FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Argument Retrieval for Controversial Questions

Bernardo Costa Moreira

Mestrado em Engenharia Informática e Computação

Supervisor: Henrique Lopes Cardoso

Second Supervisor: Bruno Martins

Third Supervisor: Fábio Goularte

July 22, 2022

Argument Retrieval for Controversial Questions

Bernardo Costa Moreira

Mestrado em Engenharia Informática e Computação

July 22, 2022

Abstract

We are constantly exposed to significant amounts of information, misinformation, and fake news. When in need of defending a point of view regarding a controversial topic, one needs to have strong supportive arguments which must relate to the topic, be coherent and based on information from reliable sources.

Natural language processing has made it possible to program models to interpret and understand text, generate human-readable text, as well as to rank and retrieve text excerpts according to their relevance for particular questions. In this context, the Touché task which was presented within the scope of the Conference and Labs of the Evaluation Forum (CLEF), focuses on argument retrieval for controversial questions. In a world full of information sources and unfounded arguments, argument retrieval is a current hot topic as it focuses on creating models that can retrieve coherent and strong premises from a text. This promising technology could help individuals form an informed opinion about a controversial topic or support a particular stance on a debate.

The Touché task offers a dataset provided by *args.me* (a search engine for arguments, meant to be used by anyone to support people forming an opinion on a controversial topic). This dataset was gathered from five different debate portals, and it contains 387 740 arguments. The 2022 edition of the Touché Task 1 focuses on retrieving pairs of sentences. Human assessors manually evaluate the retrieved pairs of sentences by analyzing three characteristics: whether each sentence is argumentative, coherent, and forms a summary of the corresponding arguments.

Thus, the main goal of this research project is to develop a competitive model for the Touché task. To do so, we used Pyserini sparse and hybrid search to address both the 2021 and 2022 editions of Touché Task 1. In the 2021 edition, the hybrid search achieved the best results from all of our approaches, ranking fourth in relevance (0.6863) and sixth in quality (0.8116). However, due to the computational and space requirements of this approach, we decided not to use it in the 2022 edition of the task. Instead, we chose to compete with a sparse search and achieved scores of 0.772 for quality, 0.651 for relevance, and 0.378 for coherence. Our results are limited by our data arrangement process. The purpose of the task was to retrieve the most argumentative and relevant pairs of sentences, which could be formed with sentences from the same argument or not. Our approach focused on forming all sentence pairs possible within the same argument, therefore missing out on many possible pairs of sentences. Nevertheless, we have achieved, for the 2022 Task, rank third, fourth and fifth in relevance, coherence and quality. This might be indicative that based on the adopted evaluation method, it is hard to find methodologies that go beyond simple retrieval techniques.

Keywords: Natural Language Processing, Argument Retrieval, Pyserini, Sparse search, Hybrid search

Resumo

Atualmente, estamos constantemente expostos a quantidades significativas de informação, desinformação, e notícias falsas. Quando é necessário defender um ponto de vista sobre um tema controverso, é preciso ter argumentos fortes que apoiem a sua posição. Tais argumentos devem estar relacionados com o tema e basear-se em informação boa e fiável. O processamento de linguagem natural tornou possível treinar modelos para interpretar, compreender e gerar texto legível por humanos, assim como recuperar excertos de texto classificados de acordo com a relevância para questões particulares. Neste contexto, a tarefa Touché, proposta no âmbito da Conferência e Laboratórios do Fórum de Avaliação (CLEF), uma conferência internacional líder anual, centra-se na recuperação de argumentos para questões controversas. Ora, a recuperação de argumentos é um tema quente neste momento pois, num mundo cheio de fontes de informação e argumentos infundados, centra-se na criação de modelos que possam recuperar premissas coerentes e fortes a partir de um texto. Esta tecnologia promissora poderia ajudar os indivíduos a formar uma opinião informada sobre um tema controverso ou apoiar uma posição particular num debate.

Assim, esta tarefa Touché oferece um conjunto de dados fornecido por "args.me" (ou seja, um motor de busca de argumentos, destinado a ser utilizado por qualquer pessoa para apoiar pessoas que formem uma opinião sobre um tema controverso). Este conjunto de dados foi recolhido a partir de cinco portais de debate diferentes, e contêm 387 740 argumentos. A edição de 2022 da Tarefa Touché 1 centra-se na recuperação de pares de frases. Os avaliadores humanos avaliam manualmente os pares de frases recuperados, analisando três características: se cada frase é argumentativa, coerente, e forma um resumo dos argumentos correspondentes.

Assim, o principal objectivo deste projecto de investigação é desenvolver um modelo competitivo para a tarefa Touché. Para o fazer, utilizámos a pesquisa Pyserini esparsa e híbrida para abordar as edições 2021 e 2022 da Tarefa Touché 1. Na edição 2021, a pesquisa híbrida obteve os melhores resultados de todas as nossas abordagens, classificando-se em quarto lugar em relevância (0,6863) e sexto em qualidade (0,8116). Contudo, devido às exigências computacionais e de espaço desta abordagem, decidimos não a utilizar na edição de 2022 da tarefa. Em vez disso, optámos por competir com uma pesquisa esparsa e atingimos pontuações de 0,772 em qualidade, 0,651 em relevância, e 0,378 em coerência. Os nossos resultados são limitados pelo nosso processo de disposição de dados. O objectivo da tarefa era recuperar os pares de frases mais argumentativos e relevantes, que poderiam ser formados com frases do mesmo argumento ou não. A nossa abordagem centrou-se na formação de todos os pares de frases possíveis dentro do mesmo argumento, faltando, portanto, muitos pares de frases possíveis. No entanto, alcançámos, para a Tarefa 2022, terceiro, quarto e quinto lugares em relevância, coerência, e qualidade. Isto pode ser indicativo de que, com base no método de avaliação adoptado, é difícil encontrar metodologias que vão para

além de simples técnicas de recuperação.

Keywords: Processamento de Linguagem Natural, Recuperação de Argumentos, Pyserini, pesquisa esparsa, pesquisa híbrida

Acknowledgements

I want to start by thanking my supervisors, Bruno Martins and Fábio Goularte, who supported and guided me in this project, and especially my supervisor, professor Henrique Lopes Cardoso, for being always available and having the patience to enlighten me in moments where inspiration was lacking.

I also want to thank all my colleagues with whom I shared good times. To my friends since day one, Paquito, Landau, and Filipe, for the unforgettable moments, night outs, study sessions, gym sessions, and some retakes.

To Migas, Tono, Duda, Maria, and Kika, thank you for all the moments of relaxation and comfort we shared over the years.

To my friends Claudia, Vitor, Bernas, and Dantas for sharing the pain of working on projects until the last minute and always being available to help me. The "thanks" to Claudia comes with a complaint for constantly nagging me to work on projects I didn't want to do and for being such a perfectionist.

To the 2022 Tese-Udos "Queima das fitas" tent for providing me with an unforgettable last "Queima".

Finally, to all my family who was able to put up with me during many exam seasons, and throughout this dissertation. Especially my parents, without whom I would never have been able to finish my master's (mainly because I wouldn't have money to pay the tuition fees or the rent of the house).

Bernardo Moreira

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

Alan Turing

Contents

List of Figures

List of Tables

Abbreviations

AC	Argument Components
ADORE	Algorithm for Directly Optimizing Ranking pErformance
BoW	Bag of Words
CBoW	Continuous Bag-of-words
CLEF	Conference and Labs of the Evaluation Forum
CNN	Convolutional Neural Networks
DR	Dense Retrieval
FNC-1	Fake News Challenge Stage 1
HNSW	Hierarchical Navigable Small World Graphs
IR	Information Retrieval
LMs	Language models
LSTM	Long Short-Term Memory
MLM	Masked Language Modeling
MLP	Multilayer Perceptron
NDCG	Normalized Discounted Cumulative Gain
NLI	Natural Language Inference
NLP	Natural Language Processing
NNLM	Neural Net Language Model
NSP	Next Sentence Prediction
PD	Pyserini dense retrieval
PH	Pyserini hybrid retrieval
PH5	Pyserini hybrid retrieval with docT5query query expansion
PS	Pyserini sparse retrieval
PS5	Pyserini sparse retrieval with docT5query query expansion
PSArgQ	Pyserini sparse retrieval and argument quality model for re-ranking
PSArgQD	Pyserini sparse retrieval and argument quality model for re-ranking draw scores only
PSCE	Pyserini sparse retrieval and cross encoder for re-ranking
PSCED	Pyserini sparse retrieval and cross encoder re-ranking draw scores only
STAR	Stable Training Algorithm for dense Retrieval
START	Stable Training Algorithm for Dense Retrieval
ST-AD	sentence-transformers.all-distilroberta-v1
ST-AM	sentence-transformers.all-mpnet-base-v2
ST-DB	sentence-transformers.distilbert-base-nli-stsb-mean
ST-PA	sentence-transformers.paraphrase-albert-small-v2
SVM	Support Vector Machine
TCT-T	castorini.tct_colbert-v2-hnp-msmarco
TGA	Topic-Grouped Attention

Chapter 1

Introduction

The concept of Natural Language Processing (NLP) is currently present in almost everything that is performed on the web. There are many fields within the scope of NLP, and Information Retrieval (IR) is the one in which this work focuses. IR is the process of accessing and retrieving the most pertinent information given a query by a user. This work concerns retrieving relevant arguments (a collection of premises and conclusions) on controversial topics according to quality and relevance. For example, if a user searches for "war in Afghanistan" and the retrieved argument is "Freedom of speech should apply to teachers as much as anyone else", it has no utility for the user, but if the argument retrieved is the "Saddam Hussein is gone and Iraq is now functioning as one of very few democracies in the Middle East", it provides relevant information to the user.

1.1 Context

The Touché task[1], presented at the Conference and Labs of the Evaluation Forum (CLEF), focuses on retrieving arguments for controversial topics. Argument retrieval is a prominent research area since it focuses on constructing models that can return coherent and strong arguments from textual documents which is of great need in today's world. Despite the availability of good and well-established information sources, there are still vast amounts of unsupported and inauthentic information being shared every day, which creates a barrier to accessing reliable information. The technology behind Touché task has the potential to filter out all the unreliable information and therefore assist people in forming an opinion on a contentious topic, or defending their position in a discussion.

The Touché sub-task 1 also offers a dataset with data collected from "args.me"[2], which is a search engine for arguments. Its goal is to assist individuals in formulating their perspectives on controversial matters. This search engine searches for pro and con arguments for a particular issue on trustworthy web platforms and evaluates their importance using computational argumentation research methodologies.

[1] https://webis.de/events/touche-22/
[2] https://www.args.me/index.html

Table 1.1: Example of input arguments for Touché Task 1. Both arguments are related with the same topic but reach different conclusions.

Fields	**Argument 1**	**Argument 2**
Id	Sf9294c83-Af186e851	Sf9294c83-A9a4e056e
Topic	the War in Iraq was Worth the Cost	the War in Iraq was Worth the Cost
Stance	PRO	PRO
Conclusion	the War in Iraq was Worth the Cost	Saddam Hussein is gone and Iraq is now functioning as one of very few democracies in the Middle East
Premise	His removal provides stability and security not only for Iraq but for the Middle East as a region	It's important to be clear that this debate is looking at the results of the Iraq war and, by any definition Iraq is in a much more stable and secure position than it was in 2003 when American, British and other international troops arrived in the country. Whatever one thinks of the initial justifications for the war there is no doubt that the country, the region and the world are better and safer places without Saddam Hussein[i]. It is easy to criticize the allies but it is worth bearing in mind that the alternative was leaving in power a man who had committed genocide was a vicious and brutal dictator under whose regime extra-judicial execution and detention, mass-murder and torture were commonplace[ii]. [i] Richard Miniter. "Was the Iraq War Worth It?". Hudson New York. 2 September 2010. [ii] Interview with Donald Rumsfeld. Inside Politics. NPR. 14 February 2011.

Data was gathered from many different debate portals, containing 387 740 arguments that will be used to train and test the developed models; two of these arguments are represented in Table 1.1.

For the Touché Task 1 [3], the goal is to, given a query, retrieve the most argumentative, coherent, and relevant pairs of sentences, where each sentence must be able to summarize the respective topic it is inserted in. To participate in the task, the teams must submit their results file, see Figure 1.1, which must follow the TREC format:

qid stance pair rank score tag

where the **qid** is the query id, the **stance** is the stance of the sentence pair (pro or con), **pair** is the id of the sentence pair (sentece1 id, sentence2 id), and the **rank** is the sentence-pair place in the final results retrieved. The **score** is the score that the retrieval model assigned for the sentence pair, and the **tag** is the team's tag, which in our case is Bruce-Banner.

[3] https://webis.de/events/touche-22/shared-task-1.html

```
1 PRO S71152e5e-A66163a57__CONC__1,S71152e5e-A66163a57__PREMISE__2 1 17.89 myGroupMyMethod
1 PRO S6c286161-Aafd7e261__CONC__1,S6c286161-Aafd7e261__PREMISE__1 2 16.43 myGroupMyMethod
1 CON S72f5af83-Afb975dba__PREMISE__1,S72f5af83-Afb975dba__CONC__1 3 16.42 myGroupMyMethod
...
```

Figure 1.1: Example of output for Touché Task 1.

1.2 Motivation

Controversial topics are often sensitive topics; individuals might have completely different opinions, and therefore it is imperative that, when discussing these topics, access to relevant, coherent and reliable information is ensured. The inauthentic information must be filtered out, which is time-consuming and sometimes impossible since this entails the individual having to assimilate all the available information and determine its authenticity and relevance. This has motivated the development of automated programs to search, filter, and retrieve arguments, ranked by their pertinence and quality.

There are already some platforms for this purpose, for example, “args.me”. However, like with any technology, improvements can be made to provide better results and a better user experience.

1.3 Goals

This dissertation has two main goals. The first is to develop an argument retrieval technique for the 2021 edition of the Touché Task 1, where the goal is to retrieve relevant arguments for a given query. The second goal is to participate in Touché Task 1 2022 [12] with an improved approach to the technique used for the 2021 edition since the purpose of the task has suffered a fundamental change. The goal of the 2022 task 1 is to retrieve and classify pairs of relevant sentences from a collection of arguments, in response to a user query. This task aims to aid a user in searching for statements supporting their stance or just searching for a general overview regarding a specific topic.

In alignment with the above mentioned goals, we pursue the following research questions in this work:

- Compared to state-of-the-art approaches from Touché Task 1 2021 edition, can a Pyserini sparse search approach achieve competitive results?
- Despite the focus for the 2022 edition of the Touché Task 1 being different from the 2021 edition, can a simple IR approach based on the construction of documents as sentence pairs obtain a competitive model for the Touché Task 1 2022?

1.4 Contributions

We experimented with our approach based on Pyserini hybrid search, with the evaluation files of the 2021 edition of Touché Task 1, and achieved scores of 0.8116 for quality and 0.6863 for relevance. Compared to the other teams' results, our team would be placed fourth and sixth in relevance and quality, respectively. Therefore, answering our first research question, we conclude that the Pyserini sparse search compared to other state-of-the-art approaches from Touché Task 1 2021 edition can achieve competitive results.

We participated in the 2022 edition of the Touché Task 1 with a Pyserini sparse search approach, achieving scores of 0.772 for quality, 0.651 for relevance and 0.378 for coherence. With these results we achieved third place in relevance, fourth in coherence and fifth in quality. In this context, we submitted a paper reporting our approach to the Touché Task 1 2022, as shown in Appendix A.5, and it was accepted [77].

1.5 Document Structure

This document is organized into six additional chapters. Chapter 2 introduces relevant concepts to better understand the domain of this work and the rest of the document. Chapter 3 presents state-of-the-art models for the tasks of information retrieval, stance detection, and for addressing the task of the last edition of Touché Task on argument retrieval for controversial questions. These models will be used to establish a comparison to the model that is to be developed for this work. Chapter 4 describes the available datasets for the Touché Task and complementary data used to develop an argument quality model. Chapter 5 describes the approaches that we have followed to tackle this problem. Chapter 6 describes all the experiments performed and respective results. Lastly, Chapter 7 provides the main conclusions of this work and includes suggestions for future work.

Chapter 2

Background

This chapter presents all relevant information needed to comprehend subjects such as text representation, information retrieval, and argumentation in natural language.

2.1 Deep Learning

Deep learning is a field that falls under the umbrella of machine learning and it is based on artificial neural networks. Many well-known deep learning architectures are applied to different fields such as natural language processing, computer vision or speech recognition.

Neural networks are inspired by the functioning of the human brain, composed of interconnected neurons forming a network. These networks take data as an input, recognize the patterns, and then predict the output [78] [109]. One of the simplest neural network structures is the Multilayer Perceptron (MLP) [85][92][79][2] ilustred in Figure 2.1.

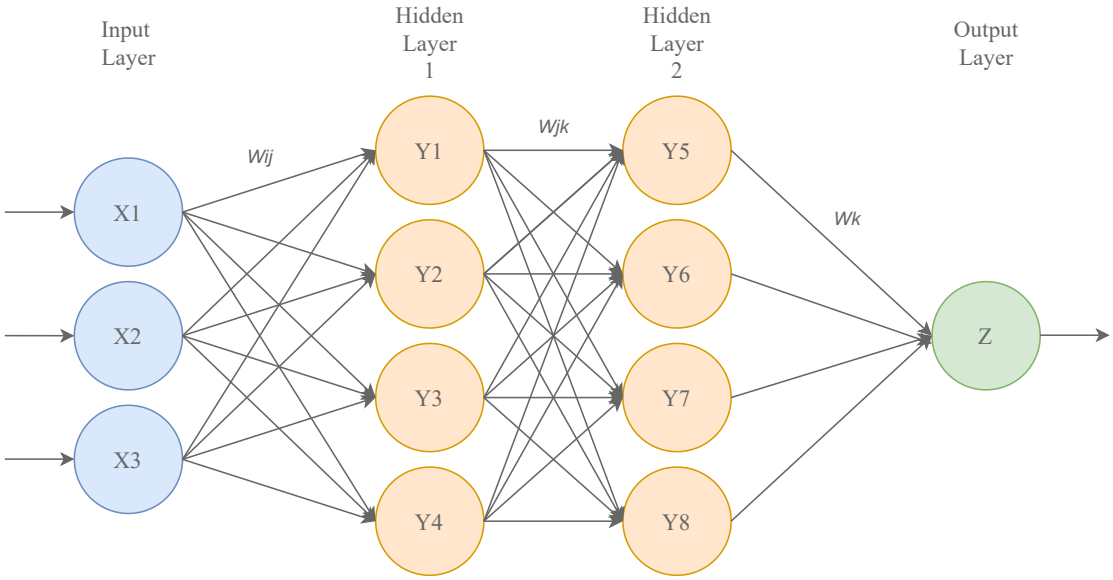


Figure 2.1: Example of a feed forward architecture with two hidden layers with four neurons each.

2.1.1 Multilayer Perceptron

MLP [94], comprises the input layer, the output layer, and in between, one or more hidden layers. These layers have neurons, which are connected from lower layers to higher layers, and have associated weights.

The input to this network is the vector from the data to be classified, and the output is the prediction of the class of the data. This model can be mathematically defined as :

$$Z = \sigma(\sum_{i=1}^{n} w_{ij}x_i + b) \tag{2.1}$$

where σ is the activation function, x the vector of inputs, w the vector of weights and b the bias.

The training stage for the MLP is done through back-propagation. This method is used to adapt the network in order to minimize the difference between the predicted output and the desired output. This minimization is achieved using gradient descent, an algorithm used to perform optimization [93]. This algorithm is a way to minimize an objective function by updating the parameters of the model in the opposite direction of the objective function parameters. An example of a gradient descent algorithm is batch gradient descent:

$$\theta = \theta - \eta \cdot \nabla(\theta)J(\theta) \tag{2.2}$$

where θ is the model's parameters, η is the learning rate and it controls the step size of the process, $J(\theta)$ is the objective function and $\nabla(\theta)J(\theta)$ is the gradient of the objective function.

Let us consider an example where a neural network is trained to predict whether a text's sentiment is positive or negative. The input layer receives the sentence and feeds the neurons with the embeddings from each term. Each neuron's connection has a weight and is connected to the hidden layer, where most computations are made. The data is propagated forward through the network, and when it reaches the output layer, the network predicts if the text's sentiment is positive or not. In a training phase, the network has access to the desired output. If the predicted output does not match the desired one, the model will calculate the magnitude error to perform back-propagation to change the neuron's weights.

2.1.2 Transformer Models

The Transformer [101] is a neural network architecture, and it is entirely based on attention mechanisms.

To better understand transformers, the concept of attention should first be understood. The weighted sum of the vector representations of a layer, also known as attention [8], can be used to provide interpretability to a neural model. This attention mechanism allows for the model to have long-term memory, which can be used by the model to " look" at the previous tokens.

First, the query key and value vectors for each word embedding must be calculated. These three vectors are calculated by multiplying the embedding by three matrices that were trained during the training stage.

Then the attention mechanism performs the dot product between the specific query and each key vector to calculate a score value. The scores are then passed through a softmax algorithm to generate a set of weights:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.3}$$

where Q is the query vector, K the key vector, V the value vector and d_k the dimensions of the queries and keys. Afterwards, it calculates the attention value as the sum of the weighted value vectors to retain the more relevant words for the specific query. Figure 2.2 represents an example of an attention mechanism.

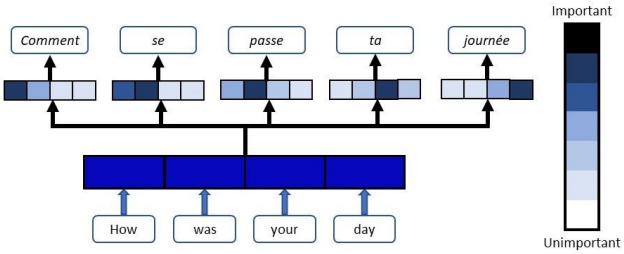


Figure 2.2: Example of attention mechanism. In this translation example, the more relevant words are mapped and assigned weights which are represented as colors. The darker the color, the more relevant the word is. [(2022, 7 may 2022). Retrieved from https://blog.floydhub.com/attention-mechanism/]

The transformer is a type of neural network architecture, represented in Figure 2.3, and it is composed of an encoder and a decoder.

To transform the input and output tokens to vectors of the same dimension as the model, two embedding layers (input embedding and output embedding) are included in the model. To provide information about the token's position in the sequence, so that the model benefits from the order sequence, positional encodings are supplied after the embeddings. These positional encodings have the same dimension as the model and are calculated with sine and cosine functions:

$$P(pos,2i) = \sin(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{2.4}$$

$$P(pos,2i+1) = \cos(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{2.5}$$

where d is the model's dimension, i the indices of each of the positional embedding dimension and pos is the position.

