Martin-Luther-Universität Halle-Wittenberg
Natuwissenschaftliche Fakultät III
Studiengang Informatik

Precision-Oriented Argument Retrieval

Bachelor's Thesis

Danik Hollatz

1. Referee: Prof. Dr. Matthias Hagen
2. Referee: M. Sc. Alexander Bondarenko

Submission date: May 10, 2022

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Halle, May 10, 2022

.....
Danik Hollatz

Abstract

Argumentation plays an enormous role when it comes to building an opinion on a controversial topic or making personal choices. The Internet provides a huge amount of opinions of other people as well as argumentative texts, where among useful and qualitative good argumens, there are also to find one-sided, populist, or faked ones.

In this thesis, we create multiple precision-oriented approaches for the task of argument retrieval. For this purpose, we use (1) models deployed in TARGER to find premises and claims in arguments, (2) the pre-trained DeepCT model to extract and use semantic importance of words in arguments, and (3) create 9 different corpora to fine-tune the DeepCT model and use them for further experiments.

For our experiments and evaluation, we use args.me corpus and relevance judgments offered by Touché lab. Using manually annotated judgments we compare our results with other research teams, that have taken participation in Touché task 1: argument retrieval for controversial questions in the years 2020 and 2021.

Compared with the results of participants of Touché task 1 in the years 2020 and 2021, our fine-tuned models outperform the most effective participant approaches in the nDCG@5 measurement.

Contents

Chapter 1

Introduction

> "An information retrieval system will tend not to be used whenever it is more painful and troublesome for a customer to have information than for him not to have it."
>
> *Calvin N. Mooers, Zator Technical Bulletin*

Much of the arguments available on the web can be faked, biased, or populist, which is why a study field argument retrieval plays a big role in today's world [55]. We believe that this implies that the retrieval process of argumentative texts should not only consider the used words in documents, but also their semantic importance in the arguments. In this thesis, we aim to create precision-oriented argument retrieval approaches, which also take into account the semantic importance of the words in arguments.

Argument retrieval is a research area that deals with sorting argumentative texts by their relevance to a query on various topics [22]. Modern web search engines are working incredibly well and are able to retrieve information in a blink of an eye, but we still can observe some lack of quality of retrieval, due to the one-sidedness of retrieved documents. As the work of Ajjour et al. [2] states, the internet is filled with one-sided documents, because of various reasons such as propaganda or marketing. Due to the way the web engines, work, they will tend to retrieve documents that are similar to the query in used words, but they do not consider the semantic importance of the used words, which may lead to arguments similar to the query, but irrelevant ones retrieved. Because of that behavior, we aim to consider the semantic importance of the words in documents when retrieving arguments.

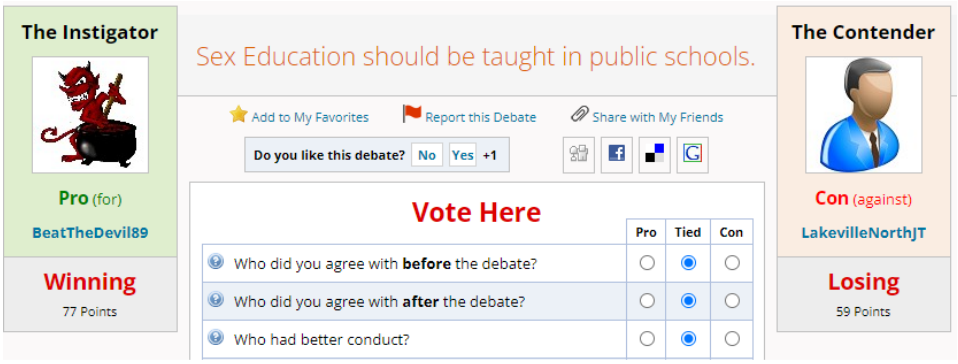


Figure 1.1: Example of a debate results on topic "Sex Education should be taught in public schools" between internet users.

The performance of modern search engines is highly influenced by bag-of-words document representations, but they only consider the frequency of the terms in the documents [17]. To increase the precision and quality of retrieved documents, in this thesis we aim to use and create approaches, that differ from frequency-based schemes. For this purpose we use (1) pre-trained deep learning network called "DeepCT" created by Dai and Callan [17] on MS-MARCO-Doc corpus[1] to estimate and use the semantic importance of terms in documents, (2) create 9 different corpora based on args.me corpus [2] to fine-tune the DeepCT model, (3) extract premises and claims from arguments via models deployed in TARGER [13] and dismiss the rest of the argument for further experiments. Args.me corpus crawls arguments from websites, dedicated to debate and contains almost 400,000 arguments (cf. example 1.1 of debate topic and outcomes, and an example 1.2 of an argument in the first debate round)

In Section 2 we overview study fields that influence argument retrieval, such as information retrieval, argument search, argument mining, deep learning, and the usage of deep learning models in the field of argument retrieval, as well provide a background to the used retrieval models and query expansion methods. We analyze various types of retrieval models and the most common approaches used in the collaborative platform for researchers named "Touché" [7–9].

In Section 3, we analyze the args.me corpus, describe how we use it to create corpora to fine-tune DeepCT models, and provide an example of a training sample.

Section 4 introduces our precision-based approaches, how we tune the parameter of retrieval models, which models from TARGER we use, and how

[1] `https://microsoft.github.io/TREC-2019-Deep-Learning/`

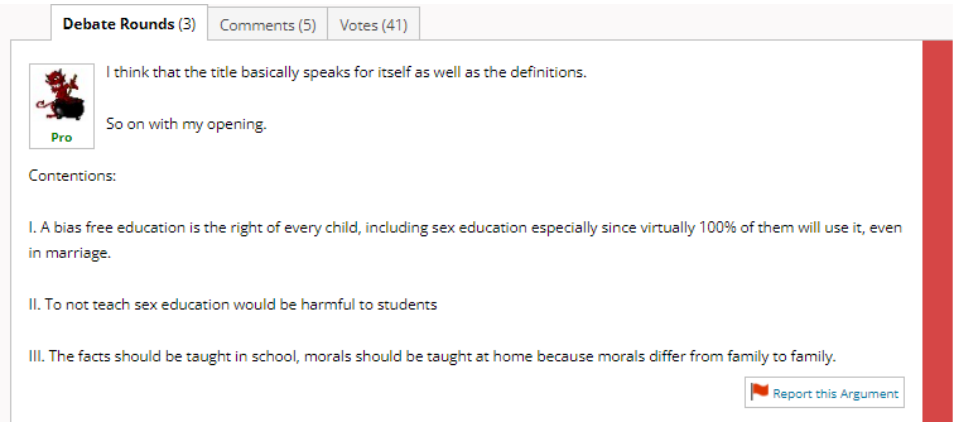


Figure 1.2: Example of an argument in the first round of the debate.

their outputs look like, as well we provide an explanation of how we fine-tune DeepCT models, what frameworks we use, analyze the output of DeepCT models and interpret the training plots of various DeepCT models.

In Section 5 we evaluate our approaches. Touché provides manually annotated relevance judgments for part of the documents contained in args.me corpus, which we use to evaluate our approaches. Touché lab has hosted two events in the years 2020 and 2021 with two tracks: (1) argument retrieval for controversial questions, and (2) argument retrieval for comparative questions. We provide an evaluation of our approaches in nDCG@5, nDCG@25, and Bpref metrics in rankings that were created using all documents from args.me corpus, and in rankings that consist only from judged documents. We compare our approaches with the best nDCG@5-score, using only judged documents, achieved by research teams, that have taken participation in the first track of Touché. We outperform the most effective participant approach in the year 2020 by 0.04, achieving an nDCG@5-score of 0.86, and outperform the best team in the year 2021 by 0.01, achieving the nDCG@5-score of 0.73. We also evaluate our approaches on all topics of the years 2020 and 2021 and their corresponding judgments.

Chapter 2

Background and Related Work

This chapter provides an overview of relevant work in the fields of information retrieval, argument retrieval, argument search, argument mining, deep learning, and deep learning models used in argument and information retrieval. We first overview a study field of information retrieval, since it is the research field, from which other fields of studies such as argument retrieval, argument search, and study of retrieval models are derived from. Then we describe the field of argument search and how the argument search engine work since it requires a knowledge of multiple fields related to the field of argument retrieval, to build one. Afterward, we describe two main aspects of any search engine: argument mining and argument retrieval. We describe the basics of deep learning and overview (1) the use of neural networks in retrieval systems, (2) how it can influence the precision of retrieved documents, (3) its advantages and disadvantages compared to the sparse retrieval systems, (4) the architecture of common deep learning model used for natural language processing—BERT.

2.1 Information Retrieval

The research field of information retrieval has come a long of way study. With the advance of computers and their capability of storing large amounts of data, it has become necessary to be able to quickly retrieve relevant documents from data saved on a computer. Starting from the 1950s, information retrieval has been dealing with the task of finding relevant information in document collections that satisfy the human-defined information need expressed as search queries [15, 49]. Among the first retrieval models $tf \cdot idf$ [28] and BM25 [47], also known as "sparse" retrieval models, were proposed. They are called so because they represent text input in one-dimensional vectors with the size of the vocabulary, in such a way, that in the case if a word from vocabulary appears in text, its entry in the vector will have a non-zero value (e.g. showing how many

times a certain word appears in the text), and otherwise zero. Since the size of vocabulary and following the size of the vector is much greater than the number of unique words in the text, most of the values in such representations will have the value of zero and only a few of them will have a value different from zero, hence called "sparse" [5]. Due to the architecture of these models, they might not consider a relevant document as such, because it contains words that are synonyms to the words used in a query but are not the same. This motivates an idea to consider not only the words themselves but also to try to extract and use the meaning of the words and hence increasing the precision of such approaches. This can be achieved using deep learning models and so-called "dense" retrieval models, which use deep learning systems.

With the growth of machine learning, there also was found a place for it in approaches in the field of information retrieval. Another type of retrieval models is based on "dense" retrieval, such approaches exploit the embeddings of text inputs created by deep learning systems, such as BERT [18], GPT-2 [46], and others. Embeddings are representations of words encoded in the form of a real-valued vector in such a way that words that have similar meanings are expected to be close to each other in the embedding space [57]. According to the work of Karpukhin et al. [29], dense models may create embeddings that are "close" to each other in the embedding space, even when they are created of texts that differ in used words, but are semantically similar to each other, while sparse retrieval models would have difficulties retrieving such context. However, dense models require additional time and resources, due to the need of using GPUs (Graphics processing units), which makes them not always the best solution to use.

Sparse and dense retrieval models, being complementary to each other [5], give an intuition of combining both these approaches, leading to the creation of so-called "hybrid" models. These models use fast and exact-matching proposed by sparse retrievers and they close the semantic gap between query and document if such appears. Work of Arabzadeh et al. [5] compare above-mentioned approaches in query latency and effectiveness of approach; in effectiveness and used resources of sparse models and improved effectiveness when using dense hybrid models; in used deep learning models for dense models.

