Czech
Technical University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Multi-stage Methods for Document Retrieval in the Czech Language

Master Thesis of
Bc. Barbora Dědková

Supervisor: **Ing. Jan Drchal, Ph.D.**
Field of study: **Open Informatics**
Subfield: **Artificial Intelligence**
August 2021

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Dědková Barbora** Personal ID number: **466307**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence**

II. Master's thesis details

Master's thesis title in English:

Multi-stage Methods for Document Retrieval in the Czech Language

Master's thesis title in Czech:

Víceúrovňové metody pro document retrieval nad českými texty

Guidelines:

The task is to develop methods of document retrieval to be deployed in the fact-checking scenario. Focus on Czech language models. Explore combinations of methods for two (or even more) retrieval stages: document preselection and reranking. Consider both classic NLP approaches (e.g., BM-25) and state-of-the-art Transformer networks.
1) Familiarize yourself with document retrieval algorithms aimed for the fact-checking task (and the related question-answering).
2) Work with the Czech Wiki FEVER and ČTK data, both supplied by the supervisor.
3) Select appropriate methods and modify them to treat the supplied data. Experiment with different setups such as two-tower architectures and cross-attention networks. Compare different loss functions.
4) Evaluate and compare the methods. Focus on precision and recall measures.

Bibliography / sources:

[1] Das, Rajarshi, et al. "Multi-step retriever-reader interaction for scalable open-domain question answering." arXiv preprint arXiv:1905.05733 (2019).
[2] Pang, Liang, et al. "Deeprank: A new deep architecture for relevance ranking in information retrieval." Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 2017.
[3] Chang, Wei-Cheng, et al. "Pre-training tasks for embedding-based large-scale retrieval." arXiv preprint arXiv:2002.03932 (2020).
[4] Thorne, James, et al. "FEVER: a large-scale dataset for fact extraction and verification." arXiv preprint arXiv:1803.05355 (2018).
[5] Thorne, James, et al. "The fact extraction and verification (fever) shared task." arXiv preprint arXiv:1811.10971 (2018).

Name and workplace of master's thesis supervisor:

Ing. Jan Drchal, Ph.D., Department of Theoretical Computer Science, FIT

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **21.02.2021** Deadline for master's thesis submission: _____

Assignment valid until: **19.02.2023**

Ing. Jan Drchal, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

Acknowledgements

I would like to thank the supervisor of this thesis, Ing. Jan Drchal, Ph.D. Thank you for your patience, guidance and advice through out my work at the AIC Fact-checking project and thank you for the opportunity. Thanks to all my coworkers for always finding the time to help and encourage me. I am also very grateful to my family and my boyfriend for supporting me in the most difficult times of my studies.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used. I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague, 13. August 2021

Abstract

This work tackles the task of large-scale document retrieval by utilizing multi-stage methods for document retrieval. It combines two retrieval stages: document preselection and document reranking. It considers classic DR approaches such as TF-IDF and BM25 in the first stage and modern Transformer networks in the second stage. It evaluates and compares the Cross-Attention and Two-Tower architectures in various setups.

Keywords: document retrieval, fact-checking, transformers, multi-stage retrieval, TF-IDF, BM25

Supervisor: Ing. Jan Drchal, Ph.D.

Abstrakt

Tato práce se zabývá úkolem vyhledávání dokumentů ve velkém měřítku pomocí vícesúrovňových metod pro hledání dokumentů. Kombinuje dvě fáze vyhledávání: předvýběr dokumentů a přeřazení dokumentů. Zvažuje klasické přístupy vyhledávání dokumentů jako jsou TF-IDF a BM25 v první fázi a moderní Transformers sítě ve druhé fázi. Vyhodnocuje a porovnává architektury Cross-Attention a Two-Tower v různých nastaveních.

Klíčová slova: vyhledávání dokumentů, fact-checking, transformers, víceúrovňové vyhledávání, TF-IDF, BM25

Contents

Figures

Tables

Chapter 1
Introduction

With every journal, article, sentence we read, we stand in front of a decision. A decision whether to believe the information we have just read or not trust it and possibly find more information on the topic. That may seem rather critical-minded but with the ever-increasing number of information available online the need for verification surges.

Misinformation and *disinformation* are a rising challenge of the digital time we are living in. Both are an act of spreading false, inaccurate, or misleading information. However the first is without the intention of misleading, the second is knowingly sharing the information with an ulterior motive such as economic gain, public manipulation, and others. From disinformation it is a small step to *fake news*, which is the same concept but sensational, often emotionally charged, and can be completely fabricated.

The terms misinformation, disinformation, malinformation, or in general terms regarding the veracity and overall flow of information and communication need clear definitions, especially recently with the COVID19 pandemic [Baines et al., 2020] and associated infodemic (ubiquity of false or misleading information in digital and physical environments during a disease outbreak) [WHO, 2021] in mind.

The solution to this sociological problem is complex. To prevent the spread of false information we need the right mix of interventions, including support of independent journalists and fact-checking platforms, efforts to boost digital media literacy, new policies to prevent harm in the digital information ecosystems, etc. Fact-checking (performed by humans or automated) is one of many ways to tackle the problem.

1.1 Fact-checking

Fact-checking is a process of verification of factual information. It seeks to classify a claim (fact) as truthful, false, or another category suitable for the task at hand. For some, fact-checking means verifying the truthfulness of factual political statements such as numerical information, historical facts, or past acts of politicians. For the English-speaking world, it is the site PolitiFact[1], and for the Czech-speaking, it is Demagog[2]. In the case of Demagog, there are four categories: true, false, misleading, and unverifiable. PolitiFact's six categories form a scale of how truthful the information is.

[1] https://www.politifact.com/
[2] https://demagog.cz/

(a): PolitiFact's *Truth-O-Meter*

(b): Demagog's verification examples labeling *true*, *unverifiable*, and *false* in this order.

Figure 1.1: Examples of classification labels for political claims.

Another Czech fact-checking site example is Manipulátoři[3]. It is also dedicated to verifying political statements or whole debates but it does not use the classifying approach as the previous two. They find and reveal false claims and disinformation and justify the reasoning behind it but do not try to select labels.

All three sites perform their fact-checking manually, which means collecting all the evidence and composing it to assess the truthfulness of a claim. It would be highly beneficial to automate the task or at least parts of it.

Another examples of English fact-checking sites are FactCheck.org[4] that monitors the factual accuracy of political speeches, debates, or news stories. Or Snopes[5] that provides evidence for fact-checking urban legends, folklore, rumors, and misinformation.

[3] https://manipulatori.cz/
[4] https://www.factcheck.org/
[5] https://www.snopes.com/

1.2 Automated Fact-checking

The process of fact-checking requires a combination of various NLP (Natural Language Processing) tasks ranging from document retrieval, question answering, natural language inference, etc. There are many approaches to the task and also many levels of automation. The work of [Thorne and Vlachos, 2018] provides an overview of common concepts, datasets, and modelling approaches of automated fact-checking. They discuss what *inputs*, *output*, and *evidence* the automated pipeline expects and produces.

The input is the information that is being fact-checked. The most popular input is simply a textual claim. There can be various types such as numerical claims, position statements, entity properties, quote verifications, etc. Various approaches may process the claim differently, eg. using Name Entity Recognition for disambiguation, while others rely only on correct evidence retrieval.

The output can be as simple as labeling a claim true or false or as complex as producing a justification along with labeling where the number of labels can be arbitrary.

Some approaches may use knowledge graphs as the sources of evidence, others a less structured data such as news articles, scientific journals, encyclopedia articles, etc.

This thesis focuses on the **evidence retrieval** part of the fact-checking pipeline. We examine document retrieval methods that select documents relevant to a query. The query is the information intended for fact-checking and the documents are paragraphs or sentences, that contain the evidence required for the justification of the verification.