The encoder comprises six identical layers, each with two sub-layers: multi-head attention and feed-forward network. Vaswani et. al employed a residual connection [40] followed by layer normalization [7] around each of the sub-layers [101]. Residual connections are a type of shortcut connection, connections skipping one or more layers, that learn residual functions concerning the

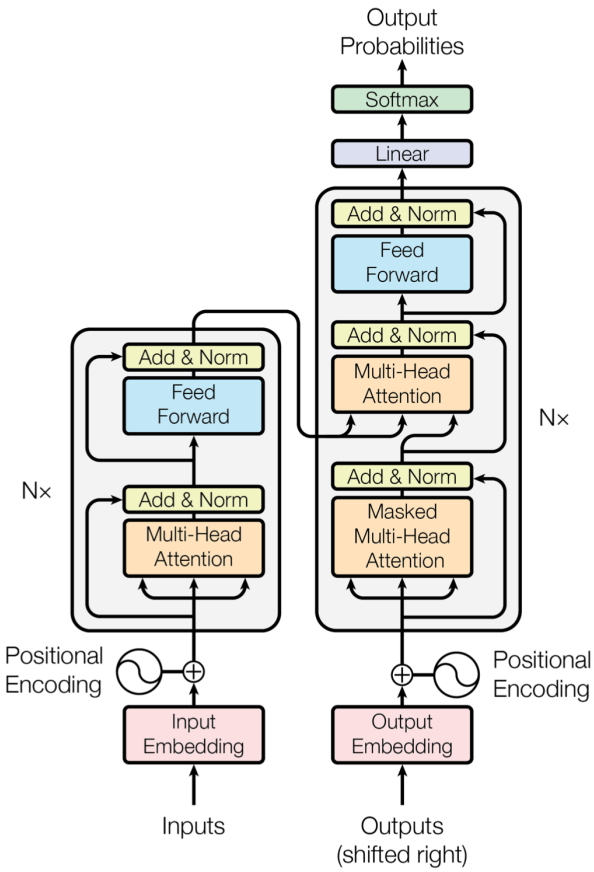


Figure 2.3: The Transformer - model architecture. [Vaswani et al. [101]]

layer inputs instead of unreferenced functions. The stacked nonlinear layers fit another mapping of $F(x) := H(x) - x$, where $H(x)$ is the desired underlying mapping and $F(x) + x$ is the transformation of the original mapping, see Figure 2.4. The intuition is that optimizing the residual mapping is simpler than optimizing the original, unreferenced mapping. With layer normalization, the mean and variance used for normalization are directly estimated from the accumulated inputs to the neurons inside a hidden layer; therefore, the normalization does not add any new dependencies between training examples:

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \tag{2.6}$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l \, \mu^l)^2} \tag{2.7}$$

where H is the number of hidden units in a layer, a^l the vector representation of the totalled inputs to the neurons in that layer and σ^l and μ^l the layer normalization statistics.

The queries, keys, and values are linearly projected N times by the multi-head attention, see in Figure 2.5. These queries, keys, and values were previously computed for each word by mul-

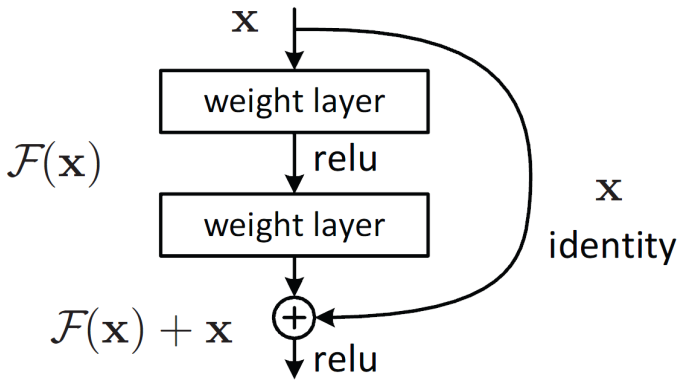


Figure 2.4: Residual learning [40].

tiplying the embedding by three matrices formed throughout the training procedure. The linear projection is conducted with the help of separate learned linear projections employed to carry out the attention function. The outputs of the attention functions are then concatenated and linearly projected to yield the final attention results.

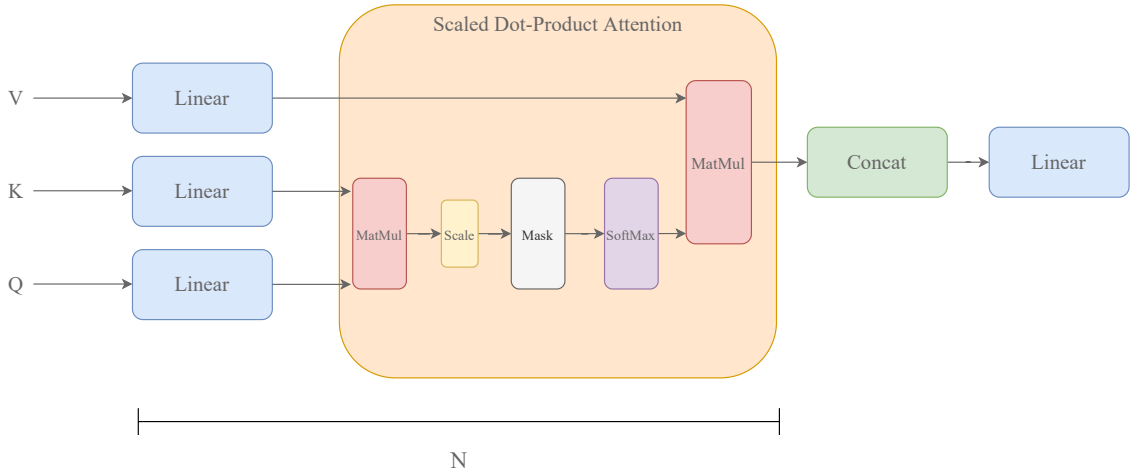


Figure 2.5: Multi-head Attention with N attention layers [101].

The feed-forward network is position-wise and fully connected, and it is applied to each position, consisting of a linear transformation, a ReLU activation, and another linear transformation. The following equation represents this feed-forward network :

$$FFN(x) = max(0, xW1 + b1)W2 + b2 \tag{2.8}$$

The decoder, like the encoder, is composed of six identical layers. Each layer contains the same two sub-layers that the encoder does, with residual connections and layer normalization, and an extra sub-layer which performs multi-head attention to the encoder output. The masked

multi-head attention is a multi-head attention sub-layer modified to ensure that the predictions for a position p are not based on information at positions higher than p.

Finally, after the decoder, the model converts the decoder output to estimated next-token probabilities via a linear transformation followed by a softmax function.

2.1.3 Dual Encoders and Cross Encoders

This subsection compares dual and cross encoders to analyze each approach's process to produce a similarity score between sentences. As the name suggests, dual encoders are composed of two encoders that produce a sentence embedding for each sentence in a pair of sentences. Each encoder receives a sentence as an input and these sentences will be embedded to be compared using cosine similarity. When it comes to cross encoder, both sentences are fed into an encoder, which produces an output from zero to one representing their similarity. It's imperative to mention that cross encoders don't create a sentence embedding in contrast with dual encoders.

An example of dual encoders can be seen in Figure 2.6.

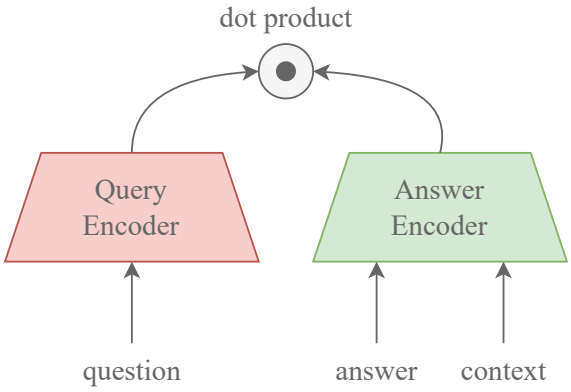


Figure 2.6: Example of a dual encoder usage. In this work [1], the dual encoder structure is being used, so that question and answer text are encoded independently.

Both have different uses. For example, the cross encoder can be used for sentence-pair scoring and dual encoders when a sentence embedding is needed.

Figure 2.7 depicts the structure of a cross-encoder model. The Cross-encoder executes self-attention between the context and the candidate using a single transformer, resulting in a richer extraction mechanism than the dual encoder. The context and candidate are enclosed by the special token [S] and concatenated into a single vector encoded by a single transformer. The transformer's first output determines the context-candidate embedding:

$$y_{ctx,cand} = first(T(ctx, cand)) \tag{2.9}$$

where ctx is the context, $cand$ the candidate and $T(x)$ is the output of a transformer. Another advantage of the dual encoder is that because the candidate label can attend to the input context

during the transformer layers, the Cross-encoder can give a candidate-sensitive input representation [47].

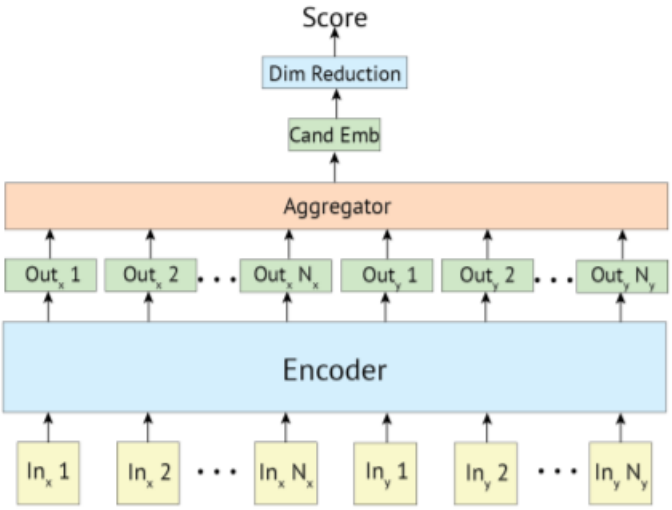


Figure 2.7: Example of cross-encoder structure [47]. The Cross-encoder combines context and candidate encoding resulting in richer context-candidate interactions at the cost of slower computation.

In Chidambaram et al. research, dual encoders for cross-lingual multi-task were found to achieve strong performance results in language and cross-lingual transfer learning [19]. The first encoder consists of a sentence encoder that passes sentence embeddings to the transformer encoder after embedding words and characters.

Even though dual and cross encoders are different from each other, it does not mean that they can't be used together.

Wu et al. developed a BERT-based entity linking model using dual-encoders and cross-encoders [107]. From a very high-level perspective, context and entities are inputted in the bi-encoder, then encoded into dense vectors. After this process, the output from the bi-encoder is passed to the cross-encoder for ranking. In one transformer, the cross-encoder encodes context and entity, then applies an additional linear layer to compute the final score for each pair.

2.2 Text Representation Methods

Text representation is fundamental for Information Retrieval. By text representation it is understood the mathematical representation of unstructured data, so that the computer can process it.

Although, text representation is extremely important in Natural Language Processing, which aim to analyze language, the machine learning architectures used for this purpose cannot understand the text as humans do. To feed some knowledge to these machine learning architectures, the

text that humans can read needs to be converted to vectors of numerical values, thus representing information to the models that can then perform tasks like classification or regression.

Many techniques have been proposed to represent text [108]; some examples are Bag Of Words (BoW) [37], TF-IDF Vectorization, and word embeddings models such as Word2Vec [97], GloVe [80], fastText [11], and BERT [23].

BOW focuses on capturing the term frequency of a word in a document. In this model, the text is represented as a multiset of its words (bag), and it is widely used in document classification, where the frequency of each word is used as a feature, to find relevant documents to the user's query [73].

TF-IDF Vectorization is a technique based on the BoW model. It considers the importance of a word in a document, allowing TF-IDF to perform better at finding relevant documents than BoW. Nevertheless, unlike Word2Vec, it cannot capture the meaning of the words.

Word2Vec uses a neural network on a vast text corpus to learn word associations. Two new model architectures were proposed [74] to minimize computational complexity for learning distributed representations of words. These architectures are CBoW and continuous Skip-Gram.

Continuous Bag-of-words (CBoW) is similar to a neural net language model (NNLM) but removes the non-linear hidden layer. In CBoW, all the words share the projection layer, and this model predicts the current word based on the context from a window of history and future words at the input.

The continuous Skip-Gram model is similar to CBoW. However, instead of using context to predict the current word, it uses each current word to predict the surrounding words. The difference between both models can be visualized in Figure 2.8.

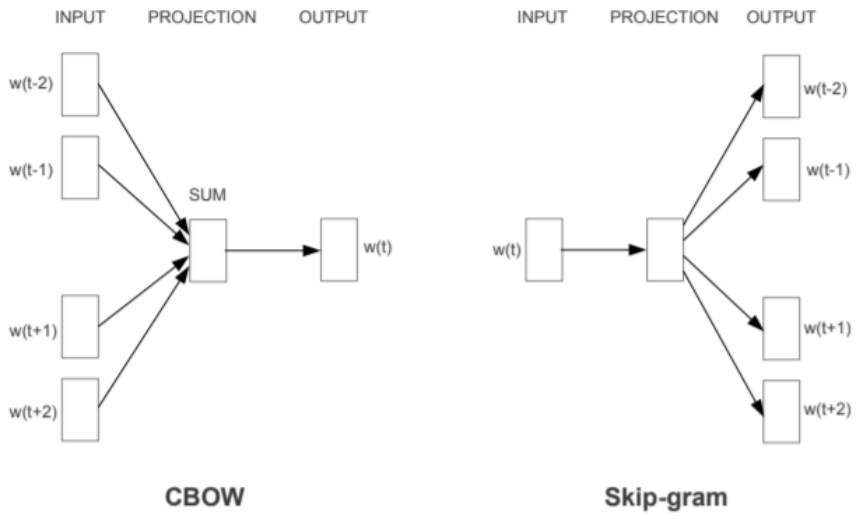


Figure 2.8: CBoW vs Skip-Gram. The Skip-gram predicts adjacent words based on the current word, whereas the CBoW architecture predicts the current word based on context [74].

Mikolov et al. later concluded that the CBoW model is better than Skip-gram on syntactic tasks but worse on semantic ones [74].

2.2.1 Glove

GloVe can be used to find relations between words, like synonyms, and consistently outperforms word2vec, achieving better results faster [80].

This technique produces word vectors for each document's word, and its central intuition is to observe that ratios of word-to-word co-occurrence probabilities can encode some meaning. To accomplish so, a vocabulary function will store and calculate each often-occurring word. Then a matrix will be formed using the word-to-word co-occurrence approach to measure the frequency of occurrence of each pair of words. Following the creation of this matrix, GloVe uses it to train the dataset, creating a word vector for each word that includes the frequencies weight for each word that occurs with other words [76].

Pennington et al. created a co-occurrence probabilities table, see Table 2.1, where it helps to visualize the Probability and Ratio for target words "ice" and "steam" with various probe words. This example was developed on a 6 billion word corpus [80].

Table 2.1: Co-occurrence probabilities table for target words *ice* and *steam* available at [80]

Probability and Ratio	k = *solid*	k = *gas*	k = *water*	k = *fashion*
P(k|*ice*)	1.9×10^{-4}	$6.6 \times 10{-}5$	3.0×10^{-3}	1.7×10^{-5}
P(k|*steam*)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(k|*ice*)/P(k|*steam*)	8.9	8.5×10^{-2}	1.36	0.96

The authors confirmed that the word ice co-occurs more often with the word solid than with gas. As for the word steam, it is the opposite; it co-occurs more with the word gas than with the word solid. Another observation that was made is that the ratio of probabilities encodes some meaning. This meaning is supported by the value of the ratio of probabilities; much greater values than 1 correlate well with properties specific to ice, and much smaller values than 1 correlate well with properties specific to steam.

2.2.2 FastText

FastText is an open-source library developed by Facebook AI Research lab and is used to build scalable solutions for text representation and classification [11] [51].

Sentence representation is done using bag of words and bag of n-grams – the usage of n-grams was necessary since bag of words is invariant to word order, and bag of n-grams captures partial information about the local word order – and subword information, linguistic units that are smaller than a word. Instead of representing a word, it breaks the word down to character n-grams and learns embeddings for them. This approach solves a problem that many NLP models have when a model has a word in a test set that was not in the train set, and the model can not create a vector

representation for the word. Since fastText uses n-grams, it can represent these words that did not exist in the train set [72].

When the number of classes is large, computing a linear classifier is computationally expensive. Therefore, Joulin et al. [51] decided to implement a hierarchical softmax [34] to improve the running time. In this structure, each node has a probability associated, representing the probability of the path from the root to that node, which means that the probability of a child node is always smaller than the probability of the parent. The use of this classifier vastly reduces the time complexity of model training.

2.2.3 BERT

BERT is a multi-layer bidirectional structure based on a Transformer encoder, meaning that it can learn context from both left and right sides of an input embedding [23]. It achieved state-of-the-art performance, being used for many problems such as Neural Machine Translation, Question Answering, Sentiment Analysis, Text Summarization, and several classification tasks.

The BERT input, as shown in Figure 2.9, is represented by the sum of the token embeddings, segment embeddings, that store information about the sentence in which the token is inserted, and position embeddings complement the token embedding providing information about the position of the token in the sentence. This sum allows the model to have a sense of order; in other words, the model knows the position of a word in the sentence.

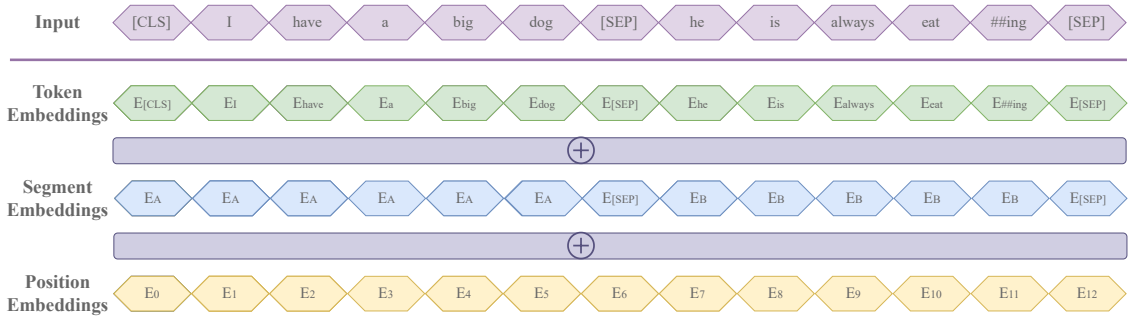


Figure 2.9: BERT input representation.

There are two different steps in this framework: pre-training and fine-tuning, see Figure 2.10. In pre-training, the model makes use of unlabeled data to train through two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

MLM consists in masking, assigning a MASK token, to 15% of the input words, after that the model runs this already masked sentence and tries to predict the masked words.

NSP is a more straightforward task that focuses on understanding sentences relations. Two sentences are given to the model and the model has to predict if they are followed by each other or not.

The fine-tunning task, unlike pre-training, uses already labeled data from the target task. The authors decided to fine-tune BERT using a self-attention mechanism. This approach joins two

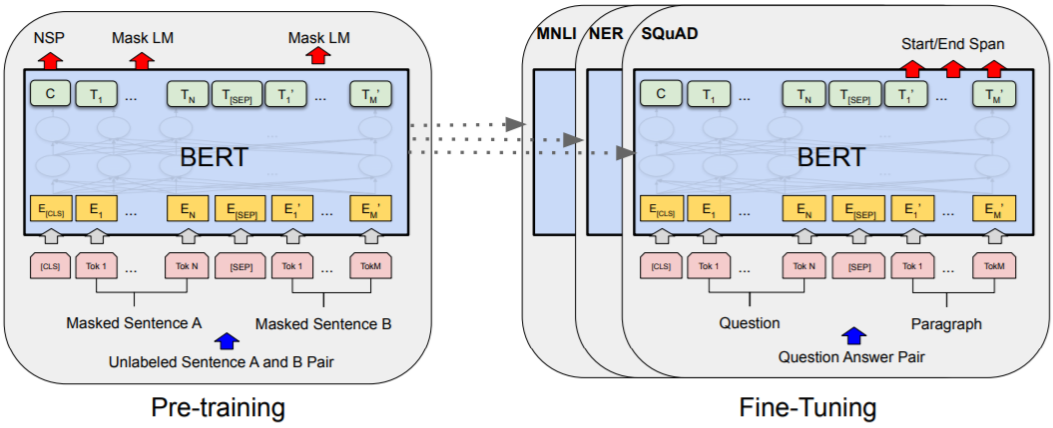


Figure 2.10: Overall pre-training and fine-tuning procedures for BERT [23]

stages: encoding text pairs and bidirectional cross attention. Merging these two stages includes bidirectional cross attention between two sentences. This model represents a considerable contribution to the NLP community since the same pre-trained bidirectional model can be used to solve many different NLP problems.

2.3 Information Retrieval

Information retrieval is the process of searching for documents, information inside a document, or even searching for other types of data, like images, audios, and videos.

Over the past decades, with the increasing amounts of data being generated, it became necessary to have easy access to organized information. This prompted the creation of IR.

IR is present in our daily lives even though we might not realize it. For example, when using web search engines, we are accessing IR systems, which when given a query, assist a user to have access to specific information. Still, it does not provide a precise answer; it provides information about the documents that might contain the required information.

An Information Retrieval Model can be defined as a set of algorithms used to rank and retrieve documents, sorted by their relevance score given the user's query. Examples of models for this process are Boolean Models [95], Vector Space Models, and Probabilistic Models [6] [43].

Boolean Models were one of the first to be used in the IR field and are still widely used. Even though they do not rank the retrieved documents by their relevance, they sort the documents by some order, for example, by date. To use these systems, the users needed to specify the information they wanted using a complex boolean (NOT; OR; AND) combination. Therefore, it is tough for a user to form a good query.

Vector Space Models are models for representing text as vectors of terms. These models analyze the similarity between the query vector and the document vector to rank a document for a

query (assigning a numeric score). Usually, to measure this similarity, it is used the cosine of the angle between the two vectors; this method is called Cosine Similarity and will be analyzed more in-depth later.

Probabilistic Models [31] aim at predicting the probability that a specific document d will be categorized as relevant or not given a query q; the probability can be represented by the following equation *P(R|q,d)*. Following the basic assumption that terms are distributed differently in relevant and non-relevant documents, it is only necessary to distinguish between documents with different sets of relevant terms, the probability becomes *P(R|q,x)*, where x is a binary vector that represents the set of terms occuring in a document, the value of the term's representation in the vector is 1 if the term is present in the document or 0 if it is absent. As a result, various documents having the same collection of terms will produce the same exact approximation of the possibility of relevance given a query.

In the context of probabilistic models, **Dirichlet** is widely used. The Dirichlet smoothing function avoids unrecognized terms having an estimated zero probability, which is accomplished by giving them small values derived from noticed terms. This model is document size-dependent since longer documents allow better probabilistic estimation. This model is given by the following equation [111]:

$$p_\mu(w|d) = \frac{c(w;d) + \mu p(w|C)}{\sum_w c(w;d) + \mu} \tag{2.10}$$

where w is the weight of a term, $c(w;d)$ the document term frequency, μ the smoothing parameters, $p(w|C)$ is the collection language model, and $p_\mu(w|d)$ is the dirichlet smoothed probability of a word seen in a document [3][111].

To provide the most relevant and accurate results, there must be a function capable of assigning a score to each outcome. It depends on the features selected to score, like similarity or word relevance in a document. Some examples of widespread scoring functions are **TF-IDF** [86], **BM25** [100], and to measure the similarity of vectors **Cosine Similarity** [83].

