Touché - Task 1 - Team Korg: Finding pairs of argumentative sentences using embeddings

Notebook for the Touché Lab on Argument Retrieval at CLEF 2022

Cuong Vo Ta[1], Florian Reiner[1], Immanuel von Detten[1] and Fabian Stöhr[1]

[1]*Leipzig University, Augustusplatz 10 Leipzig 04109 Germany* https://www.uni-leipzig.de/
Text Mining and Retrieval Group TEMIR, Leipzig University, Leipzig Germany https://temir.org/

Abstract
This notebook outlines the experiments and results for Task 1 of the *Touché Lab on Argument Retrieval at CLEF 2022* by Team Korg. ElasticSearch serves as our baseline to index the args.me corpus, extended by preprocessing steps of filtering, stemming, a custom stop-word list and WordNet-based synonyms to enrich documents. We approach the problem of finding coherent pairs of argumentative sentences by using and comparing two embedding methods, namely Doc2Vec and Sentence BERT for semantic search. In our first iteration, we use a custom-trained model for Doc2Vec and the out-of-the-box functionality of SBERT for semantic search. To refine the retrieval of meaningful sentence pairs, we incorporate the text generation functionality of GPT-2 to generate prompts as an input for the sentence embeddings. After evaluating those approaches with the Normalized Discounted Cumulative Gain and using an annotated dataset of Touché 2021, we identify Doc2Vec without text generation and a revised algorithm to match sentence pairs as our best performing approach for retrieval of argumentative sentences.

Keywords
Information retrieval, Argument retrieval, Touché Task 1, CLEF 2022, Semantic search, Doc2Vec, Sentence BERT, GPT-2, Text generation, Transformer,

1. Introduction

Today, an ever-increasing pace of news coverage can be observed, while opinions regarding controversial topics are becoming more polarized. It is important to be exposed to different views in order to form your own opinion. While it is possible to search the World Wide Web for virtually every topic, it can be quite difficult to find arguments with relevant sources, which portray different point of views. Understanding the reasoning behind two opposing points of views can be facilitated by information retrieval systems. Wachsmuth et al. [1] built an argument search engine which relies on a openly accessible index of nearly 300k web scraped arguments from different debate portals. The Touché Lab revolves around information retrieval for the arguments of the args.me corpus [2]. This years Touché Task 1 [3], held at

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
cv86jahu@studserv.uni-leipzig.de (C. Vo Ta); mai12mju@studserv.uni-leipzig.de (F. Reiner); i.vondetten@gmail.com (I. von Detten); fs72popy@studserv.uni-leipzig.de (F. Stöhr)
0000-0003-4528-5591 (I. von Detten)

the annual CLEF conference [4], is about retrieving a pro and a con argument from the corpus and forming strong argumentative sentence pairs for each of them. Our team name is "Korg".

In this paper we outline the pipeline for our argument retrieval system and the two different models of sentence embeddings we use to fulfill the task. Our argument retrieval model is based on the outcome of the last year's Touché 2021 [5], more precisely on the results of Team Elrond. Following their approach, we use ElasticSearch for our index and retrieval with the DirichletLM similarity, described in Section 3.1 [6]. In order to find a pair of fitting sentences for the retrieved result, we compare two semantic search approaches. In Section 4.2 we identify Doc2Vec [7] and Sentence-BERT (SBERT) [8] as promising models using sentence embedding. We find a relevant sentence and match it with another sentence to form a coherent pair by embedding the sentences and computing the cosine similarity of the two sentences.

Sentence embeddings are designed to find sentences with the highest similarity, therefore we have to counteract the problem of retrieving sentences with identical meaning. Therefore we develop a second approach in Section 4.3 with the text generation function from GPT-2 [9]. In this approach, we take the first retrieved sentence and give it to GPT-2 as an opening text (called prompt). Then the model generates a subsequent text output which gets passed to Doc2Vec and SBERT. With that, these models find a similar sentence in their embeddings to the generated output and form a sentence pair with the first sentence. In Section 5 we evaluate our different experiments and decide on which pipeline we use for our submission for this years Touché lab.

2. Related Work

The system described in this paper is intended to run on the TIRA platform [10, 11]. TIRA is an online platform where researches can upload and evaluate their proposed solutions for shared tasks. Shared tasks are an increasingly popular form of solving open problems in academia. Shared tasks are often conducted in the scope of conferences, especially in the field of natural language processing or machine learning[1]. The organizers usually contribute large datasets which can be used by the teams of researchers to tackle diverse problems. For the shared task of the Touché Lab, TIRA ensures that all solutions are running with the same data and produce reproducible results.

With web search engines it can be harder to get an overview about different opinions on a topic than to get the answer for a clearly stated question [12]. The field of computational argumentation deals with this and other problems by mining arguments from unstructured text and computational representation of arguments. It is essential for improving search results involving arguments [13] [14]. There are also approaches to create a *World Wide Argument Web* based on structured arguments in an *Argument Interchange Format* (AIF) [15]. In this paper we focus on the results of previous teams participating in the Touché Lab, especially of Task 1 of last years shared task [5]. The task was to retrieve arguments for "controversial questions from a focused collection of debates to support opinion formation on topics of social importance" [5].