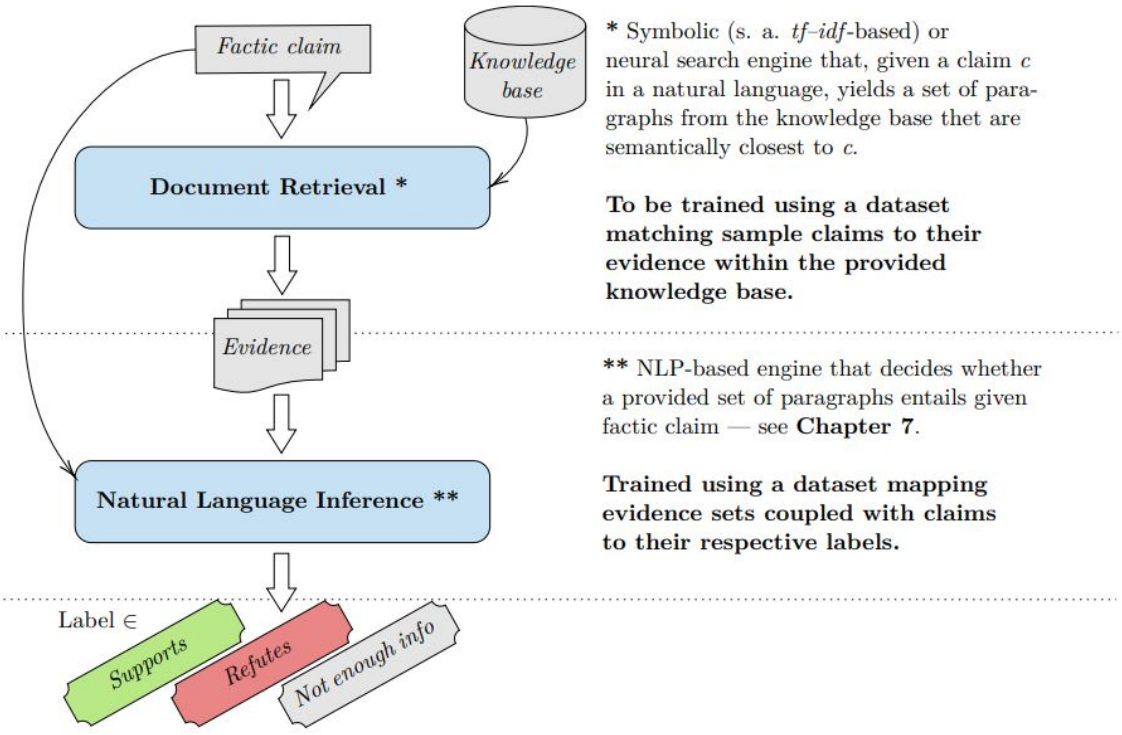

Figure 1.2: Example of a fact-checking pipeline, reprinted from [Ullrich, 2021].

1.3 Project

This thesis is a part of the AIC Fact-Checking project. The goal is to establish a strong baseline for the fact-checking task in the Czech language. The work of [Rýpar, 2021] focuses on large-scale retrieval as well as this work. [Gažo, 2021] focuses on scaling the state-of-the-art neural models for longer inputs. [Ullrich, 2021] concentrates on dataset collection and processing and establishing a baseline for natural language inference (assigning a label to the claims).

Our project group is growing in numbers and also in the number of challenges to conquer. Our focus is on journalist articles and reports however, we examine methods for DR on other datasets such as the FEVER dataset described in chapter 2.

1.4 Related Work

The multi-stage approach to document retrieval is not new. There are works already exploiting this idea. [MacAvaney et al., 2020] suggests a Precomputing Transformer Term Representations for precomputing part of document term representations before inference time and then merging them with the query representation to compute the final ranking. Their idea predominantly reduces the query-time latency which is also a goal in multi-stage DR.

The work of [Matveeva et al., 2006] is closer to our approach in the sense of reranking the documents multiple times. It presents multiple nested rankers that iteratively improve the ranking scores of the document. The rankers use the RankNet algorithm for learning ranking functions with gradient descent.

The FEVER shared tasks [Thorne et al., 2018b], [Thorne et al., 2018c], [Aly et al., 2021] inspired a lot of work in the fact-checking domain. The first task introduced the problem to the wider community and the challenges are set higher and higher with the following tasks. The newest task introduces unstructured data into the picture which until now was a great challenge. It also inspires fact-checking in other less represented languages than English. We can see that on the efforts in Danish [Binau and Schulte, 2020] fact-checking or in the surge in the creation of fact-checking datasets in other languages [Shahi and Nandini, 2020].

1.5 Thesis Outline

- **Chapter 1** introduces the fact-checking domain and its challenges. It presents some already employed fact-checking sites and reviews related work.
- **Chapter 2** familiarizes us with the two datasets used for this work. It describes the dataset collection and its properties.
- **Chapter 3** dives deep into the background of document and information retrieval, presenting classical approaches as well as modern approaches.
- **Chapter 4** describes the proposed solution to the multi-stage method approach to document retrieval. Goes through the evaluation metrics and method used in our work.
- **Chapter 5** details the setup of experiments and presents the results.

- **Chapter 6** concludes the work with a summarization of our efforts and results and discusses future work.

Chapter 2
Datasets

This chapter describes the datasets we created and used in the fact-checking domain. In the Czech language, there are not any such datasets for the task therefore, we had to search for inspiration. In the English language (and other generally more used languages than Czech), there are usually various datasets for each NLP task. We took inspiration from the **FEVER** [Thorne et al., 2018a] dataset, further described in the next section, and created a Czech version called **FEVER CS** (sec. 2.2).

Another dataset that we used is the **ČTK** dataset (sec. 2.3) created from articles provided by **Česká Tisková Kancelář** (Czech News Agency). A set of annotators from the Faculty of Social Sciences at Charles University helped us create a collection of claims (containing labels, evidence sets, etc.) such as the claims from the FEVER dataset. We used these two Czech datasets for all experiments in this thesis.

For a detailed description of creation, localization and properties of the datasets see [Ullrich, 2021].

2.1 FEVER

FEVER [Thorne et al., 2018a], the Fact Extraction and VERification dataset is a base point for many attempts to create a dataset for factual verification in other lower resource languages, for example, Danish [Binau and Schulte, 2020], [Nørregaard and Derczynski, 2021]. The FEVER shared task [Thorne et al., 2018b] also prompted many submissions, efforts, and development in this NLP task.

The FEVER dataset is created by extracting and altering sentences from Wikipedia. The sentences form claims that are subsequently verified and classified as `SUPPORTED`, `REFUTED`, or `NOT ENOUGH INFO` by annotators.

We say that a claim is `VERIFIABLE` if it is supported or refuted, otherwise it is `NOT VERIFIABLE`. When a claim is verifiable, then it has at least one *evidence set* to justify its label. A claim can have multiple evidence sets however, every set alone is sufficient to verify the claim.

The construction of the dataset consists of two parts:

1. **Claim Generation**: Extraction of information from Wikipedia where the annotators create claims from approximately 50,000 most popular pages. Each claim contains a single piece of information from a certain Wikipedia page. It can be arbitrarily complex, and the annotators can draw from additional knowledge in the form of a dictionary. The annotators also mutate the claims by rephrasing, negating, generalizing, specifying, or substituting parts of the claims, creating new ones that often need new evidence or even different labels.

2. **Claim Labeling**: Classifying a claim as `SUPPORTED` or `REFUTED` and providing the evidence. If there is no evidence or not enough to verify, the claim is classified as `NOT ENOUGH INFO`. The annotators can find the evidence on any Wikipedia page. They select one or multiple sentences that form the evidence set.

```
id: 119669
verifiable: "VERIFIABLE"
claim: "Venus takes 22 Earth days to orbit the Sun."
evidence: [[[140651, 155801, "Venus", 0]]]
label: "REFUTED"
```

Figure 2.1: FEVER annotation example.

The claims are stored in dictionaries, where each dictionary has five keys: id, claim, label, evidence, and verifiable flag. In figure 2.1 we can see an example of a dictionary representing a claim that is classified as `REFUTED`. The `evidence` is a list containing sets of evidence sentences. In the example, there is only one set that contains only one evidence sentence. The evidence sentence is structured as `[annotation_id, evidence_id, article_wikiid, sentence_index]`. Meaning that the sentence with index 0 on the page *Venus* is the evidence sentence in the example. If we look it up in the Wikipedia articles it states: *Venus orbits the Sun every 224.7 Earth days*, which counters the claim.