TF-IDF is a measure used to evaluate the relevance of a word in a document in a collection of documents. This is calculated using two different properties: **term frequency** and **inverse document frequency**.

Term frequency analyzes the importance of a word. In order to do so, it counts the instances of a word in a document. There is a problem with this analysis, there are many words that are not important for a document, but have a huge number of instances, for example: "this", "the", "is"...

Inverse document frequency was used as a solution for this, which calculates how rare or common a word in a document is by decreasing the weight value of words that appear in many documents and increasing the weight for words that are not used very often.

TF-IDF is the product of term-frequency and inverse document frequency and can be represented by the following equations [38]:

$$Y(t,d,D) = TF(t,d) \times IDF(t,D); \tag{2.11}$$

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}; \tag{2.12}$$

$$IDF(t,D) = \log(\frac{N}{|d \in D : t \in d|}), \tag{2.13}$$

where t is the term, d a document, D, the corpus and N the number of documents in the corpus.

BM25 is possibly one of the most used and essential functions in information retrieval and is used to estimate the relevance of documents to an input query. It is a non-linear combination of three document attributes: document frequency, document length, and term frequency. Traditionally, this function considered a document description restricted to two fields (body and title), but since using more fields could provide different relevance scores and increase the ranking accuracy, an extension of this function was created (BM25F) [100].

The following equation represents the BM25 score :

$$BM25 = \sum_{t \in q} TF_t \times I_t, \tag{2.14}$$

where TF_t is the term frequency already represented in the equation 2.12, and I_t [89] :

$$I_t = \log(\frac{N - df + 0.5}{df + 0.5}), \tag{2.15}$$

with df being the document frequency of term t, and N being the number of documents in the collection.

When it comes to performance, BM25 does well on single content fields. Still, it is outperformed by linear models when applied to single popularity fields (anchor text and query click information) [100].

Cosine similarity is used to measure similarity between two texts. It measures the cosine of the angle between two vectors of an inner product space. This measurement will analyze if the vectors are pointing in the same direction, which means that the closer the cosine value to 1, the greater the match between two vectors and the smaller their angle. If the value is 0, the vectors' angle is 90 degrees, having no matches between them. The cosine similarity between two vectors x and y can be calculated using the following equation:

$$similarity(x,y) = \frac{x \cdot y}{\|x\|\|y\|}. \tag{2.16}$$

2.4 Argumentation and Natural Language Processing

Arguments are used every day by almost every individual. When someone needs to take a stance, contradict someone, or persuade, that person needs to have good arguments.

Extensive research was done to help qualify and explain the process of argumentation. However, there will not be a comprehensive analysis of theoretical argumentation models since our focus is more superficial.

The data provided for this task contain many arguments. Each argument is split into sentences that represent either a conclusion or a premise. Each argument contains more information, such as the topic discussed, the stance of the argument, the source text, and the location of the argument inside the source text.

Args.me corpus has a large collection of data containing 387 740 arguments, which will be considered, evaluated, ranked, and retrieved by a model developed for the Touché Task 1. These arguments must be helpful for users during a conversation, get overviews of pros and cons, or help the user defend its stance.

Argument evaluation can be done by considering its convincingness. Generally, assigning a convincingness score to an argument is subjective since it may lack some context or might conflict with the reader's beliefs.

Habernal and Gurevych offered a novel task of predicting the persuasiveness of arguments in a pair and ranking arguments linked to a particular topic [36]. It was also mentioned by Habernal and Gurevych that a single argument's convincingness ranking would be difficult to perform since it could be affected by the annotator's bias. The authors introduced the pairwise comparison idea, where an argument is considered more convincing than the other, or they might be equally convincing.

2.5 Summary

In this Chapter, essential architectures for NLP were covered, such as Transformers and BERT. We also approached different text representation techniques, widely used in IR approaches, such as BoW, TF-IDF Vectorization, Word2Vec. It was crucial to cover these architectures since they are the foundation for the approaches described in the next Chapter.

Chapter 3

State of the Art

This chapter is divided into three sections. The first one describes proposed methods to approach sparse retrieval, while the second one presents models developed for dense retrieval. The last one describes the approaches for both the 2021 and the 2022 editions of the Touché Task 1.

3.1 Sparse Retrieval

Sparse retrieval is a technique that uses sparse representations of documents without needing to employ neural networks, e.g. BM25 ranking BoW representations.

3.1.1 LambdaBM25

LambdaBM25 [100] is a data-driven machine learning model based on the attributes of BM25, already explained in Section 2.3, training method of LambdaRank [17], and evaluation measure Normalized Discounted Cumulative Gain (NDCG) [48].

LambdaRank is a neural network learning technique based on lists and pairs. It is a RankNet extension that learns from pairs of documents per query, each with its own set of relevance labels [16]. The primary distinction between these two approaches is that LambdaRank focuses on the gradients of costs rather than on cost functions like RankNet; consequently, LambdaRank achieves a considerable speedup over RankNet.

Because it is multilayer and the truncation level can be selected to reflect how many documents are given to the user, NDCG is especially well suited to Web search applications. Burges et al. defined NDCG's formula as follows:

$$NDCG_i = N_i \sum_{j=1}^{L} \frac{2^{r(j)-1}}{\log(1+j)} \tag{3.1}$$

For LambdaBM25 project it was considered a relevance scale with 5 levels, therefore r(j) belongs 0,...,4, where r(j) is the relevance of the document at rank position j. L is the truncation level to which this normalized gain is computed, and the constant of normalization Ni was chosen so that a perfect ordering would yield $NDCG_i$ = 1.

LambdaBM25 focuses on NDGC λ-gradients. These gradients are used to organize the pair of items before starting the list's final order, and they can be thought of as document indications that indicate where they should move, where a positive lambda suggests a push towards the top and a negative suggests a push towards the bottom of the list. Burges et al. draw an analogy between documents returned for a given query and points of mass. As a result, λ_j corresponds to a force in the point of mass j M_j, implying that the sums of λ must be 0 to prevent the system from accelerating. To put it another way, if λ_A is computed based on its location concerning B, then λ_B should increase by an amount equal to and opposite to λ_A to avoid this pair from accelerating.

3.1.2 SparTerm

SpartTerm is a framework for learning sparse representations of terms throughout the whole lexicon [9]. This system develops a function to transfer the frequency-based BoW representation to a sparse term importance distribution using pre-trained language models (PLM). This allows one to use term-weighting and expansion in the same framework (see Figure 3.1).

Query	Can hives be a sign of pregnancy?	
Type	Term frequency	SparTerm
Literal term Weights	hives are caused by allergic reactions . the dryness and stretching of your skin along with other changes can make you more susceptible to experiencing hives during pregnancy . hives can be caused by an allergic reaction to almost anything . some common causes of hives during pregnancy are noted below : medicine	hives are caused by allergic reactions . the dryness and stretching of your skin along with other changes can make you more susceptible to experiencing hives during pregnancy . hives can be caused by an allergic reaction to almost anything . some common causes of hives during pregnancy are noted below : medicine
Term expansion		symptoms:1.0, women:0.99, rash:0.98, feel:0.99, causing:0.97, body:0.96, affect:0.96, baby:0.94, pregnant:0.93, **sign:0.91**, …

Figure 3.1: Comparison between BoW and SparTerm representations. This figure indicates the weight difference between the semantically important phrases. It demonstrates that the relevant weights are more profound with SparTerm and can even expand phrases not occurring in the passage [9].

SpartTerm consists of two modules (see Figure 3.2) with the same architecture as BERT: the importance predictor and the gating controller.

By displaying the semantic relevance of all terms in the vocabulary, the importance predictor integrates term weighting and expansion. This module produces a dense semantic importance distribution Ix by employing a BERT-based encoder to acquire deep embeddings with context for

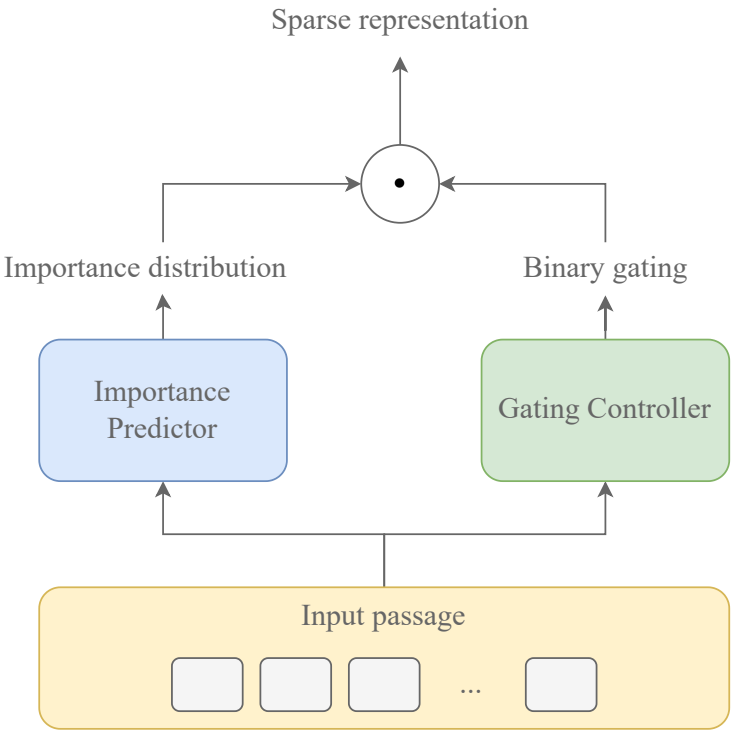


Figure 3.2: Representation of SparTerm architecture.

each word wx in passage p. Each h represents the surrounding context from a position x, offering a semantic analysis of which terms are linked to the current passage's topic :

$$I_x = Transform(h_x)E^T + b \tag{3.2}$$

where E is the shared word embedding, b the bias and $Transform$ is a linear transformation. The final importance prediction module can be obtained by simply adding all the token-wise importance distributions:

$$I = \sum_{x=0}^{L} Relu(I_x) \tag{3.3}$$

where L is the sequence length of the passage, and $Relu$ is an activation function to ensure that the importance logits do not have negative values.

The gating controller generates a binary signal describing the words to activate, to indicate the input passage. For this objective, two distinct gating controllers have been proposed: literal-only gating and expansion-enhanced gating. In literal-only gating, only terms from the original text are considered to activate in the passage representation $G(p) = BoW(p)$. On the other hand, the fundamental goal of the expansion-enhanced controller is to deal with a lexical mismatch; to do so, it generates a passage-wise dense word gating distribution G, which represents the probability of the term being in the final sparse representation.

To ensure the sparsity of G, a binary activation function *Binarizer* is applied :

$$G' = Binarizer(G) \tag{3.4}$$

With the binarized distribution G', the gating vector for expansion terms G_e is calculated:

$$G_e = G' \odot (\neg BoW(p)) \tag{3.5}$$

where $(\neg BoW(p))$ is used to ensure orthogonal to the literal-only gating. At last, by adding the expansion gatting and the literal-only gating, the final expansion-enhanced gating vector is obtained :

$$G_{le} = G_e + BoW(p) \tag{3.6}$$

3.1.3 SPLADE v2

The introduction of large pre-built language models such as BERT has re-shaped NLP and IR since these large models have shown a potent capacity to adapt to multiple tasks by making slight modifications. However, even as BOW models are still good starting points, they struggle with the long-standing issue of vocabulary mismatch, in which relevant documents may not include terms that emerge in the query. As a result, learned (neural) rankers have tried to replace these conventional approaches. Still, creating such large models presents numerous difficulties regarding efficiency and scalability. Thus, procedures in which most of the data processing could be done offline and online inference is fast are actually needed.

Given the lack of exact term matching, dense retrieval using approximated closest neighbors search has yielded excellent results while still benefiting from BoW models. As a result, there is an increasing interest in studying sparse representations for queries and documents. Thereby, models can possess desirable characteristics of BoW models such as exact-match of terms, inverted index efficiency, and interpretability. Furthermore, similar to usual expansion models in IR, these models can decrease vocabulary mismatch by modeling implicit or explicit (latent, contextualized) expansion processes.

SPLADE v2 [29], offered various enhancements to the SPLADE model [30], which is a model that learns both expansion and compression in an end-to-end technique, and these enhancements contribute to increased effectiveness or efficiency:

- Formal et al. improved efficacy by reconfiguring the SPLADE pooling mechanism, using a max pooling operation, SPLADE-max [29]:

$$w_j = max_{i \in t} log(1 + ReLU(w_{ij})) \tag{3.7}$$

where w_{ij} is the importance of a vocabulary token j for a input token i :

$$w_{ij} = transform(h_i)^T E_j + b_j, j \in \{1, ..., |V|\} \tag{3.8}$$

where $|V|$ is the vocabulary size, 30522, h_i is the input query or document sequence BERT embedding, E_j is the input embedding for token j, and b_j is a token-level bias;

- They explored a model expansion without query expansion, using a document version of SPLADE exclusively, SPLADE-doc. This model is more efficient because it relies on document term weights, permitting everything to be pre-calculated and indexed offline while still achieving competitive results :

$$s(q,d) = \sum_{j \in q} w_j^d \tag{3.9}$$

- The authors improved the efficiency of SPLADE using the distillation approach [45], obtaining nearly state-of-the-art results as well as the Benchmarking IR (BEIR) zero-shot evaluation.

As a result of these SPLADE adjustments, the BEIR benchmark improves by more than 9% compared to the leading state-of-the-art results.

3.2 Dense Retrieval

Dense retrieval calculates document relevance using dense representations. It depends on neural networks and focuses on performing a search on transformer-encoded representations.

3.2.1 STAR and ADORE

For IR, document rating is critical. Nevertheless, traditional approaches for this task have several flaws that must be addressed. The traditional model's approach would fail if the query and the document utilized different terms with the same meaning, for example, due to a vocabulary mismatch.

With recent advances in deep learning, several approaches to solving the semantic matching problem now focus on Dense Retrieval (DR) models. To better abstract the semantic meanings of queries and documents, these models aim to encode them into low-dimension embeddings.

To improve the existing strategies' performance, Zhan et al. suggest two techniques to train DR models effectively: Stable Training Algorithm for dense Retrieval (STAR) and Algorithm for Directly Optimizing Ranking pErformance (ADORE) [112].

STAR focuses on stabilizing the static hard negative sampling approach, optimizing ranking performance with static hard negatives (top documents retrieved by a warm-up model for the training queries), and stabilizing training with random negatives. It leverages a warm-up model to get the top documents for training queries (static hard negatives). This technique reuses other

document embeddings in the same batch as random negatives to improve model efficiency rather than introducing random negative documents to the input.

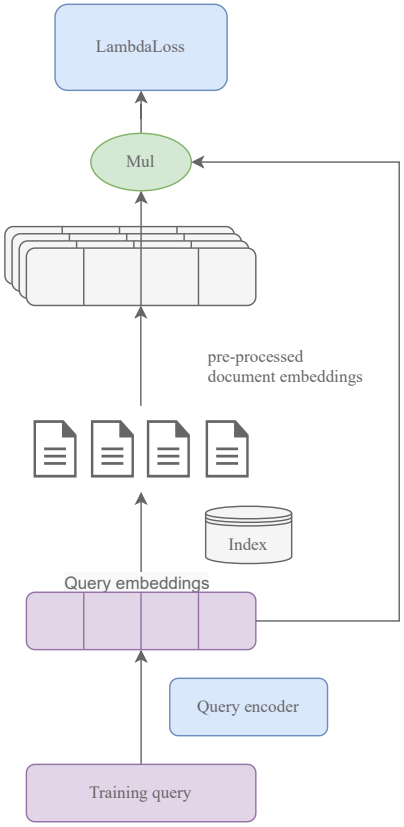


Figure 3.3: Structure of the proposed ADORE method [112].

ADORE (see Figure 3.3), is a method for training DR models with dynamic hard negative samples uses the document encoder trained by other approaches and trains the query encoder to enhance IR metrics. ADORE pre-processes the document embeddings with a pre-trained document encoder before the training stage, resulting in the document index. It encodes a series of queries and utilizes them to retrieve the top documents at each step of the training phase (dynamic hard negatives). ADORE optimizes ranking performance by retrieving at each step, as explained in [65]; for the same reason, the LambdaLoss [18]) weight derivation is used.

Zhan et al. concluded that the two proposed strategies improved performance and efficiency significantly, and that their combination provided the best retrieval performance [112].

3.2.2 ColBERT

Recently, the Information Retrieval field has seen the introduction of several neural classification models. In contrast to previous approaches that relied on hand-scraping characteristics, these models employ representations based on embedding queries and documents and directly simulate local interactions among its content material. Among them is a promising innovation that

fine-tunes deep pre-trained language models (LMs) such as BERT for estimating importance and resolving vocabulary inconsistencies between documents and queries. However, the noticeable benefits provided by these LMs come at a substantially higher computational cost. Thus, according to Hofstaer et al. [46] and MacAvaney et al. [69] BERT-based models are 100-1000 times more expensive than prior ones.

In an effort to reconcile performance with contextual relevance in IR, Khattab et al. presented ColBERT [54], a categorization model based on contextualized late interaction over BERT. ColBERT, as the name suggests, is a late interaction paradigm for estimating the relevance of a query q and a document d. Late interaction encodes q and d independently in two components of contextual embeddings, and the relevance is measured between both combinations using low-cost and pruning-friendly calculations.

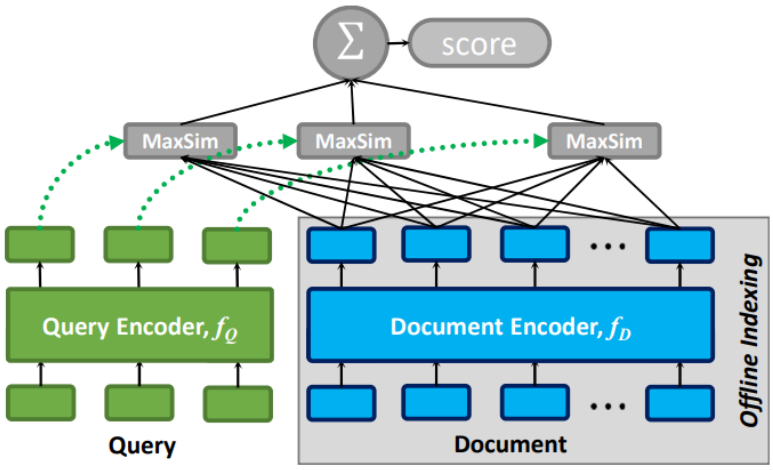


Figure 3.4: General architecture of ColBERT given a query and a document [54].

Figure 3.4 depicts the general design of ColBERT, which contains a question encoder, a document encoder, and a late interaction mechanism. Given a query q and a document d, the query encoder encrypts q into a bag of fixed-size embeddings E_q, while the document encoder integrates d into a separate bag E_d. Importantly, each embedding in E_q and E_d is integrated depending on the other terms in q or d. Furthermore, the model employing E_q and E_d computes the relevance score between q and d via the late interaction, which is calculated as a sum of maximum similarity operators – MaxSim (e.g., cosine similarity).

The following equation gives this similarity score:

$$S_{q,d} = \sum_{i=1}^{n} max_{j=1}^{m} E_{q_i} \cdot E_{d_j}^{T} \tag{3.10}$$

where Eq with length n and Ed with length m are the final vector sequences derived from q and d [53].

Provided these term scores, ColBERT calculates the document's relevance by adding the matching proof for all query terms. Furthermore, the results demonstrate that ColBERT seems to be more than 170 times faster and necessitates 14,000 fewer FLOPs per query than established BERT-based designs (see Table 3.1), all while affecting effectiveness minimally and outperforming each non-BERT baseline (see Table 3.2). As a result, ColBERT proposes a simple structure for reconciling the performance and cost of neural IR. Its late interaction approach is not limited to BERT, allowing it to be used on various architectures.

Table 3.1: Re-ranking results on MS MARCO. The re-ranking is perfomed on the top-1000 results produced by BM25. Table taken from [54].

Method	**MRR@10** (Dev)	**MRR@10** (Eval)	**Re-ranking latency** (ms)	**Flops/query**
BM25 (official)	16.7	16.5	-	-
KNRM	19.8	19.8	3	$592M(0.085\times)$
Duet	24.3	24.5	22	$159B(23\times)$
fastText+ConvKNRM	29.0	27.7	28	$78B(11\times)$
$BERT_{base}$	34.7	-	10,700	$97T(13,900\times)$
$BERT_{base}$ (The authors training)	26.0	-	10,700	$97T(13,900\times)$
$BERT_{large}$	36.5	35.9	32,900	$340T(48,600\times)$
ColBERT (over $BERT_{base}$)	34.9	34.9	61	$7B(1\times)$

Table 3.2: End to end retrieval on MS MARCO; each model retrieves top-1000. Acquired from [54].

Method	**MRR@10** (Dev)	**MRR@10** (Local Eval)	**Latency** (ms)	**Recall** @50	**Recall** @200	**Recall** @1000
BM25(official)	16.7	-	-	-	-	81.4
BM25(Anserini)	18.7	19.5	62	59.2	73.8	85.7
doc2query	21.5	22.8	85	64.4	77.9	89.1
DeepCT	24.3	-	62	69	82	91
docTTTTTquery	27.7	28.4	87	75.6	86.9	94.7
$ColBERT_{L2}(re-rank)$	34.8	36.4	-	75.3	80.5	81.4
$ColBERT_{L2}(end-to-end)$	36.0	36.7	458	82.9	92.3	96.8

3.2.3 DistilBERT

Transfer Learning approaches in Natural Language Processing have greatly increased in recent years. Nonetheless, despite significant advancements, these models have become more complex. As a result, this preference for larger models raises several concerns, including:

- The environmental cost of massively scaling these models' computational needs;
- The increasing computational and memory requirements of these models may limit their widespread implementation.

To take advantage of the larger models while avoiding the drawbacks associated with them, the method of knowledge distillation emerged. This is a compression technique where a small model - the student - is instructed to mimic the behavior of a larger model - the teacher - or a set of models [15] [44]. Thus, based on this concept, Sanh et al. [96] demonstrated that using much smaller linguistic models pre-formed with knowledge distillation, it is possible to achieve similar or even higher results than these larger models, resulting in lighter and faster models that require less expensive computational training. As a result, these smaller models retain the plasticity/versatility of larger models while remaining small enough to operate on mobile platforms, for example. Therefore, based on this approach DistilBERT (distilled version of BERT) was born.

DistilBERT is a small model that shares the same basic architecture as BERT. However, the token-type embeddings and pooler were removed, and the number of layers was reduced by a factor of 2. In addition, this smaller model preserves 97% of the language comprehension skills while being 40% smaller and 60% faster than the large model that gave rise to it, which can be observed in Table 3.3.

Table 3.3: DistilBERT and BERT performance, accuracy, size and speed comparison [96].

Model	**Score**	**IMDb Accuracy**	**params** $(\times 10^6)$	**Inf. Time** (s)
BERT-base	79.5	93.46	110	668
DistilBERT	77.0	92.82	66	410

3.2.4 MiniLMv2

Transformers that have been pre-trained have proven to be quite successful for a wide range of natural language operations. These models, however, often have a large number of attributes and are becoming increasingly sophisticated. Due to computational resource constraints and latency, these increasingly sophisticated models present challenges for fine-tuning and online serving in real-world applications.