2.2 Argument Search

Argument search is a field of study of web search engine technology, which deals with retrieving the most relevant arguments to the information need described in a query [2]. Several argument search engines have been built so far following different methodologies [31, 50, 55]. Search engines are built

differently, depending on what properties they should have, e.g. be recall- or precision-oriented [2]. The first step in building an argument search engine is acquiring data, which will be used in the engine, for instance, args.me [2] is a corpus that is consisted only of arguments and is used in building search engine args[2] in the work of Wachsmuth et al. [55]. The work by Ajjour et al. [2] describes three different argument search pipelines (cf. Figure 2.1): opposite to the args.me corpus, in the work of Stab et al. [50] arguments are being retrieved not from the corpus, which contains arguments only, but from the heterogeneous web sources. Using a deep learning system in the work of Stab et al. [50] they can distinguish between argumentative and non-argumentative units, which makes argument retrieval possible. Work of Potthast et al. [44] introduces two opposite to each other paradigms for retrieval pipelines: "mining-after-retrieval" as done in work of Wachsmuth et al. [55] and "retrieval-before-mining" which was implemented in the work of Stab et al. [50]. As described above, in the paradigm "mining-after-retrieval" arguments are first being mined and indexed offline, hence only leaving retrieving part of the pipeline to the online, whereas for the "retrieval-before-mining" paradigm heterogeneous documents are indexed and argument mining is a part of online or query time (cf. Figure 2.1). Using distant supervision and harvesting arguments in the offline phase increases the precision of retrieved documents but will decrease the recall, since some argumentative units may be skipped [2]. Corpus used in the work of Stab et al. [50] was also translated to the German language via Google translate API (Application Programming Interface) [51]. Afterward, deep neural networks were retrained and evaluated, achieving worse but comparable results and hence giving users an opportunity to find arguments on controversial topics on their website,[3] providing German and English languages to formulate a query and retrieve arguments in the same language. According to the work of Levy et al. [31], their approach is more close to the one described in the work of Ajjour et al. [2]: they also mine arguments from recognized sources, such as Wikipedia and others, and exploit deep neural networks in order to find claims and premises from the heterogeneous texts.

2.3 Argument Mining

Argument mining is the process of finding argumentative units in a free text. The resulting dataset after argument mining will heavily influence whether the resulting search engine will be recall- or precision-oriented. Since manually annotating argumentative units in texts is an expensive and time-consuming

[2] `www.args.me`

[3] `www.argumentsearch.com`

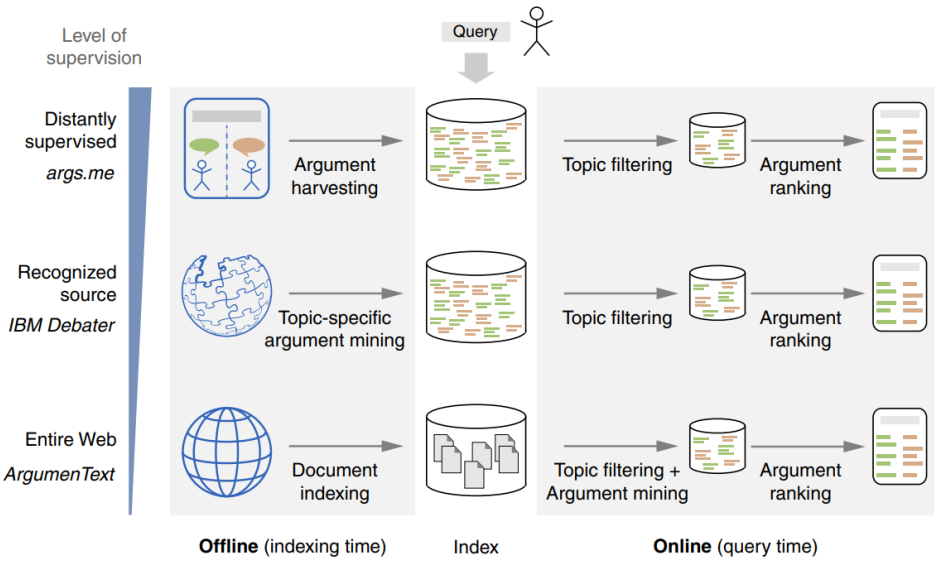


Figure 2.1: Overview of different argument search pipelines [2]. From top to bottom: work of Ajjour et al. [2] and Levy et al. [31], implementing their search engines *args* and *ProjectDebater* according to the mining-after-retrieval paradigm, whereas *ArgumenText* [50] uses retrieval-after-mining paradigm and includes mining to the query time.

process, it becomes more and more important to automize this process [34]. Argument mining is a part of the argument search pipeline and it also can be implemented differently, e.g. in works mentioned before: Levy et al. [31], Stab et al. [50]. An argument mining framework, that we use in our work, is called TARGER [13].[4] Given an input as free-text, models deployed in TARGER extract argumentative units from it, such as premises and claims. Using models deployed in TARGER, we extract premises and claims from documents from args.me corpus, dismissing the rest of the documents, and hence try to increase the precision of retrieval. One of the advantages of TARGER, compared to other argument mining frameworks, is that it provides 8 different pre-trained models, giving a user an opportunity to find out which model suits his or her needs best. These models use a deep neural network to detect argumentative units. Another framework for argument mining is called "MARGOT" and just like TARGER, MARGOT [35] provides an online web-service[5] for extracting argumentative units from free-text inputs, but in comparison, they do not use deep learning models, but approaches that are more likely to be considered as

[4] `https://demo.webis.de/targer/`
[5] `http://margot.disi.unibo.it/`

a part of machine learning. TARGER outperforms MARGOT in F1 scores of claim and evidence detection and works faster.

2.4 Argument Retrieval

One of the last steps in the argument search pipeline is to retrieve documents. To extract the most relevant arguments to the users need retrieval models are used. There are no known retrieval models that were specially created for the task of argument retrieval [44]. Search engines such as *args* and *ArgumenText* both use Lucene's implementation of BM25 retrieval model, while it is not specified what is used in *ProjectDebater*. Work of Wachsmuth et al. [55] offers an user study of different retrieval models, such as BM25 [47], Terrier's [45] implementations of DPH [3, 53], Dirichlet-smoohted Language Model [58], and $tf \cdot idf$. Based on args.me corpus, Wachsmuth et al. [55] evaluated four above mentioned retrieval approaches in 4 aspects: relevance, rhetoric, logic, and dialectic quality of retrieved arguments. As the user study shows, DPH yields to achieve overall best results among other models.

With an increasing interest in the field of argument search [19] there appear new platforms dedicated to hosting events, where research teams can compare and evaluate their approaches in argument retrieval. One of such platforms is created by Touché lab [7–9], where it is already been held 3 events, where teams could have participated in one or both following tracks:

1. Argument retrieval from corpus consisted of arguments from online debates, to provide opinions of other people on the given controversial topic.

2. Argument retrieval from corpus consisted of heterogeneous Web documents, in order to provide documents, that could help a user make a conclusion on the given comparative question.

Such platforms facilitate the growth of community interest in argument retrieval and ease the comparison of the results since the approaches are evaluated on the same corpora. Each participating team is then asked to share their approaches, hence giving a lot of insights into what kind of methods they use and creating new ways of research to be done.

Comparing pipelines of participants of Touché lab, a common approach in works of Dumani and Schenkel [20], Green et al. [25], Luu and Weder [36] was to exploit embeddings created by a pre-trained deep learning model in order to evaluate the quality of arguments and consider them in the re-ranking. Luu and Weder [36] use GPT-2 instead of BERT to create embeddings. A lot of participants also used query expansion [11, 20, 25, 36], while document

expansion was only used by one team [39]. Clustering of premises and claims was used in works of Bundesmann et al. [11], Dumani and Schenkel [20].

All the above-mentioned works are either considering qualities of arguments that are predicted with already pre-trained deep learning systems and hence increase the precision of the retrieved documents or they use techniques, that increase recall, such as query and document expansions. But none of the works has tried to fine-tune deep learning networks on the used args.me corpus. That also motivates our work to investigate the performance of the fine-tuned DeepCT model to implement a precision-oriented argument retrieval pipeline.

Retrieval models such as BM25 and $tf \cdot idf$ are based on the term frequency, although such methods also show good results, they do not consider the semantic importance of the words, which can lead to miss information from the text. The work of Dai and Callan [17] provides a solution to this problem, they fine-tuned BERT so that it could predict the semantic importance of the words in passages. After calculating the importance of the words, Dai and Callan [17] then created adjusted passages, where each word is repeated depending on its predicted importance. Words with higher importance are repeated more often and respectively words with lower importance are either not repeated at all (removed from new passage) or repeated not that often. After creating new passages and using variations of BM25 retrieval models, the approach of Dai and Callan [17] outperforms baseline, in which the index of four different corpora consists solely of the term frequency of words.

To fine-tune BERT Dai and Callan [17] added a fully-connected layer to the outputs of the pre-trained BERT model to predict the semantic importance of the words by training the added layer (cf. Figure 2.2). Dai and Callan [17] offer three different strategies of creating ground truth data:

1. Content-based weak-supervision.
2. Relevance-based supervision.
3. Weak supervision based on machine-generated pseudo-relevant feedback labels.

We use variations of the first strategy and describe it in Section 4.

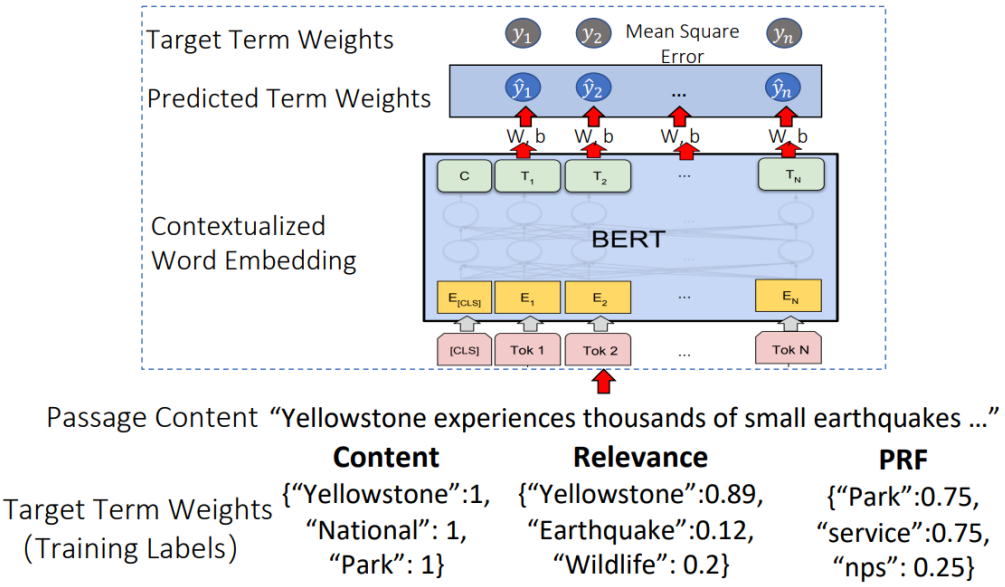


Figure 2.2: At the top is shown the architecture of DeepCT created by Dai and Callan [17] and the bottom is shown an example of arbitrary passage and the importance of the words in this passages, created by above mentioned strategies.