After the creation of the data, various data validation techniques were used. Such as selecting 1% of data to be annotated by super-annotators who did not have any time restrictions per claim. Or validation of a small amount of data by the authors.

After this process they achieved the following split sizes:

Split	SUPPORTED	REFUTED	NEI
Train	80,035	29,775	35,639
Dev	3,333	3,333	3,333
Test	3,333	3,333	3,333
Reserved	6,666	6,666	6,666

Table 2.1: FEVER split size

2.2 FEVER CS

We created a dataset deriving from the English FEVER dataset with the help of Czech Wikipedia. We adopted the FEVER annotation practices and decided to use the same three labels. We created a similar but smaller Czech verification dataset.

First, we took every evidence used in the original FEVER dataset and mapped the English Wikipedia page to the Czech one using **MediaWiki API**[1]. It utilizes the links at every Wiki page to the same page in foreign languages. The data loss was not significant because most English pages have their Czech opposite. More detail in [Ullrich, 2021].

As a knowledge base, we used a **Wiki dump**[2] from June 2020, which was parsed and tokenized to sentences. The vast majority of evidence is located in the abstracts of articles, thus we only used those resulting in every article being one paragraph long.

[1] `https://www.mediawiki.org/wiki/API:Main_page`

[2] available at `https://dumps.wikimedia.org/` or `http://bertik.net/cswiki`

Followed the translation of claims via Google Cloud Translation API, then normalization of both claims and knowledge base, and finally splitting the dataset into train, dev, and test parts. The split sizes are roughly comparable to the ones in Tab. 2.1.

Split	SUPPORTED	REFUTED	NEI
Train	53,542	18,149	35,639
Dev	3,333	3,333	3,333
Test	3,333	3,333	3,333

Table 2.2: FEVER CS split size

The format of the dataset is similar. Only difference being that the Czech dataset also keeps the original English claims in the dictionary, for better orientation between the two datasets. The complete annotation is analogous to Fig. 2.1 only with the claim divided into two entries - `claim_en` and `claim_cs`.

2.3 ČTK

Our goal is to work with and fact-check **news articles** and for that we needed another Czech dataset for verification that would not be dependant on Wikipedia this time. We decided to create a new one with a different knowledge base from the area of journalism. We took advantage of news reports from the archive provided by the **Czech News Agency** (ČTK).

In total, there are over 11 million articles from between the years 2000 and 2019. The number was reduced to 11 million from 15 million after cleaning the data. We eliminated sports results and daily news summaries due to frequent occurrences of tables, charts, and other structured data. Raw textual data fit better for our task.

2.3.1 Collection of the dataset

The collection of the dataset took place at the **Annotation platform**[3] [Ullrich, 2021] and was performed by around 170 students from the Faculty of Social Sciences at Charles University. There were three rounds of annotating during which we collected **3,293 cross-annotated claims** with labels and evidence sets. After each round, the Annotation platform was adjusted to help avoid the most common mistakes or misunderstandings of the annotators.

We followed the general directions of such dataset creation as in FEVER [Thorne et al., 2018a]. The FEVER dataset was created from the Wikipedia knowledge base and the ČTK dataset from *news reports*. Wikipedia articles often have a summary at the beginning, causing the most useful information for the formation of evidence to be located predominantly there. The news reports, on the other hand, do not share this advantage. Therefore we approached all paragraphs from the reports equally and claims along with evidence could be extracted from all. Despite these differences in data, the process could otherwise run similarly.

[3] `https://fcheck.fel.cvut.cz/site/login`

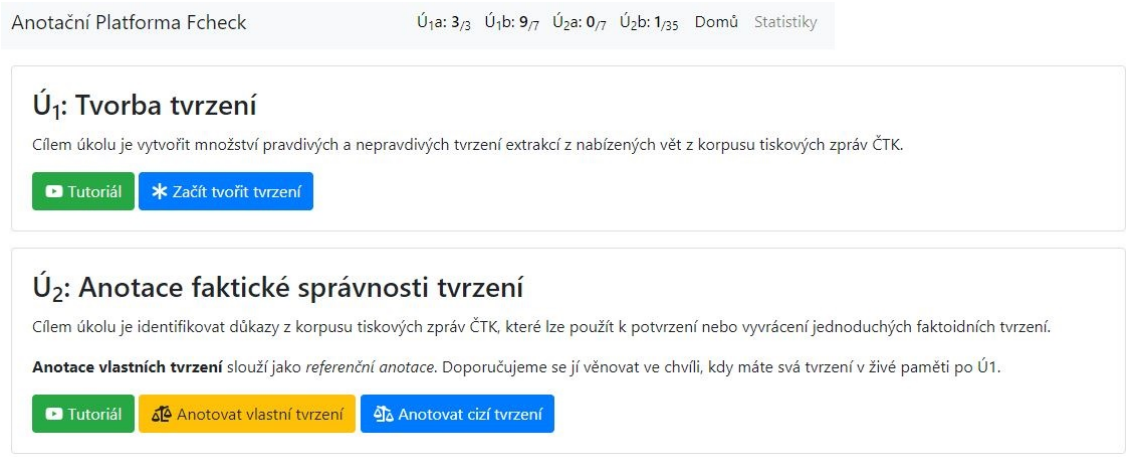

Figure 2.2: Annotation platform for the ČTK dataset. *Ú1* task - claim generation, *Ú2* task - claim labeling both with tutorials for the annotators.

Before the work of the annotators could begin, we preselected *source paragraphs* for claim generation. We skimmed the articles, selecting paragraphs that contained any verifiable information. The annotators then received these preselected texts as the initial building stone.

Firstly, the claim generation task includes claim extraction and claim mutation. During the **claim extraction** task, the annotators were presented with a paragraph from which they had to extract a single factoid claim without using their world knowledge. They could use knowledge from other paragraphs from the same article.

After the claim extraction follows **claim mutation**. The annotator has to mutate the original claim using a set of given approaches such as *generalization*, *negation*, *substitution*, *paraphrase*, and others. He can use his world knowledge as well as the source article.

Lastly, the task of **claim labeling** takes place. The annotator assigns a veracity label from the three categories `SUPPORTED`, `REFUTED`, `NOT ENOUGH INFO` as in the FEVER dataset. Then selects evidence justifying the label choice. Every annotator has to label part of claims generated by himself and part of claims generated by other annotators. This practice also ensures that every claim is annotated at least one time, but usually more.

```
id: 200
verifiable: "VERIFIABLE"
claim: "Radnice Postoloprt nechává chátrat kapli čtrnácti svatých pomocníků."
evidence: [[[-1, 33, 20010711F01495_1, -1]],
     . [[-1, 174, 20010711F01495_1, -1 ]]]
label: "REFUTED"
```

Figure 2.3: ČTK annotation example with two evidence sets. Both contain an evidence sentence from the same article.

The annotated examples are in the same format as the FEVER dataset in fig.2.1.

2.3.2 Dataset versions

After the data collection, the dataset was cleaned manually to resolve every conflict in labeling. Many were caused, e.g. by temporal reasoning which can be resolved by using timestamps. We addressed the most common problems in the annotation by informing the

annotators in the following rounds of annotations and by making the Annotation platform clearer.

The collection of the data was **cross-annotation** driven which means that each claim was labeled by more annotators. Also resulting in every verifiable claim having at least one evidence set but many having two or more. That inevitably leads to disagreements in some annotations (labeling or evidence sets).

We measured the **Fleiss** κ score [Fleiss, 1971] for the k-way inter-annotator agreement. It measures nominal scale agreement between a fixed pair of raters (annotators). We calculated it to be 0.61 for a 5-way agreement which is encouraging compared to 0.6841 for the original FEVER dataset.

The process of cleaning reduced and refined the dataset, also introducing various versions of the data. The first version after the first round of annotations was experimental and helped form the final dataset. The version used in this thesis and in all experiments in Chapter 5 is called **ČTK v2.1** and corresponds to the split size in Tab. 2.3. It was generated after all three rounds of annotations. There is also another augmented version **ČTK v2.1nli** used in [Ullrich, 2021].

Split	SUPPORTED	REFUTED	NEI
Train	1,132	519	473
Dev	100	100	100
Test	200	200	200

Table 2.3: ČTK CS split size

Chapter 3
Background

This chapter provides background to Document Retrieval (DR) methods from the classic approaches in section 3.1 to the modern Transformer networks, in section 3.2, that are a popular choice when treating any Natural Language Processing tasks nowadays.