The findings of published work [96] [105] [99] [49] use self-attention distributions to undertake knowledge distillation training, which has the disadvantage of requiring the student model to have the same number of attention heads as the teacher.

MiniLMv2 provides deep self-attention distillation, with the student model being led by self-attention distributions and value relations to mirror the teacher's self-attention configurations closely [105]. As a result, MiniLM reveals that transferring the teacher's last layer trumps layer-wise distillation.

Furthermore, using self-attention relation distillation, successfully generalized and completely simplified MiniLM, resulting in MiniLM v2 [104]. This model differs from earlier techniques in that it employs multi-head self-attention relations of pairs of queries, keys, and values to overcome the limitation on the number of students' model attention heads. The scaled dot-product of the pairs of queries, keys, and values of multiple relation heads provides the concept of multi-head self-attention values relations, as shown in Figure 3.5.

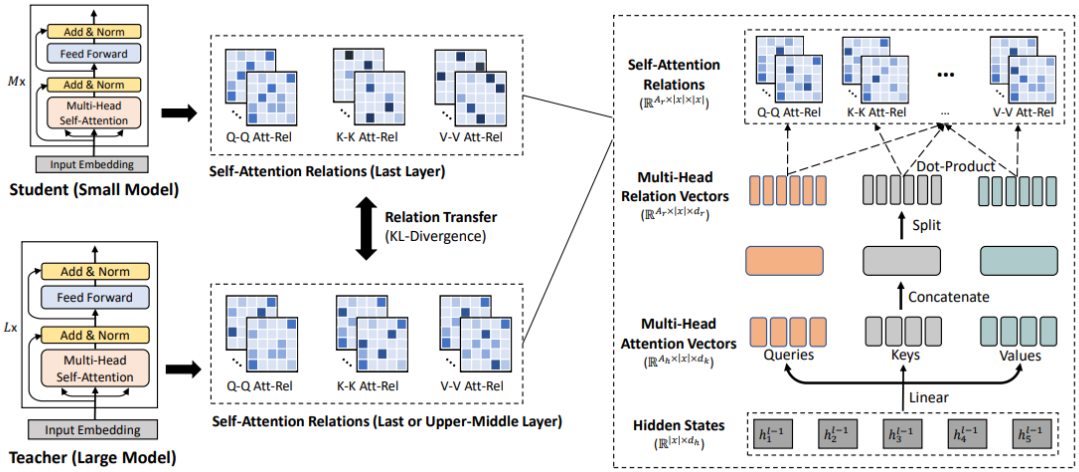


Figure 3.5: Representation of MiniLM v2 multi-head self-attention relations concept. First, the concatenation of self-attention vectors of distinct attention heads is performed to obtain the queries, keys, and values vectors of numerous connection heads. It then splits them according to the number of relation heads required. The authors chose to convey Q-Q, K-K, and V-V self-attention relations to establish an equilibrium between performance and training speed. Image taken from [104].

Moreover, the authors conducted thorough distillation experiments on a diverse group of large-sized teachers and discovered that harnessing a teacher's upper-middle layer knowledge resulted in better outcomes. The results of the experiments also demonstrated that the strategy was adequate for a wide range of monolingual and multilingual teachers in various groups, outperforming the present state of the art.

Finally, as indicated in Table 3.4, the authors conducted studies to increase the number of self-attention relationships. Even if the model's performance improves, adding more self-attention interactions increases computational costs; thus, the authors chose to transfer the Q-Q, K-K, and V-V self-attention relations to establish a balance between performance and costs.

3.2.5 Sentence-BERT

Sentence-BERT (SBERT) [87] is a pre-trained BERT network modification that employs siamese and triplet network structures to generate semantically meaningful sentence embeddings that can be compared via cosine similarity. The effort required to locate the most matching pair is reduced from 65 hours with BERT or RoBERTa to 5 seconds with SBERT while preserving accuracy.

SBERT adds a pooling procedure to the BERT/RoBERTa output to provide a fixed-sized sentence embedding. Three pooling strategies were tested: Using the CLS-token output, compute the mean of all output vectors (MEAN), and compute the vectors' maximum-over-time (MAX). Three different objective functions were used to assess the model:

Table 3.4: Data outcome of introducing more self-attention relations [104], where ($L \times H$ MiniLMv2) is the model MiniLMv2 with L layers and H hidden size. The experiments were performed on two single-sentence classification tasks (SST-2) [98], one inference task (MNLI) [106], and SQuAD2 [84], considered a significant question-answering benchmark.

Model	Teacher	SQuAD2	MNLI-m	SST-2
6×384 MiniLMv2	$BERT_{BASE}$	72.9	**82.8**	91.3
6×384 MiniLMv2 + Att-Rels	$BERT_{BASE}$	**73.3**	**82.8**	**91.6**
6×384 MiniLMv2	$BERT_{LARGE}$	74.3	83.0	91.1
6×384 MiniLMv2 + Att-Rels	$BERT_{LARGE}$	**74.7**	**83.2**	**92.4**
6×384 MiniLMv2	$RoBERTa_{LARGE}$	**76.4**	**84.4**	92.0
6×384 MiniLMv2 + Att-Rels	$RoBERTa_{LARGE}$	76.0	**84.4**	**92.1**
6×768 MiniLMv2	$BERT_BASE$	76.3	84.2	**92.4**
6×768 MiniLMv2 + Att-Rels	$BERTa_BASE$	**76.8**	**84.4**	92.3
6×768 MiniLMv2	$BERT_{LARGE}$	77.7	85.0	**92.5**
6×768 MiniLMv2 + Att-Rels	$BERT_{LARGE}$	**78.1**	**85.2**	92.5
6×768 MiniLMv2	$RoBERTa_{LARGE}$	**81.6**	87.0	**94.5**
6×768 MiniLMv2 + Att-Rels	$RoBERTa_{LARGE}$	81.2	**87.3**	94.1

- Classification Objective Function: The output is supplied by the equation:

$$o = softmax(W_t(u, v, |u - v|)), \quad W_t \in \mathbb{R}^{3n \times k} \tag{3.11}$$

  where u and v are sentence embeddings and $|u - v|$ is the element-wise difference. W_t is the trainable weight, n is the sentence embedding dimension, and k is the number of labels, as shown in Figure 3.6.

- Regression Objective function: Cosine similarity between two sentence embeddings, as shown in Figure 3.6.

- Triplet objective function: Triplet loss tunes the network for an anchor sentence a, a positive sentence p, and a negative sentence n to such an extent that the distance between the anchor and the positive sentence is smaller than the distance between the anchor and the negative sentences:

$$max(||s_a - s_p|| - ||s_a - s_n|| + \varepsilon, 0) \tag{3.12}$$

  where s_x is the sentence embedding for $x = aornorp$, $||s_a - s_p||$ is the distance between anchor sentence embeddings and positive sentence embeddings, and a margin ε which ensures that s_p is ε closer to s_a than s_n.

The authors stated that BERT is unsuitable for usage with common similarity measurements, and SBERT was designed to address this. SBERT can be utilized for computationally impossible tasks to represent using BERT. For example, computing 50 million sentence combinations with BERT would take 65 hours, but only 5 seconds with SBERT.

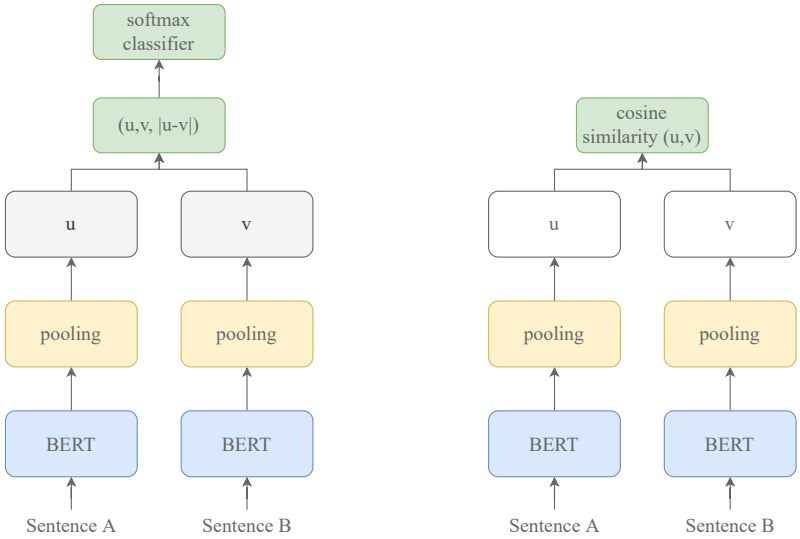


Figure 3.6: SBERT architecture examples. On the left, the SBERT architecture with the classification objective function, and on the right, SBERT architecture with the regression objective function [87].

3.3 Touché

Various teams participated in Touché Task 1, in the 2021 edition and also in the 2022 edition. Each teams' approach is then reported in a Task Overview [13] [12]. These approaches, and the data for the task are explored in this section.

3.3.1 Approaches for the 2021 edition

There have been some tested approaches in previous editions of this Touché Task 1. As described in Bondarenko et al. [13], the usual approaches are based on DirichletLM [70] [81] and BM25 [90] models combined with WordNet [27] for query expansion.

Elrond, the team with the highest relevance score, utilized a DirichletLM-based retrieval model, using also the Krovetz stemming method [56], and excluded stop words.

In the case of *Heimdall* [32], which was the team with the highest quality score, they utilized a DirichletLM but integrated with a topical relevance analysis using Universal Sentence Encoder and k-means clustering, and finally, a support vector regression model for argument quality trained on the Webis-ArgQuality-20 corpus[1].

Pippin Took uses DirichletLM as their retrieval model and WordNet-based query expansion. The team *Robin Hood* used phrase embeddings combined with a query expansion method - RM3 [59]. In this approach both the premise and the conclusion of an argument are represented in two separate vector spaces, an then uses their cosine similarity on the embedding query to calculate the arguments rank.

[1] https://zenodo.org/record/3780049

Asterix approach combines WordNet-based query expansion with BM25 retrieval model. Later, a quality-aware linear regression model is used to re-rank.

Team Dread Pirate Roberts used a DirichletLM to perform the ranking and then a LambdaMART model to re-rank the top-100 results.

At last, team *Skeletor* used a fine-tuned a BM25 model combined with the cosine similarity of passages calculated by a phrase embedding model.

3.3.2 Approaches for the 2022 edition

For the 2022 edition of the Touché Task 1 [12], the approaches focused on retrieval models such as TF-IDF, DirichletLM, BM25, and DPH [5]. The participating teams also used existing toolkits, such as Apache OpenNLP [2] for language detection, Project Debater API [10] for stance detection in arguments, and classifiers [33] [88] trained on the IBM Rank 30k corpus [35] for argument quality detection.

The tasks' baseline, *Swordsman*, employed a graph-based approach that ranked the sentences of an argument by their centrality in the argument graph as proposed by Alshomary et al. [4].

Team *D'Artagnan*, which placed fourth, seventh and fifth in relevance, quality and coherence respectively, used BM25 and DirichletLM models, stemming, n-grams, stopword removal, and query expansion with WordNet [75] and word2vec [74].

The team *Daario Naharis* developed a retrieval system using Lucene TF-IDF implementation and re-ranked the sentence pairs based on stance detection using the Project Debater API, achieving the best quality and coherence score, and ranking second in relevance.

Lucene-based approaches were developed by team *Gamora*, which placed fifth, third and seventh in relevance, quality and coherence respectively . The team added the title of a discussion and their stance on the topic to the document to obtain document-level relevance and quality scores. For document similarity, the team used BM25 and DirichletLM; for sentence agreement, the team used SBERT and TF-IDF.

The team *General Grevious* used an IR pipeline Lucene-based extended with LowerCaseFilter, EnglishPossessiveFilter, and LengthFilter. BM25 and Dirichlet-based along with RAKE [91] were used for document and sentence relevance; for re-ranking, the members relied on sentiment and readability analysis. Even though the team used re-ranking approaches, their best model solely relies on query expansion, resulting in ninth in relevance and tenth in quality and coherence.

Team *Gorgon*, placed in eighth, sixth, and eigth in relevance, quality and coherence, used Lucene for retrieval and compared BM25 and LMDirichlet similarity measures developing four different analyzers combining: LowercaseFilter, Krovetz stemmer, EnglishPossessiveFilter, and StopwordFilter. The sentence pairs were formed by creating all possible combinations of sentences within a single document.

The team *Hit Girl*, which placed sixth in relevance and coherence, and fourth in quality, proposed a combination of semantic search and argument quality agnostic models for re-ranking.

[2] https://opennlp.apache.org/

Even though argument quality re-ranking improved the quality score, it had an effect on the relevance score. As a result, the team proposed structural distance, a re-ranking method that uses fuzzy matching between the query and the sentences based on the part of speech tags.

Team *Korg* ranked eleventh in all metrics, relevance, quality and coherence. The team proposed to use ElasticSearch[3] with LM-Dirichlet similarity measure to identify the best matching argumentative sentences for a query. Doc2vec [60] trained on all sentences in the argsme corpus, or GPT-2 [82] was used to find similar sentences by direct comparison and generation. To preprocess the sentences, the team used LowercaseFilter, AsciifoldingFilter with Krovetz stemmer, and a stopword list.

To retrieve argumentative sentences, team *Pearl* used DirichletLM and DPH models in a two-stage retrieval pipeline. The argument quality scores were then used as a pre-processing step to eliminate noise examples. With this approach, the team ranked third in coherence, seventh in relevance and eighth in quality.

Finally, Elasticsearch with DirichletLM and BM25 was used for retrieval by team *Porthos*, which received the highest relevance score, and second place for both quality and coherence. Prior to retrieval, the team used the SVM of [33] and the BERT approach of [88] to remove sentence duplicates and filter irrelevant sentences by removing sentences in an incorrect language. SBERT and BERT trained for the next sentence prediction task form the sentence pairs.

3.4 Summary

This Chapter explored some proposed state-of-the-art approaches for information retrieval. The models described provided an insight on how to approach information retrieval problems, which we based our approach on, described in Chapter 5. Even though the architecture used to approach Touché Task 1 is essential, so is the data processing. Thus, the next chapter presents a comprehensive analysis of the used data.

[3] https://www.elastic.co/

Chapter 4

Data

It is critical for the development of this work to fully comprehend the data presented and to seek out additional data to improve the approaches used to address the stated problem. Hence, in this chapter, all the data used for this research project is thoroughly described, such as the data provided for Touché Task 1 and complementary data used to train an argument quality model.

4.1 Data

As mentioned in Section 1.1, the data for the Touché Task 1 was available at the webpage of the sub-task, and was gathered from four different debate portals: Debatewise (14 353 arguments) , IDebate (13 522 arguments), Debatepedia (21 197 arguments), Debate.org (338 620 arguments), and from the Canadian Parliamentary discussions (48 arguments). This data comprises 387 740 arguments, and were provided through the args.me [103] search engine.

Each argument has an identifier (id), a conclusion, a premise supporting the conclusion, and a context. The premise field contains not just the premise's text but also its stance. The context section comprises the discussion title as well as extensive information about the source and how the argument was obtained. Since each argument contains a lot of information, we chose to focus on five fields: id, conclusion, text, stance, and topic. All other contextual data was omitted to limit the amount of data and improve its quality, see Figure 4.1 and Listing 4.1.

The format of the Canadian parliamentary data is the same as stated above, but the contents of the context have non-relevant additional fields, and the discussion title field has been renamed to topic.

For the 2022 Task, an additional field – sentences – was added to each argument. Given that the Touché Task 1 for 2022 requests a pair of sentences rather than arguments, this new field was critical. It comprises the conclusion, each premise sentence, and the ids identifying each sentence.

We provide some data statistics in order to reduce the danger of unexpected data or data misunderstanding, as shown in Table 4.1 and Figures 4.2a and 4.2b.

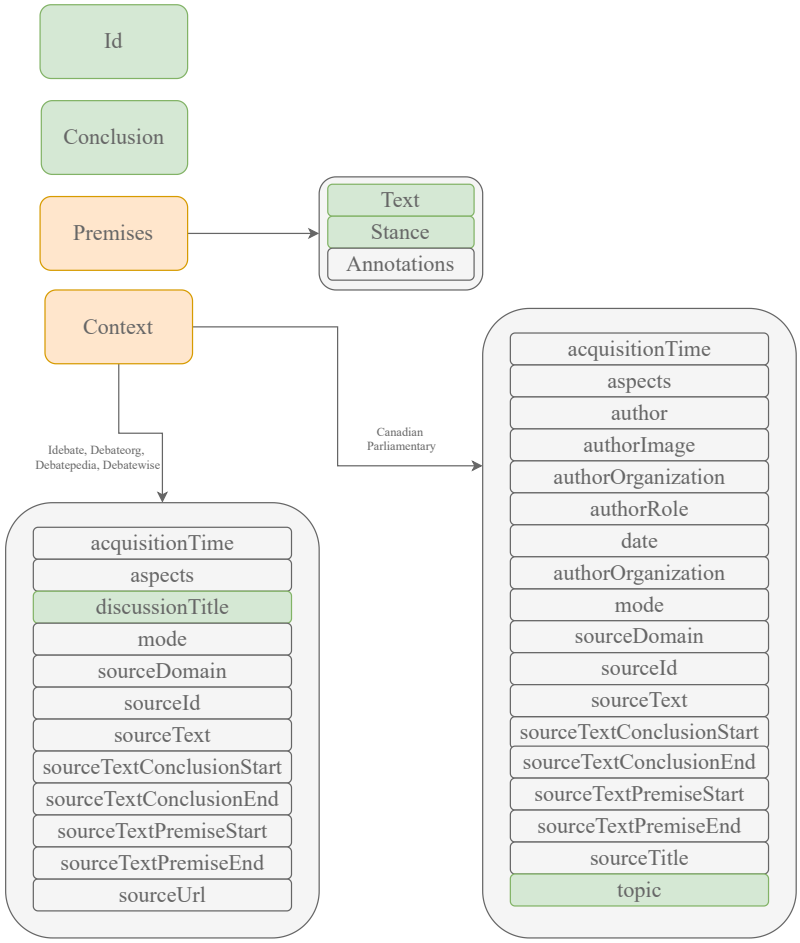


Figure 4.1: Representation of the data architecture supplied for Touché task 1.

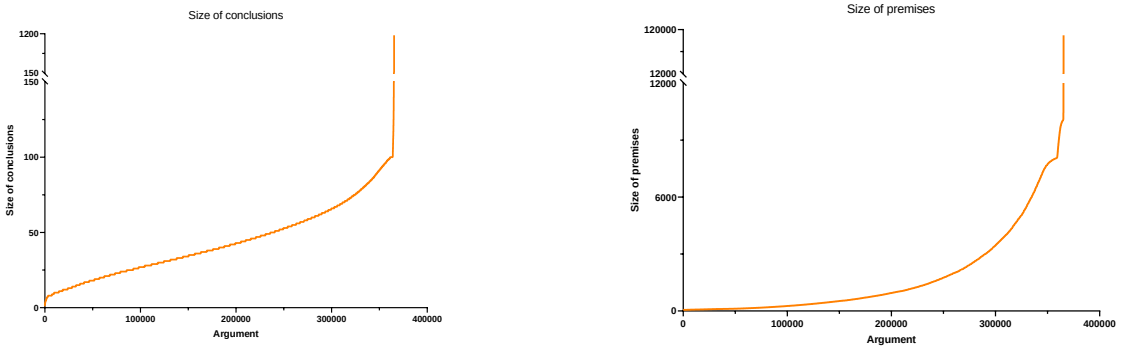
Table 4.1: Table reporting the average, max and min values of sentences, arguments and conclusions.

Parameter	**Value**
Average number of sentences per argument	16.8
Max sentences per argument	1700
Min sentences per argument	1
Average number of sentences per topic	87.7
Average number of arguments per topic	5.2
Average length of premises	1791
Smallest premise size	53
Largest premise size	106062
Average length of conclusion	43.4
Smallest conclusion size	1
Largest conclusion size	1160

Figure 4.2 illustrates the size of premises and conclusions per argument. It can be observed that the conclusions' most common values range from 0 to 60, but there are some cases with sig-

```
{
  "id": "Sf9294c83-Af186e851",
  "conclusion": "the War in Iraq was Worth the Cost",
  "premises": [
    {
      "text": "His removal provides stability and security not only for Iraq but for the Middle East as a region",
      "stance": "PRO",
      "annotations": []
    }
  ],
  "context": {
    "acquisitionTime": "2019-04-19T12:40:25Z",
    "aspects": [],
    "discussionTitle": "the War in Iraq was Worth the Cost",
    "mode": "discussion",
    "sourceDomain": "idebate",
    "sourceId": "Sf9294c83",
    "sourceText": "idebate.org Educational and informative news and resources on debate, advocacy and activism for youth. News Debatabase Events Community Media About Search form Search This House Believes that the War in Iraq was Worth the Cost The war in Iraq is likely to be remembered for decades... Login with Facebook Login with Twitter ",
    "sourceTextConclusionStart": 196,
    "sourceTextConclusionEnd": 230,
    "sourceTextPremiseStart": 8137,
    "sourceTextPremiseEnd": 8234,
    "sourceTitle": "This House Believes that the War in Iraq was Worth the Cost | idebate.org",
    "sourceUrl": "https://idebate.org/debatabase/international-middle-east-politics-terrorism-warpeace/house-believes-war-iraq-was-worth"
  }
}
```


Listing 4.1: Example of an argument from the IDebate portal.



(a) Size of conclusions per argument.

(b) Size of premises per argument.

Figure 4.2: Size of conclusions and premises per argument.

nificantly different values comprehended between 100 and 1160. Regarding the size of premises, the most common length is smaller than 6000, but in some cases, these premises might reach length values from 6000 to 106062.

The task also included input data containing the topics used as queries in addition to the data containing the arguments. This file enclosed the title and the number of the query for the 2021

edition, as shown in Listing 4.2; for the 2022 edition, it featured the title, number of the query, topic description, and narrative, as shown in Listing 4.3. The title is the question itself, the description provides an explanation and a context for the given question, and the narrative describes the desired outcome.

```
<topic>
    <number>51</number>
    <title>Do we need sex education in schools?</title>
</topic>
```

Listing 4.2: Example of a topic in the 2021 edition input data.

```
<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>
```

Listing 4.3: Example of a topic in the 2022 edition input data.

4.2 Complementary Data

Since one of our approaches focuses on pre-training a regression model to predict argument quality scores, we needed data with premises text and associated quality scores to train our model. Because the data provided did not have information regarding argument quality scores, we had to use complementary data. This complementary data was crawled from IBM Debater[1] and SIGIR-19-ArgSearch [2], as show in Table 4.2 and 4.3, respectively.

The IBM debater comprises 5300 arguments, divided by discussion title, and each argument has a text and a quality rank in the range [0, 1].

The SIGIR dataset was processed before being concatenated with the IBM debater. The conclusion, the ranked premise, and four evaluation criteria are included in the SIGIR dataset: relevance, rhetorical quality, dialectical quality, and logical quality. The metrics are scored on a scale of 1 (very bad) to 4 (very good); since this rank structure does not match the preceding dataset, we

[1] https://research.ibm.com/haifa/dept/vst/debating_data.shtml
[2] https://github.com/webis-de/SIGIR-19

Table 4.2: Example of IBM Debater data.