2.5 Retrieval Models and Query Expansion

This section provides a brief overview of the RM3 query expansion method and retrieval models used in this thesis.

2.5.1 BM25 Retrieval Model

BM52 is a probabilistic retrieval model, which estimates the relevance score of a document based on query terms, frequency of terms in document, document frequency and document length. For the query Q and for the document D, relevance score can be calculated by the following equation [16]:

$$\sum_{i \in Q} \log \frac{(r_i + 0.5) \,/\, (R - r_i + 0.5)}{(n_i - r_i + 0.5) \,/\, (N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1)\, f_i}{K + f_i} \cdot \frac{(k_2 + 1)\, qf_i}{k_2 + qf_i},$$

where the summation is done over all terms i in the query Q, r_i is the number of relevant documents containing term i, n_i is the number of documents where term i occurs, N is the total number of documents in the corpus, R is the number of relevant documents for this query, frequency of the term i in the document is saved in variable f_i and in the query is saved in qf_i, and k_1, k_2 are

variables whose values should be found empirically, and K is defined as follows:

$$K = k_1 \left((1 - b) + b \cdot \frac{dl}{avdl} \right),$$

where b is a parameter that also should be found empirically, dl represents the length of the document, and $avdl$ shows the average length of documents in the corpus.

2.5.2 Dirichlet-smoothed Language Model

To estimate the relevance score of a document Dirichlet-smoothed language model uses knowledge about how many times a word from the query appears not only in the documents, but also in the entire corpus, as:

$$\sum_{i \in Q} \log \frac{f_i + \mu \frac{cf_i}{\sum_i cf_i}}{dl + \mu},$$

where Q, i, f_i, and dl are defined the same as in the last section, μ is a smoothing variable that should be found empirically, and cf_i represents the frequency of a term in the entire corpus [4].

Among the advantages of the Dirichlet-smoothed language model it never assigns zero probability to a term, it considers the frequency of a word in the entire corpus, and it normalizes the documents and because of that short and long documents are estimated comparably equally [4].

2.5.3 RM3 Query Expansion

Query expansion methods were created with the idea of extending original queries with additional terms so that relevant documents would be retrieved with a higher probability [12]. Extend terms are derived either from feedback documents that were created by users (for example judgments) or from the retrieved most relevant documents to the query [24].

Relevance feedback through a relevance model RM1 calculates the weight of a certain term t via following equations:

$$\mathrm{RM}\,1(t, R) = \sum_{d \in R} P(t \mid d) \frac{\mathrm{score}(i, d)}{\sum_{d' \in R} \mathrm{score}\,(i, d')},$$

where R is a set of relevance feedback document to the query i, $P(t \mid d)$ represents the probability of term t occurring in document d and $score(i, d)$ represents the estimated relevance score of d to the original query i.

Relevance model RM1 is improved via linear combination of RM1 weight and query term weight as:

$$\mathrm{RM}\,3(t, R) = \lambda \cdot \mathrm{RM}\,1(t, R) + (1 - \lambda) \cdot P(t \mid i),$$

where $P(t \mid i)$ is the probability that term t occurs in the query i and λ in range $[0, 1]$ is the feedback impact. In the PyTerrier implementation of RM3, it considers only top M expansion terms with the highest weights to avoid efficiency issues of retrieval for very long queries.

2.6 Deep Learning

This section is heavily inspired by the book of Chollet [14] and provides a brief introduction to the history of artificial intelligence, the machine learning paradigm, and how machine learning approaches work. In this thesis, we use deep learning systems for argument retrieval via fine-tuning the BERT model, which is used in DeepCT architecture, and by using models deployed in TARGER. According to Chollet [14], the history of artificial intelligence begins in the 1950s, at the time when a group of pioneers decided to investigate if computers could "think" in some way. The main goal of artificial intelligence is to automatize tasks that are performed by humans, it is a broad field, which includes in it machine learning and deep learning, but also has a wide range of other fields included.

Machine learning is a programming paradigm that differs from the "classical" programming paradigm. Classical programming is based on a set of rules, defined by the programmer and followed by them. As for machine learning, this set of rules is created by the computer itself, while machine learning systems are rather "trained" than defined with rules. Such a system requires a lot of data relevant to the task it should "learn" to solve, the rules in such systems are created by the influence of training samples and adjusting of the model to minimize the loss.

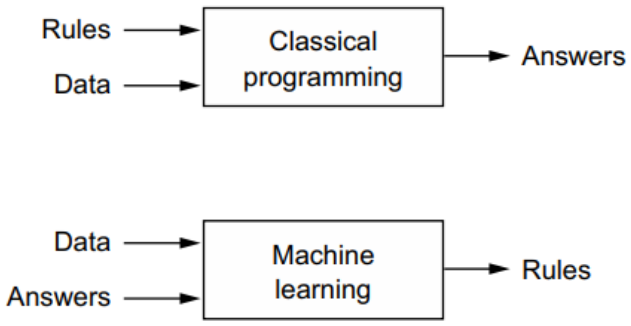


Figure 2.3: Paradigms of classical programming and machine learning. Figure 1.2, page 4 [14].

Both machine learning and deep learning are having the same goal: to learn to produce useful representations of the data. This means, that given input data, both machine and deep learning approaches learn how to transform input data into some representations, which then is transformed into the meaningful (for the humans) output.

Deep learning is considered to be a subfield of machine learning, the difference between machine and deep learning consists in the fact that the systems of the latter are built with multiple "layers", in other words, some functions that create representations and adapt to the data. The "depth" of the model is defined by the number of layers it is built with, often involving tens or hundreds of them.

Layers are parameterized by their weights. Weights are the core of any deep learning system and are represented by a bunch of real numbers. While being trained, weights are being adjusted, so that a deep learning network minimizes the loss function (in other literature also often called "cost function" or "error objective function").

Loss functions are necessary to understand how good or bad the model works. Considering the output of the deep or machine learning system, a loss function returns a score that defines how far is the prediction from the real data (also called "ground truth" data). Depending on the loss score, weights are accordingly adjusted. The process of adjusting the weights of the models is called "backpropagation" and is performed by its optimizer. We omit the description of what is optimizer and how does backpropagation work due to it not being the main objective of our research and requires a lot of additional explanations.

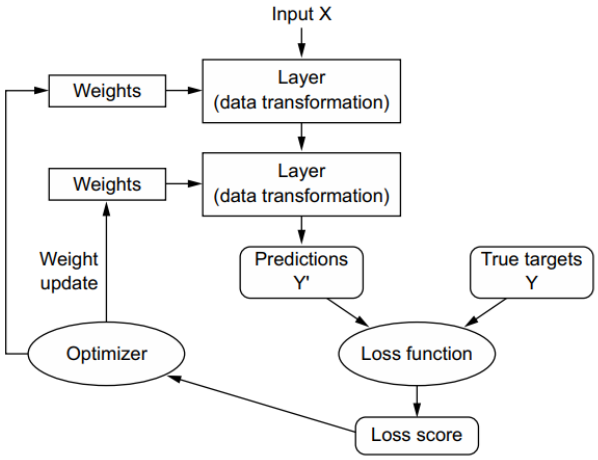


Figure 2.4: Training process of neural network, consisted of data transformations through layers, receiving the predictions, comparing them to the true targets or ground truth, calculating the loss score via loss function and updating the weights of the layers with optimizer. Figure 1.9, page 10 [14].

Deep learning systems made it possible to automize and achieve much better results in such tasks as image classification, speech and handwriting transcription, text-to-speech conversion, and many other fields [14].

2.6.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a language representation model, created by Devlin et al. [18]—a group of researchers from Google. Language models can be seen as a function that estimates the probability of certain linguistic units, such as symbols, words, or sequences [15]. At the time of publishing, BERT was a breakthrough model, achieving state-of-the-art results on eleven natural language processing tasks. The main idea, which differs BERT from other language models, is the bidirectional training of an attention mechanism—Transformer [54]. The idea behind the attention mechanism is to recalculate the representation of the sequence, produced by other layers in the deep learning model. It is generally accepted, that this attention mechanism is what makes BERT outperform the old state-of-the-art approaches [48]. The word "bidirectional" here means that the context of the token or word in the sequence is created depending on the tokens both from the left and right side of him, while older approaches were considering either the tokens from the left or the right sight of the certain token.

Another interesting idea in the training process of BERT is its prediction goal. Compared to ELMo [27], which was also a breakthrough architecture at the time it was published, BERT possesses two prediction goals: Masked Language Model (MLM) and Next Sentence Prediction (NSP). While the training process for ELMo is built by predicting each next word in the sequence, thus recreating the whole sequence, in the case of BERT, 15% of tokens are being randomly masked and the goal is to predict the masked tokens.

As written above, another prediction goal of BERT is Next Sentence Prediction. The idea behind this task is to predict whether the second sentence of the given pair of sentences is subsequent to the first one. In the training phase, 50% of the given pairs are subsequent and the other 50% are not.

Work of Rogers et al. [48] gives an insightful overview of research on what is actually learned by embeddings created by BERT and what is the advantage of MLM and NSP objectives. Work of Mickus et al. [41] states that due to the NSP objective representations of tokens created by BERT are dependent on the position of the token in a sentence and through MLM objective BERT shows knowledge for semantic roles [23].

What also makes BERT especially good is its ease to fine-tune. For example, by taking the pre-trained model and adding a linear layer to the model, we can train the model for the classification task, e.g. sentiment analysis. Similar to this approach, in Section 4 we describe how we fine-tune the BERT model in our argument retrieval pipeline.

Chapter 3

Data and Data Preprocessing

This chapter describes the corpus that is used for argument retrieval experiments including its characteristics, data sample examples, and data processing. We also give an overview of how training examples for the DeepCT model [17] are created.

3.1 Data and Data Access

In our work, we use the args.me corpus that consists of 387,740 arguments from 59,637 debates crawled from 4 online debate portals [2]. Each argument or data entry in args.me corpus consists of a conclusion, one or more premises, and their respective stances: either supporting or attacking the given conclusion. Additionally, the context such as the text of the debate in which premises and claims appear and meta information such as acquisition time, the title of the debate, and its URL are provided; for each argument, a unique ID is assigned. To the date of writing, args.me is one of the largest corpora created for argument retrieval.[6]

Instead of using json files downloadable from online resources, we use ir_datasets—a package for Python created by MacAvaney et al. [37]. We found it easier to work with args.me dataset via mentioned package, since it provides useful features such as selecting ranges of data, fast lookup of documents content, much easier access to data in comparison to reading json files, and others [37].