Document retrieval in information science is the task of matching a user request (query) against a document. The required output of a DR pipeline is a relevance-ranked list of top k documents selected from a broad collection of documents. The structure of the documents varies from newspaper articles, website pages to database records. The query can also be arbitrarily complex, from one word to multiple sentences long. DR systems are used daily, for example, in the form of web search engines.

A document retrieval system consists of two main tasks:

1. Matching - Finding relevant documents to the query.
2. Ranking - Evaluating and sorting the matching documents.

3.1 Traditional Document Retrieval

In the traditional document retrieval pipeline (see Fig.3.1), some processes such as *document processing* can run in advance before the query is posed. After normalizing the document to a predefined format, it is broken down into retrievable units. Those can be paragraphs, chapters, or even full documents. That is followed by deleting *stop words* (i.e., prepositions, conjunctions, articles) and *stemming* (removing suffixes), which results in higher frequency counts due to the lack of all morphological variants of a word. The last step is producing an *inverted file*, a sorted array consisting of all indexable terms, their weights, and links to the documents.

After document processing, we also need a representation of the query. The *query processing* is done in real-time and partly consists of similar steps as document processing. It begins with *tokenization* of the query terms (deletion of stop words, stemming, and phrase recognition), followed by their *expansion*. The expansion introduces synonyms to the terms and also highly associated terms. It improves the recall measure but might reduce precision depending on the length of the query.

The query and document representations are vectors with information about term frequency and weights. Having these vectors, we search the inverted file for documents containing any of the query's terms. We compute a *similarity score* between the query vector and the candidate document vectors using different matching functions depending on the DR model we are using. Finally, we order the documents decreasingly by the similarity score and return the list to the user. The user then has the opportunity to modify the query, thus starting the pipeline anew.

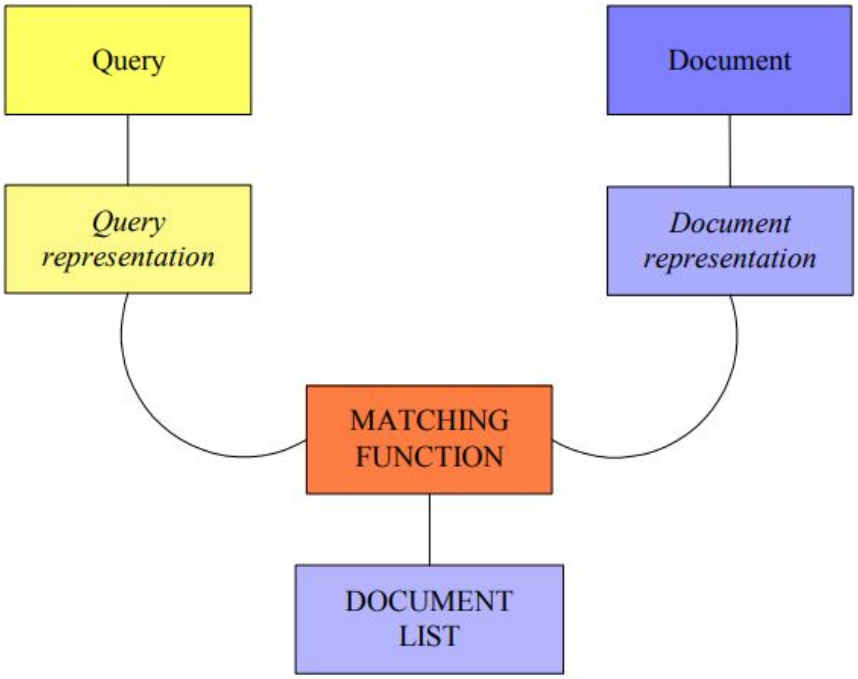

Figure 3.1: Document retrieval pipeline [Liddy, 2005].

There are four different classic document retrieval models based on various approaches and theories:

- **Boolean Model** - simplest and oldest DR model based on classical set theory
- **Vector Space Model** - algebraic model, where documents and queries are represented as vectors
- **Probabilistic Model** - model, that relies on probabilistic theories such as Bayes' theorem
- **Language Model** - statistical method based on determining a probability of a sequence of words

Follows the presentation of the four theoretical models along with their advantages and disadvantages at the end of each subsection [Pannu et al., 2014].

3.1.1 Boolean Model

The Boolean Model is based on set theory, where documents are represented as a set of terms and queries as logical expressions. It does not use term frequencies and term weights. The query representation is formed by terms linked with boolean operators AND, OR, and NOT. The relevance of a document is a result of a logical evaluation of the query, with respect to the document. Each term in the query is assigned a logical value of 1 or 0 depending on whether it is located in the document or not. When the logical expression is evaluated to 1, the document is considered relevant.

An example of a query for which documents, that contain pair of $term_1$ and $term_2$ or pair of $term_1$ and $term_3$, should be retrieved:

$$query = (term_1 \text{ AND } term_2) \text{ OR } (term_1 \text{ AND } term_3).$$

A document has to fulfill this logical formula; otherwise, it will not be considered relevant even if it contains some of the terms but not all and might be relevant.

The Boolean retrieval model does not provide a ranked list of documents due to the binarity of the results. In order to relevance rank the documents, we need to relax the logical interpretations of the boolean operators and use them as distances, which can be ordered. Another disadvantage of the model is that the terms are not weighted, meaning that the retrieved document has to contain exactly the terms specified by the query logical formula. This causes the model to be more suitable for data retrieval rather than information retrieval.

3.1.2 Vector Space Model

The Vector Space Model (VSM) [Salton et al., 1975] is an algebraic model in which the query and documents are represented by vectors. The vectors exist in a term space of the size of all unique terms in the document collection. The similarity between a query and a document is determined by the closeness of the vectors. We measure the closeness using the angle between the normalized vectors either by cosine similarity or dot product. We typically use weighted vectors, which we obtain by *document indexing*.

Document indexing is the task of representing a document by a vector of terms that appear in the document, where each term has a weight denoting its importance. The creation of the vector includes steps such as stop word removal and stemming, mentioned at the beginning of the chapter. Universally, the terms are weighted using **Term Frequency** (TF) and **Inverse Document Frequency** (IDF) vectors. TF is a frequency with which a term t appears in document d, calculated as the raw count of the term t over the sum of all other term t' counts in d:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}. \tag{3.1}$$

IDF represents the importance of a term t with respect to all documents from the collection D. It is calculated as a logarithm of the inverse fraction of documents that contain t over the total number N of documents in D:

$$idf(t,D) = \log \frac{N}{n_t}, \tag{3.2}$$

where n_t is the number of documents containing t.

The final weight for each term in a vector representing a document is the multiple of the term frequency value with the value from the inverse document frequency vector:

$$tf\text{-}idf(t,d,D) = tf(t,d) \cdot idf(t,D). \tag{3.3}$$

With this calculation, longer documents have an advantage. They get higher relevance because each term from the query appears more times. By dividing each term weight by the vector length, we normalize the vectors. Then the length of the document no longer affects the result.

The best indexing terms are those that appear with high frequency in a low number of documents. When a term does not appear in a document then it is assigned a weight of 0 since the first multiple in equation 3.3 is 0.

We have indexed all documents in our collection, and we do the same with our query (only the normalization is not necessary), obtaining both document and query representation. Next, we take our *matching function* (see Fig.3.1) and calculate a score for every query-document pair to rank the documents. The function calculates the closeness between the two vectors P (document), Q (query) using **cosine similarity**. It is the cosine of the angle θ between the vectors, calculated as the dot product of the vectors divided by the magnitude of each of the vectors:

$$cos(\theta) = \frac{P \cdot Q}{\|P\|\|Q\|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}}, \tag{3.4}$$

where p_i and q_i are weights of individual terms of the vectors. When both vectors are normalized, their lengths are equal to 1, and we are left only with the dot product.

The Vector Space Model is commonly used for its consistency in performance over many collections of documents. Another advantage of the VSM, is that it allows partial matching as opposed to the Boolean binary model. It can also easily adapt its parameters, including the term weighting scheme. However, it relies on information from the document collection and when the collection is changed, it has to recompute all weights.