[1]for examples of shared tasks on international conferences, see https://www.statmt.org/wmt21/, https://pan.webis.de/clef19/pan19-web/ and https://alt.qcri.org/semeval2019/index.php?id=tasks

This years task is extending on this and asks the participating teams to additionally retrieve a pair of argumentative sentences for positive and negative stances of an argument. Solutions from Touché 2021 covered a wide range of approaches and performance for the task. To retrieve arguments, the retrieval model based on DirichletLM proved to perform better than other models [16]. Multiple teams worked on evaluating the impact of preprocessing by means of query expansion, word stemming and WordNet-based synonyms. The approaches involved different teams choosing semantic search as a method to retrieve relevant results. Preprocessing proved to be a important step to increase relevance and quality of the search results. Semantic search delivered mixed results, which were influenced by the choice of model and training data.

To compare sentences from arguments for the sentence pair retrieval two technologies are used in this paper: Sentence BERT (SBERT) [8] and Doc2Vec [7]. SBERT is a modification of the BERT network, a neural network designed for natural language processing (NLP) tasks like language modeling and next sentence prediction [17]. SBERT has a better performance on sentence-pair regression tasks and can for example be used to find similar sentence pairs in a huge collection of sentences. Doc2Vec is based on Word2Vec, a neural network with word embeddings in vector space, where vectors are representing words. The embedding is designed to solve problems like calculating the similarity of words. This is usually done by measuring the cosine similarity of the respective vectors [18, 19]. Doc2Vec extends the vector representation from words to documents of arbitrary size.

The generative pre-trained transformer (GPT-2) is a language model developed by OpenAI [2] which uses a deep neural network to perform several text-related tasks like translation or text summarization [9]. In this paper GPT-2 is used to generate follow-up sentences based on sentences from arguments.

3. Methodological Approach

In the following section, the methods we use will be described briefly. To provide context, we introduce our pipeline and present the concept of semantic search and sentence embeddings. We rely on these concepts to match sentences which should represent an argument in its entirety and should form a coherent pair at the same time. Then, we will cover the technologies we use in detail: ElasticSearch, Doc2Vec, SBERT and GPT-2.

3.1. Indexing

We use ElasticSearch to create an index of all arguments from the args.me corpus. In order to improve the out-of-the-box functionality of ElasticSearch we implement a preprocessing pipeline before building the index. For optimal results, we follow the approach from group Elrond from Touché 2021. They performed well in relevance and quality scores, and were placed among the best in both categories. They used a combination of several preprocessing steps, which are: filters, namely Asciifolding and Lowercase, and stemming with the Krovetz algorithm [5, 20]. Then we add an additional step and remove stop words with a custom stop

[2]https://openai.com/

word list. As the last step in our pipeline we follow the approach of Team Elrond again and enrich our documents with WordNet-based synonyms [21, 5].

3.2. Retrieval

To retrieve arguments based on the given query, we also use ElasticSearch. We use the LM Dirichlet similarity to score matching documents because previous contributions from Touché which used ElasticSearch reported best results with LM Dirichlet similarity [22, 5, 23]. Figure 1 provides an overview of the pipeline of our retrieval system up to this point: The args.me corpus serves as a base for indexing and we search for relevant arguments using ElasticSearch. We use the top results as a starting point to find sentences which form a coherent pair. For this, we experiment with different methods which we introduce in the following sections.

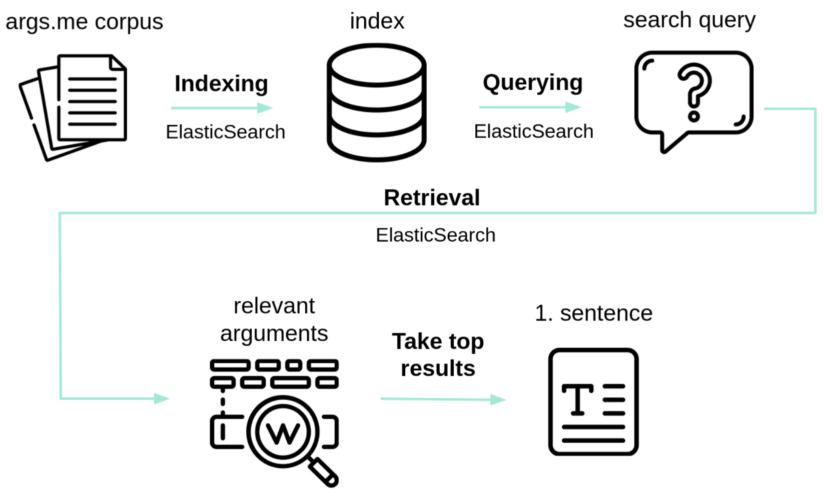

Figure 1: The setup of the retrieval system to retrieve relevant arguments up to a point where an initial sentence can be used to find a pair of argumentative sentences.

3.3. Semantic Search

Semantic search describes the process of finding meaning in a text. This meaning could refer to different parts of the search process, like understanding the query, the data or representing knowledge in a suitable way for retrieval [24]. So instead of the more traditional search engines which try to find documents based on lexical matches, semantic search can also find synonyms [25].