Id	Rank	Text
38	0.3	A free people have free will. A free will as long as it does not infringe on the rights of others. To Gamble should be legal as long as it does not effect others rights.
42	0.615384615	A government shouldn't interfere with how citizens spend their money
52	0.25	A person who wins while gambling can have very large financial benefits that can change their life for the better.
56	0.166666667	A solution is to develop a form of gambling entertainment that limits financial and emotional risk.
69	0.7	Able individuals should be allowed to make their own choices. There is no government or other institution better placed to evaluate the pros and cons of a decision than the affected individual.

normalized these metrics by adding all the values and dividing the total by 16, yielding weights ranging from 0 to 1.

Table 4.3: Example of SIGIR data, where **T** represents the *Topic ID*, **D** the *Discussion ID* , **A** the *Argument ID*, **A?** *Is Argument?*, **S** the *Stance*, **R** the *Relevance*, **LQ** the *Logical Quality*, **RQ** the *Rhetorical Quality*, **DQ** the *Dialectical Quality*, **C** the *Conclusion*, **P** the *Premise* and **Com** the *Comment*

T	D	A	A?	S	R	LQ	RQ	DQ	C	P	Com
1	6608	1	True	Con	2	1	2	1	Abortion Should Be Banned	I used to be against abortion until I met you.	personal attack
1	8865	50	True	Pro	3	4	2	3	Abortion should be legal	Abortion should be legal because if a woman do ...	repetitive
1	9560	50	True	Pro	3	2	1	2	Abortion Should Be Banned	All abortion shouldn't be banned However in m...	NaN
1.	9560	70	True	Pro	4	4	4	3	Abortion Should Be Banned	Abortion (elective abortions) should be banned..	NaN
1	11158	5	True	Con	3	2	3	3	Why Abortion should be banned	What about the trauma rape victims go through?...	NaN

Chapter 5

Approach

As described in the Chapter 3, various approaches to tackle information retrieval tasks are available. This chapter describes the strategies chosen to approach the 2021 and 2022 editions of the Touché Task 1.

5.1 Addressing the Touché Task 1 2021

For the Touché Task 1, from 2021, the goal was: given a question on a controversial topic, retrieve relevant arguments from the input data. To do so, we explored various methods from the argument retrieval field and developed an argument quality model and a cross-encoder for similarity analysis.

We chose to use Pyserini [62], a python toolkit for information retrieval search. This toolkit allows performing a sparse, dense, and hybrid search.

Our methods range from utilizing a model or a combination of models to retrieve and rate the quality of the arguments:

- Pyserini sparse retrieval (PS)
- Pyserini dense retrieval (PD)
- Pyserini hybrid retrieval (PH)
- Pyserini sparse retrieval with docT5query query expansion (PS5)
- Pyserini hybrid retrieval with docT5query query expansion (PH5)
- Pyserini sparse retrieval and argument quality model for re-ranking (PSArgQ)
- Pyserini sparse retrieval and argument quality model for re-ranking draw scores only (PSArgQD)
- Pyserini sparse retrieval and cross encoder for re-ranking (PSCE)
- Pyserini sparse retrieval and cross encoder re-ranking draw scores only (PSCED)

To conduct the experiments, all the methodologies mentioned above used the same data; this data was structured in different layouts (Figure 5.1) to compare the quality and relevance scores of the information retrieved:

- Premise and conclusion concatenated (PremConc)
- Premise, conclusion and topic concatenated (PremConcTopic)
- Premise, conclusion and DocT5query concatenated (PremConcDT5)

The PremConcDT5 is an approach that concatenates the PremConc approach with the generated queries using docTTTTTquery model with the PremConc as input, described in Subsection 5.1.6.

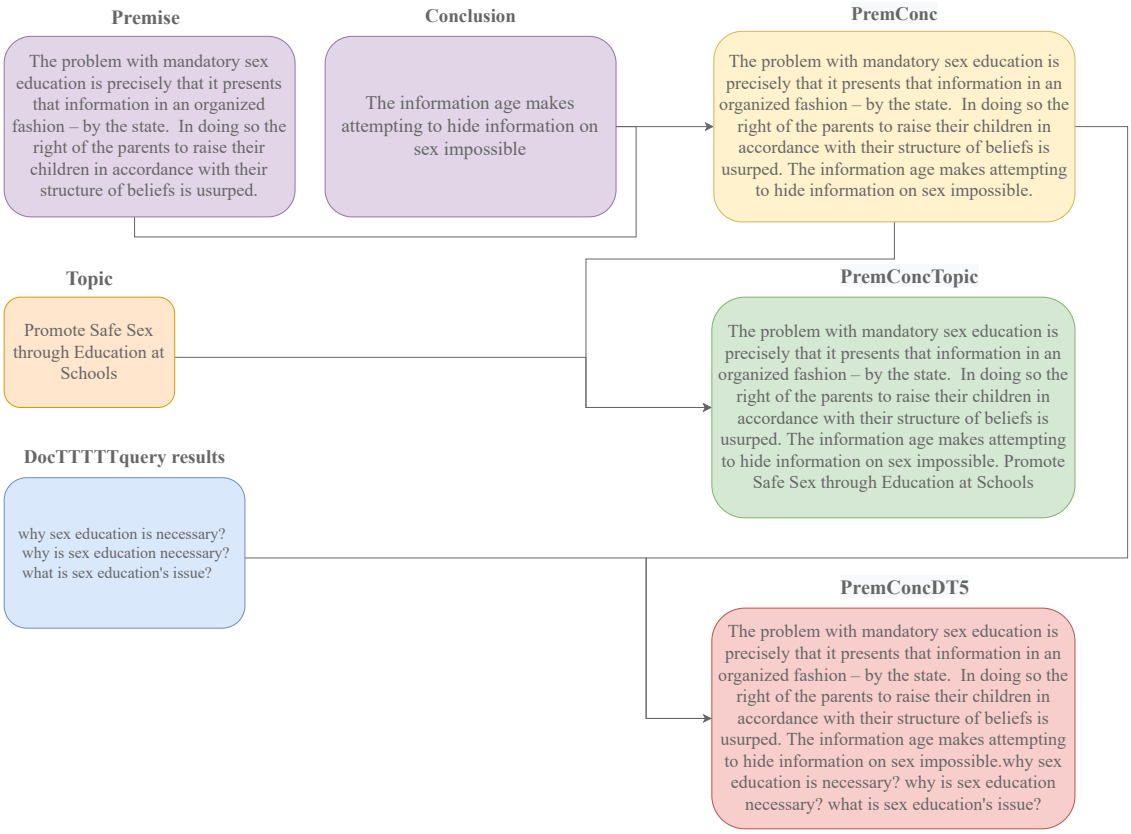


Figure 5.1: Representation of the three different arrangements of sentences used: PremConc, PremConcTopic, and PremConcDT5

To evaluate our results on the available evaluation files for the 2021 task edition, we used the trec_eval tool[1], which is the standard tool used by the TREC[2] community for assessing a run given the subjects and evaluation files.

[1] https://github.com/usnistgov/trec_eval
[2] https://trec.nist.gov/

5.1.1 Pyserini sparse retrieval

Our first approach focuses on Pyserini's sparse retrieval -– via integration with the Anserini IR toolkit built on the Lucene search library [39], both for indexing the data and performing the search since it performs retrieval with BM25 ranking using bag-of-words representations.

The Pyserini toolkit receives the data as an input, which it indexes and utilizes to search for the results for a given query.

Both the index and search processes use the Lucene library. For the search process, we specified how many hits (returned documents) for each question were needed (1000) and which scoring function to use, which in our case is BM25 scoring using bag-of-words representations, as shown in Figure 5.2.

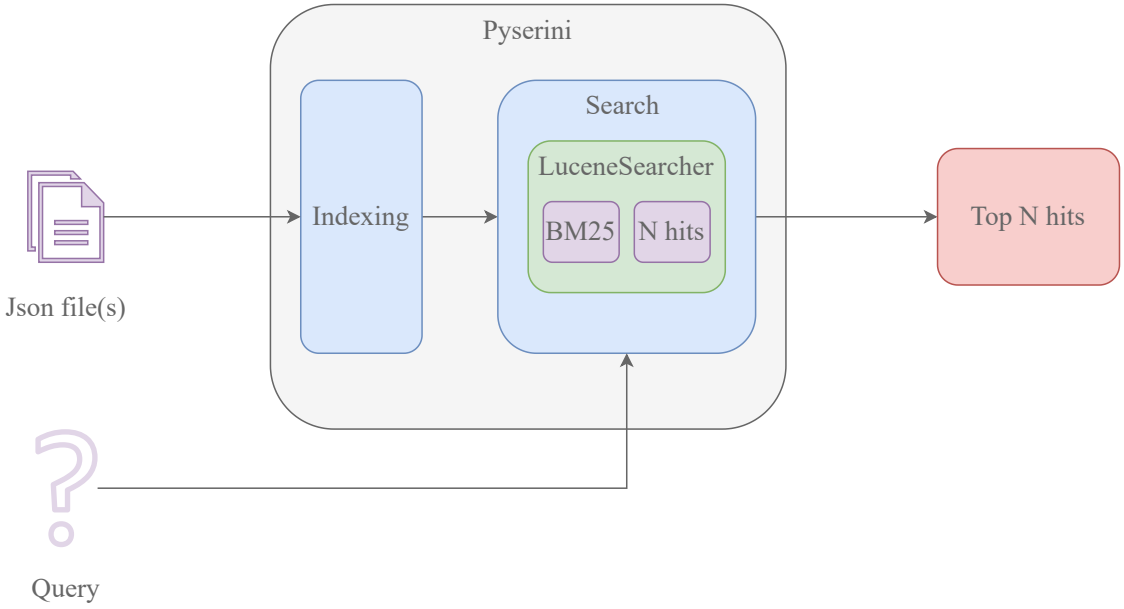


Figure 5.2: Representation of Pyserini sparse search. Given the input data in JSON format, and a query, this method will start by indexing the input data, upon which it will perform the search. We specified the scoring function (BM25) and the pretended number of hits (N hits). LuceneSearcher will retrieve the top N documents for the input query.

5.1.2 Pyserini dense retrieval

Pyserini incorporates Facebook's FAISS [3] library for efficient similarity search across dense vectors [50], which in turn incorporates the Hierarchical Navigable Small World Graphs (HNSW) library [71], see Figure 5.3. HNSW is a fully graph-based solution for the approximate K-nearest neighbor search based on navigable small-world graphs. It constructs a multi-layer system consisting of a hierarchical set of proximity graphs. This method enables low-latency querying to facilitate dense and hybrid retrieval.

[3] https://github.com/facebookresearch/faiss

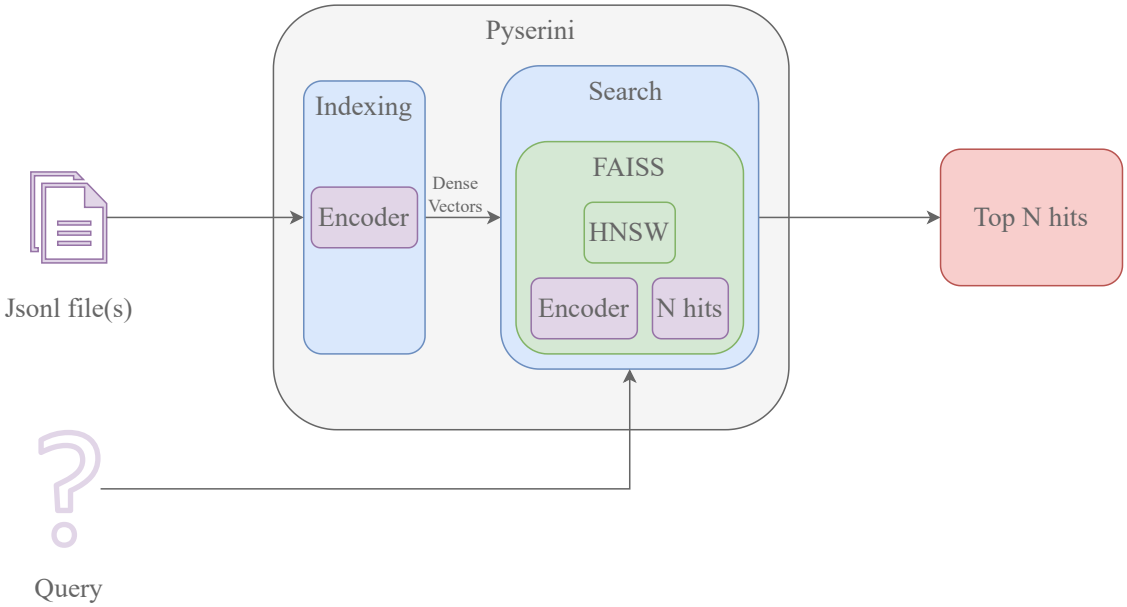


Figure 5.3: Representation of Pyserini dense search. Given the input data in JSON lines format and a query, this method will start by indexing the input data, producing dense vectors, using the chosen encoder. The search will then be performed upon the dense vectors, using the specified encoder to convert the query to the exact representation of the documents. It will return the top N hits, also specified by the user.

For this approach, we had to specify the query encoder. We decided to use different query encoders, variants of colBERT and distilBERT to observe the performance differences:

- castorini/tct_colbert-v2-hnp-msmarco [4]
- sentence-transformers/distilbert-base-nli-stsb-mean-tokens [5]

5.1.3 Pyserini hybrid retrieval

The Pyserini's hybrid search combines the previously stated sparse and dense approaches. The HybridSearcher class was used to execute the search, which mixes the output of the LuceneSearcher and FaissSearcher.

Since this retrieval is based on sparse and dense retrieval, we must provide our scoring function and query encoder. For the scoring function, we chose BM25 with bag-of-words representations. For the query encoder, we decided to experiment with a broader range of encoders than for dense retrieval:

- castorini.tct_colbert-v2-hnp-msmarco [6] (TCT-C)

[4] https://huggingface.co/castorini/tct_colbert-v2-hnp-msmarco
[5] https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-mean-tokens
[6] https://huggingface.co/castorini/tct_colbert-v2-hnp-msmarco

- sentence-transformers.all-distilroberta-v1 [7](ST-AD)
- sentence-transformers.all-mpnet-base-v2 [8](ST-AM)
- sentence-transformers.distilbert-base-nli-stsb-mean [9] [87](ST-DB)
- sentence-transformers.paraphrase-albert-small-v2 [10] [87] (ST-PA)

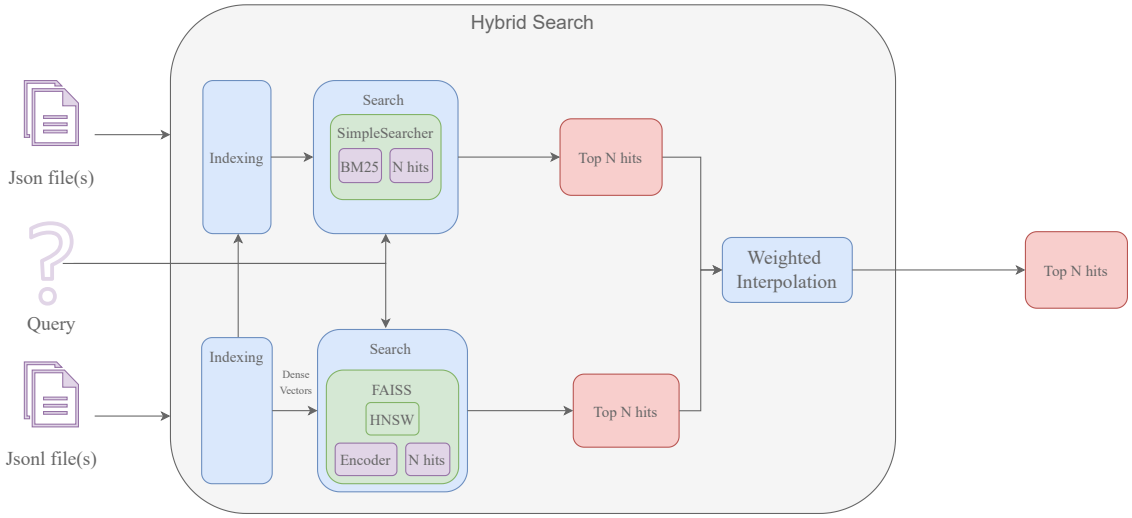


Figure 5.4: Representation of Pyserini hybrid search. The hybrid search combines the sparse and the dense search. After having the top N hits from both searches, it will perform a weighted interpolation to produce the final top N results.

As we can see in Figure 5.4, Pyserini hybrid search utilizes both Pyserini sparse and dense search. We specify our input data for both the query and the encoder for the dense search. The model indexes the documents; these indexes are used to search the results for the query. In dense search, it is also necessary to specify an encoder to convert the query to the exact representation of the documents. After the search step, the model will use the top N hits from both searches to perform the weighted interpolation:

$$\phi(q,d) = \begin{cases} \alpha \cdot \phi_{sp}(q,d) + \min\limits_{d \in \mathcal{D}_{ds}} \phi_{ds}(q,d), & if d \notin \mathcal{D}_{ds} \\ \alpha \cdot \min\limits_{d \in \mathcal{D}_{sp}} \phi_{sp}(q,d) + \phi_{ds}(q,d), & if d \notin \mathcal{D}_{sp} \\ \alpha \cdot \phi_{sp}(q,d) + \phi_{ds}(q,d), & otherwise \end{cases} \tag{5.1}$$

where d is the document, q the query, the hyperparameter α whose optimal value for BM25 is 0.10, $\mathcal{D}_{sp}$ and $\mathcal{D}_{ds}$ are the sparse and dense representations of the top N retrieved passages,

[7] https://huggingface.co/sentence-transformers/all-distilroberta-v1
[8] https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[9] https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-mean-tokens
[10] https://huggingface.co/sentence-transformers/paraphrase-albert-small-v2

with relevance scores, $\phi_{sp}(q, d \in \mathcal{D}_{sp})$ and $\phi_{ds}(q, d \in \mathcal{D}_{ds})$, respectively [63]. After the weighted interpolation, the model retrieves the top N passages according to their scores.

5.1.4 Argument quality model

Argumentation quality is crucial for argument mining, debate technology, and other similar applications [102]. We developed a model to predict quality scores for individual arguments to supplement our sparse retrieval approach.

This model employs a SimpleTransformers library pre-train distilbert classification model that has been fine-tuned using the complementary data presented in Section 4.2. This model was trained as a regression model to produce a score in the range [0, 1]. This training procedure used an AdamW optimizer with the epsilon default value (1e-8), to improve numerical stability, and 100 epochs. The learning rate was 5e-5, the train batch size was 16, the evaluation batch size was 8, and the weight decay was 0.1. This model achieved a mean squared error of 0.0429.

AdamW [68] is an improvement of the optimizer Adam, by modifying the weight decay implementation, decoupling weight decay from the gradient update. Weight decay is a form of regularization to lower the chance of the model fitting the training data perfectly, consequently, the model would not accurately predict data which was not used during the training stage - overfitting.

The developed argument quality model was then used to re-rank the arguments retrieved by Pyserini's sparse search; this method was employed with two variants:

Re-ranking argument quality model: We began by using Pyserini sparse to extract the top 2000 argument for all specified queries; next, all retrieved arguments are parsed into our new model, which assigns a quality score to each input. Then, we sort the arguments in descending order for each topic using the quality ratings already determined.

Argument quality model for re-ranking draw scores only: We utilized Pyserini sparse search, as stated above, to extract the top 2000 for all input queries. Then, rather than re-ranking all of the arguments, just those returned by Pyserini with tied scores are processed into our model to break the tie.

5.1.5 Cross-encoder

Cross-encoders, as discussed in Section 2.1.3, are encoders that take two input sentences and produce an output value ranging from zero to one, expressing the similarity between these two sentences.

We chose to use a SentenceTransformers distilroberta-based cross-encoder [11] that had been pre-trained on the STS benchmark dataset [12]. Similar to the argument quality model, this cross encoder was utilized after the Pyserini's sparse search to perform re-ranking. Although this strategy is similar to argument quality models, it will rank arguments based on their semantic resemblance to the query topic rather than judging argument quality.

[11] https://huggingface.co/cross-encoder/stsb-distilroberta-base
[12] http://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark

5.1.6 DocT5query

The Pyserini toolkit includes an expansion model, docTTTTTquery [13], otherwise known as docT5-query, that focuses on predicting queries that an input document may answer. Since our work required us to retrieve arguments for a specific query, we believed it was critical to test this expansion model.

In contrast to the strategies outlined before, this expansion model was used before the search. This model was used to augment the data by feeding it the input arguments and generating possible queries for these arguments, see Table 5.1 . We then concatenated the argument with the potential questions.

We conducted sparse and hybrid searches on the new data after this concatenation.

Table 5.1: Example of generated queries using DocT5query for a single premise.

Premise	**Generated Query 1**	**Generated Query 2**	**Generated Query 3**
It's important to be clear that this debate is looking at the results of the Iraq war and, by any definition Iraq is in a much more stable and secure position than it was in 2003 when American, British and other international troops arrived in the country. Whatever one thinks of the initial justifications for the war there is no doubt that the country, the region and the world are better and safer places without Saddam Hussein[i]. It is easy to criticize the allies but it is worth bearing in mind that the alternative was leaving in power a man who had committed genocide was a vicious and brutal dictator under whose regime extra-judicial execution and detention, mass-murder and torture were commonplace.	why did iraq get a war started	why iraq war was necessary	how does the war in iraq affect the united states.

5.2 Addressing the Touché Task 1 2022

This year, Touché task 1 announced a similar task to the last year's, with a significant difference. Instead of returning the best arguments for a query, the participating teams had to form pairs of sentences and retrieve those pairs.

This means that the structure of data used as input for this year was different. For this task, our approach starts by forming the possible pairs of sentences, see Figure 5.5. The sentence pairs were not restricted to sentences from the same argument. Nevertheless, our team only chose to form all possible sentence pairs between sentences from the same argument.

