is an arbitrary decision based both on a tradeoff to keep embeddings size small enough and on the fact that they should contain the most significant values in terms of *context classification*.
   Moreover, this is the best performing configuration according to the BERT authors themselves [11].
6. Perform a first screening, keeping only the tokens among the 10 that have similarity score above 85%, then:
   - If at least one token has passed the screening, it means that the context here is quite clear to BERT and so it is *reliable*. As a consequence, we screen once again the 10 tokens with a lower similarity threshold set to 75%, then of these "very good" tokens we keep only the best K ones, according to their scores, where K is a parameter that is set depending on how many tokens have been replaced with [MASK] at step 3, in order to provide an upper bound to the total number of queries that can be generated by the algorithm for each input query, which is set to 5000 by default.
   - If only the original token has passed the screening, it means that none of the proposed 10 tokens were good enough, so it means that BERT is dealing with an ambiguous context. For this reason, we ask BERT to generate consecutively batches of 20 tokens up to 100 in total, until at least one of these 20 has similarity score above 75%. Again, of this list of "less good" tokens we keep only the best K ones.
7. Provided the lists of new tokens for each position of [MASK], compute their cartesian product to compose the list of new queries for the current query. At this step, the number of new queries depends on the upper bound parameter seen before, so it can be in the order of thousands.
8. From the list of new queries we take a set of *max_n_query* queries at random, where *max_n_query* is the second input parameter. For consistency between runs, the random seed is fixed, so that same inputs to the function always correspond to the same list of queries.

This entire procedure is performed on each query of the list provided as the first input parameter, so that all new queries generated for each query end up in the same output list.

In our pipeline the function needs to be called just once: as mentioned before, what it returns is a list of small lists (each of which is associated to a topic) with, at most, *max_n_query* new queries each. Then this list is handed to another function that writes the queries in a properly formatted temporary .xml file, which is eventually read by the Lucene-based searcher.

3.4. Argument Quality

3.4.1. The Argument Quality Dataset

To build our argument quality predictor, we decided to use the Argument Quality Dataset of Gienapp et al. [16], as it contains 1271 arguments extracted from the args.me corpus with each having scores for rhetorical, logical, dialectical, and overall quality as well as topical relevance. Quality scores were obtained by almost 42k pairwise judgements. To simplify the approach, we made our model predict only the overall quality, which can be obtained as the length of the vector with rhetorical, logical and dialectical quality as components.

3.4.2. Adapting the BERT Family to the Argument Quality Dataset

Building on Gretz et al. [14], we decide to explore the possibility of using pre-trained language models for predicting the overall quality of an Argument. We make a distinction in the training process of models made of a transformer-based encoder and a traditional feedforward neural network, as already alluded to above in the description of previous work:

- Adaptation: while keeping the weights of the encoder freezed, only the dense layer on top is trained on the new task;
- Fine-tuning: the whole model is fine-tuned on the new task.

Both approaches have advantages and disadvantages, and in the end we decided to go with the first approach as it reduces model complexity (lowering the possibility of overfitting) and requires less computational resources. In addition to this, as reported in [17] fine-tuning seems to be more appropriate when pre-training task and transfer tasks are closely related. This was not the case as BERT [11] for example is pre-trained on MLM and *Next Sentence Prediction (NSP)* (both can be framed as classification tasks) and the target task required to produce a real-valued score (regression task), so we decided to proceed with adaption.

The model works as follows: given an argument (truncated at 512 tokens), an encoded representation is computed using the [CLS] embedding of one of the models mentioned below. Finally, this representation is passed to a feedforward neural network of depth 3 which computes the MSE loss w.r.t. the original target value.

For the dataset, we used a train-validation-test split of 80%-10%-10%. Both hyperparameter selection and training were performed in a deterministic way by setting a specific seed.

3.4.3. Model Selection

Differently from Gretz et al. [14], we decided to explore 4 different models of the BERT family: BERT [11], DistilBERT [18], RoBERTa [19] and ALBERT [20]. BERT is by far the most famous approach of using the transformer encoder to create bi-directional contextualized word representations, and several variants have been published due to its omnipresence in state-of-the-art NLP.

For example RoBERTa changes the pre-training by removing NSP but is trained on a much wider dataset with larger batch sizes. In addition to this, dynamic masking is applied: instead of "masking" always the same tokens (i.e. replacement with [MASK] special token, removal or substitution), different tokens are masked for each epoch.

ALBERT and DistilBERT can be considered lightweight versions of BERT, i.e. models created from BERT with the aim to reduce the number of parameters while maintaining or even surpassing the original performance. The first was able to greatly reduce the number of parameters by using factorized embedding parameterization (a way to reorganize the embedding matrix) and cross-layer parameter sharing. NSP, which was originally thought to be one of the key aspects of the success of BERT, is now regarded as unnecessary due to its semplicity and for this reason ALBERT substitutes it with *Sentence Order Prediction (SOP)*. DistilBERT on the other hand, is an approximate version of BERT, obtained using knowledge distillation, a technique which lets a larger model "teach" a smaller one, called the student. DistilBERT retains 95% performance of the original model, but uses only half the number of parameters and consequently runs 60% faster.