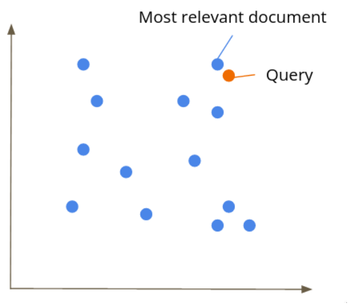

Figure 2: A schematic illustration of a vector space in two dimensions. The query of a search, marked as the orange dot, is embedded and relevant documents are closer than irrelevant documents. Figure adapted from [25]

The basic idea is that known data is embedded into a vector space, as seen in Figure 2. A text of arbitrary length is represented in the vector space. In the same way, a search query can be embedded into the same vector space and a vector can be inferred for the new data. It is possible to find the closest entry by calculating the euclidean distance between arbitrary vectors. This entry should then have a high semantic overlap with the query [25]. We use semantic search to compare the sentences we want to pair with the goal to find the most similar sentences. Next, we describe the two methods we used to retrieve sentence pairs: Doc2Vec and Sentence BERT. The two methods are based on the aforementioned concept of embedding documents of different length, in our case sentences, into a vector space and finding documents which are similar in their semantics.

3.3.1. Doc2Vec

Doc2Vec is an unsupervised machine learning model that represents documents by a dense vector. It was first introduced in 2014 from Mikolov and Le and builds upon Word2Vec [26, 7]. Word2Vec assumes that words can be embedded into the vector space and the resulting vectors can be used to measure semantic similarity. This understanding of semantic similarity or meaning is based on the bag-of-word model. Doc2Vec aims to overcome the major weaknesses of bag-of-word models: Assuming a sentence can be represented by their respective bag of words implies that the meaning is independent from the word order. Therefore, when using Word2Vec, the word ordering within the respective document is lost. Additionally, the context and semantics of the words are usually ignored and semantic features of sentences like negation or irony or cannot be taken into account [26]. To retrieve argumentative sentences for positive and negative stances, it is needed to represent this context. Using Word2Vec leads to a scenario where different sentences could have the same representation as long as those sentences contain the same words.

To solve this issue and take the context of the sentence as well as the word order into account, Mikilov and Le add a new vector to a document of variable length. The document vectors serve as information about the context of a word when using the embedding. It is combined with the respective word vectors of the document by averaging or concatenating. The *Paragraph id*, as seen in Figure 3, serves as a representation of the context for the paragraph or document, respectively.

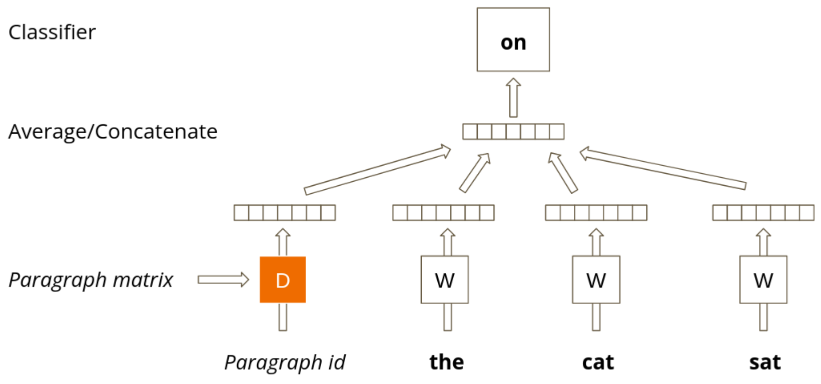

Figure 3: Doc2Vec's framework for learning paragraph vectors. The *Paragraph id* represents the context of a paragraph and is used as additional information to calculate embeddings for single words. Figure adapted from [26]

This document vector D is then used for the training of the word vectors W and holds the document representation. The authors report improvement for information retrieval and classification over other methods based on word embeddings and bigram embeddings alone. We choose Doc2Vec to retrieve coherent sentence pairs because the meaning of documents of variable length, in our case sentences, is represented and takes the sentence as a whole into account instead of every word, independent from the context. Therefore the similarity score produced by Doc2Vec promises to result in sentences which are similar in meaning rather than semantic similarity only.

3.3.2. Sentence BERT

BERT is a language representation model that was first introduced in 2018. For creating this language model, BERT uses deep bidirectional representations from unlabeled text. Bidirectional indicates that the model predicts a word based on what words it precedes and follows. With this approach, BERT managed to achieve success on several state-of-the-art tasks. This includes tasks like question answering and language inference [8].

For semantic similarity search, on the other hand, BERT is not so well suited. This is due to BERTs architecture, which comes with a high computational overhead. To overcome this bottleneck, Sentence-BERT (SBERT) was introduced in 2019 [25]. SBERT is a modification of the existing pretrained BERT network, which allows us "to derive semantically meaningful sentence embeddings" [25]. This is achieved by adding a pooling operation to the output of BERT and results in a fixed sized sentence embedding. Furthermore, siamese and triplet networks are created to fine-tune BERT and ensure that the produced embeddings are "semantically meaningful and can be compared with cosine-similarity" [27, 25].