[13] https://github.com/castorini/docTTTTTquery

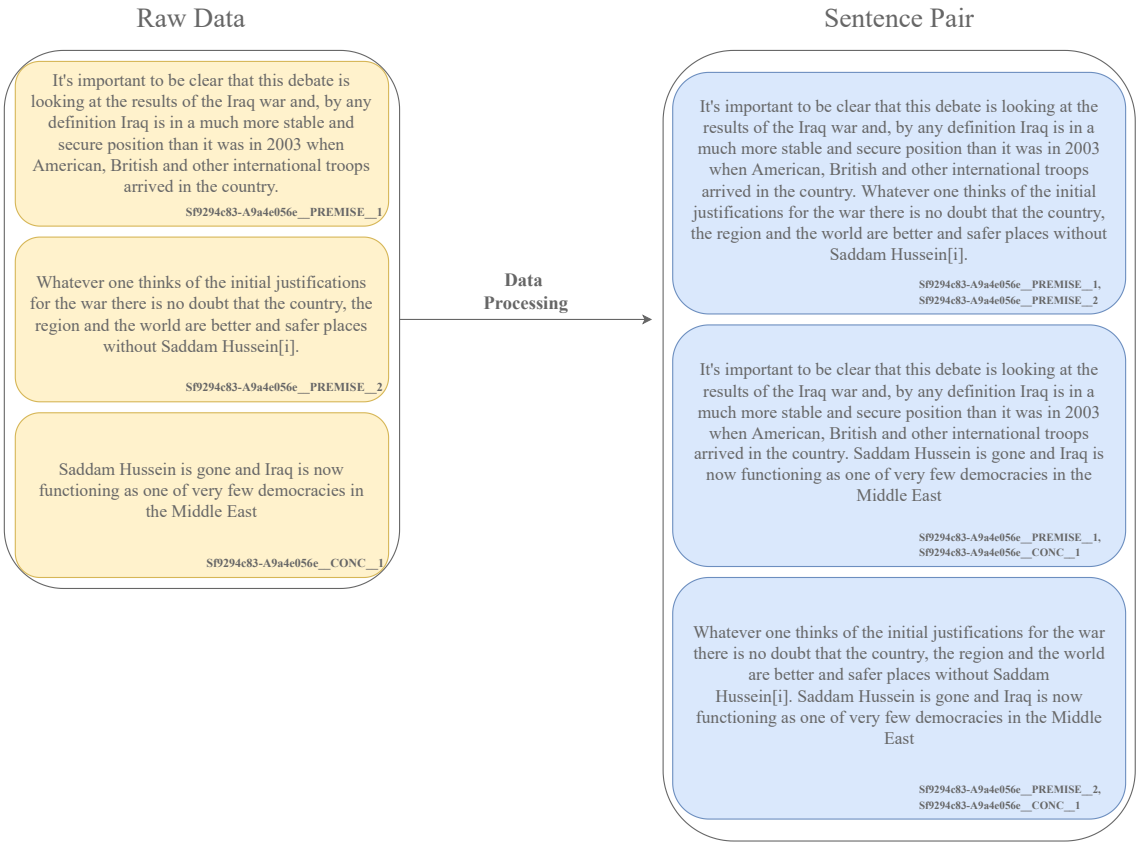


Figure 5.5: Processing raw data to form pairs of sentences. Each sentence contains an id and content; both the sentence id and the sentence content are concatenated to form the sentence pair. The number of pairs formed depends on the number of sentences in the array.

The queries were also processed, since we decided to experiment with two variations of the query data instead of using only the title of the query. An example of a query is represented in Listing 4.3:

- Title only;
- Title, description and narrative concatenated.

After having this processed data, we examined our experiments for the last 2021 task and decided to follow the Pyserini sparse search. Even though the Pyserini hybrid search achieved marginally higher scores, the computational costs were considerably higher compared to Pyserini sparse. Since we had numerous data to index and search, the optimal method to follow was the Pyserini sparse.

As shown in Figure 5.6 the dataset provided for the Touché Task 1 is pre-processed, where the noise data is filtered out, and all the sentence pairs from the same argument are formed. The query file is also processed, resulting in two variants: a query with the title and a query with title, description, and narrative. With these pairs of sentences and queries, as described in Subsection 5.1.1 the

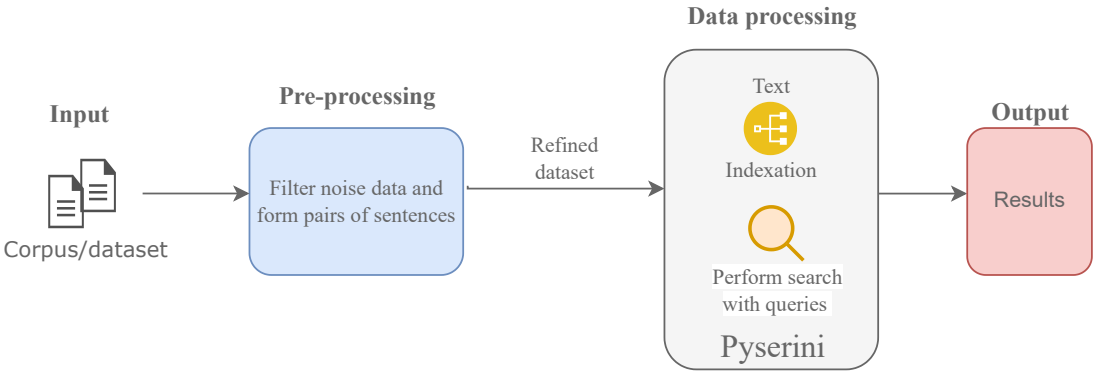


Figure 5.6: Approach to the Touché Task 1 2022. We start by filtering the irrelevant data and forming the possible pairs of sentences from the same argument, and generate the queries to be used, with title only and with title, description and narrative 4.3. With these pairs, and topics created, we use Pyserini sparse to index and search the top N results for the given queries.

Pyserini search starts by indexing the input data, upon which it searches for the top N documents for a given query.

To conduct the experiments, similar to the approach for the 2021 edition, we used different data layouts:

- Sentence-pairs (SP);
- Sentence-pairs concatenated with topic (SPT);
- Query title only (QT);
- Query title with description and narrative concatenated (QTDN).

5.3 Summary

Summarizing, we used a variety of approaches to solving the 2021 and 2022 editions of Touché Task 1. Overall the methods range from Pyserini sparse, dense, and hybrid search. Pyserini sparse search was complemented with an argument quality model and a cross encoder. All the approaches use different structures of data, and for the 2022 edition, the queries were also used with two additional designs. Table 5.2 summarizes all the approaches used for the 2021 and 2022 editions of Touché Task 1. These approaches have been tested and the results are described in Chapter 6.

Table 5.2: Overview of the approaches. (v1) represents the pyserini_run_v1, (v2) pyserinir_run_v2, (v3) pyserini_run_v3, and (v4) pyserini_run_v4

Touché Task 1 edition	Approach	Encoder	Data Structure	Query Structure
2021	PS	-	PremConc	QT
2021	PS	-	PremConcTopic	QT
2021	PS5	-	PremConcDT5	QT
2021	PSArgQ	-	PremConc	QT
2021	PSArgQ	-	PremConcTopic	QT
2021	PSArgQD	-	PremConc	QT
2021	PSArgQD	-	PremConcTopic	QT
2021	PSCE	-	PremConc	QT
2021	PSCE	-	PremConcTopic	QT
2021	PSCED	-	PremConc	QT
2021	PSCED	-	PremConcTopic	QT
2021	PD	TCT-C	PremConc	QT
2021	PD	TCT-C	PremConcTopic	QT
2021	PD	ST-AD	PremConc	QT
2021	PD	ST-AD	PremConcTopic	QT
2021	PH	TCT-C	PremConc	QT
2021	PH	TCT-C	PremConcTopic	QT
2021	PH	ST-AD	PremConc	QT
2021	PH	ST-AD	PremConcTopic	QT
2021	PH5	ST-AM	PremConcDT5	QT
2021	PH	ST-DB	PremConc	QT
2021	PH	ST-DB	PremConcTopic	QT
2021	PH	ST-PA	PremConc	QT
2021	PH	ST-PA	PremConcTopic	QT
2022 (v1)	PS	-	SP	QTDN
2022 (v2)	PS	-	SP	QT
2022 (v3)	PS	-	SPT	QTDN
2022 (v4)	PS	-	SPT	QT

Chapter 6

Experimental Evaluation

This chapter starts by reporting the experiments performed on the developed argument quality model using different base models. Furthermore, all the experiments performed on the 2021 and 2022 editions of the Touché Task 1 are reported. For the 2021 edition, several approaches were tested, such as Pyserini sparse, Pyserini sparse with argument quality model for re-ranking, Pyserini sparse with cross encoder for re-ranking, Pyserini dense, and Pyserini hybrid. The 2022 edition experiments only used Pyserini sparse search with different data layouts.

6.1 Argument quality model

We trained the argument quality model four times with a different pre-trained model (ALBERT [58], DistilBERT [96], BERT [23], RoBERTa [66]). All the models were trained on the complementary data described in Chapter 4 with an AdamW optimizer with the epsilon default value (1e-8), to improve numerical stability, and 100 epochs. The learning rate was 5e-5, the train batch size was 16. To evaluate these models, we tested the quality and relevance results produced for the Touché Task 1, re-ranking the draw scores from the Pyserini sparse search.

Table 6.1: Mean squared error of different models for argument quality.

Base model	**Mean Squared Error**
bert-base-uncased [1]	0.0407
albert-base-v2 [2]	0.0357
distilbert-base-uncased [3]	0.0429
roberta-base [4]	0.0362

In Table 6.1 we report the mean squared error results for all four fine-tuned models on the complementary data. All the models achieve similar mean squared errors comprised between 3.6% and 4.3%. Thus, we decided to use the argument quality model trained with the *distilbert-base-uncased* model, since it is lighter and faster, to perform the experiments for the Touché Task 1 2021, with different approaches (PSArgQ, PSArgQD) and different data layouts (PremConc, PremConcTopic), reported in Section 6.2.

6.2 Touché Task 1 2021

The first goal of this dissertation was to develop a competitive approach to participate in Touché Task 1. Therefore, we decided to use the 2021 evaluation data to experiment with our techniques and evaluate our performance on the task. This evaluation data was provided by the task organizers, and contained the top arguments, manually labeled by human assessors, both for their general topical relevance and for the rhetorical quality. For both quality and relevance, the metric used was the Normalized Discounted Cumulative Gain (NDCG).

First, we experimented with the Pyserini sparse search, see Table 6.2.

Table 6.2: Quality and relevance results of the Pyserini sparse search with different approaches.

Approach	**Data layout**	**Quality** NDCG@5	**Relevance** NDCG@5
PS	PremConc	**0.8116**	0.6742
PS	PremConcTopic	0.7881	**0.6838**
PS5	PremConcDT5	0.7500	0.6479
PSArgQ	PremConc	0.0126	0.0069
PSArgQ	PremConcTopic	0.0128	0.0069
PSArgQD	PremConc	0.8115	0.6742
PSArgQD	PremConcTopic	0.7892	0.6834
PSCE	PremConc	0.0333	0.0219
PSCE	PremConcTopic	0.0334	0.0219
PSCED	PremConc	0.8113	0.6738
PSCED	PremConcTopic	0.7892	0.6821

We noticed that using premise and conclusion concatenated provided a better quality score than premise conclusion and topic, but using the latter results in a better relevance score. This occurs because introducing the topic sentence to the premise will negatively affect the semantic quality of the sentence since these sentences might not perfectly fit together. On the other hand, concatenating this topic adds information to the premise; this means that it provides more details on the discussion topic the sentence is inserted in, resulting in a better relevance score. We also conclude that using the argument quality or cross encoder to re-rank all the retrieved arguments by Pyserini sparse affects our approach. Since the arguments from the *args.me* are extensive, this model can not effectively perform the re-ranking since it was trained on small sentences, thus producing low quality and relevance scores (0.0126, 0.0069). Using the argument quality or cross encoder to re-rank the draw scores is also not adequate for the reasons mentioned above but produces higher scores (0.8115 for quality and 0.6742 for relevance) than the re-rank for all the retrieved arguments because there is a small number of arguments with tied scores. After the Pyserini sparse search, we decided to test the dense search with two different encoders, see Table 6.3.

The dense Pyserini search did not enhance the previous approach, resulting in low quality and relevance scores. These poor results were expected since the Pyserini toolkit explicitly mentions that support for the custom collections is less mature for dense retrieval. Another disadvantage

Table 6.3: Quality and relevance results of the Pyserini dense search with different encoders.

Encoder	**Data layout**	**Quality** NDCG@5	**Relevance** NDCG@5
castorini/tct_colbert-v2-hnp-msmarco	PremConc	0.4907	0.4516
castorini/tct_colbert-v2-hnp-msmarco	PremConcTopic	0.4907	0.4517
distilbert-base-nli-stsb-mean-tokens	PremConc	0.1570	0.1175
distilbert-base-nli-stsb-mean-tokens	PremConcTopic	0.1569	0.1177

while using Pyserini dense search is the index size, since the indexes produced by the dense search are considerably larger than those generated with a sparse approach [62].

When it comes to the usage of *ST-DB*, this model provided even worst results than expected because, as described in this model's hugging face page[5], it produces sentence embeddings of low quality; therefore this model is deprecated.

Since the results of the dense search were not promising, we decided to test the Pyserini hybrid approach, see Table 6.4.

Table 6.4: Quality and relevance results of the Pyserini hybrid search with different encoders.

Encoder	**Data layout**	**Quality** NDCG@5	**Relevance** NDCG@5
TCT-T	PremConc	0.6597	0.6182
TCT-T	PremConcTopic	0.6337	0.6031
ST-AD	PremConc	**0.8116**	0.6745
ST-AD	PremConcTopic	0.7863	**0.6863**
ST-AM	PremConcDT5	0.7500	0.6476
ST-DB	PremConc	0.7711	0.6623
ST-DB	PremConcTopic	0.7362	0.6476
ST-PA	PremConc	0.7663	0.6404
ST-PA	PremConcTopic	0.7438	0.6321

Compared to the other approaches' results, we concluded that using Pyserini hybrid search, with *ST-AD* encoder was the best option for the task.

A pre-trained distilroberta-base model was used to generate *ST-AD*, which was fine-tuned on a 1B sentence pairs dataset, this 1B dataset is the result of the concatenation from multiple datasets, see Table A.1. The model was trained with a contrastive learning objective: given a sentence from the pair, it should predict which of a set of randomly picked other sentences was paired with it in 1B dataset used for fine-tuning. This model's intended usage is to generate a vector containing semantic information for a given input text, thereby performing information retrieval, clustering, and sentence similarity tasks. As already mentioned, *ST-DB* produced low-quality sentence embeddings. Therefore this model is labeled as deprecated. The *ST-AM* was designed to work on single sentences and short paragraphs. During training, the hyperparameter "sequence length" was set at 128 tokens. Because our data sequence length can immense large values, as

[5] https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-mean-tokens

shown in Section 4.1, this model is not the best choice for this purpose. When it comes to *ST-PA* the maximum sequence length for this model is 256 and was trained on a smaller dataset compared to *ST-AD*, which was trained on a dataset with 1 billion data, and its maximum sequence length is 512; therefore, *ST-AD* is more suitable for the task.

Last year, twenty-two teams participated in the task, to which we will be comparing our results. Each team could submit five runs which means that each could have five different pairs of scores (Quality - Relevance), but only the best for each metric would count. If there were draw scores, the other pair value would be used to break the tie (e.g., When evaluating teams' quality scores, if two teams have the same quality performance, the team with the best relevance score would be placed first than the one with the lowest relevance score.)

Table 6.5: Our team's quality and relevance performance compared to the participating teams' scores in the 2021 Touché Task 1.

Team name	**Quality** nDCG@5	**Team name**	**Relevance** nDCG@5
Heimdall	0.841	Elrond	0.720
Skeletor	0.827	Pippin Took	0.705
Asterix	0.818	Robin Hood	0.691
Elrond	0.817	**Our Approach**	**0.686**
Pippin Took	0.814	Asterix	0.681
Our Approach	**0.812**	Dread Pirate Roberts	0.678
Goemon Ishikawa	0.812	Skeletor	0.667
Hua Mulan	0.811	Luke Skywalker	0.662
Dread Pirate Roberts	0.810	Shanks	0.658
Yeagerists	0.810	Heimdall	0.648
Robin Hood	0.809	Athos	0.637
Luke Skywalker	0.808	Goemon Ishikawa	0.635
Macbeth	0.803	Jean Pierre Polnareff	0.633
Athos	0.802	Swordsman	0.626
Jean Pierre Polnareff	0.802	Yeagerists	0.625
Swordsman	0.796	Hua Mulan	0.620
Shanks	0.795	Macbeth	0.611
Blade	0.763	Blade	0.601
Little Foot	0.718	Deadpool	0.557
Batman	0.695	Batman	0.528
Deadpool	0.679	Little Foot	0.521
Gandalf	0.603	Gandalf	0.486
Palpatine	0.562	Palpatine	0.401

Upon analyzing Table 6.5, we concluded that if our team had participated in the task from the 2021 edition, we would have placed sixth in quality and fourth in relevance out of twenty-three participating teams.

6.3 Touché Task 1 2022

Given the good results of our approach on the 2021 evaluation data, our team decided to experiment with four different data combinations with the same retrieval model: Pyserini sparse search.

The Pyserini hybrid search achieved similar results as Pyserini sparse search on the 2021 edition's data, 0.8116 and 0.6863 for quality and relevance, respectively, and this approach demands large amounts of computational costs. Therefore we chose to follow the most balanced approach, i.e., the approach with the best cost/performance ratio, Pyserini sparse search.

Our team participated in the touché task and was limited to five submitted runs. We decided to submit four different runs with Pyserini sparse search and four different combinations of data :

- pyserini_sparse_v1 (v1): pairs of sentences, and query with title, description and narrative
- pyserini_sparse_v2 (v2): pairs of sentences, and query with title only
- pyserini_sparse_v3 (v3): pairs of sentences with topic concatenated, and query with title, description and narrative
- pyserini_sparse_v4 (v4): pairs of sentences with topic concatenated, and query with title only

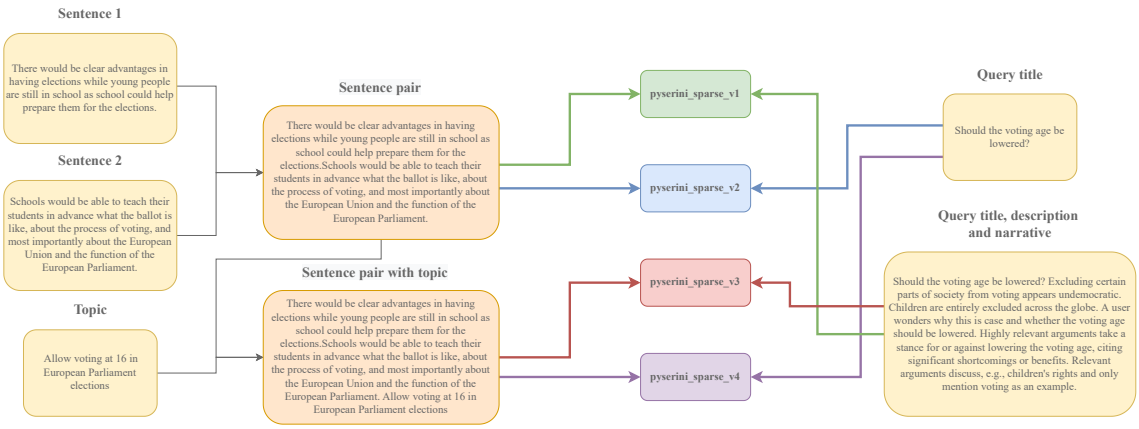


Figure 6.1: Representation of the four combinations of data used with an example.

Figure 6.1 provides an example of the four different combinations of data. Recently, the task organizers made available the evaluation files, upon which we performed experiments to evaluate our approach performance, see Table 6.6. The evaluation files contained the top pairs of sentences ranked according to their general topical relevance, their argument quality and their coherence.

Even though we used the same argument retrieval models as the one used in experiments performed with 2021 data, the results were significantly worse for this edition. The results were affected by our data arrangement approach. It was possible to form pairs of sentences with sentences from different arguments. However, we chose to create all the possible pairs of sentences

Table 6.6: Quality, relevance and coherence results for our four runs submitted in Touché Task1 2022.

Run	**Quality** NDCG@5	**Relevance** NDCG@5	**Coherence** NDCG@5
pyserini_sparse_v1	**0.772**	0.641	**0.378**
pyserini_sparse_v2	0.701	0.580	0.353
pyserini_sparse_v3	0.760	**0.651**	0.354
pyserini_sparse_v4	0.709	0.586	0.357

within the same argument, which led to our model's inability to retrieve some of the most relevant pairs of sentences. Nevertheless, with our four submitted runs, we placed third in relevance, fourth in coherence, and fifth in quality, as shown in Table 6.7, where our team tag is **Bruce Banner**. In Appendix A.3, A.2, and A.4 all the runs submitted by the participating teams are reported.

Table 6.7: Touché Task 1 2022 Quality, Relevance and Coherence teams scores.

Team	**Relevance** NDCG@5
Porthos	0.742
Daario Naharis	0.683
Bruce Banner	**0.651**
D Artagnan	0.642
Gamora	0.616
Hit Girl	0.588
Pearl	0.481
Gorgon	0.408
General Grievous	0.403
Swordsman	0.356
Korg	0.252

Team	**Quality** NDCG@5
Daario Naharis	0.913
Porthos	0.873
Gamora	0.785
Hit Girl	0.776
Bruce Banner	**0.772**
Gorgon	0.742
D Artagnan	0.733
Pearl	0.678
Swordsman	0.608
General Grievous	0.517
Korg	0.453

Team	**Coherence** NDCG@5
Daario Naharis	0.458
Porthos	0.429
Pearl	0.398
Bruce Banner	**0.378**
D Artagnan	0.378
Hit Girl	0.377
Gamora	0.285
Gorgon	0.282
Swordsman	0.248
General Grievous	0.231
Korg	0.168

6.4 General Discussion

In this chapter, we experimented with the before-proposed approaches. We concluded that, even though we experimented with the Pyserini sparse search and with the combination of this search with a cross encoder and an argument quality model, the best result for the 2021 edition of Touché Task 1 was obtained with the Pyserini hybrid search with ST-AD.

As for the 2022 edition of the Touché Task 1 the results submitted to the task were obtained using Pyserini sparse search, with different combinations of data and queries.

Regarding Pyserini sparse search, we also concluded that using data composed of premise and conclusion concatenated results in a better quality score but worse relevance score than premise, conclusion, and topic concatenated. An overview of the performed experiments is shown in Table 6.8.

Table 6.8: Overview of the experiments.

Touché Task 1 edition	**Approach**	**Encoder**	**Data Structure**	**Query Structure**	**Quality NDCG@5**	**Relevance NDCG@5**	**Coherence NDCG@5**
2021	PS	-	PremConc	QT	0.8116	0.6742	-
2021	PS	-	PremConcTopic	QT	0.7881	0.6838	-
2021	PS5	-	PremConcDT5	QT	0.7500	0.6479	-
2021	PSArgQ	-	PremConc	QT	0.0126	0.0069	-
2021	PSArgQ	-	PremConcTopic	QT	0.0128	0.0069	-
2021	PSArgQD	-	PremConc	QT	0.8115	0.6742	-
2021	PSArgQD	-	PremConcTopic	QT	0.7892	0.6834	-
2021	PSCE	-	PremConc	QT	0.0333	0.0219	-
2021	PSCE	-	PremConcTopic	QT	0.0334	0.0219	-
2021	PSCED	-	PremConc	QT	0.8113	0.6738	-
2021	PSCED	-	PremConcTopic	QT	0.7892	0.6821	-
2021	PD	TCT-C	PremConc	QT	0.4907	0.4516	-
2021	PD	TCT-C	PremConcTopic	QT	0.4907	0.4517	-
2021	PD	ST-AD	PremConc	QT	0.1570	0.1175	-
2021	PD	ST-AD	PremConcTopic	QT	0.1569	0.1177	-
2021	PH	TCT-C	PremConc	QT	0.6597	0.6182	-
2021	PH	TCT-C	PremConcTopic	QT	0.6337	0.6031	-
2021	PH	ST-AD	PremConc	QT	0.8116	0.6745	-
2021	PH	ST-AD	PremConcTopic	QT	0.7863	0.6863	-
2021	PH5	ST-AM	PremConcDT5	QT	0.7500	0.6476	-
2021	PH	ST-DB	PremConc	QT	0.7711	0.6623	-
2021	PH	ST-DB	PremConcTopic	QT	0.7362	0.6476	-
2021	PH	ST-PA	PremConc	QT	0.7663	0.6404	-
2021	PH	ST-PA	PremConcTopic	QT	0.7438	0.6321	-
2022 (v1)	PS	-	SP	QTDN	**0.772**	0.641	**0.378**
2022 (v2)	PS	-	SP	QT	0.701	0.580	0.353
2022 (v3)	PS	-	SPT	QTDN	0.760	**0.651**	0.354
2022 (v4)	PS	-	SPT	QT	0.709	0.586	0.357

Chapter 7

Conclusions

Individuals can access information in a fraction of a second. For this purpose, it is crucial to guarantee that the information retrieved to the user is coherent, relevant, and accurate.