To perform hyperparameter selection, an exhaustive grid search was performed over the following set of parameters for 2 epochs for each model:

- Model: we used the following HuggingFace [21] implementations of the aforementioned models (see the model card on HuggingFace for more details):
  - bert-base-uncased
  - distilbert-base-uncased
  - albert-base-v2
  - roberta-base
- Learning rate $lr \in [0.005, 0.001, 0.0005, 0.0001, 0.00001]$
- Adam weight decay [22] $w \in [0.0005, 0.0001, 0.00001]$
- Batch size $b \in [8, 16, 32]$
- Dropout probability $p \in [0, 0.1, 0.2, 0.5]$, if set to 0 a simple feedforward neural network with ReLU activations was used otherwise AlphaDropout [23] was applied to the [CLS] embedding and to the hidden activation of the feedforward neural network. In that case ReLUs were substituted with SeLUs [23].

For all configurations we tracked the R^2 score on both training set and validation set (which was checked 4 times per epoch), and picked the best combination for each of the 4 models

according to R^2 on validation set. The 4 selected models were then trained for 30 epochs, using early stopping with a patience of 6 and by saving the best model according to validation R^2. Finally, the model performance was assessed on the remaining test set. Every experiment was logged using the online W&B logger [2] [24].

3.5. Ranking Functions

We decided to apply the argument quality re-ranking only to $D^q_{n\_rerank}$, i.e. the top *n_rerank* arguments according to the relevance score for each query q. For the final list of results, we had to combine relevance and quality scores. We tried three different ranking functions, using a parameter $\alpha \in [0, 1]$ that represented the importance of the quality score for a document d with relevance score $r(d)$ and a quality score $q(d)$:

- Normalization function: w.r.t. to a query q, we re-ranked the list by normalizing relevance and quality scores:

$$R(q,d) = (1-\alpha)\, r_{norm} + \alpha\, q_{norm} = (1-\alpha)\, \frac{r(d)}{\max_{d' \in D^q_{n\_rerank}} r(d')} + \alpha\, \frac{q(d)}{\max_{d' \in D^q_{n\_rerank}} q(d')} \quad (1)$$

- Sigmoid Function: in order to compress relevance and quality scores, with decided to use a re-scaled sigmoid function with a scale parameter β, $\sigma(\beta x) := \frac{1}{1+e^{-\beta x}}$. The function used was:

$$R(q,d) = (1-\alpha)\, \sigma(\beta r(d)) + \alpha\, \sigma(\beta q(d)) \quad (2)$$

  The parameter β was used to counter the typical "squishing" of the sigmoid, which maps large positive or negative values into close output values (closer to 1 or 0 respectively), while values near the origin can assume a wider spectrum of values. As β tends to 0, the steepness of the sigmoid decreases and allows for a wider neighbourhood of values centred in the origin to get more diverse values.

- Hybrid Function: in this case, we applied the sigmoid to the quality scores while normalizing relevance scores:

$$R(q,d) = (1-\alpha)\, r_{norm} + \alpha\, \sigma(\beta q(d)) = (1-\alpha)\, \frac{r(d)}{\max_{d' \in D^q_{n\_rerank}} r(d')} + \alpha\, \sigma(\beta q(d)) \quad (3)$$

4. Experimental Setup

Our setup makes use of GNU Make to quickly run repetitive tasks like building the project, indexing the Args.me corpus, executing our program, and evaluating the peformance.

The Makefile uses the environment variables defined in the .env file, which can be customized by any user (the file is not tracked) in order to change the parameters of the program or to define some environment-specific variables (e.g. where the program can find the Args.me corpus).

[2] To see further details and logs about the training process, see here

After executing the project with the command make run, the main.py program will create a folder containing the run-file (used to evaluate the performance), the arguments used to execute the program, and a file containing the nDCG@5 score in its name.

In order to test our system, we produced several runs - each with its own nDCG@5 - by using W&B Sweeps[3] which allowed us to test several combinations of hyperparameters in a grid-search fashion. Of these we selected the 10 most interesting runs: we performed this choice not only by considering their nDCG@5 score, but also their diversity in terms of hyperparameters. The limited amount of runs selected is mainly due to the necessity of performing hypothesis testing through Student's *t*-test, in order to see if two given runs differ significantly. To keep under control the amount of type I errors over the whole test we need to apply Bonferroni correction, which consists in lowering the significance level α_t, whose value is commonly set to 5%, to α_t/m (about 0.11% for 10 runs), where m is the number of all possible pairs of runs. As a consequence, the greater m is, the smaller α_t/m becomes, so to contextually avoid a huge increase in type II errors we decided to stick with this amount of runs as a tradeoff. To produce the performance plots and test for statistical significance of the runs, we employed two custom Matlab scripts:

- *box_plot_with_mean*: produces the box plots of the 10 selected runs.
- *students_t_test*: computes the *p-values* of *t*-test between each pair of runs among the 10 selected runs, and returns only pairs that are significatively different.