SBERT is easily accessible over a python framework, which is based on PyTorch and Transformers [28, 29]. The framework allows the use of different pretrained models to train sentence embeddings on over 100 languages [25]. The model we use (all-MiniLM-L12-v2) is based on Microsoft's MiniLM and finetuned on more than 1 billion sentence pairs [30, 25]. The authors describe it as a general purpose model. This was a reason for us to not fine-tune the model any further for the training of our sentence embeddings.

3.4. GPT-2

GPT-2 is the abbreviation for Generative Pre-trained Transformer 2 which is an unsupervised transformer language model created by OpenAI in 2019. It is most used to translate texts, answer questions, summarize passages and to generate text output [31]. OpenAI web scraped 45 million outbound links from the social media platform Reddit to gather training data for the GPT-2 model. In order to assess the quality of a shared URL they used links which had at least 3 upvotes from the community. The final corpus consists of slightly over 8 million documents for a total of 40 GB of text [9]. The architecture of GPT-2 is based on Transformer [29]. OpenAI released four different model sizes of GPT-2 for free use which are compared in Table 1 [9].

Name	Parameters	Layers	d_{model}
SMALL	117M	12	768
MEDIUM	345M	24	1024
LARGE	762M	36	1280
EXTRA LARGE	1542M	48	1600

Table 1
Architecture hyperparameters for the publicly available four model sizes of GPT-2 [9]

We use GTP-2 to generate follow-up sentences for the initially retrieved sentences from the corpus with the goal of finding more coherent sentence pairs. This leads to the pipeline displayed in Figure 4, where we use the generated sentence from GTP-2 as input to retrieve a similar sentence with Doc2Vec and SBERT.

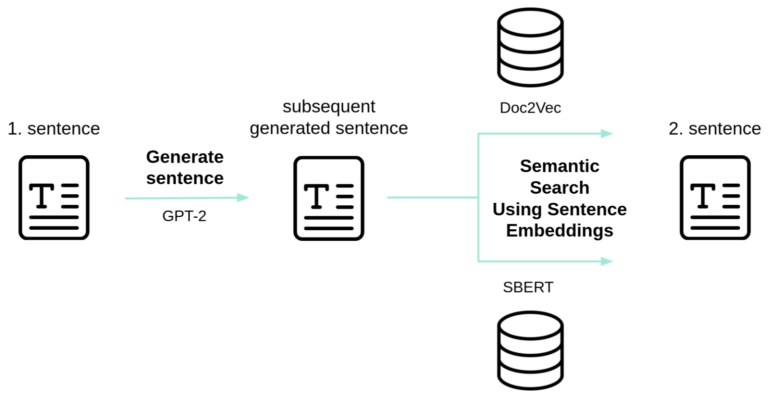
1. sentence
Generate sentence
GPT-2
subsequent generated sentence
Doc2Vec
Semantic Search Using Sentence Embeddings
SBERT
2. sentence

Figure 4: Continuation of the pipeline including text generation to find a fitting second argument.

4. Experiments

In the following sections we describe our approach in more detail and explain our experiments. We outline the pipeline for our retrieval system, as well as the setup and preprocessing steps. This includes a detailed description of the data we rely on and our steps to create an index. ElasticSearch serves as a baseline for indexing and retrieval of arguments. Then we describe our process of developing our pipeline with Doc2Vec, SBERT and text generation with GPT-2 from a prototype to a more refined system for retrieving argumentative sentences. In the next section we evaluate the results of the different approaches.

4.1. Setup: Dataset, Indexing and Argument Retrieval

For our experiments, we used the args.me corpus [2]. This corpus consists of 387 606 arguments. Each argument consists of an id, conclusion, premises (modeled as *stance* and *text*), context and sentences (modeled as *sourceTitle*, *sourceId*, *nextArgumentInSourceId*, *sourceUrl*, *discussionTitle*, *previousArgumentInSourceId*, *acquisitionTime*). The arguments were crawled from four different platforms, namely: Debatewise, IDebate.org, Debatepeia and Debate.org.
As described in Section 3.1 and 3.2, we use ElasticSearch for indexing and argument retrieval. We work with the configuration used by Team Elrond from the previous Touché task 1 [5]. With this setup we can query ElasticSearch to retrieve relevant arguments. Next, we present our approaches to pair fitting sentences.

4.2. Approach 1: Semantic Search using Doc2Vec and a matching algorithm

With this approach, the conclusion and each sentence of a retrieved argument is embedded by Doc2Vec. The Doc2Vec model was trained on the corpus of args.me [2]. We choose one sentence as a document in the context of Doc2Vec and train the model on all sentences of the corpus. We preprocess the sentences with two filters: First, we remove all sentences with less than three words, because they carry very little to no meaning. Then, we remove all sentences containing a hyperlink, because in most cases these sentences refer to a source to support their argument, but carry no meaning in themselves. For matching two sentences, we first experiment with a naive approach to look for the most similar sentence to the conclusion in the whole corpus. The conclusion is inferred as a vector in our Doc2Vec model and the most similar sentence from the corpus is located by computing the cosine similarity of the inferred vector. The quality of the matches is varying immensely and the two sentences only supported each other in a few cases, because the sentences from the whole corpus are retrieved and are often seemingly similar, but covering very different topics. Therefore we only look for matching sentences within the sentences of an argument.