For the evaluation of our approaches, we use two sets of controversial topics (see example in Listing 3.1), saved in xml format. These sets were used in Touché in the years 2020 and 2021 for the task of argument retrieval for controversial questions. For each topic, we aim to retrieve the most relevant documents from

[6] `https://webis.de/data/args-me-corpus.html`

the entire args.me corpus. To measure the performance of retrieval for each set of topics we use the corresponding set of relevance judgments created by human assessors, named qrels. Qrels are saved in Text REtrieval Conference (TREC) format and can be easily read by PyTerrier [38][7]—a Python framework that we use for retrieval and for evaluation of approaches. Documents annotated in qrels with relevance judgments have labels of -2, 0, 1, and 2, where -2 means that the document is spam to the query, and the higher the label gets, the more relevant to the query the document is.

Listing 3.1: Example of controversial topic. Element "number" represents the serial number of topic. Element "title" displays the controversial topic. Element "description" is a possible context of search and element "narrative" describes what kind of documents are considered relevant for this topic.

```
<topic>
<number>1</number>
<title>Should teachers get tenure?</title>
<description> A user has heard that some countries do give teachers tenure
    and others don't. Interested in the reasoning for or against tenure, the
     user searches for positive and negative arguments. The situation of
    school teachers vs. university professors is of interest. </description>
<narrative> Highly relevant arguments make a clear statement about tenure
    for teachers in schools or universities. Relevant arguments consider
    tenure more generally, not specifically for teachers, or, instead of
    talking about tenure, consider the situation of teachers' financial
    independence. </narrative>
</topic>
```

3.2 Data Preprocessing

To prepare data for fine-tuning the DeepCT model, we first split the premises of debates into smaller passages, so that they can be used as input to DeepCT. First, we create 3 datasets, by using different documents from the original corpus:

1. All documents available, even ones that are judged—"All documents".
2. All documents available, except those that are judged—"With pools".
3. All documents available, except those that are judged and except top-50 documents that were retrieved by participants of Touché 2021—"Without pools".

Since we split the texts into smaller passages, the number of training samples is increased (cf. Table 3.1). From each of the above-mentioned datasets, we create

[7] https://github.com/terrier-org/pyterrier

Table 3.1: Number of passages in the original dataset and after splitting into smaller passages.

Data	Number of passages
Original	387,740
All documents	831,758
With pools	819,181
Without pools	717,551

another 3 datasets, that are used for training of DeepCT model and differ in used reference field: (1) topics, (2) conclusions, (3) both topics and conclusions. To create training samples for the network we use the content-based weak-supervision strategy proposed by Dai and Callan [17]. We first remove all stop-words, using a set of stop-words that was chosen by Natural Language Toolkit (nltk) [6]—a Python toolkit, both from passages and corresponding reference field. Afterward, we apply stemming via PorterStemmer [43] implementation in nltk to all tokens from text passages and reference fields. Each training sample (cf. example of training sample 3.3) consists of a passage, its serial number, its ID (in the form "{ID from args.me}___{serial_number}"), and its ground truth labels. The latter is based on the reference field and is done according to content-based weak supervision described by Dai and Callan [17]: if there is a stem of a word from a passage and this stem also appears in stems created from the corresponding reference field, this word is then considered as an important one and will have a value of 1.0 in ground truth field of training sample (cf. Table 3.2). As Table 3.2 shows, since it is not always the case, that stems of words in passages are also intersecting with stems of words in the reference field, some of the training samples do not possess any tokens as ground truth data. According to Dai and Callan [17], such global derivation of term weights for tokens from the entire document might be seen to be not as effective, as finding local important words for each passage. However, the DeepCT model is still able to find locally important words and if a passage introduces noise in the data, the DeepCT model will assign 0 weights to tokens from these passages.

Table 3.2: Overview of empty training's samples for 9 different datasets. Column "Empty" shows number of training samples that have 0 important tokens, column "At least one" shows number of training samples that contain at least one important word. Column "Total" indicates the overall amount of training's sample in the corresponding dataset.

Data	Reference Field	Empty	At least one	Total
With pools	Conclusions	189,654	629,527	819,181
With pools	Topics	190,007	629,174	819,181
With pools	Topic and conclusions	187,041	632,140	819,181
All documents	Conclusions	191,328	640,430	831,758
All documents	Topics	191,651	640,107	831,758
All documents	Topic and conclusions	188,663	643,095	831,758
Without pools	Conclusions	173,933	543,618	717,551
Without pools	Topics	174,385	543,166	717,551
Without pools	Topic and conclusions	171,707	545,844	717,551

Table 3.3: Example of training sample created using conclusion as reference field, which is defined as "Domestic Violence Awareness should be increased".

Passage	"As I mentioned in my previous claim, raising awareness can make the perpetrator feel targeted and cause an increase in abuse. Rather than raise awareness for drugs and alcohol, it would encourage the perpetrator to take out his anger because of the people who judge him/her. Humans can be deceitful, cynical, and untrustworthy. According to the Centers for Disease Control and Prevention, "Every minute, about 20 people are physically abused by an intimate partner in the U.S.," (1). The sad truth is that when victims decide to stand up for themselves and ask for help, they cannot all be helped. Currently, there are not enough funds to help everyone and increasing awareness will not benefit victims because they will be risking their lives to receive help but not be able to receive it on time. The National Network to End Domestic Violence reported that "more than 22,000 calls were answered by local domestic violence hotlines, and on that same day, more than 9,500 requests for services were unmet due to inadequate funding or staff available to assist these survivors" (2).",
Position	1
ID	S21b181c6-A10c0da05___1
Ground truth	"violence": 1.0, "awareness": 1.0, "increase": 1.0, "increasing": 1.0, "domestic": 1.0

Chapter 4

Experiments

This chapter describes our experiments for precision-oriented argument-retrieval. The experiments are based on trying different retrieval models on args.me corpus, we transform the corpus with models deployed in TARGER, transform the corpus with pre-trained on MS-MARCO-Doc corpus DeepCT model, fine-tune DeepCT model on different variations of args.me corpus, and analyze the outputs of our approaches.

4.1 Parameter Tuning

For every variation of the dataset that we created, we compare the effectiveness of argument retrieval with four common retrieval models: BM25, BM25 with RM3, Dirichlet-smoothed LM (DLM), DLM with RM3. Parameters of retrieval models have a strong influence on the effectiveness of IR [32] and hence we tune them via grid-search and two-fold-cross-validation, which are implemented in the PyTerrier framework. The two folds that we use for cross-validation are two sets of search topics that are used respectively in Touché task 1 for the year 2020 and for the year 2021. We find the best Parameters for the following retrieval models in the following ranges, inspired by the work of Lin [33]:

- BM25, b from [0.15, 0.75] with a step size of 0.2, k_1 from [0.6, 4.4] with a step size of 0.6 and k_2 for following values: [2,5,8,10].
- DLM, the smoothing parameter μ from [0,10000] with a step size of 250.
- RM3 variant of retrieval models, number of terms M to add to the query from [4, 16] with a step size of 2, number of feedback documents N from [4, 10] with a step size of 2 and the relevance of the original query λ from [0.2, 1] with a step size of 0.2.

Table 4.1: Overview of statistics of original args.me corpus and after extracting premises and claims via models deployed in TARGER. Values are given in the number of tokens. St.D stays for standard deviation.

Data	Embeddings	Max	Min	Mean	Median	St.D
args.me	-	16,751	1	317	140	406
Combined	fastText	15,955	0	212	92	285
Essays	fastText	15,154	0	219	94	294
IBM	fastText	16,529	1	311	138	399
WebD	dependency	15,252	0	30	0	77
WebD	fastText	15,263	0	29	2	69
Essays	dependency	10,393	0	173	71	237

When using the query expansion method of RM3 we tune only the parameters of RM3 and use the default parameters of the used retrieval model, that were assigned by the creators of PyTerrier.

4.2 Transforming Data with TARGER

To make the retrieval more precision-oriented we want to identify from arguments only its premises and claims, dismissing the rest of the argument. For this purpose, we use 6 different models deployed in TARGER, which can identify premises and claims from arguments. Each model differs in the dataset that was used in the training of the model and in the pre-trained set of word embeddings. Models we use are trained on the persuasive essays (Essays) [21], web discourse (WebD) [26], IBM Debater [31] or dataset combined of three above-mentioned datasets (Combined) and use fastText [42] or dependency-based embeddings [30].

We use API provided by TARGER to use its deployed models, which expect a raw text as input and output a list of word-level tokens with labels saved in Inside-outside-beginning format for premises and claims, as well as the confidence score for each label. After extracting premises and claims from each document (cf. examples 4.1 and 4.2 of extracted premises and claims from passage in training sample 3.3) of original args.me corpus, we create 6 different corpora (overview in Table 4.1) that we use for further evaluation.

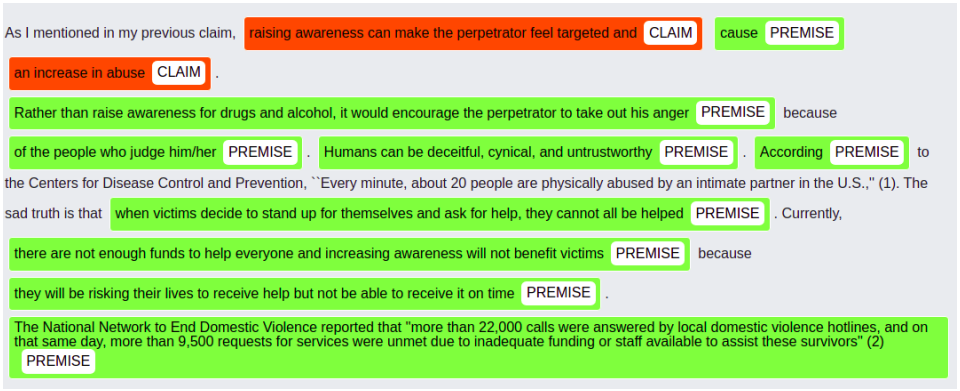


Figure 4.1: Highlighted premises and claims retrieved with web-interface of TARGER using model trained on Essays dataset and using fastText embeddings.

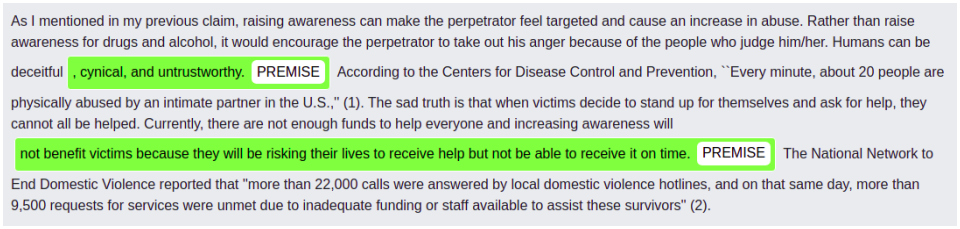


Figure 4.2: Highlighted premises and claims (none) retrieved with web-interface of TARGER using model trained on WebD dataset and using dependency embeddings.