Both scripts require 3 *.csv* files:

- *ndcg.csv*: a matrix where rows are topics and columns are runs; each cell contains the nDCG@5 score for a run on a given topic;
- *runs.csv*: an array of strings containing the identifiers of the runs; elements in the array are in the same order as columns on *ndcg.csv*;
- *topics.csv*: an array of strings containing the identifiers of the topics, so in our case they are basically the numbers between 1 and 50; elements in the array are in the same order as rows on *ndcg.csv*.

The repository containing all the code and scripts is hosted on BitBucket[4]. The main evaluation measure that we considered is the nDCG@5 score, because it is the measure used by the Touché Task 1 evaluation commitee to rank the proposed systems.

5. Results and Discussion

5.1. Indexing and Searching

We tried optimizing indexing and searching without using query expansion and argument quality re-ranking, so we decided to run both indexer and searcher on a diverse enough set

[3]Results are publicly available and can be explored here

[4]https://bitbucket.org/upd-dei-stud-prj/seupd2021-yeager/src/master/

of configurations in order to keep the one which lead to best overall results, considering the nDCG@5 score as the main performance measure. No statistical relevance analysis has been involved in this phase, since our objective was to find the best configuration to use as a baseline to be compared against all the other runs, which in turn included also query expansion and argument quality re-ranking.

At first we compared two retrieval functions: BM25 and LMDirichletSimilarity. The results were that the latter almost doubled the score, so all of the following tests have been performed using uniquely LMDirichletSimilarity.

We also tested different configurations for the Lucene's Analyzer, which is responsible for the tokenization of the body and the title of an argument. We found that:

- Any kind of stemmer significantly degraded the performance.
- By removing the stop words filter we consistently improved the nDCG@5 score.
- The English possessive filter slightly reduced the performance.

Therefore, our Analyzer made use of just a tokenizer and a lowercase filter.

Finally, we optimized the boosts assigned to a match of the query with the title and with the body. We fixed the body boost to 1.0 and tried different values for the title boost and we found something unexpected: the higher was the boost assigned to a title match, the lower was the score. As a consequence, as shown in Table 4, we decided to keep only runs with title boost t set to 0.

5.2. Query Expansion

More than one approach was tested before landing on the one described in section 3.3.

These were the most promising ones:

- *Back-Translation*, also called *Spinning*, is a tecnique which consists in using a multilingual translation model to translate a sentence into a foreign language and back again into the original one. An immediate variant may also extend the chain of translations to get more diverse results.

  First we proved the effectiveness of the approach using TextBlob library[5], which for translation task relies on Google Translate API, so it could not be used for the final submission. It lead to some very good results, especially when translating from English to languages of distant families, like Arabic, Russian or Japanese.

  Unfortunately, switching to HuggingFace's Transformers pretrained models (e.g. *mbart-large-cc25* and *mt5-small*), we faced some issues that prevented us from deeply testing this approach, but it is certainly one that we are willing to develop further.

- Since replacing naively all query terms with their synonyms (provided by the lexical database WordNet) lead, as expected, to most of the new queries being meaningless or out of context, we tried to use BERT to generate some substitute candidates for the same

[5] https://textblob.readthedocs.io/en/dev/

should insider trading be allowed ?
should insider trade be forbidden ?
should insider trade be prohibited ?
should insider trading be permitted ?
should insider trading be legal ?
should insider trade be tolerated ?
should insider trading be banned ?
should insider trade be acceptable ?
should insider trading be outlawed ?
should insider trading be illegal ?

Table 1
Examples of useful new queries

terms and then check if these proposed words were also included in the synonyms list. The idea behind it was to introduce BERT's contextual awareness in the algorithm, but proved to be not suitable for most topics, where it just could not generate any new query;

- The last approach was based on the same intuition behind the one explained in section 3.3, so it exploited again BERT's ability to generate meaningful word embeddings. The main difference was that, after generating the first batch of 10 candidates for each [MASK] position, we immediately combined them into the list of new queries and we computed the embeddings associated to each entire new query. Specifically, to get a single vector we decided to average the second to last hidden layer of each token of the query, thus producing 768-dimensional embeddings. This strategy aimed at taking into account the meaning of the query as a whole, so that we could compute the cosine similarity between new queries embeddings and the original query embedding, in order to keep only the top K ones, for example.