There are many tasks and challenges in this field, which have been motivating the development of state-of-the-art approaches. Despite the increasing amount of attention being drawn to the topic of IR, there is still room for many growth and improvements. Thus, in this work, the main goal is to improve the quality of information retrieved for controversial topics, given the importance of providing correct information.

Various approaches to document retrieval were proposed, tested, and compared in recent works. Therefore, this project's development strategy is based on current state-of-the-art published approaches and combined various methods to achieve state-of-the-art performance and contribute to the NLP community.

The best results for the 2021 edition of the Touché Task 1 resulted from the Pyserini hybrid search, ranking fourth in relevance and sixth in quality. Nevertheless, when performing the experiments for the 2022 edition, our team concluded that, since we had to index large amounts of data, the most balanced approach regarding performance and computational costs is the Pyserini sparse search.

We also conclude that our approach, which we thoroughly explained in the paper submitted to the Touché Task 1 2022 [77], would benefit from a different data arrangement. This data arrangement should contain sentence pairs formed with sentences from other arguments or from the same argument instead of using an approach like ours, forming all possible pairs of sentences within the same argument. This approach resulted in large amounts of data ceasing our plans to use the Pyserini hybrid search. This approach also led to the impossibility of retrieving high-quality and relevant sentence pairs from different arguments; therefore, our data retrieval performance is limited. Nevertheless, in our best run, we achieved 0.772, 0.651, and 0.378 for Quality, Relevance, and Coherence scores, respectively. With these results, we achieved third place in relevance, fourth in coherence, and fifth in quality.

As future work, we think that there are several enhancements to be made, such as :

- Develop a different data arrangement process that allows sentence pairs to be formed with sentences from different arguments;
- Develop a considerably smaller sentence-pairs dataset to use Pyserini hybrid search, e. g. use Pyserini sparse to search the top 1000 documents and then use Pyserini hybrid to re-rank those retrieved documents;
- Improve the argument quality regression model;

References

[1] Amin Ahmad, Noah Constant, Yinfei Yang, and Daniel Cer. Reqa: An evaluation for end-to-end answer retrieval models. *arXiv preprint arXiv:1907.04780*, 2019.

[2] Ashikin Ali, Rozaida Ghazali, and Mustafa Mat Deris. The wavelet multilayer perceptron for the prediction of earthquake time series data. In *Proceedings of the 13th International Conference on Information Integration and Web-based applications and services*, pages 138–143, 2011.

[3] Wafa'Za'al Alma'aitah, Abdullah Zawawi Talib, and Mohd Azam Osman. Towards adaptive structured dirichlet smoothing model for digital resource objects. *Multimedia Tools and Applications*, 80(8):12175–12194, 2021.

[4] Milad Alshomary, Nick Düsterhus, and Henning Wachsmuth. Extractive snippet generation for arguments. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1969–1972, 2020.

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

[6] Giambattista Amati. *Information Retrieval Models*, pages 1976–1981. Springer New York, New York, NY, 2018.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. Sparterm: Learning term-based sparse representation for fast text retrieval. *arXiv preprint arXiv:2010.00768*, 2020.

[10] Roy Bar-Haim, Yoav Kantor, Elad Venezian, Yoav Katz, and Noam Slonim. Project debater apis: Decomposing the ai grand challenge. *arXiv preprint arXiv:2110.01029*, 2021.

[11] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[12] 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 *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 13th*

International Conference of the CLEF Association (CLEF 2022), Lecture Notes in Computer Science, page to appear, Berlin Heidelberg New York, September 2022. Springer.

[13] 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.

[14] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[15] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[16] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[17] Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19, 2006.

[18] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[19] Muthuraman Chidambaram, Yinfei Yang, Daniel Cer, Steve Yuan, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Learning cross-lingual sentence representations via a multi-task dual-encoder model. *CoRR*, abs/1810.12836, 2018.

[20] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online, July 2020. Association for Computational Linguistics.

[21] William Coster and David Kauchak. Simple English Wikipedia: A new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[22] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. Ms marco: Benchmarking ranking models in the large-data regime. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1566–1576, 2021.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[24] Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

[25] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165, 2014.

[26] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: long form question answering. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3558–3567. Association for Computational Linguistics, 2019.

[27] Christiane D. Fellbaum. Wordnet : an electronic lexical database. *Language*, 76:706, 2000.

[28] Katja Filippova and Yasemin Altun. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[29] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. Splade v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086*, 2021.

[30] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292, 2021.

[31] Norbert Fuhr. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3):243–255, 06 1992.

[32] Lukas Gienapp. Quality-aware argument retrieval with topical clustering. *Working Notes of CLEF*, 2021.

[33] Lukas Gienapp, Benno Stein, Matthias Hagen, and Martin Potthast. Efficient pairwise annotation of argument quality. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5772–5781, 2020.

[34] Joshua Goodman. Classes for fast maximum entropy training. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, volume 1, pages 561–564. IEEE, 2001.

[35] Shai Gretz, Roni Friedman, Edo Cohen-Karlik, Assaf Toledo, Dan Lahav, Ranit Aharonov, and Noam Slonim. A large-scale dataset for argument quality ranking: Construction and analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7805–7813, 2020.

[36] Ivan Habernal and Iryna Gurevych. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599, Berlin, Germany, August 2016. Association for Computational Linguistics.

[37] Yaakov HaCohen-Kerner, Daniel Miller, and Yair Yigal. The influence of preprocessing on text classification using a bag-of-words representation. *PLOS ONE*, 15(5):1–22, 05 2020.

[38] Ari Aulia Hakim, Alva Erwin, Kho I Eng, Maulahikmah Galinium, and Wahyu Muliady. Automated document classification for news article in bahasa indonesia based on term frequency inverse document frequency (tf-idf) approach. In *2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–4, 2014.

[39] Erik Hatcher and Otis Gospodnetic. *Lucene in action (in action series)*. Manning Publications Co., 2004.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[41] Matthew Henderson, Paweł Budzianowski, Inigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulić, et al. A repository of conversational datasets. *arXiv preprint arXiv:1904.06472*, 2019.

[42] Christopher Hidey and Kathleen McKeown. Identifying causal relations using parallel wikipedia articles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1424–1433, 2016.

[43] Djoerd Hiemstra. *Information Retrieval Models*, pages 1–19. Wiley, United States, October 2009.

[44] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[45] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666*, 2020.

[46] Sebastian Hofstätter and Allan Hanbury. Let's measure run time! extending the ir replicability infrastructure to include performance aspects. *arXiv preprint arXiv:1907.04614*, 2019.

[47] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*, 2019.

[48] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA, 2017.

[49] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[50] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[51] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[52] Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. Gooaq: Open question answering with diverse answer types. *arXiv preprint arXiv:2104.08727*, 2021.

[53] Omar Khattab, Christopher Potts, and Matei Zaharia. Relevance-guided supervision for openqa with colbert. *Transactions of the Association for Computational Linguistics*, 9:929–944, 2021.

[54] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[55] Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*, 2018.

[56] Robert Krovetz. Viewing morphology as an inference process. *Artificial intelligence*, 118(1-2):277–294, 2000.

[57] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[58] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[59] Victor Lavrenko and W Bruce Croft. Relevance-based language models. In *ACM SIGIR Forum*, volume 51, pages 260–267. ACM New York, NY, USA, 2017.

[60] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[61] Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115, 2021.

[62] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362, 2021.

[63] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. Distilling dense representations for ranking using tightly-coupled teachers. *arXiv preprint arXiv:2010.11386*, 2020.

[64] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[65] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[66] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[67] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online, July 2020. Association for Computational Linguistics.

[68] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[69] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1101–1104, 2019.

[70] David JC MacKay and Linda C Bauman Peto. A hierarchical dirichlet language model. *Natural language engineering*, 1(3):289–308, 1995.

[71] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[72] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[73] Michael Frederick McTear, Zoraida Callejas, and David Griol. *The conversational interface*, volume 6. Springer, 2016.

[74] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[75] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[76] Shapol M Mohammed, Karwan Jacksi, and Subhi RM Zeebaree. Glove word embedding and dbscan algorithms for semantic document clustering. In *2020 International Conference on Advanced Science and Engineering (ICOASE)*, pages 1–6. IEEE, 2020.

[77] Bernardo Moreira, Henrique Lopes Cardoso, Bruno Martins, and Fábio Goularte. Team bruce banner at touché 2022: Argument retrieval for controversial questions. In Guglielmo Faggioli, Nicola Ferro, Allan Hanbury, and Martin Potthast, editors, *Working Notes of CLEF 2022 – Conference and Labs of the Evaluation Forum*, 2022.

[78] Berndt Müller, Joachim Reinhardt, and Michael T Strickland. *Neural networks: an introduction*. Springer Science & Business Media, 1995.

[79] Leonardo Noriega. Multilayer perceptron tutorial. *School of Computing. Staffordshire University*, 2005.

[80] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[81] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 275–281, New York, NY, USA, 1998. Association for Computing Machinery.

[82] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[83] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. In *The 7th International Student Conference on Advanced Science and Technology ICAST*, volume 4, page 1, 2012.

[84] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[85] Hassan Ramchoun, Youssef Ghanou, Mohamed Ettaouil, and Mohammed Amine Janati Idrissi. Multilayer perceptron: Architecture optimization and training. 2016.

[86] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[87] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[88] Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. Classification and clustering of arguments with contextualized word embeddings. *arXiv preprint arXiv:1906.09821*, 2019.

[89] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 2004.

[90] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[91] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1(1-20):10–1002, 2010.

[92] Dennis W Ruck, Steven K Rogers, and Matthew Kabrisky. Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*, 2(2):40–48, 1990.

[93] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[94] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[95] Gerard Salton, Edward A Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[96] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[97] Anita Kumari Singh and Mogalla Shashi. Vectorization of text documents for identifying unifiable news articles. *International Journal of Advanced Computer Science and Applications*, 10(7), 2019.

[98] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[99] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

[100] Krysta Svore and Christopher Burges. A machine learning approach for improved bm25 retrieval. pages 1811–1814, 01 2009.

[101] 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.

[102] Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187, 2017.

[103] 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 Kevin Ashley, Claire Cardie, Nancy Green, Iryna Gurevych, Ivan Habernal, Diane Litman, Georgios Petasis, Chris Reed, Noam Slonim, and Vern Walker, editors, *4th Workshop on Argument Mining (ArgMining 2017) at EMNLP*, pages 49–59. Association for Computational Linguistics, September 2017.

[104] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. Minilmv2: Multihead self-attention relation distillation for compressing pretrained transformers. *arXiv preprint arXiv:2012.15828*, 2020.

[105] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

[106] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[107] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Zero-shot entity linking with dense entity retrieval. *CoRR*, abs/1911.03814, 2019.

[108] Jun Yan. *Text Representation*, pages 3069–3072. Springer US, Boston, MA, 2009.

[109] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[110] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

[111] Chengxiang Zhai. *Risk minimization and language modeling in text retrieval*. PhD thesis, PhD thesis, Carnegie Mellon University, 2002.

[112] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, 2021.

[113] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Appendix A

Appendix

A.1 Datasets used for the 1B dataset

Table A.1: Multiple datasets used to form the 1B dataset [1].

Dataset	Number of training tuples	Paper
Reddit comments(2015-2018) [2]	726,484,430	[41]
S2ORC *Citation pairs (Abstracts)* [3]	116,288,806	[67]
WikiAnswers [4]	77,427,422	[25]
PAQ (Question, Answer) [5] pairs	64,371,441	[61]
S2ORC Citation pairs (Titles) [6]	52,603,982	[67]
S2ORC (Title, Abstract) [7]	41,769,185	[67]
Stack Exchange (Title, Body) pairs [8]	25,316,456	-
MS MARCO triplets [9]	9,144,553	[22]
GOOAQ: Open Question Answering with Diverse Answer Types [10]	3,012,496	[52]
Yahoo Answers (Title, Answer) [11]	1,198,260	[113]
Code Search [12]	1,151,414	-
COCO Image captions [13]	828,395	[64]
SPECTER citation triplets [14]	684,100	[20]
Yahoo Answers (Question, Answer) [15]	681,164	[113]
Yahoo Answers (Title, Question) [16]	659,896	[113]
SearchQA [17]	582,261	[24]
Eli5 [18]	325,475	[26]
Flickr 30k [19]	317,695	[110]
Stack Exchange Duplicate questions (titles) [20]	304,525	-
AllNLI (SNLI and MultiNLI) [21] [22]	277,230	[14] [106]
Stack Exchange Duplicate questions (bodies) [23]	250,519	-
Stack Exchange Duplicate questions (titles+bodies) [24]	250,460	-
Sentence Compression [25]	180,000	[28]
Wikihow [26]	128,542	[55]
Altlex [27]	112,696	[42]
Quora Question Triplets [28]	103,663	-
Simple Wikipedia [29]	102,225	[21]
Natural Questions (NQ) [30]	100,231	[57]
SQuAD2.0 [31]	87,599	[84]
TriviaQA [32]	73,346	-
Total	**1,124,818,467**	

A.2 Touché Task 1 2022 all submitted runs relevance results

Team	Tag	Mean nDCG@5	CI95 Low	CI95 High
Porthos	scl_dlm_bqnc_acl_nsp	0.742	0.670	0.807
Daario Naharis	INTSEG-Letter-no_stoplist-Krovetz-Icoef-Evidence-Par	0.683	0.609	0.755
Daario Naharis	INTSEG-Run-Whitespace-Krovetz-Stoplist-Pos-Evidence-icoeff-Sep	0.676	0.587	0.762
Daario Naharis	INTSEG-Run-letter-english-2-20-no_stoplist-pos-evidence-icoef-An	0.670	0.589	0.751
Bruce Banner	Bruce-Banner_pyserinin_sparse_v3	0.651	0.573	0.720
D Artagnan	seupd2122-6musk-kstem-stop-shingle3	0.642	0.575	0.705
Bruce Banner	Bruce-Banner_pyserinin_sparse_v1	0.641	0.575	0.705
Daario Naharis	INTSEG-Whitespace-Stoplist-Krovetz-Icoef-Sep	0.629	0.549	0.706
Gamora	seupd2122-javacafe-gamoraHeuristicsOnlyQueryReductionDoubleIndex	0.616	0.551	0.687
D Artagnan	seupd2122-6musk-stop-wordnet-kstem-dirichlet	0.608	0.542	0.682
D Artagnan	seupd2122-6musk-stop-kstem-concsearch	0.591	0.514	0.667
Hit Girl	Io	0.588	0.515	0.657
Gamora	seupd2122-javacafe-gamoraStandardDoubleIndex	0.588	0.518	0.655
Bruce Banner	Bruce-Banner_pyserinin_sparse_v4	0.586	0.509	0.658
D Artagnan	seupd2122-6musk-word2vec-sentences-kstem	0.585	0.506	0.661
Gamora	seupd2122-javacafe-gamoraHeuristicsDoubleIndex	0.584	0.512	0.656
Hit Girl	Ganymede	0.583	0.513	0.650
Bruce Banner	Bruce-Banner_pyserinin_sparse_v2	0.580	0.507	0.654
Hit Girl	Jupiter	0.560	0.484	0.631
Hit Girl	Europa	0.546	0.477	0.615
Gamora	seupd2122-javacafe-gamora_tfidf_kstemstopengpos_multi_YYY	0.516	0.446	0.581
Gamora	seupd2122-javacafe-gamora_sbert_kstemstopengpos_multi_YYY	0.497	0.407	0.586
Pearl	PearlBlocklist_WeightedRelevance	0.481	0.399	0.560
Pearl	PearlArgRank8040_WeightedRelevance	0.479	0.403	0.556
Pearl	PearlArgRank7530	0.470	0.391	0.547
Pearl	PearlBlocklist	0.466	0.380	0.549
Pearl	PearlArgRank8040	0.465	0.389	0.551
Gorgon	GorgonA2Bm25	0.408	0.354	0.461
Daario Naharis	INTSEG-Run-Whitespace-Porter-Wordnet-Pos-no_stoplist-tfidf-An	0.406	0.305	0.510
General Grievous	seupd2122-lgtm_QE_NRR	0.403	0.335	0.471
General Grievous	seupd2122-lgtm_NQE_NRR	0.402	0.335	0.476
Gorgon	GorgonA1Bm25	0.396	0.350	0.442
Gorgon	GorgonBasicBM25	0.387	0.330	0.439
General Grievous	seupd2122-lgtm_NQE_NRR_ONLY_TITLE	0.386	0.314	0.451
General Grievous	seupd2122-lgtm_QE_NRR_ONLY_TITLE	0.386	0.317	0.450
Gorgon	GorgonKEBM25	0.378	0.329	0.428
Swordsman	baseline_swordsman	0.356	0.296	0.412
Gorgon	GorgonBasicLMD	0.315	0.269	0.362
D Artagnan	seupd2122-6musk-stop-kstem-basic	0.300	0.229	0.369
Korg	korg9000	0.252	0.187	0.318
Porthos	scl_dlm_bqnc_acl_nsp_100_test	0.244	0.215	0.275

Figure A.1: Touché Task 1 2022 all submitted runs relevance results [33].

A.3 Touché Task 1 2022 all submitted runs quality results

Team	Tag	Mean nDCG@5	CI95 Low	CI95 High
Daario Naharis	INTSEG-Letter-no_stoplist-Krovetz-Icoef-Evidence-Par	0.913	0.870	0.947
Daario Naharis	INTSEG-Run-letter-english-2-20-no_stoplist-pos-evidence-icoef-An	0.898	0.855	0.941
Daario Naharis	INTSEG-Run-Whitespace-Krovetz-Stoplist-Pos-Evidence-icoeff-Sep	0.896	0.841	0.944
Porthos	scl_dlm_bqnc_acl_nsp	0.873	0.825	0.913
Gamora	seupd2122-javacafe-gamoraHeuristicsOnlyQueryReductionDoubleIndex	0.785	0.729	0.848
Gamora	seupd2122-javacafe-gamoraHeuristicsDoubleIndex	0.779	0.716	0.835
Daario Naharis	INTSEG-Whitespace-Stoplist-Krovetz-Icoef-Sep	0.776	0.712	0.839
Hit Girl	Ganymede	0.776	0.707	0.840
Bruce Banner	Bruce-Banner_pyserinin_sparse_v1	0.772	0.702	0.830
Bruce Banner	Bruce-Banner_pyserinin_sparse_v3	0.760	0.680	0.832
Gamora	seupd2122-javacafe-gamora_tfidf_kstemstopengpos_multi_YYY	0.755	0.686	0.823
Gamora	seupd2122-javacafe-gamora_sbert_kstemstopengpos_multi_YYY	0.743	0.656	0.823
Gorgon	GorgonA2Bm25	0.742	0.700	0.786
D Artagnan	seupd2122-6musk-stop-wordnet-kstem-dirichlet	0.733	0.676	0.787
Gamora	seupd2122-javacafe-gamoraStandardDoubleIndex	0.731	0.672	0.786
Gorgon	GorgonA1Bm25	0.729	0.686	0.774
D Artagnan	seupd2122-6musk-kstem-stop-shingle3	0.728	0.657	0.794
D Artagnan	seupd2122-6musk-stop-kstem-concsearch	0.727	0.659	0.786
Hit Girl	Jupiter	0.725	0.651	0.796
Gorgon	GorgonKEBM25	0.724	0.677	0.769
D Artagnan	seupd2122-6musk-word2vec-sentences-kstem	0.723	0.659	0.787
Hit Girl	Europa	0.721	0.643	0.793
Hit Girl	Io	0.719	0.643	0.797
Bruce Banner	Bruce-Banner_pyserinin_sparse_v4	0.709	0.624	0.783
Bruce Banner	Bruce-Banner_pyserinin_sparse_v2	0.701	0.610	0.783
Gorgon	GorgonBasicBM25	0.685	0.634	0.734
Gorgon	GorgonBasicLMD	0.679	0.634	0.726
Pearl	PearlArgRank7530	0.678	0.609	0.744
Daario Naharis	INTSEG-Run-Whitespace-Porter-Wordnet-Pos-no_stoplist-tfidf-An	0.671	0.585	0.753
Pearl	PearlBlocklist_WeightedRelevance	0.670	0.605	0.734
Pearl	PearlBlocklist	0.670	0.605	0.729
Pearl	PearlArgRank8040_WeightedRelevance	0.670	0.601	0.735
Pearl	PearlArgRank8040	0.668	0.595	0.737
Swordsman	baseline_swordsman	0.608	0.543	0.671
General Grievous	seupd2122-lgtm_QE_NRR	0.517	0.444	0.583
General Grievous	seupd2122-lgtm_NQE_NRR	0.517	0.442	0.591
General Grievous	seupd2122-lgtm_NQE_NRR_ONLY_TITLE	0.475	0.387	0.559
General Grievous	seupd2122-lgtm_QE_NRR_ONLY_TITLE	0.475	0.392	0.555
Korg	korg9000	0.453	0.384	0.529
D Artagnan	seupd2122-6musk-stop-kstem-basic	0.441	0.357	0.517
Porthos	scl_dlm_bqnc_acl_nsp_100_test	0.274	0.247	0.301

Figure A.2: Touché Task 1 2022 all submitted runs quality results [34].