To get the most meaningful sentences and to ensure the sentences are forming a coherent pair, we develop the following matching algorithm: All sentences of an argument are matched with each other and the cosine similarity of the respective vectors is calculated. Then, we calculate the cosine similarity for each sentence to the conclusion. Here, our reasoning to compare each sentence to the conclusion is the following: We assume the conclusion carries the summarized meaning of the argument, therefore we want to retrieve sentences which are most similar to the conclusion and carry the most meaning or the most important part of the argument, respectively. In a last step, we average the distance between the two sentences and the distance of each of the two sentences to the conclusion. Then the sentence pair with the highest averaged cosine similarity forms our top argumentative sentence pair.

4.3. Approach 2: Text-Generation

As an alternative approach we use GPT-2's text generation functionality to find a second sentence. We use the sentence from the results of ElasticSearch as input for GPT-2. It generates a subsequent text output. This output is used as input for semantic search in our Doc2Vec and SBERT models. In the end the most similar sentence to the generated output will be matched with the initial sentence and form a sentence pair. The idea is that GPT-2 can generate a sentence which is a coherent follow-up to the first sentence and that our two models can find the most similar sentence to the generated one.

Just using the first argument as an opening text (referred to as *prompt*) leads to mixed quality of the generated texts. In order to stabilize the quality we follow the approach of Akiki and Potthast from the Touché Lab 2020 [23]. They used the text generation functionality for a query expansion to achieve better retrieval results. By embedding the query in an argumentative structure, as seen in Table 2, they steered the language model into a output which resembles a opinion more closely than a simple statement [32]. To represent arguments for the same topic, but with opposing stances, they created positive, negative and neutral prompts as shown in Table 2. This stabilizes the text generation and leads to results which carry a stronger pro or con sentiment. Using the same prompts as Akiki and Potthast, we insert the conclusion from the respective argument into the outlined argumentative structure seen in Table 2 and use the modified prompt as input for GPT-2.

Stance	Prompt
Positive	- What do you think? <argument> - Yes because ..
	- What do you think? <argument> - The answer is yes ...
Negative	- What do you think? <argument> - No because ...
	- What do you think? <argument> - The answer is no ...
Neutral	- What do you think? <argument> - I don't know ...
	- What do you think? <argument> - Not sure ...

Table 2
Prompts for different stances to get a variety of argumentative output. The pattern of a question, the argument and an affirmative or dissenting beginning of a sentence lead to texts which closer to opinions in an arguments than factual statements.

Further the choice of the decoding method of GTP-2, which decodes the representation from the model into text, and their respective parameters are important for the outcome of the text output. These methods decide how incoherent, repetitive or generic the generated text is by examining which token could follow another token [33]. For our purpose we used the following three sampling methods for decoding as they provided the best output:

Temperature Sampling With this approach, a probability distribution is shaped through temperature [34]. The parameter $t \in [0, 1)$ steers the distribution towards high probability tokens or low probability tokens. A low value improves the generation quality but decreases the diversity of the output. A high value regularizes the generation by making the model less certain of its top choices [32, 35]. We used a temperature of 1.4.

Top-K Sampling In this sampling method, the neural language model distribution gets truncated to a set of size k of most likely tokens. These tokens will then be filtered and the probability mass is redistributed among those k next tokens [35, 36]. We chose a k of 75.

Nucleus Sampling Nucleus sampling is a stochastic decoding method. Like top-k sampling, this method is also truncating the language model distribution. It chooses from the smallest possible set of words whose cumulative probability exceeds the probability p. The rest of the tokens get discarded afterwards. It is also called top-p sampling [35]. We set the probability to $p = 0, 6$.

With these 3 decoding strategies and the six prompts we generate 18 text outputs with a length of 75 tokens for each passed argument. In order to figure out which generated output would fit best to the passed argument, we compute the cosine similarity for each sentence with the first argument. Some cosine similarity scores are quite high because GPT-2 was just repeating the passed argument in it's output (Table 3). Therefore we set a threshold to remove sentences which were too similar to the prompt. Among the remaining generated sentences, the best one is passed to Doc2Vec and BERT for semantic search.

With this approach we encounter the problem that we always have a conclusion as part of the sentence pair and that some conclusions are quite short, therefore carrying little meaning. To counteract this, we implement the following step in our pipeline:

Doc2vec The results of ElasticSearch, the most relevant arguments to a query, are passed on to Doc2Vec. For each sentence of the argument, GPT-2 generates a subsequent text output. Afterwards we compute the cosine similarity of each text output with each sentence of the respective argument and use the most similar sentence pair.

SBERT The conclusion of the best argument retrieved by ElasticSearch is used as input for SBERT. The language model embeds the conclusion and retrieves the five sentences with the highest semantic similarity. For each of these sentences, GPT-2 generates a subsequent text output. These text outputs are again passed on to SBERT which searches similar sentences. The

Generated Output	**Score**
What do you think? We need more sex education in schools. No because that's what the kids do. We need more sex education in schools. Do you agree with the idea that we should have sex education in schools? No. We should have sex education in schools. Do you agree with the idea that we should have sex education in schools?	0.8109
What do you think? We need more sex education in schools. I don't know what to say. I think that sex education is just not enough. If you don't get it right, it's not going to work. I think that if we are really concerned about the sex education system, we need to have a national sex education program, where all ...	0.7838
What do you think? We need more sex education in schools. Not sure ive seen any evidence that sex education is helping to reduce sexual violence.	0.7396

Table 3
This table displays an example of the top three out of 18 sentences ordered by the cosine similarity to the query "*sex education in schools*". The expanded query is italic and everything following is generated by GPT-2.

result of SBERT are then matched with the first argument and the most similar sentence pair form the desired argumentative sentence pair.