  This gave some good results, but required to compute the embeddings and cosine similarity, both quite computationally expensive operations, for up to few thousands times for each query to expand. As a consequence, it proved to be an infeasible approach.

Regarding the results obtained with the selected query expansion method, we want to report some remarkable phenomena:

- Very short queries were very hard to expand effectively, because when the mask was applied on them, BERT was left with a very ambiguous 2-3 words sentence, so it required more time to even attempt to find good enough substitutes and at the end it often failed to find any new query.

  As an example, the topic *"Is homework beneficial?"* was expanded like this: *"is homework acceptable ?"*, *"is homework great ?"*, *"is homework appropriate ?"*, thus failing to find any valid substitute to *homework*.

- Some queries lead to useful and consistent results. You can see an example with the 10 new queries obtained from one of Touchè 2020 topics in Table 1.

- In constrast, queries with hyphens were problematic (see Table 2)

is vaping with e - cigarettes safe ?
is vaping with anti - commerce okay ?
is vaping with mini - com okay ?
is vaping with a - messaging possible ?
is vaping with no - books dangerous ?
is vaping with re - reader safe ?
is vaping with e - book dangerous ?
is vaping with z - commerce illegal ?
is vaping with no - mail allowed ?
is vaping with half - readers bad ?

Table 2
Examples of useless new queries due to hyphens

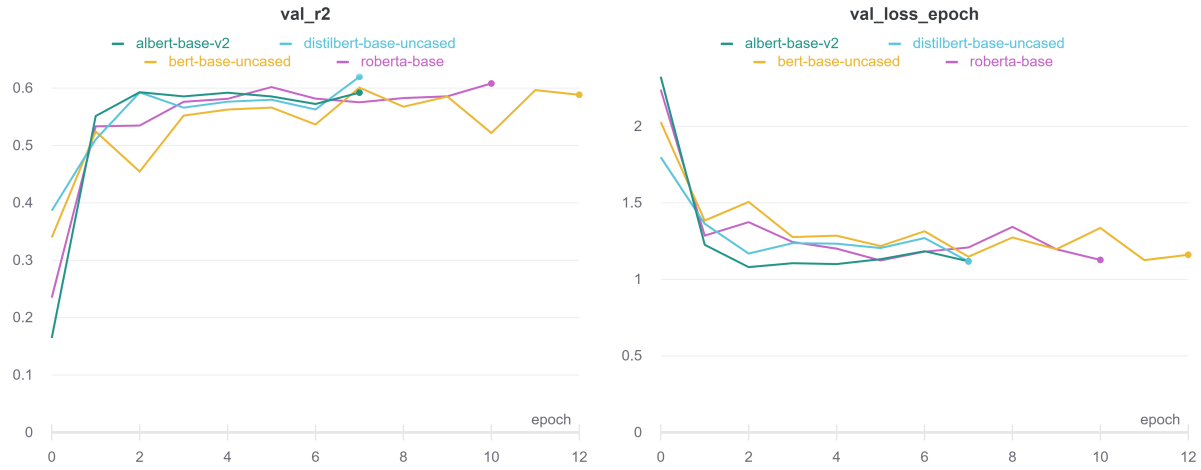
should abortion be legal ?
should divorce be allowed ?
should divorce be outlawed ?
should abortion be illegal ?
should abortion be outlawed ?
should abortion be acceptable ?
should abortion be compulsory ?
should abortion be banned ?
should divorce be prohibited ?
should divorce be acceptable ?

Table 3
Examples of peculiar term substitution operated by BERT

- We also noticed that some specific pairs of words are consistently recognized by BERT as very similar. In the example in Table 3, the terms *abortion* and *divorce*, although being far from having the same meaning, are being replaced in new queries interchangeably. This is almost certainly due to bias in BERT's pre-training dataset, where these two words used to appear together very often.