4.3 Training and Inference of DeepCT Models

We use an official implementation of the DeepCT model, which gives easy access to training and further inference of the model.[8] Implementation is written in Python 3.0 and used an open-source framework for machine learning TensorFlow 1.15.0 [1]. For the train we set the batch size to 16, the maximum length of passage to 512 tokens, the learning rate to 0.00002, the number of epochs to 3 and use the pretrained BERT-base model [52]. As the training curves of DeepCT models on the aforementioned datasets, we created show, they can quickly adapt to the new objective and displays that can predict the semantic importance of words well, since the loss drops rapidly at first and then continue to get lower over time (cf. Table 4.3). After inference (cf. distribution of most repeated words after DeepCT inference Figure 4.3) of variations of DeepCT models on args.me corpus, we create 9 different corpora (overview in Table 4.2) that we use for further evaluation.

[8] `https://github.com/AdeDZY/DeepCT`

Table 4.2: Overview of statistics on original args.me corpus and after inference of variations of DeepCT models. St.D stays for standard deviation.

Data	Reference Field	Max	Min	Mean	Median	St.D
args.me	-	16,751	1	317	140	406
MARCO	-	1,171	0	297	294	119
With pools	Conclusions	23,802	0	510	297	613
With pools	Topics	22,723	0	511	296	614
With pools	Topics and Conclusions	22,979	0	522	323	609
All documents	Conclusions	17,656	0	529	327	621
All documents	Topics	15,815	0	508	298	616
All documents	Topics and Conclusions	16,764	0	534	334	611
Without pools	Conclusions	22,979	0	522	323	609
Without pools	Topics	23,802	0	510	297	613
Without pools	Topics and Conclusions	23,742	0	541	338	621

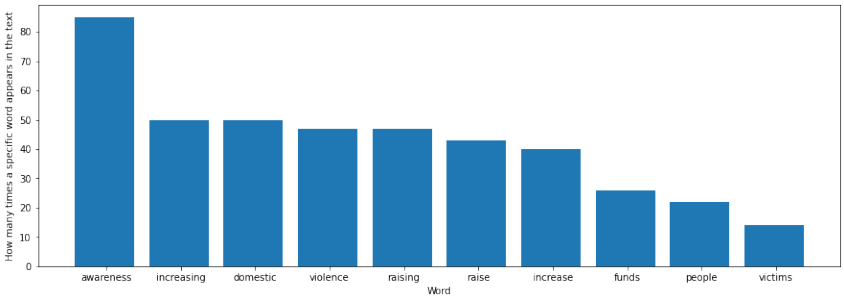
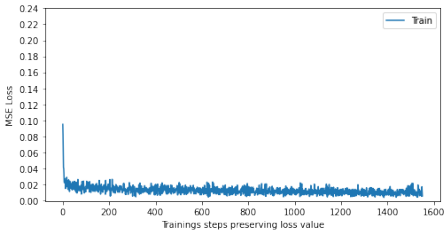
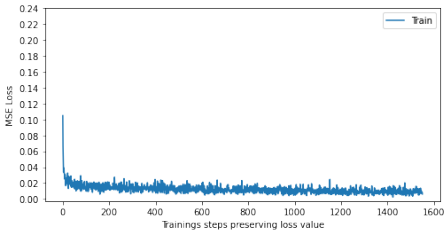


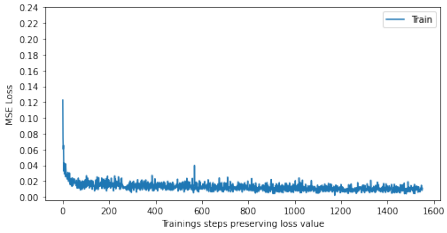
Figure 4.3: Overview of most repeated words in passage from training's sample 3.3 after inference of DeepCT model trained on all documents with conclusions as reference field.



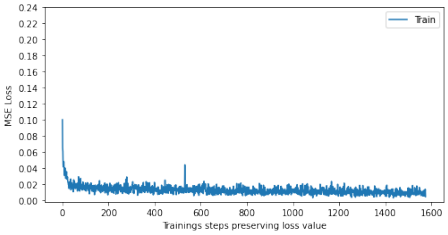
(a) Data: With pools, Reference Field: Conclusions



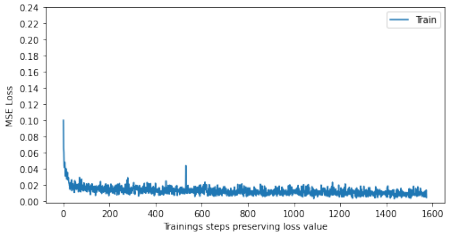
(b) Data: With pools, Reference Field: Topics



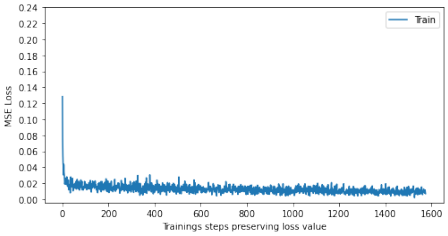
(c) Data: With pools, Reference Field: Conclusions and Topics



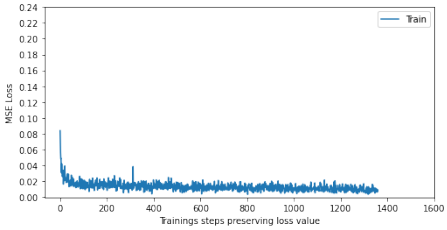
(d) Data: All documents, Reference Field: Conclusions



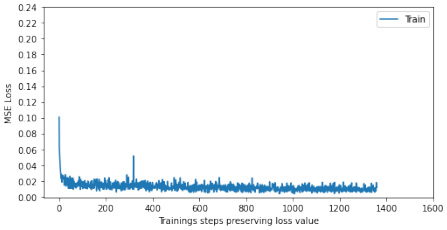
(e) Data: All documents, Reference Field: Topics



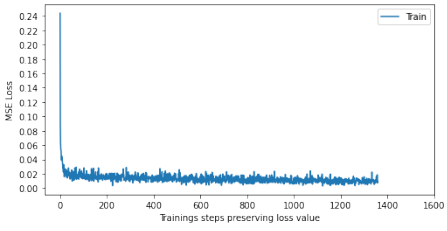
(f) Data: All documents, Reference Field: Conclusions and Topics



(g) Data: Without pools, Reference Field: Conclusions



(h) Data: Without pools, Reference Field: Topics



(i) Data: Without pools, Reference Field: Conclusions and Topics

Table 4.3: Training curves of variations of DeepCT models.

Chapter 5

Evaluation

In this chapter, we evaluate the effectiveness of argument retrieval on args.me corpus, after it was transformed via models deployed in TARGER [13] and after inference of trained DeepCT [17] models. We describe metrics we use to measure the effectiveness, such as Normalized Discounted Cumulative Gain (nDCG) and Bpref, and interpret the results of our approaches.

5.1 Evaluation Metrics

To evaluate the effectiveness of retrieval models we first have to obtain a ranking of documents and then find out if the retrieved documents are relevant to the given query. To measure the effectiveness of retrieval models we use nDCG metric [40, 56], because this is a precision-oriented metric and it takes into account the annotated relevance of the documents and their positioning in the ranking. Normalized Discounted Cumulative Gain is based on two other metrics: Cumulative Gain (CG) and Discounted Cumulative Gain (DCG).

Cumulative Gain is a sum of annotated relevance's rel of first p retrieved documents in obtained ranking:

$$\mathrm{CG_p} = \sum_{i=1}^{p} rel_i$$

However, the distribution of relevant and irrelevant documents in the ranking is not considered by CG. This motivates an idea for the DCG metric: relevance of a document in ranking is divided by the logarithm of its position, thus penalizing the score of DCG if relevant documents appear low in the ranking and motivating to create such ranking, where relevant documents will be placed first in the ranking. A common approach is to use the natural logarithm of the

position:

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\ln(i+1)}$$

Since for some queries it is easier to find relevant documents and for some it is harder, this motivates a nDCG metric, which normalizes the DCG score using an ideal discounted cumulative gain(iDCG):

$$\mathrm{IDCG_p} = \sum_{i=1}^{|REL_p|} \frac{rel_i}{\log_2(i+1)}$$

IDCG sorts all relevant documents by their relevance and hence produces the best possible DCG score, where REL_p is the sorted list of relevant documents in the corpus up to the position p. Having iDCG- and DCG-scores, we can compute the nDCG-score for the first p documents:

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

Since we do not have the complete judgments in the qrels files, we also use the Bpref metric to evaluate the effectiveness of our approaches. Bpref is used in situations when the relevance judgments are not complete: it measures the ranking based on whether judged relevant documents are placed above judged non-relevant documents,

$$\text{Bpref } = \frac{1}{R} \sum_{r} 1 - \frac{|\ n \text{ ranked higher than } r\ |}{R}$$

with R defined as a number of relevant documents to the query, r is a relevant document, and n is a judged nonrelevant document in the first R retrieved documents [10].

5.2 Retrieval Comparison

To compare the results achieved based on corpora of original args.me corpus and after its being transformed with trained DeepCT models, we compare retrieved documents for the topic “Should teachers get tenure?” using BM25 retrieval model with default parameters and using DeepCT model, which was trained with conclusions as reference field and on data without pools. Further, in text, we call the args.me corpus after it was transformed with DeepCT model as “DeepCT corpus”, ranking based on DeepCT corpus as “DeepCT ranking” and ranking based on original args.me corpus as ”original ranking”.

As we investigate the first six documents in the DeepCT ranking and the original ranking (cf. Table 5.1), we see that the rankings do not differ much and contain the same documents but in a slightly different order (cf. Table 5.2). As Table 5.2 shows, the BM25 retrieval model on the DeepCT corpus ranks 3 documents higher and 3 documents lower, when compared to the original ranking. However, the difference in positions of documents placed higher is bigger, than the difference in positions of documents placed lower. For example, while relevant documents in the DeepCT ranking get higher in 5, 1, and 4 positions respectively, documents that gets a lower position, compared to the original ranking, are only placed 1 position below. For this particular example, that means, that the BM25 retrieval model applied to the DeepCT corpus is able to give relevant documents higher positions in ranking, which makes the overall effectiveness of BM25 better. Nevertheless, for the first 25 documents in the DeepCT ranking, one relevant document is missing, but since this document is also placed in position 24 in the original ranking, this does not make a big difference in the effectiveness of this approach. There were also no relevant documents found in the DeepCT ranking that would not be in the original ranking.

We investigate the reason for such re-ranking based on the first document in the DeepCT ranking, which was placed 5 positions above, compared to the original ranking. This document is rather big and contains 1,629 tokens in it. In this document words, whose stem is "teacher" are repeated 47 times and words that stem from "tenur" are repeated 35 times. After transforming this document with the DeepCT model, words whose stem is "teacher" are repeated 941 times, and words whose stem is "tenure" are repeated 918 times. In this case, DeepCT found words, that overlap with words from the topic, as the most important ones and repeated them more, than any other words in the document. Since BM25 is a frequency-based retrieval model, such repetition of tokens in the document made this document more relevant for BM25, as it was before in args.me corpus.