5. Evaluation And Discussion

We use a a two-step approach for our own evaluation of the results of our retrieval system. In the first step we evaluate the results of the argument retrieval and in a second step we evaluate the sentence pairs generated from the results of the argument retrieval. This ensures that the sentence pairs are generated out of a well evaluated baseline of argument retrieval. We use the mean Normalized Discounted Cumulative Gain (NDCG) to measure the quality for both steps over several topics [37]. This measure is also used to compare the different approaches for the Touché 2021 tasks for argument retrieval [5]. So we are also able to compare our results of the argument retrieval to the results of the previous results of Touché 2021. We also evaluate the system in the context of this years Touché Lab. The results from the TIRA platform will be discussed at the end of this section.

5.1. Argument retrieval evaluation

We use a dataset with graded results for 50 example topics for the evaluation of the results from the argument retrieval. The dataset was also used for the relevance judgments for the Touché 2021 task 1 [38]. The entries in this dataset contain a topic id, the argument id and a grade which indicates how relevant a argument is for a topic. The grades can have the values -2 non-argument (spam) and from 0 (not relevant) to 3 (highly relevant).

In Table 4 the results for different approaches on argument retrieval are displayed. The baseline retrieval uses the fields *conclusion*, *premises.text* and *sentences.sent_text* from the argument corpus with the weight of one for each field and LMDirichlet as retrieval model [6]. For the optimized baseline approach the weights of the fields which are used for retrieval are optimized (in a range from zero to three) with a bayesian optimizer[39]. The measure of the optimization is the Mean NDCG value as described before. The expanded index approach is the one which is described in Section 3.1. In the optimized expanded index approach the weights for the search fields are also optimized with a bayesian optimizer and in the heavily optimized expanded approach the optimization is made with more iterations but as described in Table 4 the NDCG values are staying the same.

Compared to the Touché 2021 task 1 approaches with the best relevance scores (Elrond with a NDCG@5 value of 0.720 and Pippin Took with a NDCG@5 value of 0.705) the optimized expanded index approach (The expanded index approach from section 3.1 with optimized weigths of the search fields) scores slightly better.

Method	**Mean NDCG**	**Mean NDCG@5**
Baseline	0.8385	0.6754
Baseline, optimized	0.8559	0.7116
Expanded Index	0.8506	0.7166
Expanded Index, optimized	0.8624	0.7417
Expanded Index, heavily optimized	0.8624	0.7417

Table 4
The Mean NDCG and NDCG@5 values for different approaches on argument retrieval

5.2. Sentence pair evaluation

For the evaluation of the retrieved sentence pairs the evaluation data has to be created manually. Therefore a dataset similar to the dataset for the argument retrieval evaluation is created based on the retrieved sentence pairs for a topic. Then the results are reviewed by hand and are graded by the relevance to the topic and how good the sentences fit together in an argumentative way. To keep the effort within an acceptable range the evaluation is made for ten top results of each of ten topics.

As shown in Table 5 the Doc2Vec method without text generation from GPT-2 returns the best results for the sentence pair retrieval. The baseline approach uses random pairs of sentences of the first ten results. It is striking that the Bert approach performs even worse than the random pairs approach.

Method	Mean NDCG	Mean NDCG@5
Random Pairs	0.1775	0.1121
Doc2Vec	0.5572	0.3644
Doc2Vec + GPT-2 text generation	0.3235	0.2283
Bert + GPT-2 text generation	0.05	0.05

Table 5
The Mean NDCG and NDCG@5 values for different approaches of sentence pair retrieval

5.3. Final evaluation on TIRA

In the previous section, the results of the teams own evaluation were described. The retrieval of sentence pairs is also evaluated in the context of Touché 2022 [3]. For this, we are provided with the TIRA platform [11]. Researchers can submit their solution for shared tasks and evaluate the results on TIRA. We choose the configuration which performed best in our own evaluation, that is Doc2Vec without GTP-2 text generation. The evaluation results for our submitted run, named *korg9000*, can be seen in Table 6.

Category	Mean NDCG@5	CI95 Low	CI95 High
Relevance Evaluation	0.252	0.187	0.318
Quality Evaluation	0.453	0.384	0.529
Coherence Evaluation	0.168	0.117	0.223

Table 6
The evaluation results for the run on the TIRA platform by category

Our retrieval system performed poorly both in comparison to our own evaluation and to the results of the other participants of the shared task. The Mean NDCG@5 for relevance is 0.252, compared to the best performing run from Team Porthos with a Mean NDCG@5 of 0.742. The Mean NDCG@5 for quality is 0.453. The best performing team in this category is Daario Naharis with a Mean NDCG@5 of 0.913. For the coherence evaluation, our run scores a Mean NDCG@5 of 0.168, while the best performing team, again Daario Naharis, has a score of 0.458.