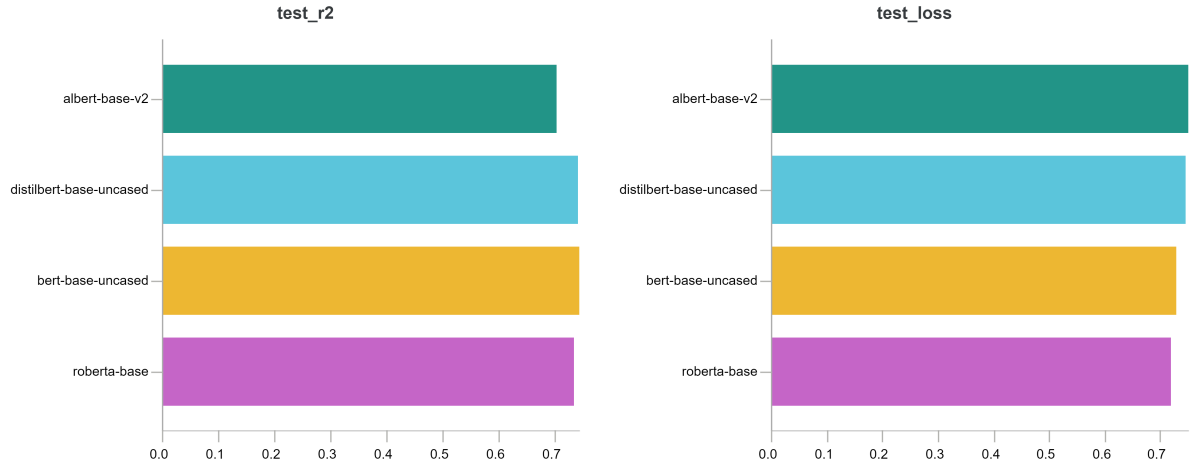
5.3. Argument Quality

The four final models had very similar results in term of validation R^2, between 0.6 and 0.62 as can be seen in 1a. This is also true for the shape of the validation loss 1b. Also on the test set the four models had very similar results, with BERT and DistilBERT being slightly better than the other two (2a, 2b). Compared to results on the same dataset obtained by [25], we were able to produce a more powerful regressor, since they report that the MSE loss of their best (ensemble) model on the same dataset is 1.322 for combined quality, while on the test set our best model is capable of achieving a value of 0.718. This is to be expected, as they used a simple SVR using pre-processed textual features while in our case we could count on the full word-level attention typical of transformer models.



(a) (b)

Figure 1: Plots of Validation R^2 and loss as functions of epochs. Some curves are shorter due to early stopping.



(a) (b)

Figure 2: Bar plots of test R^2 and loss

5.4. Overall Retrieval Pipeline

Table 4 shows the runs in decreasing order of nDCG@5 score for different combinations of parameters. The baseline score, obtained using Dirichlet similarity, is not very far from the best score of our solution, which presents an improvent of 0.67%. We expected this difficulty in improving the baseline with deep learning methods as this was also proven in Touché 2020 and may be due to the difficulty of developing efficient modules for argument quality prediction and query expansion.

Run	AQE	α	β	t	max_docs	n_{rerank}	quality model	$R(q,d)$	nDCG@5
lunar-sweep-201	no	0.75	-	0	100	5	BERT	normalize	**0.8279**
chocolate-sweep-50	no	0.1	2	0	100	5	BERT	sigmoid	0.8273
volcanic-sweep-138	no	0.5	0.8	0	100	5	BERT	sigmoid	0.8271
lunar-sweep-58	no	0.1	0.2	0	100	5	RoBERTa	hybrid	0.8230
vague-sweep-rev-204	no	0.75	1.1	0	100	5	ALBERT	sigmoid	0.8229
lucene-baseline	no	0	-	0	100	-	-	normalize	0.8224
sage-sweep-rev-160	no	0.5	1.5	0	100	5	RoBERTa	sigmoid	0.8093
distinctive-sweep-110	no	0.1	0.8	0	100	15	ALBERT	hybrid	0.7992
good-sweep-85	yes	0.1	0.3	0	100	15	DistilBERT	hybrid	0.6857
expert-sweep-rev-95	yes	0.1	0.3	0	100	20	RoBERTa	hybrid	0.6801

Table 4
List of 10 selected runs, sorted in decreasing order of nDCG@5. Runs sent to Touché are highlighted (the baseline is in light blue)

As it is clear from the last two rows of Table 4, we didn't get any improvement by using query expansion. This was an expected outcome since, as described in 5.2, some queries are well expanded, so they managed to produce a set of synonyms that allow the searcher to retrieve a wider range of arguments that fall in the same topic, while others change their original meaning, leading to a score that is lower than what they would have achieved without these incoherent queries. As mentioned before, this phenomenon is linked both to the size of the original query and to how easily replaceable its terms are, so even setting a higher number of generated queries would not have improved the expansion quality in any noticeable way.

A few combinations of parameters allowed the quality re-ranker to slightly improve the baseline score. This happens when the re-ranker permutates the top-5 arguments, i.e. without changing the set of arguments that influences nDCG@5.

As mentioned in 5.1, the boost assigned to matches occuring on the title of an argument didn't lead to better results: one reason may be that this boost pushes up arguments that inherit the title from the parent discussion without containing a relevant body. This issue may be solved by a quality model that is able to significantly distinguish good arguments from bad arguments, but the quality scores assigned to the retrieved arguments are often not very far from the neutral score. For this reason, we did not log any test with titleboost different than 0 to concentrate on other parameters.