Table 5.1: First 6 retrieved documents using BM25 retrieval model, used on args.me corpus after inference of DeepCT model ("DeepCT") and on original args.me corpus ("Original"). Each document is assigned with a unique identifier and its manually annotated relevance ("label"). If document has no annotated relevance, its entry is filled with "-".

DeepCT		Original	
id	**label**	**id**	**label**
Sc065954f-A6deb09b6	2	S51530f3f-Ae32a4a1b	-
Sc065954f-Ae72bc9c6	2	Sb0680508-Aa5189771	2
Sb0680508-Aa5189771	2	Sc065954f-Ae72bc9c6	2
Sc065954f-A24a16870	2	Sc065954f-A24a16870	2
S51530f3f-A4715d76f	-	Sff0947ec-A46d54897	2
Sff0947ec-A46d54897	2	Sc065954f-A6deb09b6	2

Table 5.2: Relevant documents that have changed their positions in ranking based on args.me corpus after and before inference of DeepCT model. In columns "DeepCT" and "Original" are shown the positions of documents in corresponding rankings, in column "Δ" is shown the change of the position of a document in DeepCT ranking compared to its position in Original ranking.

id	**DeepCT**	**Original**	**Δ**
Sc065954f-A6deb09b6	0	5	+5
Sc065954f-Ae72bc9c6	1	2	+1
Sc065954f-Ac3a1cfc1	13	17	+4
Sb0680508-Aa5189771	2	1	-1
Sff0947ec-A46d54897	5	4	-1
Sc065954f-A39b0539e	8	7	-1

5.3 Results

To evaluate our approaches we use the nDCG@5, nDCG@25, and Bpref measurements. We also differ between the two rankings: in the first case ("All documents") ranking is containing all documents, both judged and unjudged, in the second case ("Unjudged removed"), we remove all the unjudged documents from the ranking because that is how the participants approaches in Touché were measured and allows us to compare them with our approaches in Table 5.3. Bpref is presented only once since its value is the same for both of the aforementioned cases. For each retrieval model, we present 3 tables with results: the first table is based on topics both from the years 2020 and 2021 from Touché task 1, the second table is based only on topics from the year 2020 and the third one is based on topics from the year 2021. Each table represents achieved results on argument retrieval based on (1) original args.me corpus, (2) on args.me corpus after it was transformed models deployed in TARGER—"Targers", which differ

in data they trained on and used embeddings, (3) on args.me corpus after if was transformed with pre-trained on MS-MARCO-Doc corpus DeepCT model and models we fine-tuned on corpora that we created based on args.me corpus and described in Section 3—"DeepCT", which differ in data they were trained on and used reference field (cf. Tables from Table 5.4 to Table 5.15).

Comparing the results achieved by various retrieval models, we can see the following patterns: (1) retrieval based on original corpus after transforming it with models deployed in TARGER makes the effectiveness of retrieval models most often worse, except a few times, when it outperforms in nDCG@5 for all documents (cf. Table 5.7) and in nDCG@25 also for all documents (cf. Table 5.9). Models based on the WebD dataset tend to delete the whole or most of the text documents in args.me corpus (cf. Table 4.1), since they do not find any premises or claims in the documents and thus achieve the worst results overall. Another 4 models do not dismiss that much of a text in documents and hence achieve results close to the retrieval based on the original args.me corpus. They can be used in cases when there is a need to reduce the space required to save the corpus and the effectiveness of the model is not crucial, e.g. model trained on Combined corpus achieves on average slightly lower scores in used metrics but reduces the mean amount of tokens in documents from 317 to 212; (2) after transforming args.me corpus with DeepCT models, this approaches always achieve better results in Bpref, nDCG@5, and nDCG@25 when ranking is based only on ranked documents. For the case, when retrieval is done using all documents, about a half of DeepCT-based approaches outperform other retrieval approaches, based on original args.me corpus or sometimes based on corpus after transforming it with one of the models deployed in TARGER; (3) all of the approaches based on fine-tuning DeepCT models outperform in all metrics results achieved by pre-trained on MS-MARCO-Doc corpus DeepCT model; (4) best results in nDCG@5 and in nDCG@25 based on only judged documents are retrieved via DLM or Dirichlet-smoothed Language Model with RM3 retrieval approaches, while best Bpref is achieved by BM25 with RM3 retrieval models (cf. Table 5.3). In Table 5.3 we choose the model with the best performance based on the score achieved in the nDCG@5 measurement based on only judged documents since it is how the approaches of participants of Touché were judged. In the following tables we name MS-MARCO-Doc corpus as "MARCO".

Table 5.3: Overview of approaches that achieved best results in nDCG@5-measurement for each retrieval model and for each year. Column "Transformed with" shows a model which was used to transform original args.me corpus. Occasionally, the best results were achieved only after transformation with DeepCT models. For the years 2020 and 2021 for comparison, we provide the best-achieved result from Touché participants. Column "Ref. Table" stays for Reference Table and references to all results of the corresponding approach.

	Retrieval Model	Transformed with	Unjudged removed	All documents		Ref. Table
			nDCG@5	nDCG@5	Bpref	
2020	BM25	DeepCT: With pools + Topics & Concl.	0.79	0.42	0.71	5.8
	BM25 with RM3	DeepCT: With pools + Topics & Concl.	0.82	0.40	**0.77**	5.11
	DLM	DeepCT: With pools + Topics & Concl.	0.79	**0.45**	0.68	5.5
	DLM with RM3	DeepCT: All documents + Conclusions.	**0.86**	0.42	0.71	5.14
	Best Touché	-	0.80	-	-	-
2021	BM25	DeepCT: Without pools + Topics & Concl.	0.72	0.61	0.74	5.9
	BM25 with RM3	DeepCT: With pools + Topics & Concl.	0.71	0.53	**0.74**	5.12
	DLM	DeepCT: All documents + Topics & Concl.	**0.73**	**0.61**	0.72	5.6
	DLM with RM3	DeepCT: All documents + Conclusions.	0.68	0.51	0.73	5.15
	Best Touché	-	0.72	-	-	-
2020+2021	BM25	DeepCT: All documents + Topics & Concl.	0.73	0.49	0.72	5.7
	BM25 with RM3	DeepCT: With pools + Topics & Concl.	0.73	0.47	**0.76**	5.10
	DLM	DeepCT: With pools + Topics & Concl.	**0.74**	**0.52**	0.70	5.4
	DLM with RM3	DeepCT: Without pools + Conclusions.	0.74	0.45	0.71	5.13

Table 5.4: Effectiveness of the tuned Dirichlet-smoothed Language Model approach on topics of years 2020 and 2021, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	0.57	**0.47**	0.59	0.70	0.64
	Data	**Embeddings**					
Targers	WebD	dependency	0.35	0.26	0.41	0.58	0.48
	Combined	fastText	0.54	0.42	0.58	0.69	0.62
	Essays	dependency	0.52	0.40	0.57	0.69	0.62
	Essays	fastText	0.52	0.42	0.59	0.69	0.64
	IBM	fastText	**0.57**	0.46	0.59	0.70	0.64
	WebD	fastText	0.33	0.24	0.40	0.59	0.47
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.35	0.31	0.61	0.59	0.61
	With pools	Conclusions	0.53	0.44	0.70	0.72	0.70
	With pools	Topics	0.52	0.44	0.69	0.73	0.70
	With pools	Topic & Concl.	0.52	0.44	**0.70**	**0.74**	**0.71**
	All documents	Conclusions	0.54	0.45	0.70	0.74	0.71
	All documents	Topics	0.53	0.45	0.70	0.73	0.71
	All documents	Topic & Concl.	0.52	0.45	0.70	0.74	0.71
	Without pools	conclusions	0.52	0.44	0.70	0.72	0.70
	Without pools	Topics	0.50	0.43	0.69	0.71	0.70
	Without pools	Topic & Concl.	0.54	0.44	0.70	0.73	0.71

Table 5.5: Effectiveness of the tuned Dirichlet-smoothed Language Model approach on topics of the year 2020, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		All documents			Unjudged removed	
			nDCG@5	nDCG@25	Bpref	nDCG@5	nDCG@25
Original		args.me	**0.50**	**0.38**	0.57	0.77	0.68
	Data	**Embeddings**					
Targers	WebD	dependency	0.34	0.26	0.46	0.69	0.55
	Combined	fastText	0.43	0.34	0.56	0.75	0.65
	Essays	dependency	0.43	0.33	0.55	0.77	0.66
	Essays	fastText	0.44	0.36	0.59	0.78	0.69
	IBM	fastText	0.50	0.38	0.57	0.77	0.68
	WebD	fastText	0.27	0.21	0.43	0.69	0.53
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.25	0.25	0.60	0.61	0.63
	With pools	Conclusions	0.45	0.37	**0.68**	0.78	0.74
	With pools	Topics	0.45	0.36	0.66	0.78	0.73
	With pools	Topic & Concl.	0.45	0.37	0.68	**0.79**	0.74
	All documents	Conclusions	0.46	0.37	0.67	0.79	0.74
	All documents	Topics	0.45	0.37	0.68	0.79	0.74
	All documents	Topic & Concl.	0.43	0.36	0.67	0.79	0.74
	Without pools	Conclusions	0.41	0.35	0.68	0.78	**0.74**
	Without pools	Topics	0.40	0.36	0.68	0.77	0.74
	Without pools	Topic & Concl.	0.44	0.36	0.68	0.78	0.74

Table 5.6: Effectiveness of the tuned Dirichlet-smoothed Language Model approach on topics of the year 2021, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	0.63	**0.55**	0.62	0.66	0.66
	Data	**Embeddings**					
Targers	WebD	dependency	0.35	0.26	0.36	0.53	0.45
	Combined	fastText	0.63	0.51	0.60	0.66	0.64
	Essays	dependency	0.60	0.47	0.58	0.65	0.63
	Essays	fastText	0.60	0.48	0.59	0.63	0.64
	IBM	fastText	**0.64**	0.54	0.61	0.66	0.65
	WebD	fastText	0.39	0.27	0.37	0.57	0.47
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.45	0.37	0.62	0.60	0.64
	With pools	Conclusions	0.60	0.51	0.72	0.71	0.71
	With pools	Topics	0.60	0.52	0.71	0.72	0.72
	With pools	Topic & Concl.	0.60	0.52	**0.72**	0.72	0.73
	All documents	Conclusions	0.61	0.52	0.72	0.72	0.73
	All documents	Topics	0.61	0.53	0.72	0.72	0.73
	All documents	Topic & Concl.	0.61	0.53	0.72	**0.73**	**0.74**
	Without pools	Conclusions	0.62	0.52	0.72	0.70	0.72
	Without pools	Topics	0.61	0.51	0.71	0.70	0.71
	Without pools	Topic & Concl.	0.64	0.53	0.72	0.72	0.73