5.4. Discussion of the final results

The approach of our experiments outlined in this paper is to compare different types of semantic search. Within our own evaluation, we were able to compare the retrieved sentence pairs of Doc2Vec and SBERT. The results were mixed in both cases. This can be attributed to the differences between the two methods: While it was easy to train the Doc2Vec model ourselves, the model was not designed to work with short sentences, but with paragraphs in mind. For SBERT, it was outside of the scope of this notebook to customize the training, which could have led to the poor results. We use the same pipeline to generate the input for the semantic search, but we could have benefited from a step-wise evaluation of each step of the pipeline. Furthermore more runs on the TIRA platform with different configurations would have been helpful to test out our approaches. Then the strengths and weaknesses of the respective approaches could have been better analysed and combined.

6. Conclusion

To solve Touché task 1, we developed a system to find pairs of argumentative sentences for positive and negative stances towards an arguement. We worked with the args.me corpus and two different approaches: Doc2Vec and Sentence BERT, and combining both of them with GPT-2, finally reembedding the resulting sentences. For the retrieval of arguments we use ElasticSearch. In the end we compare and evaluate our results with the NDCG and Mean NDCG.

We show the differences in how the sentences are paired have a great impact on the overall results of the retrieval system. The Doc2Vec approach delivers acceptable results after reworking the sentence pairing algorithm to only use sentences from the respective argument. Furthermore using text generation to improve the input for the sentence embedding models does not provide better results by default. On the contrary, the results for Doc2Vec are better without the additional step of generating a more elaborate prompt. The quality of the sentence pairs depends on the quality of the generated sentences from GPT-2. Using GPT-2 in this context needs more fine-tuning to reduce the randomness in the generated texts. This could be a starting point for further improvements as it is possible to use GPT-2 to generate more than one sentence and check if one of those sentences fits better. This is contrasted by the negative impact on performance by the sentence generation, which slows down the entire pipeline. Doc2Vec without additional text generation delivers the best performance in the context of our own evaluation.

Unfortunately, Sentence BERT performs really poorly in our experiments. The SBERT approach in combination with the text generation using GPT-2 has a significant lower Mean NDCG/Mean NDCG@5 score in the evaluation than the baseline approach with random sentence pairs. The poor results could stem from not further fine-tuning SBERT or not choosing the right model for this task. The algorithm to match the two desired sentences needs to be revised to produce better results. There is more room for further improvements of our pipeline, like other steps in preprocessing, using part of speech tags or trying to find a subject object relationship between argumentative sentence pairs to get a better understanding of the context of a sentence.

Acknowledgments

Thanks to Theresa Elstner for the extensive feedback on the draft of this notebook and to the team of Temir https://temir.org/people.html for supporting us with the upload of our software on the Tira https://tira.io/ platform to evaluate our retrieval system.

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: K. Ashley, C. Cardie, N. Green, I. Gurevych, I. Habernal, D. Litman, G. Petasis, C. Reed, N. Slonim, V. Walker (Eds.), 4th Workshop on Argument Mining (ArgMining 2017) at EMNLP, Association for Computational Linguistics, 2017, pp. 49–59. URL: https://www.aclweb.org/anthology/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] A. Bondarenko, M. Fröbe, J. Kiesel, S. Syed, T. Gurcke, M. Beloucif, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2022: Argument Retrieval, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. 13th International Conference of the CLEF Association (CLEF 2022), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2022, p. to appear.

[4] Clef initiative, 2010. URL: http://www.clef-initiative.eu/, accessed: 2022-02-27.

[5] A. Bondarenko, L. Gienapp, M. Fröbe, M. Beloucif, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2021: Argument Retrieval, in: K. Candan, B. Ionescu, L. Goeuriot, H. Müller, A. Joly, M. Maistro, F. Piroi, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. 12th International Conference of the CLEF Association (CLEF 2021), volume 12880 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg New York, 2021, pp. 450–467. URL: https://link.springer.com/chapter/10.1007/978-3-030-85251-1_28. doi:10.1007/978-3-030-85251-1_28.

[6] C. Zhai, J. Lafferty, A study of smoothing methods for language models applied to ad hoc information retrieval, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01, Association for Computing Machinery, New York, NY, USA, 2001, p. 334–342. URL: https://doi.org/10.1145/383952.384019. doi:10.1145/383952.384019.

[7] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013. arXiv:1301.3781.

[8] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019.

[10] T. Gollub, B. Stein, S. Burrows, D. Hoppe, Tira: Configuring, executing, and disseminating information retrieval experiments, in: 2012 23rd International Workshop on Database and Expert Systems Applications, 2012, pp. 151–155. doi:10.1109/DEXA.2012.55.

[11] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA Integrated Research Architecture, in: N. Ferro, C. Peters (Eds.), Information Retrieval Evaluation in a Changing World, The Information Retrieval Series, Springer, Berlin Heidelberg New York, 2019. doi:10.1007/978-3-030-22948-1_5.

[12] M. Pasca, Web-based open-domain information extraction, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 2605–2606. URL: https://doi.org/10.1145/2063576.2064034. doi:10.1145/2063576.2064034.