Figure 3 summarizes the performance of the 10 selected runs on each topic. This allows us to see that, while at least ten topics lead consistently to optimal performance among all the runs, a few others perform especially bad. Let's discuss some examples of the latter case:

- Topic 8 ("Should abortion be legal?") gets a nDCG@5 score of 0 on both runs which include query expansion (*good-sweep-85* and *expert-sweep-rev-95*). This is clearly linked to how this topic is expanded, as we have already shown in Table 3, and a good motivation for this behaviour has already been discussed in 5.2.

- Also topic 10 ("Should any vaccines be required for children?") manages to get a nDCG@5 of 0 on the same runs. Having a look at how query expansion worked on this topic, we think it may be due to the term *required* being replaced interchangeably with *mandatory*

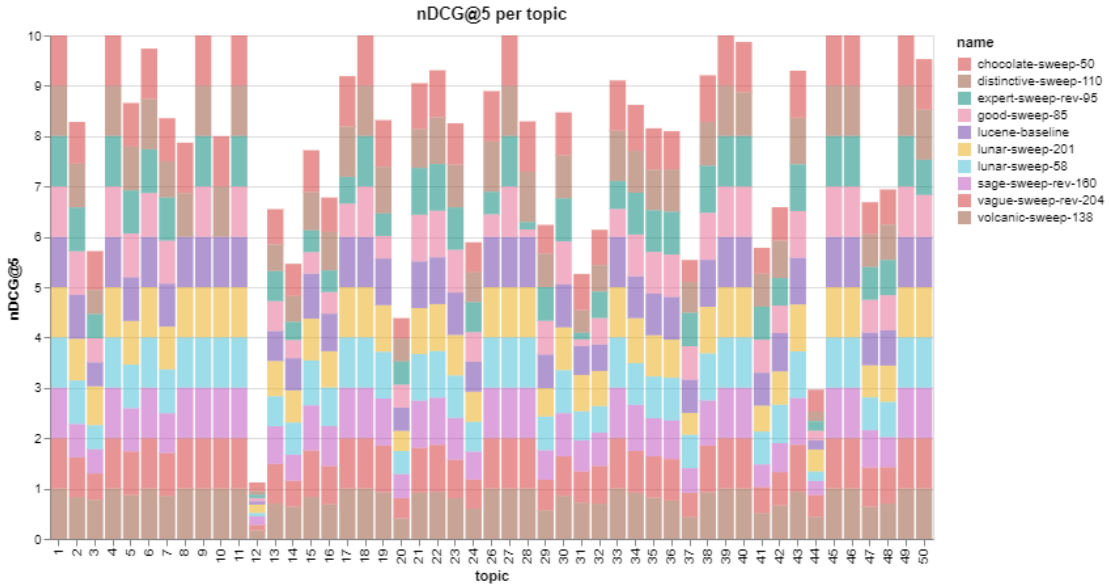


Figure 3: Bar plot of nDCG@5 scores for each of the 10 selected runs grouped by topic

and *recommended*, whose meanings are similar but their lexical nuances really make the difference in this specific context.

- On the other hand, we observe that topics such as 12 ("Should birth control pills be available over the counter?"), 20 ("Is drinking milk healthy for humans?") and 44 ("Should election day be a national holiday?") perform uniformly bad among all of the selected runs. We suggest it could be the result of a misunderstanding of the expression *over the counter*, which is possibly not very common, but also of the odd specificity of these questions. For example, in topic 20 the presence or absence of the term *milk* changes the meaning of the topic completely, making it difficult to retrieve actually relevant documents.

The data of Table 4 and the plot of Figure 3, together with some additional visualizations, can be explored in our W&B Report[6].

5.5. Statistical Significance Analysis

In Figure 4, we plotted the box plots of the runs listed in Table 4: by looking at the plot we can group together the first eight runs and the last two. In the first group, we noticed the following points:

- The mean presents a more or less constant behaviour;

[6]W&B Report available here

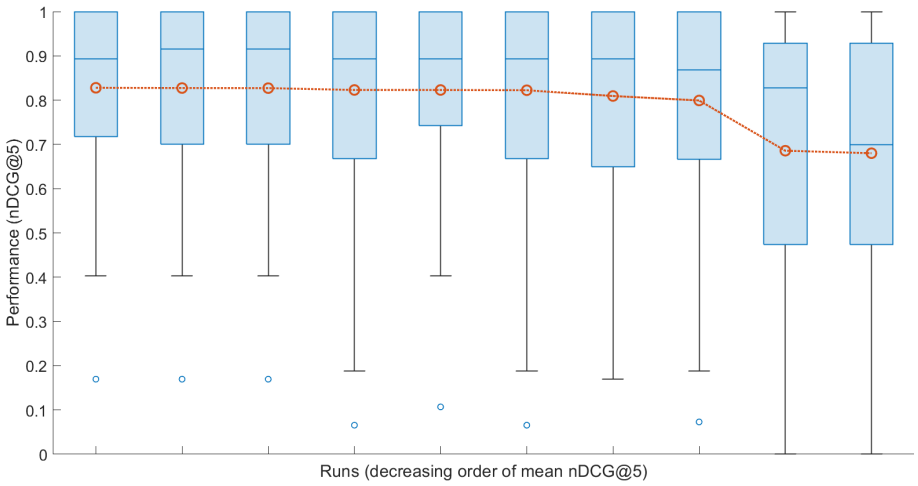


Figure 4: Box plot of the runs presented in Table 4. Red dots indicate average nDCG@5