Table 5.7: Effectiveness of the tuned BM25 approach on topics of years 2020 and 2021, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	0.52	**0.45**	0.63	0.70	0.67
	Data	**Embeddings**					
Targers	WebD	dependency	0.31	0.25	0.42	0.61	0.51
	Combined	fastText	0.52	0.43	0.60	0.71	0.66
	Essays	dependency	**0.54**	0.41	0.57	0.70	0.63
	Essays	fastText	0.51	0.42	0.59	0.69	0.65
	IBM	fastText	0.52	0.44	0.63	0.70	0.67
	WebD	fastText	0.31	0.24	0.41	0.59	0.49
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.33	0.30	0.62	0.56	0.60
	With pools	Conclusions	0.49	0.44	0.72	0.72	0.71
	With pools	Topics	0.50	0.44	0.71	0.73	0.72
	With pools	Topic & Concl.	0.51	0.44	**0.72**	0.73	0.71
	All documents	Conclusions	0.51	0.44	0.72	0.73	0.72
	All documents	Topics	0.50	0.43	0.72	0.72	0.71
	All documents	Topic & Concl.	0.49	0.44	0.72	**0.73**	**0.72**
	Without pools	Conclusions	0.49	0.43	0.72	0.72	0.71
	Without pools	Topics	0.50	0.44	0.72	0.72	0.71
	Without pools	Topic & Concl.	0.51	0.44	0.72	0.73	0.72

Table 5.8: Effectiveness of the tuned BM25 approach on topics of the year 2020, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	0.38	0.35	0.64	0.76	0.71
	Data	**Embeddings**					
Targers	WebD	dependency	0.26	0.22	0.47	0.72	0.58
	Combined	fastText	0.39	0.34	0.61	0.77	0.69
	Essays	dependency	0.45	0.34	0.55	0.77	0.67
	Essays	fastText	0.40	0.34	0.59	0.77	0.70
	IBM	fastText	0.37	0.34	0.64	0.77	0.71
	WebD	fastText	0.24	0.19	0.44	0.67	0.54
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.26	0.26	0.62	0.61	0.64
	With pools	Conclusions	0.38	0.35	0.70	0.78	0.75
	With pools	Topics	0.42	**0.36**	0.70	0.78	0.75
	With pools	Topic & Concl.	0.42	0.36	**0.71**	**0.79**	**0.76**
	All documents	Conclusions	0.41	0.36	0.70	0.79	0.75
	All documents	Topics	**0.43**	0.35	0.70	0.77	0.75
	All documents	Topic & Concl.	0.39	0.36	0.70	0.78	0.75
	Without pools	Conclusions	0.38	0.35	0.70	0.78	0.75
	Without pools	Topics	0.40	0.35	0.70	0.77	0.75
	Without pools	Topic & Concl.	0.40	0.35	0.70	0.78	0.75

Table 5.9: Effectiveness of the tuned BM25 approach on topics of the year 2021, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		All documents			Unjudged removed	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	**0.66**	0.55	0.62	0.67	0.66
	Data	**Embeddings**					
Targers	WebD	dependency	0.37	0.28	0.37	0.56	0.50
	Combined	fastText	0.65	0.51	0.60	0.68	0.66
	Essays	dependency	0.62	0.48	0.58	0.67	0.65
	Essays	fastText	0.61	0.49	0.59	0.66	0.65
	IBM	fastText	0.66	**0.55**	0.62	0.68	0.66
	WebD	fastText	0.38	0.28	0.37	0.58	0.51
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.40	0.35	0.62	0.53	0.61
	With pools	Conclusions	0.58	0.52	0.73	0.69	0.72
	With pools	Topics	0.58	0.51	0.73	0.71	0.73
	With pools	Topic & Concl.	0.60	0.51	0.73	0.71	0.73
	All documents	Conclusions	0.61	0.52	0.73	0.71	0.73
	All documents	Topics	0.58	0.50	0.73	0.70	0.72
	All documents	Topic & Concl.	0.59	0.52	0.73	0.71	0.73
	Without pools	Conclusions	0.59	0.52	0.73	0.71	0.72
	Without pools	Topics	0.61	0.52	0.73	0.71	0.72
	Without pools	Topic & Concl.	0.61	0.53	**0.74**	**0.72**	**0.73**

Table 5.10: Effectiveness of the tuned BM25 with RM3 approach on topics of years 2020 and 2021, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	**0.46**	0.41	0.68	0.71	0.69
	Data	**Embeddings**					
Targers	WebD	dependency	0.16	0.14	0.40	0.52	0.47
	Combined	fastText	0.38	0.36	0.64	0.63	0.65
	Essays	dependency	0.36	0.33	0.61	0.66	0.64
	Essays	fastText	0.41	0.36	0.64	0.67	0.66
	IBM	fastText	0.45	0.41	0.68	0.70	0.68
	WebD	fastText	0.17	0.15	0.40	0.55	0.47
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.34	0.31	0.65	0.57	0.62
	With pools	Conclusions	0.46	**0.41**	0.76	0.72	0.73
	With pools	Topics	0.44	0.40	0.75	0.72	0.73
	With pools	Topic & Concl.	0.47	0.42	**0.76**	**0.76**	**0.74**
	All documents	Conclusions	0.45	0.40	0.75	0.73	0.73
	All documents	Topics	0.42	0.39	0.75	0.71	0.73
	All documents	Topic & Concl.	0.42	0.39	0.74	0.72	0.73
	Without pools	Conclusions	0.43	0.41	0.76	0.71	0.72
	Without pools	Topics	0.44	0.41	0.76	0.72	0.73
	Without pools	Topic & Concl.	0.42	0.40	0.76	0.71	0.73

Table 5.11: Effectiveness of the tuned BM25 with RM3 approach on topics of the year 2020, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	0.36	0.35	0.71	0.77	0.73
	Data	**Embeddings**					
Targers	WebD	dependency	0.14	0.13	0.45	0.55	0.49
	Combined	fastText	0.28	0.30	0.66	0.70	0.68
	Essays	dependency	0.27	0.28	0.63	0.72	0.67
	Essays	fastText	0.32	0.31	0.66	0.74	0.70
	IBM	fastText	0.34	0.35	0.71	0.76	0.72
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.26	0.26	0.67	0.62	0.67
	With pools	Conclusions	0.38	0.36	0.77	0.80	0.81
	With pools	Topics	0.39	0.36	0.77	0.81	0.81
	With pools	Topic & Concl.	**0.40**	**0.37**	**0.77**	**0.82**	**0.81**
	All documents	Conclusions	0.37	0.35	0.76	0.81	0.81
	All documents	Topics	0.36	0.34	0.76	0.78	0.80
	All documents	Topic & Concl.	0.36	0.34	0.76	0.80	0.81
	Without pools	Conclusions	0.38	0.36	0.77	0.80	0.81
	Without pools	Topics	0.37	0.35	0.76	0.79	0.81
	Without pools	Topic & Concl.	0.36	0.35	0.77	0.79	0.81

Table 5.12: Effectiveness of the tuned BM25 with RM3 approach on topics of the year 2021, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	**0.56**	**0.47**	0.65	0.65	0.65
	Data	**Embeddings**					
Targers	WebD	dependency	0.19	0.15	0.35	0.51	0.47
	Combined	fastText	0.47	0.41	0.63	0.57	0.62
	Essays	dependency	0.45	0.39	0.60	0.59	0.63
	Essays	fastText	0.50	0.42	0.62	0.60	0.63
	IBM	fastText	0.55	0.47	0.65	0.64	0.65
	WebD	fastText	0.22	0.18	0.36	0.57	0.49
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.43	0.36	0.63	0.56	0.62
	With pools	Conclusions	0.53	0.47	0.75	0.67	0.70
	With pools	Topics	0.49	0.45	0.74	0.67	0.71
	With pools	Topic & Concl.	0.53	0.47	0.74	**0.71**	0.72
	All documents	Conclusions	0.52	0.46	0.74	0.68	0.71
	All documents	Topics	0.48	0.45	0.75	0.67	0.72
	All documents	Topic & Concl.	0.48	0.44	0.73	0.68	0.70
	Without pools	Conclusions	0.49	0.46	0.74	0.67	0.71
	Without pools	Topics	0.51	0.46	**0.75**	0.70	**0.72**
	Without pools	Topic & Concl.	0.48	0.45	0.75	0.67	0.70

Table 5.13: Effectiveness of the Dirichlet-smoothed Language Model with RM3 approach on topics of years 2020 and 2021, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	0.50	0.36	0.57	0.67	0.60
	Data	**Embeddings**					
Targers	WebD	dependency	0.32	0.23	0.42	0.55	0.45
	Combined	fastText	0.47	0.35	0.56	0.65	0.59
	Essays	dependency	0.42	0.31	0.53	0.63	0.55
	Essays	fastText	0.48	0.37	0.56	0.66	0.60
	IBM	fastText	0.50	0.38	0.58	0.68	0.61
	WebD	fastText	0.29	0.22	0.39	0.54	0.43
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.32	0.28	0.63	0.63	0.63
	With pools	Conclusions	0.44	0.36	0.71	0.67	0.60
	With pools	Topics	0.48	0.39	0.72	0.72	**0.71**
	With pools	Topic & Concl.	0.42	0.36	0.70	0.71	0.70
	All documents	Conclusions	0.44	0.37	0.70	0.72	0.69
	All documents	Topics	0.46	0.37	0.72	0.72	0.71
	All documents	Topic & Concl.	**0.50**	**0.40**	**0.74**	0.71	0.68
	Without pools	Conclusions	0.45	0.38	0.71	**0.74**	0.69
	Without pools	Topics	0.46	0.39	0.72	0.73	0.68
	Without pools	Topic & Concl.	0.46	0.38	0.71	0.72	0.70

Table 5.14: Effectiveness of the Dirichlet-smoothed Language Model with RM3 approach on topics of the year 2020, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	0.45	0.35	0.58	0.78	0.69
	Data	**Embeddings**					
Targers	WebD	dependency	0.35	0.25	0.48	0.68	0.55
	Combined	fastText	0.41	0.30	0.54	0.75	0.65
	Essays	dependency	0.37	0.29	0.53	0.73	0.62
	Essays	fastText	0.45	0.35	0.58	0.79	0.69
	IBM	fastText	0.45	0.35	0.58	0.78	0.69
	WebD	fastText	0.27	0.21	0.44	0.66	0.50
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.25	0.25	0.64	0.66	0.67
	With pools	Conclusions	0.39	0.35	0.72	0.84	0.77
	With pools	Topics	0.45	0.37	0.74	0.83	0.79
	With pools	Topic & Concl.	0.41	0.34	0.69	0.84	0.76
	All documents	Conclusions	0.42	0.35	0.71	**0.86**	0.77
	All documents	Topics	0.46	0.38	0.75	0.84	0.80
	All documents	Topic & Concl.	**0.49**	**0.38**	**0.75**	0.85	**0.81**
	Without pools	Conclusions	0.39	0.35	0.70	0.82	0.76
	Without pools	Topics	0.41	0.36	0.74	0.85	0.81
	Without pools	Topic & Concl.	0.38	0.34	0.71	0.85	0.78

Table 5.15: Effectiveness of the Dirichlet-smoothed Language Model with RM3 approach on topics of the year 2021, nDCG- and Bpref-scores are calculated using all documents in the ranked lists and after removing documents without relevance judgments.