3.1.3 Probabilistic Model

The Probabilistic Model aims to rank documents according to the probability of their relevance to the user query. It depends on estimations and probabilities and operates under certain assumptions. The first assumption is that terms appear independently of each other in the collection. The second is that they are scattered differently in relevant documents than in nonrelevant documents.

If we had a set of relevant and nonrelevant documents, we could estimate the probability of a term appearing in a relevant document. Since we do not have these sets, we estimate the probability by counting the number of documents in which the term appears and in which it does not. The estimate is further refined based on the distribution of the term in the documents.

Similarily to VSM, we create a vector of term weights where the weight for i-th term is:

$$w_i = \log \frac{p_i(1-q_i)}{q_i(1-p_i)}, \tag{3.5}$$

where p_i is the probability that a document containing term i is relevant and q_i is the probability that a document contains the term but it is not relevant. It is then calculated as:

$$p_i = \frac{r_i}{R},$$

$$q_i = \frac{n_i - r_i}{N - R},$$

where:

- n_i is the number of documents containing term i
- r_i is the number of relevant documents containing term i
- N is the number of documents

- R is the number of relevant documents.

As mentioned earlier, R is not known; therefore, we assume that p is constant (e.g., 0.5) and q is estimated by the values from Inverse Document Frequency. Now eq. 3.5 can be rewritten as:

$$w_i = \log \frac{r_i(N - R - n_i + r_i)}{(n_i - r_i)(R - r_i)}$$

but with the assumption that p_i is 0.5, the number of relevant documents containing i and nonrelevant containing i is equal. That leads to $n_i - r_i = 0$ being in the denominator of the fraction. To overcome this issue, we can add 0.5 to each term and we get:

$$w_i = \log \frac{(r_i + 0.5)(N - R - n_i + r_i + 0.5)}{(n_i - r_i + 0.5)(R - r_i + 0.5)}, \tag{3.6}$$

a term weight that can be used similarily to IDF vector in VSM.

Among the advantages of the Probabilistic Model are its theoretical foundation, the ability to rank the documents by their probability of relevance, and good performance over many collections. The biggest disadvantage is the unrealistic assumption that terms occur independently

BM25

BM25, Best Match 25, or Okapi weighting scheme [Robertson and Zaragoza, 2009] is a ranking function rooted in probabilistic document retrieval, which also takes advantage of the TF-IDF model. The relevance of document d (from collection D) to a query q is calculated by the following formula [Manning et al., 2008]:

$$BM25(q,d,D) = idf(q,D) \cdot \frac{tf(q,d) \cdot (k_1 + 1)}{k_1 \cdot (1 - b + b \cdot \frac{L_d}{L_{avg}}) + tf(q,d)} \cdot \frac{(k_3 + 1) \cdot tf(q,d)}{k_3 + tf(q,d)}, \tag{3.7}$$

where *idf* and *tf* were already introduced (sec. 3.1.2), L_d is the length of document d and L_{avg} is the average length of a document. Remaining k_1 and k_3 are positive tuning parameters and b is another tuning parameter. k_1 calibrates the document term frequency scaling. When $k_1 = 0$, then no term frequency is used, corresponding to a binary model. On the other hand, higher values of k_1 correspond to using a raw frequency count. The k_3 parameter calibrates the term frequency scaling of the query. The last parameter b ($\in < 0, 1 >$) determines the scaling by document length. When equal to 0, no length normalization is used, when equal to 1, the parameter fully scales the term weight by the document length. The reasonable values of the three parameters, achieved by experimentation [Manning et al., 2008], are between 1.2 and 2 for k_1 and k_3 and 0.75 for b.

The BM25 weighting scheme is widely used for its good performance across a range of collections [Jones et al., 2000]. It performs especially well in the TREC evaluations [Voorhees et al., 2005].

3.1.4 Language Model

The last model, Language Model (LM), ranks documents based on the probability that they generated the query. Each document has its own language model and the goal is

to determine the probability that a query was generated by that model. The retrieved documents are then sorted by the probability. LMs are used in various areas of NLP and in information retrieval it is called *query-likelihood* retrieval model. There are also various types of language models:

- **Unigram** also known as **bag-of-words** model assigns probabilities to different terms, where the probability of each term depends on the terms probability of occurrence in the document summing over all term probabilities in a document to 1. In other words, it uses only the count of the terms in a document and estimates each term independently, not taking advantage of the context. The probability of a query q is then the multiples of all probabilities of the query's terms:

$$P(q) = \prod_{t \in q} P(t). \tag{3.8}$$

- **N-gram** as opposed to unigram takes advantage of the context by conditioning on the previous n - 1 terms. It predicts the next item in the sequence using conditional probabilities. It approximates the probability of occurrence of the i-th word w_i in a context to a probability of its occurrence in a shorter context, consisting only of preceding n - 1 terms. Then the probability of a query of m terms would be:

$$P(t_1, \ldots, t_m) \approx \prod_{i=1}^{m} P(t_i | t_{i-(n-1)}, \ldots, t_{i-1}). \tag{3.9}$$

  Unigram is then a special case of an n-gram. We denote **bigram** and **trigram** as n-grams where $n = 2$ and $n = 3$ respectively since they are the most used n-grams along with unigram.

- The **Bidirectional** model, unlike n-grams, analyzes the context in both directions and not only backwards, which increases result accuracy. It is utilized in machine learning and speech generation applications.

- The **Exponential** model combines feature functions and n-grams in an equation to compute the rank of a document. The model is designed to maximize cross entropy.

- Lastly, the **Continuous space** or **Neural network** model uses continuous representations of embeddings of words (the process of assigning a weight to a word). It represents the words as a non-linear combination of weights in a neural network. This model is especially useful when the dataset is very large and usually it contains rare words which cause problems in a linear model (n-gram).

3.2 Transformers

The last type of language model, the Neural network based language model, leads us to the currently popular and state-of-the-art NLP models. The probabilistic LM (in Sec. 3.1.4) encounters two problems. A **context problem** of the n-gram, where the probability depends only on the n words which is often not enough for complex texts. When n grows, it widens the context but the number of word permutations grows exponentially making it impossible to store. This creates the second problem - **sparsity**. Most of the permutations of words never occur and are still stored.

Neural networks solve the sparsity problem by the way they encode the input in **embeddings of words** (continuous vectors) that also encode semantic relationships. To solve the context problem, we need a system to learn which words are more important than others. That is where **Recurrent Neural Networks** (RNNs) come in.

3.2.1 From RNNs to Transformers

Recurrent neural networks are good at processing sequential data like audio [Chung et al., 2014], video, or most importantly text. In NLP tasks, they can be used for machine translation [Cho et al., 2014b], sentiment classification [Tang et al., 2015], question answering [Iyyer et al., 2014], etc.

Regardless the specific task, the concept is always the same. On an example of a sentence, the RNNs take the first word, pass it through the network and get an output, like a classic feedforward network. They take the second word and feed it to the network along with the hidden state from the previous pass. They obtain a new hidden state to pass with the next word and so on until the end of the sentence. This means that the context problem is solved because they take *every word* into account.

However due to the *backpropagation of the gradient* in the network (the vanishing gradient [Hochreiter, 1998]), the words at the beginning of the sentence start to matter less. This is known as *short-term memory* and is the reason for the arrival of **LSTMs** (Long Short-Term Memory Units [Hochreiter and Schmidhuber, 1997]) and **GRUs** (Gated Recurrent Units [Cho et al., 2014a]).

LSTMs solve the vanishing gradient problem by preserving the error through backpropagation. They contain information in *gated cells* where it can be stored, written to, or read from. Each cell contains a *forget gate*, *input gate*, and *output gate*. Throughout the training of the network, the cells learn what information to store, write or forget. The gated units help maintain a more constant error and they allow the recurrent networks to learn over many timesteps.

GRUs are LSTMs without the output gate which means that the output from a memory cell writes itself directly to the larger net. GRUs contain an *update gate* (acts similar to the forget and input gate) and a *reset gate* (acts as another forget gate). Both GRUs and LSTMs learn long-term dependencies using the gated units. The units learn what information to add or remove to the hidden state, solving the short-term memory issue. The GRU tends to be quicker to train, although it is good practice to train both a GRU and an LSTM to decide which is more suitable for a given task.