- The *Inter-Quartile Range (IQR)* increases from left to right: this implies that the performance becomes less concetrated around the mean and thus that the system presents greater variance;
- The length of the lower whisker increases from left to right: given the limited amount of topics, it means that for some of the topics, from run 4 onwards, there is a significative drop in performance;
- The first three runs have a more concentrated distribution, but still present an outlier, which very likely corresponds to topic 12.

In the last two runs, which are those with query expansion, we see a considerable drop in mean nDCG@5 together with a large IQR. The lower whisker touches the minimum value of 0 and since the first quartile is just below 0.5, we can assume that almost 25% of the observations (i.e. topics) have a score below 0.5. As argued above, this can be justified with the great variability that query expansion introduces.

Table 5 consists of all the pairs of runs which have proved to be significatively different one from the other, according to Student's t-test performed on all the possible pairs of runs among the 10 selected ones. As mentioned in section 4, we considered a significance level $\alpha_t = 0.001111$ to keep the *Family-wise Error Rate* lower than 5%, so all pairs of runs whose *p-value* is higher than α_t are not considered to be significatively different. To be more precise, the smaller the *p-value* associated to a pair of runs is, the more likely it is that the topic-wise nDCG@5 scores belonging to these two runs come from different distributions.

We can clearly see that the selected runs are split between those applying *Query Expansion*, which are *good-sweep-85* and *expert-sweep-rev-95*, and those not using it, confirming that this

Run 1	Run 2	p-value
good-sweep-85	vague-sweep-rev-204	0.000972
good-sweep-85	volcanic-sweep-138	0.000742
good-sweep-85	lunar-sweep-58	0.000674
good-sweep-85	lunar-sweep-201	0.000706
good-sweep-85	chocolate-sweep-50	0.000724
good-sweep-85	lucene-baseline	0.000706
expert-sweep-rev-95	vague-sweep-rev-204	0.000720
expert-sweep-rev-95	volcanic-sweep-138	0.000549
expert-sweep-rev-95	lunar-sweep-58	0.000497
expert-sweep-rev-95	lunar-sweep-201	0.000522
expert-sweep-rev-95	chocolate-sweep-50	0.000535
expert-sweep-rev-95	lucene-baseline	0.000521

Table 5
List of pairs of runs which are significatively different, according to Student's *t*-test

parameter is by far the most relevant when deciding the outcomes of a run. Another interesting result is that even *sage-sweep-rev-160* and *distinctive-sweep-110* failed to be considered diverse enough from all of the others. To be fair, we performed the *t*-test again with slightly higher value of α_t to find out that *distinctive-sweep-110* ended up with a rather low *p-value* of 0.001377 w.r.t. *lunar-sweep-58* and of 0.001595 w.r.t. *lucene-baseline*, pointing out that also the n_{rerank} parameter has a significative impact on results.

6. Conclusions and Future Work

As noted in [15], the usage of state-of-the-art architectures for argument retrieval in the previous edition of the challenge only slightly improved the results obtained by the DirichletLM baseline and our results seem to point in that direction. Nevertheless, we believe this to be due to the difficulty of the task and that deep-learning based approaches could provide interesting insights and improvements over traditional baselines.

First of all, it would be interesting to use transformer-based models also for the retrieval part: in particular, similarly to what was done last year in [26], it could be interesting to design a vector space IR model based on BERT, which could produce document embeddings fine-tuned for this task. For example, BERT models tailored to produce embeddings of long spans of text such as SBERT [27] could be deployed on this task. This integration would be orthogonal to our work and subsitute the DirichletLM retrieval model.

As far as query expansion is concerned, we believe that at least one of the approaches which we briefly discussed in 5.2 is worth some additional effort: *Back-translation* proved to be a really promising way of exploiting pretrained transformers-based models to generate differently phrased sentences, while preserving most of the original meaning, so we are eager to investigate on this tecnique. Another method that we think should lead to more stable query expansion results is the one about partial embeddings-masking which was proposed by [12], variants of which are substantially unlimited.

Regarding argument quality prediction, it could be interesting to study the performance of

our model in a fine-tuning approach instead of the adaptation one we used. In addition to this, it could be interesting to make the re-ranker predict the three types of quality (logical, dialectical and rhetorical) to see whether it would be possible for them to capture these three aspects independently. The re-ranking of results could then be performed based only on one of these qualities or on the overall quality score. Clearly, this adds a substantial layer of difficulty for the language model but could yield interesting details on how it internally represents these sophisticated features of language.