	Corpus		**All documents**			**Unjudged removed**	
			nDCG@5	**nDCG@25**	**Bpref**	**nDCG@5**	**nDCG@25**
Original		args.me	**0.54**	0.38	0.56	0.62	0.59
	Data	**Embeddings**					
Targers	WebD	dependency	0.29	0.21	0.35	0.51	0.44
	Combined	fastText	0.53	0.40	0.57	0.63	0.61
	Essays	dependency	0.46	0.34	0.53	0.59	0.58
	Essays	fastText	0.51	0.37	0.55	0.61	0.59
	IBM	fastText	0.54	0.41	0.57	0.63	0.61
	WebD	fastText	0.31	0.22	0.35	0.56	0.47
	Trained on data	**Reference field**					
DeepCT	MARCO	Title	0.39	0.32	0.62	0.63	0.64
	With pools	Conclusions	0.49	0.38	0.71	0.68	0.68
	With pools	Topics	0.51	0.42	0.71	0.66	0.68
	With pools	Topic & Concl.	0.43	0.37	0.71	0.66	**0.68**
	All documents	Conclusions	0.47	0.38	0.69	0.68	0.68
	All documents	Topics	0.45	0.37	0.70	0.67	0.68
	All documents	Topic & Concl.	0.51	0.41	**0.73**	**0.68**	0.68
	Without pools	Conclusions	0.50	0.42	0.72	0.66	0.68
	Without pools	Topics	0.52	0.42	0.70	0.66	0.67
	Without pools	Topic & Concl.	0.53	**0.42**	0.71	0.66	0.68

Chapter 6

Conclusions

In this thesis, we contribute various approaches to the task of argument retrieval, while we aim to increase the precision of retrieved documents, by taking into account the semantic importance of the words in documents. For this purpose, we used a pre-trained on MS-MARCO-Doc corpus DeepCT model, as well as fine-tuned DeepCT model on nine different corpora, that we created based on args.me corpus and we also extracted premises and claims via models deployed in TARGER, which we further used for experiments and evaluation.

Using fine-tuned DeepCT models we achieved better results in the nDCG@5 metric on ranking based only on judged documents, compared to the most effective participants approach in Touché task 1 in the year 2020 by 0.04, achieving an nDCG@5-score of 0.86 and outperforming the most effective participants approach in the year 2021 by 0.01, achieving the nDCG@5-score of 0.73. Fine-tuning of DeepCT model on args.me corpus has also shown a high influence on the effectiveness of retrieval in Touché, since our fine-tuned models achieved much better results, compared to the pre-trained model on MS-MARCO-Doc corpus. Approaches based on models deployed in TARGER have shown little to no improvements compared to the results of according retrieval models used on original args.me corpus, but we believe they might be used for the reduction of required space to save the corpora when the effectiveness is not crucial.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] Yamen Ajjour, Henning Wachsmuth, Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. Data acquisition for argument search: The args. me corpus. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 48–59. Springer, 2019.

[3] Giambattista Amati. Frequentist and bayesian approach to information retrieval. In *European Conference on Information Retrieval*, pages 13–24. Springer, 2006.

[4] Jesse Anderton. Lecture notes in cs6200: Information retrieval, 2020. URL `https://www.ccs.neu.edu/home/vip/teach/IRcourse/1_retrieval_models/slides/language_models.pdf`.

[5] Negar Arabzadeh, Xinyi Yan, and Charles LA Clarke. Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2862–2866, 2021.

[6] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[7] Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. Overview of Touché 2020: Argument Retrieval. In Avi Arampatzis, Evangelos Kanoulas, Theodora Tsikrika, Stefanos Vrochidis, Hideo Joho, Christina Lioma, Carsten Eickhoff, Aurélie Névéol, Linda Cappellato, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 11th International Conference of the CLEF Association (CLEF 2020)*, volume 12260 of *Lecture Notes in Computer Science*, pages 384–395, Berlin Heidelberg New York, September 2020. Springer. doi: 10.1007/978-3-030-58219-7_26. URL `https://link.springer.com/chapter/10.1007/978-3-030-58219-7_26`.

[8] Alexander Bondarenko, Lukas Gienapp, Maik Fröbe, Meriem Beloucif, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. Overview of Touché 2021: Argument Retrieval. In K. Selçuk Candan, Bogdan Ionescu, Lorraine Goeuriot, Henning Müller, Alexis Joly, Maria Maistro, Florina Piroi, Guglielmo Faggioli, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 12th International Conference of the CLEF Association (CLEF 2021)*, volume 12880 of *Lecture Notes in Computer Science*, pages 450–467, Berlin Heidelberg New York, September 2021. Springer. doi: 10.1007/978-3-030-85251-1_28. URL `https://link.springer.com/chapter/10.1007/978-3-030-85251-1_28`.

[9] Alexander Bondarenko, Maik Fröbe, Johannes Kiesel, Shahbaz Syed, Timon Gurcke, Meriem Beloucif, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. Overview of Touché 2022: Argument Retrieval. In Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty, editors, *Advances in Information Retrieval. 44th European Conference on IR Research (ECIR 2022)*, Lecture Notes in Computer Science, Berlin Heidelberg New York, April 2022. Springer.

[10] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR*, 2004.

[11] Maximilian Bundesmann, Lukas Christ, and Matthias Richter. Creating an argument search engine for online debates. In *CLEF (Working Notes)*, 2020.

[12] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *Acm Computing Surveys (CSUR)*, 44 (1):1–50, 2012.

[13] Artem Chernodub, Oleksiy Oliynyk, Philipp Heidenreich, Alexander Bondarenko, Matthias Hagen, Chris Biemann, and Alexander Panchenko. Targer: Neural argument mining at your fingertips. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 195–200, 2019.

[14] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[15] Hinrich Schütze Christopher D. Manning, Prabhakar Raghavan. Introduction to information retrieval. URL `https://nlp.stanford.edu/IR-book/`. [Online; Accessed on December 8, 2021].

[16] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading, 2010.

[17] Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*, pages 1897–1907, 2020.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] Lorik Dumani and Ralf Schenkel. 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*, pages 957–960, 2019.

[20] Lorik Dumani and Ralf Schenkel. Ranking arguments by combining claim similarity and argument quality dimensions. In *CLEF (Working Notes)*, 2020.

[21] Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1002. URL `https://aclanthology.org/P17-1002`.

[22] Saeed Entezari and Michael Völske. Argument retrieval using deep neural ranking models. In *CLEF (Working Notes)*, 2020.

[23] Allyson Ettinger. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48, 2020.

[24] Maik Fröbe, Sebastian Günther, Alexander Bondarenko, Johannes Huck, and Matthias Hagen. Using keyqueries to reduce misinformation in health-related search results. 2022.

[25] Tommaso Green, Luca Moroldo, and Alberto Valente. Exploring bert synonyms and quality prediction for argument retrieval. 2021.

[26] Ivan Habernal and Iryna Gurevych. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179, April 2017. doi: 10.1162/COLI_a_00276. URL `https://aclanthology.org/J17-1004`.

[27] Suzana Ilić, Edison Marrese-Taylor, Jorge A Balazs, and Yutaka Matsuo. Deep contextualized word representations for detecting sarcasm and irony. *arXiv preprint arXiv:1809.09795*, 2018.

[28] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[29] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[30] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-2050. URL `https://aclanthology.org/P14-2050`.

[31] Ran Levy, Ben Bogin, Shai Gretz, Ranit Aharonov, and Noam Slonim. Towards an argumentative content search engine using weak supervision. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2066–2081, 2018.

[32] Dan Li and Evangelos Kanoulas. Bayesian optimization for optimizing retrieval systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 360–368, 2018.

[33] Jimmy Lin. The neural hype and comparisons against weak baselines. In *ACM SIGIR Forum*, volume 52, pages 40–51. ACM New York, NY, USA, 2019.

[34] Marco Lippi and Paolo Torroni. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):1–25, 2016.

[35] Marco Lippi and Paolo Torroni. Margot: A web server for argumentation mining. *Expert Systems with Applications*, 65:292–303, 2016.

[36] Thi Kim Hanh Luu and Jan-Niklas Weder. Argument retrieval for comparative questions based on independent features. 2021.

[37] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. Simplified data wrangling with ir_datasets. In *SIGIR*, 2021.

[38] Craig Macdonald and Nicola Tonellotto. Declarative experimentation ininformation retrieval using pyterrier. In *Proceedings of ICTIR 2020*, 2020.

[39] Alina Mailach, Denise Arnold, Stefan Eysoldt, and Simon Kleine. Exploring document expansion for argument retrieval. *Working Notes of CLEF*, 2021.

[40] Vijay Mhaskar. Measuring search relevance using ndcg. `https://blog.thedigitalgroup.com/measuring-search-relevance-using-ndcg`, 2015. [Online; accessed 25-April-2022].

[41] Timothee Mickus, Denis Paperno, Mathieu Constant, and Kees Van Deemter. What do you mean, bert? assessing bert as a distributional semantics model. *arXiv preprint arXiv:1911.05758*, 2019.

[42] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

[43] Martin F Porter. An algorithm for suffix stripping. *Program*, 1980.

[44] Martin Potthast, Lukas Gienapp, Florian Euchner, Nick Heilenkötter, Nico Weidmann, Henning Wachsmuth, Benno Stein, and Matthias Hagen. Argument search: assessing argument relevance. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1117–1120, 2019.

[45] I Qunis, G Amati, V Plachouras, B He, C Macdonald, and C Lioma. A high performance and scalable information retrieval plateform. In *SIGR workshop on open source information retrieval*, 2006.

[46] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[47] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[48] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[49] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[50] Christian Stab, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna 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*, pages 21–25, 2018.

[51] Chris Stahlhut. Searching arguments in german with argumentext. In *DESIRES*, page 104, 2018.

[52] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*, 2019.

[53] Rekha Vaidyanathan, Sujoy Das, and Namita Srivastava. A study on retrieval models and query expansion using prf. *International Journal of Scientific & Engineering Research*, 6(2):13–18, 2015.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[55] Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59, 2017.

[56] Wikipedia. Discounted cumulative gain — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Discounted%20cumulative%20gain&oldid=1069780045`, 2022. [Online; accessed 25-April-2022].

[57] Wikipedia. Word embedding — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Word%20embedding&oldid=1084612624`, 2022. [Online; accessed 01-May-2022].

[58] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, volume 51, pages 268–276. ACM New York, NY, USA, 2017.