Even with more evolved versions of recurrent networks such as Chung's Gated Feedback RNN [Chung et al., 2015] or Schuster's Bidirectional RNN [Schuster and Paliwal, 1997], they still have limitations in remembering long-term dependencies which stem from their sequential nature. It causes long training times because there is no possibility for parallelization. The **transformer architecture** solves this problem with a mechanism called **Attention**.

3.2.2 Attention

The **attention mechanism** introduced by [Bahdanau et al., 2016] and [Luong et al., 2015] ensures that the model learns which input deserves the most attention, in other words, which part of the input is most relevant. In an example of machine translation (Fig. 3.3), there are two sentences in different languages forming a matrix. Each pixel in the matrix shows the correlation between the source and target word.

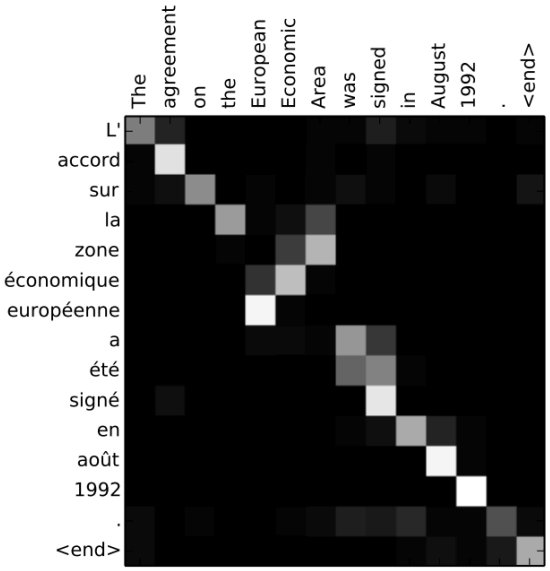

Figure 3.2: Attention mechanism on a machine translation example, reprinted from [Bahdanau et al., 2016].

When we put the same sentence on the sides of the matrix, it is called **self-attention**. It shows how parts of the sentence relate to others. E.g., a pronoun referring to a noun mentioned elsewhere in the sentence.

To calculate self-attention, we need three vectors for each input vector - **query**, **key**, and **value** (q, k, v). We obtain them by multiplying the embedding of the input vector with a query, key, and value **weight matrices** (W^Q, W^K, W^V) that we trained during the training process. To calculate the attention score between two words a and b, we take the dot product between the query vector q_a and the key vector k_b, divide it by the square root of the dimension of the key vector d_k and pass the result through a *softmax function*. Finally, multiply that by the value vector v_b. We usually calculate this for all words (embeddings) in the input by grouping the vectors in matrices:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{3.10}$$

where Q, K, and V are the matrices composed of the query, key, and value input vectors respectively. The output are the attention scores for the input sentence.

In [Vaswani et al., 2017], they present:

1. **Scaled dot-product attention**, which is the self-attention described above (Equation 3.10). There are two most common attention functions - **additive** and **dot-product**. The paper argues that dot-product attention is faster and more space-efficient due to optimized matrix multiplication. The scaled index can take on different values; Vaswani suggests the square root of the key vector dimension which leads to having more stable gradients.

2. **Multi-head attention** adds an arbitrary number of other weight matrices. Then the scaled dot-product attention is calculated for each of these weight matrices. It helps to project the input embeddings into different representation subspaces. Instead of one vector of attention scores (as in a single calculation of the attention function), we obtain as many vectors as the number of attention heads (number of weight matrices).

We concatenate the vectors and multiply the resulting matrix with an additional weight matrix W^O. We are left with a single matrix capturing the information from all attention heads which we can pass to the rest of the network.

$$\begin{aligned} \text{MultiHeadAttention}(Q,K,V) &= \text{Concat}(\text{head}_1,...,\text{head}_h)W^O, \\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \end{aligned} \tag{3.11}$$

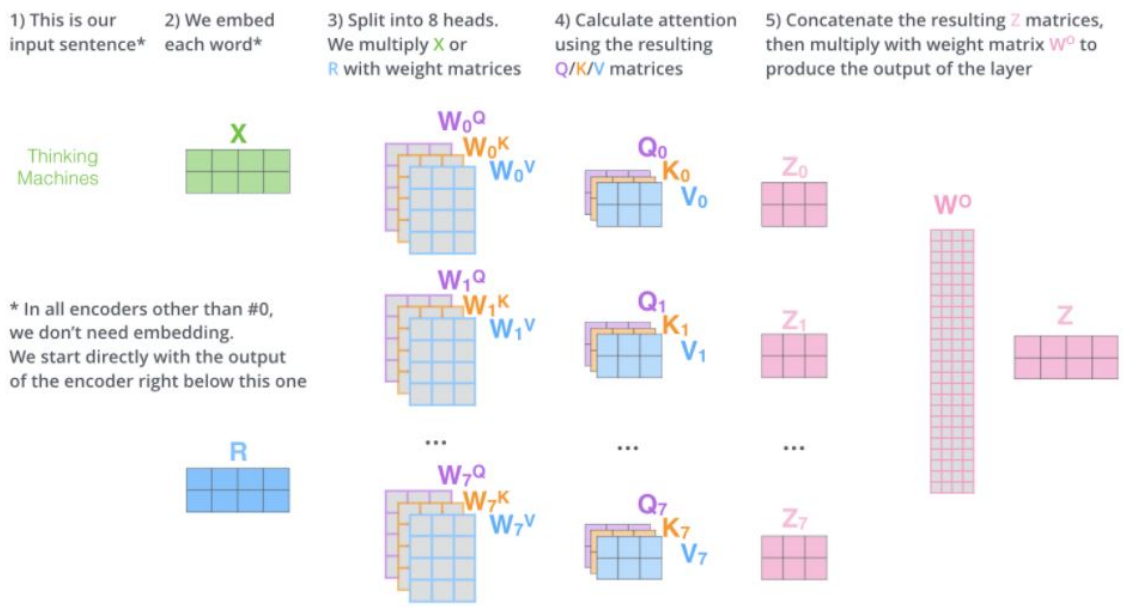

Figure 3.3: Multi-head attention calculation, reprinted from [Alammar, 2018].

3.2.3 Transformer Architecture

The transformer architecture introduced in [Vaswani et al., 2017] (in contrast with RNNs) makes parallelization possible and employs the attention mechanisms described in Sec. 3.2.2. Most of the state-of-the-art models for NLP tasks are utilizing this architecture, including OpenAI's [1] **GPT** models (newest GPT-3 [Brown et al., 2020]) or Google's **BERT** [Devlin et al., 2019].

The transformer adopts the **encoder-decoder architecture**. The **encoder** (left block in Fig. 3.4) consists of a self-attention mechanism and a feedforward network. There are six identical layers composed of two sub-layers: a multi-head self-attention layer and a fully connected feedforward network. Both sub-layers end with *layer normalization* and have a *residual connection* around them.

The first layer of the encoder takes embeddings of the input sentence as well as positional encoding containing information about the order of the sentence. The initial input embedding is the result of adding the sentence embedding with the positional encoding vector. The positional encoding adds meaningful distances between the embedding vectors once they are projected into the query, key, and value vectors (Fig. 3.3) and are used for the calculation of self-attention.