A.4 Touché Task 1 2022 all submitted runs coherence results

Team	Tag	Mean nDCG@5	CI95 Low	CI95 High
Daario Naharis	INTSEG-Run-Whitespace-Krovetz-Stoplist-Pos-Evidence-icoeff-Sep	0.458	0.389	0.525
Daario Naharis	INTSEG-Letter-no_stoplist-Krovetz-Icoef-Evidence-Par	0.444	0.375	0.508
Porthos	scl_dlm_bqnc_acl_nsp	0.429	0.353	0.509
Daario Naharis	INTSEG-Run-letter-english-2-20-no_stoplist-pos-evidence-icoef-An	0.407	0.331	0.489
Pearl	PearlArgRank7530	0.398	0.311	0.485
Pearl	PearlArgRank8040	0.396	0.311	0.481
Pearl	PearlBlocklist	0.392	0.307	0.475
D Artagnan	seupd2122-6musk-kstem-stop-shingle3	0.378	0.311	0.452
Bruce Banner	Bruce-Banner_pyserinin_sparse_v1	0.378	0.300	0.459
Hit Girl	Ganymede	0.377	0.303	0.456
Pearl	PearlBlocklist_WeightedRelevance	0.369	0.287	0.450
Pearl	PearlArgRank8040_WeightedRelevance	0.369	0.291	0.443
Hit Girl	Io	0.365	0.302	0.430
D Artagnan	seupd2122-6musk-stop-wordnet-kstem-dirichlet	0.358	0.292	0.427
Bruce Banner	Bruce-Banner_pyserinin_sparse_v4	0.357	0.273	0.446
Bruce Banner	Bruce-Banner_pyserinin_sparse_v3	0.354	0.272	0.444
Bruce Banner	Bruce-Banner_pyserinin_sparse_v2	0.353	0.283	0.433
Hit Girl	Europa	0.349	0.287	0.415
D Artagnan	seupd2122-6musk-stop-kstem-concsearch	0.336	0.270	0.400
D Artagnan	seupd2122-6musk-word2vec-sentences-kstem	0.333	0.274	0.400
Hit Girl	Jupiter	0.330	0.269	0.394
Daario Naharis	INTSEG-Whitespace-Stoplist-Krovetz-Icoef-Sep	0.288	0.216	0.361
Gamora	seupd2122-javacafe-gamora_sbert_kstemstopengpos_multi_YYY	0.285	0.203	0.373
Gorgon	GorgonKEBM25	0.282	0.233	0.335
Gamora	seupd2122-javacafe-gamoraHeuristicsOnlyQueryReductionDoubleIndex	0.276	0.204	0.347
Gamora	seupd2122-javacafe-gamora_tfidf_kstemstopengpos_multi_YYY	0.276	0.200	0.372
Gorgon	GorgonBasicBM25	0.274	0.210	0.334
D Artagnan	seupd2122-6musk-stop-kstem-basic	0.273	0.207	0.346
Gamora	seupd2122-javacafe-gamoraHeuristicsDoubleIndex	0.272	0.203	0.343
Gorgon	GorgonA2Bm25	0.259	0.209	0.314
Swordsman	baseline_swordsman	0.248	0.193	0.303
Gorgon	GorgonA1Bm25	0.246	0.197	0.301
General Grievous	seupd2122-lgtm_QE_NRR	0.231	0.162	0.313
General Grievous	seupd2122-lgtm_NQE_NRR	0.228	0.164	0.299
Gorgon	GorgonBasicLMD	0.225	0.162	0.289
General Grievous	seupd2122-lgtm_NQE_NRR_ONLY_TITLE	0.220	0.158	0.283
General Grievous	seupd2122-lgtm_QE_NRR_ONLY_TITLE	0.219	0.160	0.283
Daario Naharis	INTSEG-Run-Whitespace-Porter-Wordnet-Pos-no_stoplist-tfidf-An	0.203	0.137	0.280
Gamora	seupd2122-javacafe-gamoraStandardDoubleIndex	0.195	0.139	0.252
Korg	korg9000	0.168	0.117	0.223
Porthos	scl_dlm_bqnc_acl_nsp_100_test	0.105	0.070	0.144

Figure A.3: Touché Task 1 2022 all submitted runs coherence results [35].

A.5 Submitted paper describing our team approach for the Touché Task 1 2022

Team Bruce Banner at Touché 2022: Argument Retrieval for Controversial Questions

Notebook for the Touché Lab on Argument Retrieval at CLEF 2022

Bernardo C. Moreira[1], Henrique Lopes Cardoso[1,2], Bruno Martins[3,4] and Fábio Goularte[3]

[1]*Faculdade de Engenharia da Universidade do Porto, Porto, Portugal*
[2]*Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC), Porto, Portugal*
[3]*INESC-ID, Lisboa, Portugal*
[4]*Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal*

Abstract

Argument retrieval is a prominent topic in the context of current natural language processing applications. The task focuses on creating models that can retrieve coherent and strong arguments from textual sources. This technology can help individuals build an informed opinion about a controversial topic or support a particular stance on a debate. In this context, the Touché Task 1 was proposed within the scope of Conference and Labs of the Evaluation Forum 2022 (CLEF 2022), based on argument retrieval for controversial questions. We chose to compete with a sparse search. Despite ranking 3rd on relevance, 5th on quality and 4th on coherence, we concluded that our results are limited by our data arrangement process. The purpose of the task was to retrieve the most argumentative and relevant pairs of sentences, which could be formed with sentences from the same argument or not. Our approach focused on forming sentence pairs from the same argument, and achieved scores of 0.772 for quality, 0.651 for relevance, and 0.378 for coherence.

Keywords

Argument retrieval, Controversial questions, Sparse representation

1. Introduction

Currently, Natural Language Processing (NLP) is prevelant in almost everything done on the web. There are many research fields within the scope of NLP, such as Information Retrieval (IR), a process of accessing and retrieving the most pertinent information given a user query. More specifically, the focus of this work is on retrieving arguments for controversial topics.

Applications dealing with natural language use machine learning models for different purposes, such as speech analysis and understanding, generation of human-readable text, and information retrieval. The Touché lab[1], proposed in the scope of the Conference and Labs of

CLEF'22: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
up201604014@fe.up.pt (B. C. Moreira); hlc@fe.up.pt (H. Lopes Cardoso); bruno.g.martins@tecnico.ulisboa.pt (B. Martins); fabio.goularte@inesc-id.pt (F. Goularte)
0000-0002-3307-0078 (B. C. Moreira); 0000-0003-1252-7515 (H. Lopes Cardoso); 0000-0002-3856-2936 (B. Martins); 0000-0003-4378-8734 (F. Goularte)

CEUR Workshop Proceedings (CEUR-WS.org)

[1]https://webis.de/events/touche-22/

the Evaluation Forum (CLEF), focuses on the task of retrieving arguments. The goal of the Touché lab is to motivate the NLP community to develop or improve upon existing technologies for argument mining and argument analysis. This year, this lab organized three shared tasks: Task 1 focused on argument retrieval for controversial questions; Task 2 on argument retrieval for comparative questions; and Task 3 focused on image retrieval for arguments.

Our team focused on the first task, where the goal is to retrieve and rank relevant sentence pairs from a collection of arguments, given a query about a controversial topic. Touché Task 1 at CLEF 2022 [1] is similar to the Touché Task 1 at CLEF 2021 [2], where given a query systems should retrieve relevant arguments. In contrast, in the 2022 edition, the participating teams have to retrieve a pair of sentences gathered from the arguments collection. This significant alteration increases the complexity of the task since the pair of sentences must be coherent, each sentence in the pair must be argumentative, and together the sentences should provide a summary of the argument from which they are retrieved. For example, if a user searches for "Libertarianism", an example of an output would be the pair of sentences "For example, Jim Babka, from Libertarian organization Downsize DC said they have had some very successful alliances with groups who would never describe themselves as conservative" and "Libertarians cooperate with many non right wing organizations".

The results obtained on preliminary experiments with data from the 2021 edition were encouraging; therefore, we decided to follow the same approach to the 2022 edition, which relies on Pyserini sparse search with BM25, further detailed in Section 3.

We have submitted four different runs for this shared task and leveraged the available evaluation files for the 2022 Task 1 to evaluate our approach. We achieved in our best run 0.772, 0.651, and 0.378 for Quality, Relevance, and Coherence scores, respectively. Human assessors manually define these metrics and evaluate the output's quality, relevance, and coherence, given a specific query. With these results we achieved third place in relevance, fourth in coherence and fifth in quality.

2. Background

Argument retrieval is a topic of growing interest. There have been some tested approaches in previous editions of this Touché task 1. As described in Bondarenko et al. [2], the usual approaches are based on DirichletLM [3] [4] and BM25 [5] models combined with WordNet [6] for query expansion.

BM25 is possibly one of the most used and essential functions in information retrieval and is used to estimate the relevance of documents to an input query. It is a non-linear combination of three document attributes: document frequency, document length, and term frequency.

Elrond, the team with the highest relevance score, utilized a DirichletLM-based retrieval model, using also the Krovetz stemming method [7], and excluded stop words. In the case of *Heimdall* [8], which was the team with the highest quality score, they utilized a DirichletLM but integrated with a topical relevance analysis using Universal Sentence Encoder and k-means clustering, and finally, a support vector regression model for argument quality trained on the Webis-ArgQuality-20 corpus[2].

[2] https://zenodo.org/record/3780049

Touché Task 1 2022 submissions were evaluated manually, by humam assessors, based on each submitted run's sentence pairs relevance and quality scores. This entails analyzing if each sentence is argumentative, if both sentences in the pair are coherent and whether this pair of sentences form a summary of the respective arguments.

3. Approach

We make use of Pyserini [9], a python toolkit for information retrieval search. Our approach focuses on Pyserini's sparse retrieval – via integration with Anserini IR toolkit built on the Lucene search library [10], both for indexing the data and performing the search, since it performs retrieval with BM25 ranking using bag-of-words representations.

We used last year's evaluation files since we didn't have this year's data to calculate our approach's quality and relevance scores. To assess our runs on these evaluation files, we used the trec_eval[3] tool which is the standard tool used by the TREC[4] community for evaluating a run, given the topics and the evaluation files. Figure 1 shows our approach composed by five steps: (1) Input, (2) Pre-processing, (3) Data, (4) Data processing, and (5) Output.

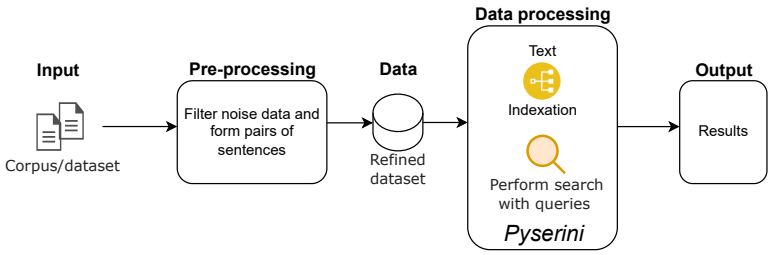


Figure 1: Our approach to the Touché Task 1 problem.

Our team submitted four different runs. All these runs were generated using Pyserini sparse representations, given the good results in the experiments performed on last year's data. All four runs submitted were generated using two variations of data and two variations of queries, as described in Section 3.3.

Since there was no available method to test the approaches before submitting, we needed to test the planned strategies with the available data from last year's task. With this in mind, we performed a series of experiments using the quality and relevance data available which provided an overview of our approaches.

The data was organized in two forms to evaluate the best arrangement of data for this task. The argument layouts tested were: (PC) Conclusions and Premises concatenated and (PCT)

[3] https://github.com/usnistgov/trec_eval
[4] https://trec.nist.gov/

Conclusions, Premises, and Topic concatenated.

In Table 1, our team results for each argument layout are compared with the top 3 results from last year's edition, both for quality and relevance. We noticed that using premise and conclusion concatenated provided a better quality score than premise conclusion and topic, but using the latter results in a slightly better relevance score. This occurs because introducing the topic sentence to the premise will negatively affect the semantic quality of the sentence since these sentences might not be perfectly aligned. On the other hand, concatenating this topic adds information to the premise; this means that it provides more details on the discussion topic the sentence is inserted in, resulting in a better relevance score.

Table 1
Quality and relevance NDCG@5 scores for our approaches, and to the top 2 teams from last year's competition.

Approach	Quality	Relevance
PC	**0.8116**	0.6742
PCT	0.7881	**0.6838**

Team Name	Quality
Heimdall	**0.841**
Skeletor	0.827

Team Name	Relevance
Elrond	**0.720**
Pippin Took	0.705

3.1. Input

The data provided for this year's task is separated into two different files. The first one is the data to be used for retrieval, and the other one contains the queries.

Each element in the data file is an *argument* containing five fields: (1) *id*, (2) *conclusion*, (3) *premises*, (4) *context*, and (5) *sentences*.

The sentences field is used for this task – the aim is to retrieve a pair of sentences formed with sentences from the same or different argument. The *sentences* field aggregates the *conclusion* of the argument and each *sentence* in its premises, each identified by a unique *id*.

The queries file contains the *topic id*, the *title* which is the question itself, the *description*, and the *narrative*. The description provides an explanation and a context for the given question, while the narrative describes the desired outcome.

3.2. Pre-Processing

Our team chose to follow the above mentioned and tested approach to the 2021 Task 1, which relies on pyserini sparse to index and perform the search of relevant documents for a given query. As it was already mentioned, this approach consists of five steps: (1) Input, (2) Pre-processing, (3) Data, (4) Data processing, and (5) Output, as show in Figure 1. The data processing phase consists in generating the data in a format amenable for an information retrieval setup. This task requires that a query's output be formed as a pair of sentences. The "sentences" field in the argument contains all the sentences and respective id; therefore, we processed this field to develop all possible pairs of sentences from the same argument.

For example, the argument with id **Sf9294c83-A9a4e056e** contains three sentences: two premises and one conclusion. After being processed, these sentences generated three different pairs of sentences (see Figure 2).

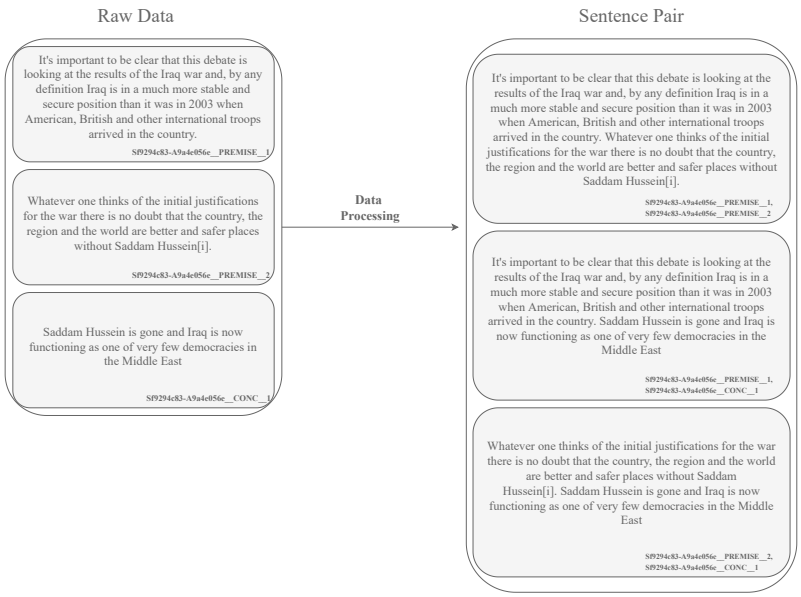


Figure 2: Processing raw data to form pairs of sentences. Each sentence contains an id and content; both the sentence id and the sentence content are concatenated to form the sentence pair. The number of pairs formed depends on the number of sentences in the array.

3.3. Data

The query data (see Listing 1) was used in two ways. The first and more straightforward consists of using only the query. The second, an expanded query version, includes the query concatenated with the narrative and the description.

```
<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>
```

Listing 1: Example of a topic in the query data.

The new generated data, which will be indexed by Pyserini, is composed of documents containing each an id and a sentence pair.

Argument sentences were paired following two different arrangements, as shown in Figure 3.

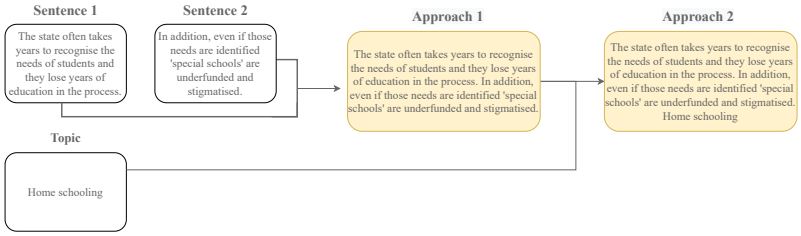


Figure 3: Representation of the two different arrangements of sentences used.

The first approach consists in using both sentences concatenated, while the second approach concatenates both the sentences and the topic.

As shown in Figure 4, with these different approaches for both queries and sentence pairs, we submitted four different runs:

- pyserini_sparse_v1: pairs of sentences, and query with title, description and narrative
- pyserini_sparse_v2: pairs of sentences, and query with title only
- pyserini_sparse_v3: pairs of sentences with topic concatenated, and query with title, description and narrative
- pyserini_sparse_v4: pairs of sentences with topic concatenated, and query with title only

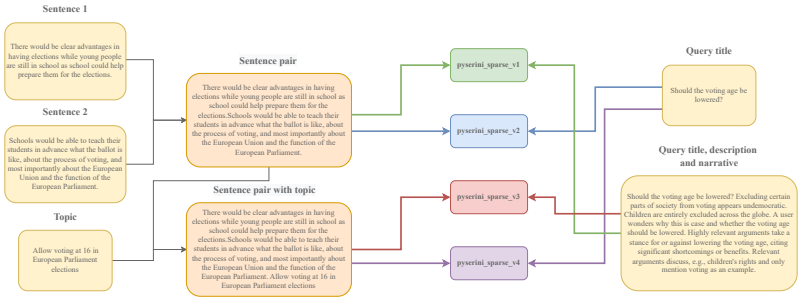


Figure 4: Representation of the four combinations of data used with an example.

3.4. Data Processing

The Pyserini toolkit, which is responsible for indexing the new data and performing the search with the given queries, allows a variety of approaches. Given our time and resource limitations, the best approach to this task is to use sparse representations. Pyserini sparse representations produces smaller indexes than dense search. Pyserini also includes a hybrid approach that makes use of both sparse and dense indexes; however, given our approach that generates a high number of sentence pairs, we did not generate dense representations for our sentence pairs, and thus do not make use of the hybrid approach.

The Pyserini toolkit receives the data as an input, which it indexes and uses to search for documents given a query. Both the index and search processes use the Lucene library; for the search process, we specified how many hits for each question we need (1000) and which scoring function to use, which in our case is BM25 with default values from Pyserini.

3.5. Output

The post-processing step consists of cleaning Pyserini's output to match the structure requested for Touché Task 1, which is the standard TREC format:

qid stance pair rank score tag

Even though the format of the document, created in the previous step, is the correct one, our data is not ready to submit. In this step, to provide all the required data for the submission, we iterate through the results file to replace the default values for stance (*Q0*) and tag (*Anserini*) produced by Pyserini. These values are replaced with the correct stance of the pair of sentences and our team tag *Bruce-Banner*, respectively, resulting in a new and valid results file.

4. Experimental Evaluation

After the release of Touché Task 1 2022 evaluation files we performed experiments to evaluate the performance of our runs (see Table 2). The evaluation files contained the top pairs of sentences, for each query, ranked according to their general topical relevance, their argument quality and their coherence. Eleven teams participated in this Task, and according to these metrics, our team **Bruce-Banner** was placed in third in relevance, fourth in coherence and fifth in quality, as shown in Table 3.

Table 2
Quality, relevance and coherence results for our four runs submitted in Touché Task1 2022.

Run	Quality NDCG@5	Relevance NDCG@5	Coherence NDCG@5
pyserini_sparse_v1	**0.772**	0.641	**0.378**
pyserini_sparse_v2	0.701	0.580	0.353
pyserini_sparse_v3	0.760	**0.651**	0.354
pyserini_sparse_v4	0.709	0.586	0.357

Table 3
Touché Task 1 2022 Quality, Relevance and Coherence teams scores.

Team	Relevance NDCG@5
Porthos	0.742
Daario Naharis	0.683
Bruce Banner	**0.651**
D Artagnan	0.642
Gamora	0.616
Hit Girl	0.588
Pearl	0.481
Gorgon	0.408
General Grievous	0.403
Swordsman	0.356
Korg	0.252

Team	Quality NDCG@5
Daario Naharis	0.913
Porthos	0.873
Gamora	0.785
Hit Girl	0.776
Bruce Banner	**0.772**
Gorgon	0.742
D Artagnan	0.733
Pearl	0.678
Swordsman	0.608
General Grievous	0.517
Korg	0.453

Team	Coherence NDCG@5
Daario Naharis	0.458
Porthos	0.429
Pearl	0.398
Bruce Banner	**0.378**
D Artagnan	0.378
Hit Girl	0.377
Gamora	0.285
Gorgon	0.282
Swordsman	0.248
General Grievous	0.231
Korg	0.168

Even though our team placed third in relevance, fourth in coherence, and fifth in quality, we believe that our approach would benefit from a different data arrangement process. In our approach, we chose to form all possible pairs of sentences within the same argument, but sentences from different arguments were also accepted for the task. However, in the quality evaluation file provided by the task, sentence pairs formed with sentences from different arguments vary between 35.6% and 46.9% across topics. Still, while our approach does provide sentence pairs that have been well-ranked, there is a limitation on the provenance of pairs we were able to form, which directly determines the retrieved entries.

5. Conclusions

Our team decided to participate in this Task with a sparse search approach. Our data processing stage influenced our approach. For this Task, it was allowed and even desired to form pairs with sentences from different arguments, but we decided to form all the possible pairs of sentences from the same arguments. Thus, various pairs of sentences were not indexed, resulting in a poorer collection of documents to retrieve.

As future work it would be interesting to:

- Form data with pairs of sentences, including pairs from different arguments. Sentence selection needs to be done carefully; otherwise, if all the possible pairs of sentences are formed, the resulting collection will be enormous, which may lead to the impossibility of using pyserini search or bring enormous computational costs;
- Follow a different approach resulting in lesser data so that pyserini hybrid retrieval can be used, i.e., using pyserini sparse to retrieve the top 2000 pairs of sentences and then using pyserini hybrid to re-rank those retrieved pairs;
- Re-rank the arguments returned with a similar score using a fine-tuned BERT model to assess argument quality [11].

It would also be interesting to analyze the other teams' approaches, such as *Porthos* and *Daario Naharis*, which achieved scores better than ours.

References

[1] 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.

[2] 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.

[3] D. J. MacKay, L. C. B. Peto, A hierarchical dirichlet language model, Natural language engineering 1 (1995) 289–308.

[4] J. M. Ponte, W. B. Croft, A language modeling approach to information retrieval, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98, Association for Computing Machinery, New York, NY, USA, 1998, p. 275–281. URL: https://doi.org/10.1145/290941.291008. doi:10.1145/290941.291008.

[5] S. Robertson, H. Zaragoza, The probabilistic relevance framework: Bm25 and beyond, Foundations and Trends® in Information Retrieval 3 (2009) 333–389. URL: http://dx.doi.org/10.1561/1500000019. doi:10.1561/1500000019.

[6] C. D. Fellbaum, Wordnet : an electronic lexical database, Language 76 (2000) 706.

[7] R. Krovetz, Viewing morphology as an inference process, Artificial intelligence 118 (2000) 277–294.

[8] L. Gienapp, Quality-aware argument retrieval with topical clustering, Working Notes of CLEF (2021).

[9] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, R. Nogueira, Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations, in: Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021), 2021, pp. 2356–2362.

[10] E. Hatcher, O. Gospodnetic, Lucene in action (in action series), Manning Publications Co., 2004.

[11] S. Gretz, R. Friedman, E. Cohen-Karlik, A. Toledo, D. Lahav, R. Aharonov, N. Slonim, A large-scale dataset for argument quality ranking: Construction and analysis, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 7805–7813.