References

[1] H. Wachsmuth, M. Potthast, K. Al-Khatib, Y. Ajjour, J. Puschmann, J. Qu, J. Dorsch, V. Morari, J. Bevendorff, B. Stein, Building an argument search engine for the web, in: Proceedings of the 4th Workshop on Argument Mining, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 49–59. URL: https://www.aclweb.org/anthology/W17-5106. doi:10.18653/v1/W17-5106.

[2] Y. Ajjour, H. Wachsmuth, J. Kiesel, M. Potthast, M. Hagen, B. Stein, Data Acquisition for Argument Search: The args.me corpus, in: C. Benzmüller, H. Stuckenschmidt (Eds.), 42nd German Conference on Artificial Intelligence (KI 2019), Springer, Berlin Heidelberg New York, 2019, pp. 48–59. doi:10.1007/978-3-030-30179-8_4.

[3] S. Robertson, H. Zaragoza, M. Taylor, Simple bm25 extension to multiple weighted fields, in: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM '04, Association for Computing Machinery, New York, NY, USA, 2004, p. 42–49. URL: https://doi.org/10.1145/1031171.1031181. doi:10.1145/1031171.1031181.

[4] R. Levy, B. Bogin, S. Gretz, R. Aharonov, N. Slonim, Towards an argumentative content search engine using weak supervision, in: Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 2066–2081. URL: https://www.aclweb.org/anthology/C18-1176.

[5] C. Stab, J. Daxenberger, C. Stahlhut, T. Miller, B. Schiller, C. Tauchmann, S. Eger, I. Gurevych, ArgumenText: Searching for arguments in heterogeneous sources, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 21–25. URL: https://www.aclweb.org/anthology/N18-5005. doi:10.18653/v1/N18-5005.

[6] H. K. Azad, A. Deepak, Query expansion techniques for information retrieval: A survey, Information Processing Management 56 (2019) 1698–1735. URL: https://www.sciencedirect.com/science/article/pii/S0306457318305466. doi:https://doi.org/10.1016/j.ipm.2019.05.009.

[7] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Huggingface's transformers: State-of-the-art natural language processing, 2020. arXiv:1910.03771.

[8] S. Kuzi, A. Shtok, O. Kurland, Query expansion using word embeddings, 2016, pp. 1929–1932. doi:10.1145/2983323.2983876.
[9] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, 2013. arXiv:1310.4546.
[10] D. Victor, Neuralqa: A usable library for question answering (contextual query expansion + bert) on large datasets, 2020.
[11] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.
[12] W. Zhou, T. Ge, K. Xu, F. Wei, M. Zhou, BERT-based lexical substitution, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 3368–3373. URL: https://www.aclweb.org/anthology/P19-1328. doi:10.18653/v1/P19-1328.
[13] H. Wachsmuth, N. Naderi, Y. Hou, Y. Bilu, V. Prabhakaran, T. A. Thijm, G. Hirst, B. Stein, Computational argumentation quality assessment in natural language, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 176–187. URL: https://www.aclweb.org/anthology/E17-1017.
[14] S. Gretz, R. Friedman, E. Cohen-Karlik, A. Toledo, D. Lahav, R. Aharonov, N. Slonim, A large-scale dataset for argument quality ranking: Construction and analysis., in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 7805–7813.
[15] L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696.
[16] L. Gienapp, B. Stein, M. Hagen, M. Potthast, Webis Argument Quality Corpus 2020 (Webis-ArgQuality-20), 2020. URL: https://doi.org/10.5281/zenodo.3780049. doi:10.5281/zenodo.3780049.
[17] M. E. Peters, S. Ruder, N. A. Smith, To tune or not to tune? adapting pretrained representations to diverse tasks, in: Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), 2019, pp. 7–14.
[18] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, arXiv preprint arXiv:1910.01108 (2019).
[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).
[20] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, in: ICLR 2020 : Eighth International Conference on Learning Representations, 2020.
[21] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.

emnlp-demos.6.
[22] D. P. Kingma, J. L. Ba, Adam: A Method for Stochastic Optimization, in: ICLR 2015 : International Conference on Learning Representations 2015, 2015.
[23] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: Advances in Neural Information Processing Systems, volume 30, 2017, pp. 971–980.
[24] L. Biewald, Experiment tracking with weights and biases, 2020. URL: https://www.wandb.com/, software available from wandb.com.
[25] M. Bundesmann, L. Christ, M. Richter, Creating an argument search engine for online debates, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_182.pdf.
[26] C. Akiki, M. Potthast, Exploring argument retrieval with transformers, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_241.pdf.
[27] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3980–3990.