[1] `https://openai.com/`

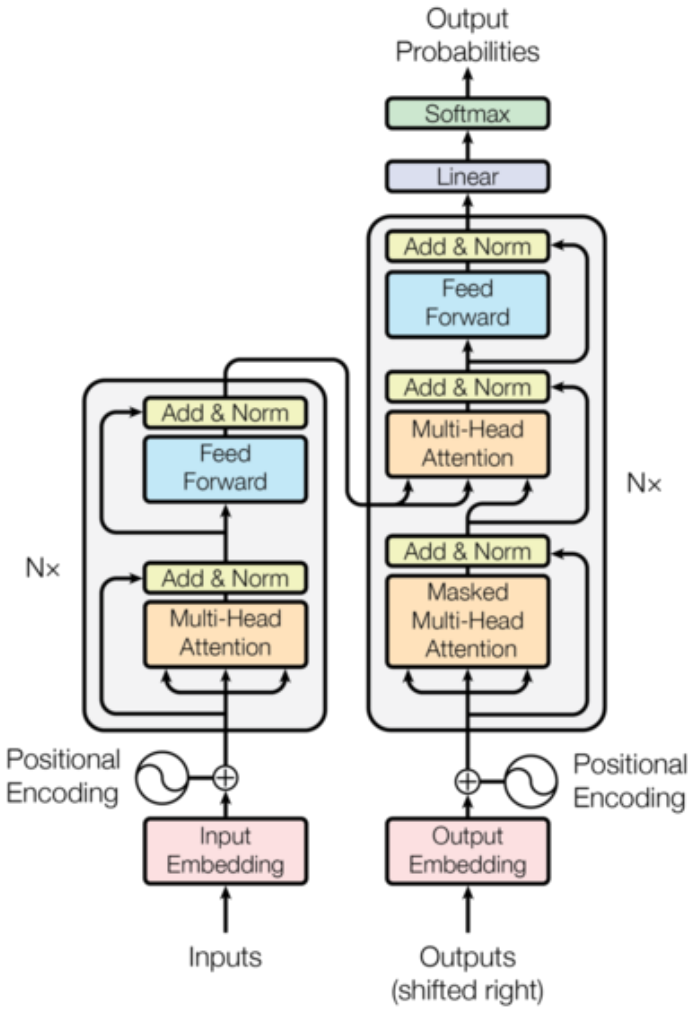

Figure 3.4: Transformer encoder-decoder architecture, reprinted from [Vaswani et al., 2017]

The **decoder** (right block in Fig. 3.4) consists of the same two parts as the encoder plus an attention mechanism over the output from the encoder. The first decoder takes embeddings of the output sequence and positional encodings, similarly to the encoder. Every layer also has a residual connection and is followed by layer normalization. The output sequence is *partially masked* to prevent the transformer from using future or current output for prediction. In addition, the last decoder is followed by a *linear layer* and a *softmax* function to generate the output probabilities over the vocabulary.

To sum up, the model perceives the entire input sequence simultaneously through the encoder and maps it to a continuous representation. The decoder generates an output sequence given the representation from the encoder. Vaswani [Vaswani et al., 2017] suggests $N = 6$ as the number of the encoder and also decoder blocks.

3.2.4 BERT

BERT, **B**idirectional **E**ncoder **R**epresentations from **T**ransformers introduced in [Devlin et al., 2019] is a language representation model based on the transformer architecture. It pre-trains deep bidirectional representations from unlabeled data by conditioning on context from both directions. The pre-trained model can then be fine-tuned to any specific task (such as question answering) with the same model architecture that only adds one

additional output layer.

According to [Devlin et al., 2019], there are two strategies for applying pre-trained language representations to downstream tasks: **feature-based** and **fine-tuning** approaches. The first one has to use task-specific architectures that contain the pre-trained representation in the form of additional features. An example of this architecture is ELMo [Peters et al., 2018]. The fine-tuning approach avoids task-specific parameters and trains all parameters while fine-tuning on a downstream task. The Generative Pre-trained Transformer models [Brown et al., 2020] take advantage of this approach.

BERT improves the fine-tuning based approach by using the **Masked Language Model** (MLM) and **Next Sentence Prediction** (NSP) pre-training tasks, further described in subsection 3.2.7. It demonstrates the importance of bidirectional pre-training in contrast to uni-directional LM utilized in the GPT models [Radford et al., 2018]. Thanks to BERT's pre-training, there is no need for task-specific architectures in the fine-tuning phase and BERT achieves high performance on a wide range of NLP tasks.

3.2.5 Cross-Attention Approach

BERT uses something called **cross-attention paradigm** [Chang et al., 2020], also called **interaction-based paradigm** in [Khattab and Zaharia, 2020]. For a query q and a document d, it can model the interaction within themselves as well as between the two of them at the same time. The input representation can represent one sentence and a pair of sentences (in our example a query and an answer) in one single token sequence. The first token of the input sequence is always a special token called **[CLS]**. The sequence ends with another special token called **[SEP]**. If we want to use two sentences, like in our example with query and document, we separate them with the same [SEP] token. The input sequence is then a concatenation of: [CLS] + q token embeddings + [SEP] + d token embeddings + [SEP] (see Fig. 3.5 on the right).

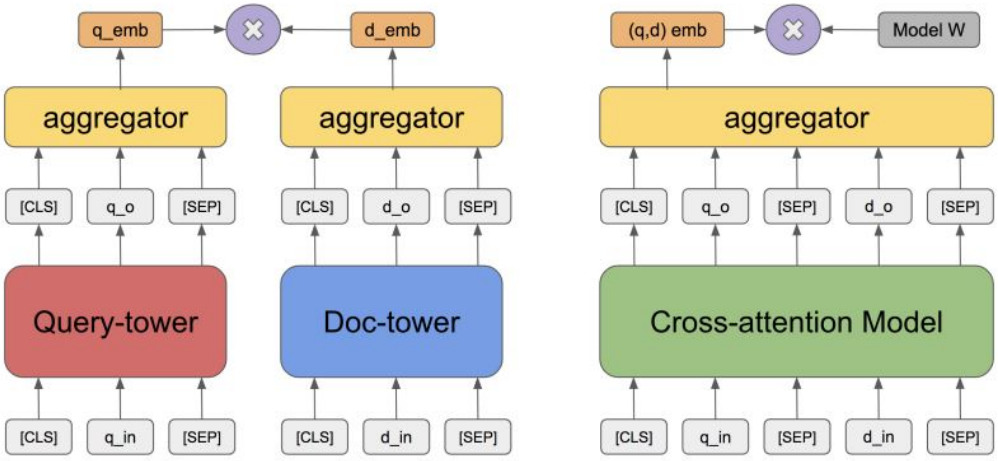

Figure 3.5: Two-Tower paradigm (left) vs. Cross-Attention paradigm (right), reprinted from [Chang et al., 2020].

The input sequence of token embeddings is further modified by adding another two embedding vectors to it. The first vector is the **segment embedding vector** which differentiates between the first and second sentence. The second vector is the **positional embeddings vector** which encodes the order of the tokens.

3.2.6 Two-Tower Approach

Another type of paradigm is the **Two-Tower paradigm** (Fig. 3.5 on the left), as it is called in [Chang et al., 2020], also known under Representation-based Similarity in [Khattab and Zaharia, 2020], Siamese network in [Das et al., 2016], or Dual-Encoder model in [Cer et al., 2018]. It is an embedding-based model that encodes the query and document separately into a similar embedding space. The input embeddings are formed in the same manner as in the cross-attention approach. The [CLS] and [SEP] tokens are used similarly at the beginning and end of the token sequence. The final input embedding is a sum of the token embedding vector with the positional and segment embedding vector again. The embeddings can be computed either by the same or a different language model, which means that the *towers* in the model can be identical or different. In this thesis, we worked only with the case of identical towers. The towers are usually based on the transformer architecture and each tower takes one of the query/document embeddings. That means that the attention is calculated within the query/document but not across both as in the cross-attention approach.

To measure the similarity between the query and document embeddings, we need a *scoring function*. It can be any similarity or distance metric or even dot-product between the vectors. An advantage over the cross-attention approach is that the document embeddings can be computed in advance. During the inference time, it is sufficient to compute the query embedding and find relevant documents with the *nearest neighbor search* using one of the similarity metrics.

3.2.7 Pre-training Tasks

The BERT model pre-training phase [Devlin et al., 2019] depends on two pre-training tasks: the token level **masked LM (MLM)** and **Next Sentence Prediction (NSP)**. [Chang et al., 2020] presents paragraph level pre-training tasks that can further improve models like BERT on top of the primary pre-training tasks. The tasks are **Inverse Cloze Task (ICT)**, **Body First Selection (BFS)**, and **Wiki Link Prediction (WLP)**. They work with Wikipedia articles as the training data.

- **Masked Language Model** - It is inspired by the Cloze Task [Taylor, 1953]. It randomly masks some tokens from the input and predicts the masked words based on the context. It fuses the context from left and right of the masked token which allows pre-training bidirectional representations. In contrast with LMs that are trained unidirectionally either from left to right or right to left. There are also LMs trained in both directions and then concatenated [Peters et al., 2018]. This task teaches the model to understand relationships between words. The BERT model adopts 15% as the probability of a token being masked and uses the BooksCorpus [Zhu et al., 2015] and English Wikipedia as the training corpus.