[13] R. Rinott, L. Dankin, C. Alzate Perez, M. M. Khapra, E. Aharoni, N. Slonim, Show me your evidence - an automatic method for context dependent evidence detection, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 440–450. URL: https://aclanthology.org/D15-1050. doi:10.18653/v1/D15-1050.

[14] T. Bench-Capon, P. E. Dunne, Argumentation in artificial intelligence, Artificial Intelligence 171 (2007) 619–641. URL: https://www.sciencedirect.com/science/article/pii/S0004370207000793. doi:https://doi.org/10.1016/j.artint.2007.05.001, argumentation in Artificial Intelligence.

[15] I. Rahwan, F. Zablith, C. Reed, Laying the foundations for a world wide argument web, Artificial Intelligence 171 (2007) 897–921. URL: https://www.sciencedirect.com/science/article/pii/S0004370207000768. doi:https://doi.org/10.1016/j.artint.2007.04.015, argumentation in Artificial Intelligence.

[16] M. Potthast, L. Gienapp, F. Euchner, N. Heilenkötter, N. Weidmann, H. Wachsmuth, B. Stein, M. Hagen, Argument search: Assessing argument relevance, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1117–1120. URL: https://doi.org/10.1145/3331184.3331327. doi:10.1145/3331184.3331327.

[17] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[18] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013. arXiv:1301.3781.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, 2013. arXiv:1310.4546.

[20] R. Krovetz, Viewing morphology as an inference process, in: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '93, Association for Computing Machinery, New York, NY, USA, 1993, p. 191–202. URL: https://doi.org/10.1145/160688.160718. doi:10.1145/160688.160718.

[21] C. Fellbaum (Ed.), WordNet: An Electronic Lexical Database, Language, Speech, and Communication, MIT Press, Cambridge, MA, 1998.

[22] C. Zhai, J. Lafferty, A study of smoothing methods for language models applied to ad hoc information retrieval, SIGIR Forum 51 (2017) 268–276. URL: https://doi.org/10.1145/3130348.3130377. doi:10.1145/3130348.3130377.

[23] A. Bondarenko, M. Fröbe, M. Beloucif, L. Gienapp, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2020: Argument Retrieval, in: A. Arampatzis, E. Kanoulas, T. Tsikrika, S. Vrochidis, H. Joho, C. Lioma, C. Eickhoff, A. Névéol, L. Cappellato, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. 11th International Conference of the CLEF Association (CLEF 2020), volume 12260 of *Lecture Notes in Computer Science*, Springer, Berlin Hei-

delberg New York, 2020, pp. 384–395. URL: https://link.springer.com/chapter/10.1007/978-3-030-58219-7_26. doi:10.1007/978-3-030-58219-7_26.

[24] H. Bast, B. Buchhold, E. Haussmann, Semantic search on text and knowledge bases, Foundations and Trends® in Information Retrieval 10 (2016) 119–271. URL: http://dx.doi.org/10.1561/1500000032. doi:10.1561/1500000032.

[25] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. arXiv:1908.10084.

[26] Q. V. Le, T. Mikolov, Distributed representations of sentences and documents, 2014. arXiv:1405.4053.

[27] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015). URL: http://dx.doi.org/10.1109/CVPR.2015.7298682. doi:10.1109/cvpr.2015.7298682.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017. arXiv:1706.03762.

[30] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL: https://arxiv.org/abs/2002.10957. doi:10.48550/ARXIV.2002.10957.

[31] C. Hegde, S. Patil, Unsupervised paraphrase generation using pre-trained language models, 2020. arXiv:2006.05477.

[32] C. Akiki, M. Potthast, Exploring Argument Retrieval with Transformers, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), Working Notes Papers of the CLEF 2020 Evaluation Labs, volume 2696, 2020. URL: http://ceur-ws.org/Vol-2696/.

[33] A. Géron, Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems, first edition, fifth release ed., O'Reilly, Beijing, 2018. URL: https://katalog.ub.uni-leipzig.de/Record/0-1640048871.

[34] D. H. Ackley, G. E. Hinton, T. J. Sejnowski, A learning algorithm for boltzmann machines, Cognitive Science 9 (1985) 147–169. URL: https://www.sciencedirect.com/science/article/pii/S0364021385800124. doi:https://doi.org/10.1016/S0364-0213(85)80012-4.

[35] A. Holtzman, J. Buys, L. Du, M. Forbes, Y. Choi, The curious case of neural text degeneration, 2020. arXiv:1904.09751.

[36] A. Fan, M. Lewis, Y. Dauphin, Hierarchical neural story generation, 2018. arXiv:1805.04833.

[37] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Trans. Inf. Syst. 20 (2002) 422–446. URL: https://doi.org/10.1145/582415.582418. doi:10.1145/582415.582418.

[38] webis.de, Touché 2021 relevance judgements, 2021. URL: https://webis.de/events/touche-22/shared-task-1.html, accessed: 2022-02-27.

[39] J. Močkus, On Bayesian Methods for Seeking the Extremum, Springer Berlin Heidelberg, Berlin, Heidelberg, 1975, pp. 400–404. URL: https://doi.org/10.1007/978-3-662-38527-2_55. doi:10.1007/978-3-662-38527-2_55.