- **Next Sentence Prediction** - Teaches the model to understand long-term dependencies across sentences (relationships between sentences). The training examples for this task are formed by two sentences. 50% of the time, the second sentence is the correct following sentence (positive example), and the other 50% is a random sentence from the corpus (negative example).

- **Inverse Cloze Task** - The original Cloze Task [Taylor, 1953] predicts a masked out text based on context. The inverse of that is predicting context based on a sentence (described in [Lee et al., 2019]). The query is a random sentence from any passage

and the document is the rest of the sentences from the same passage. It requires more than learning matching features, it needs to capture the semantic context. For example, the query sentence often does not contain the most important word and the model still needs to understand the meaning and predict the context.

- **Body First Selection** - BFS takes a random sentence from the first paragraph of an article (on Wikipedia it is the summary of the article) as a query and a random paragraph from the rest of the article as the document. It captures semantic relationships outside of the local paragraph.
- **Wiki Link Prediction** - WLP on the other hand captures inter-page semantic relationships. The query is again a random sentence from the summary. The document is a random passage from another page from where there is a hyperlink leading to the page of the query.

Chapter 4
Method

This chapter describes the proposed method and our approach to the problem, used models and architectures, and evaluation of the solution.

4.1 Multi-Stage Retrieval

When faced with a *large-scale* retrieval problem, we can simplify the problem to learning a **scoring function** that assigns a score to pairs of queries and documents. The score from the function $f(q, d)$ is a real number and the higher it is, the closer the pair is (meaning d is relevant to q). In the case of fact-checking, q is the factoid for verification, and d is a passage containing the evidence for the verification.

Using a BERT-like model based on the cross-attention architecture (Sec. 3.2.5) on this task might not be suitable considering that the scoring function has to be computed for every possible query/document combination. The corpus can count hundreds of thousands of documents or more. Therefore we use a less powerful algorithm to reduce the number of documents followed by a transformer-based model to rerank the preselected documents. The two phases of the retrieval are called differently in literature. The Retrieval phase and the Scoring phase in [Chang et al., 2020], or the Retriever and Reader in [Chen et al., 2017]. We are calling them **Document Preselection** and **Document Reranking**.

4.1.1 Document Preselection

In the first phase, we take the training data in the FEVER-like format (for both datasets feverCS and ČTK), and for every claim, we preselect 500 potentially relevant documents from the document database. 500 is the suggested upper limit in [Monz, 2003].

We work with two models in this part. The first one is **DrQA** from [Chen et al., 2017] taken from their Retrieval part of the machine reading at scale pipeline. It takes advantage of one of the classical document retrieval approaches, the TF-IDF (Sec. 3.1.2 in the previous chapter). The queries and documents are represented as TF-IDF weighted bag-of-words vectors improved by n-gram features and hashing. They use a bigram with the hashing of [Weinberger et al., 2010] to map the bigram into 2^{24} bins.

The second model is the **Anserini model** ([Yang et al., 2017], [Yang et al., 2018]) from a Python Toolkit Pyserini [Lin et al., 2021]. It works with the Lucene search library and uses the BM-25 model (Sec. 3.1.3) with the bag-of-words vector representation.

We use both of these models for the preselection of documents to reduce the searching space for the reranking phase. We also refer to them as a baseline for all our trained models.

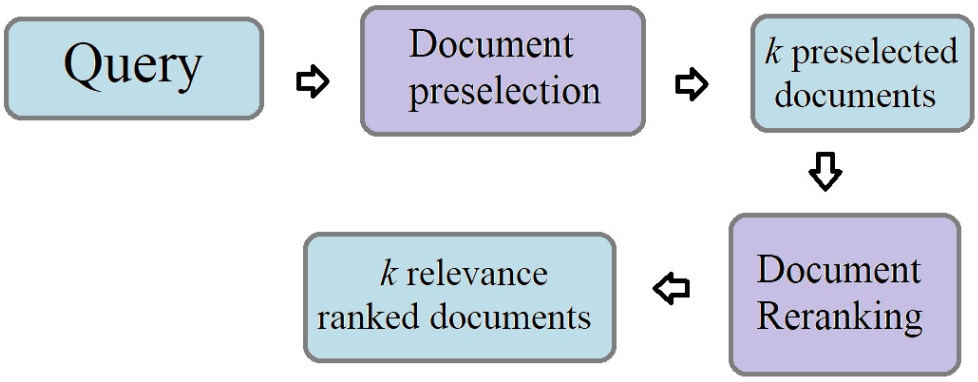

Figure 4.1: Preselection and Reranking pipeline.

4.1.2 Document Reranking

In the second phase, we train a transformer-based model on the preselected data from the first phase. There are many available pretrained transformer models in the HuggingFace library[1] [Wolf et al., 2020]. We decided on a distilled version of BERT, the **DistilBERT** [Sanh et al., 2020] from the Sentence Transformers library[2]. We train the model on two different architecture: the **two-tower** (Sec. 3.2.6) and **cross-attention** (Sec. 3.2.5). We choose the distilled version of BERT mainly because it is a smaller model which results in faster training times, especially on the cross-attention architecture.

Model distillation is one of many ways to speed up a model. It is a compression technique that transfers knowledge from a "teacher" model to a "student" model [Buciluǎ et al., 2006]. The student model learns to mimic the behavior of the teacher model. The DistilBERT model achieves 95% of BERT's performance while having 40% fewer parameters. The training time is also 60% faster than that of BERT.

The Two-Tower model firstly creates a set of training examples. There are *positive examples* formed by a claim and an evidence sentence. Regarding the *negative examples*, we differentiate between **hard negatives** and **soft negatives**. A hard negative is formed by the same claim as the positive and by one evidence from the 500 preselected. A soft negative is a random evidence from the rest of the dataset (excluding the preselected evidence for the current claim). The names follow from the expectation that a random evidence should not be considered relevant to the claim as opposed to the preselected potentially relevant evidence.

The Cross-Attention model creates the training examples by generating embeddings of the claim and evidence (in the manner described in Sec. 3.2.5). Again the positive examples labeled 1 are formed by the claims and their correct evidence and the negative labeled 0. The negatives are only created from the preselected documents that are not in the evidence set.

4.2 Evaluation

To evaluate our trained models, we used four different evaluation metrics described below.

[1] `https://huggingface.co/models`
[2] `https://github.com/UKPLab/sentence-transformers`

- **Precision** is the fraction of relevant documents to the query among all retrieved documents.

$$\text{precision} = \frac{|\text{relevant retrieved documents}|}{|\text{retrieved documents}|} \tag{4.1}$$

- **Recall** is the fraction retrieved of relevant documents among all relevant documents.

$$\text{recall} = \frac{|\text{relevant retirieved documents}|}{|relevantdocuments|} \tag{4.2}$$

- **F1** is the harmonic mean of precision and recall.

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.3}$$

- **Mean Reciprocal Rank** (MRR) calculates the reciprocal of the rank at which the first relevant document was retrieved and is averaged across queries. For one query it is $\frac{1}{\text{rank}}$ where the rank is the position of the highest-ranked (most relevant) document. For multiple queries Q, it is:

$$\text{MRR} = \frac{1}{\text{Q}} \sum_{i=1}^{\text{Q}} \frac{1}{\text{rank}_i} \tag{4.4}$$

4.3 Faiss

As we mentioned in section 3.2.6 about the Two-Tower architecture, there is a scoring function to measure the similarity between the query and document embeddings. We use the **FAISS library** [Johnson et al., 2017] by Facebook AI[3] used for similarity search over large datasets. It is a highly optimized nearest-neighbor search implementation. FAISS indexes the pairs of queries and documents and searches for the highest indexed pairs in the dataset. The search is sped up by different methods, such as *partitioning the index into Voronoi cells* or using *Product Quantization* to compress the vectors.

4.4 Losses

During the reranking phase of training on the Two-Tower architecture, we examine different types of loss functions that are calculated over the training examples. The examples are formed differently depending on the loss function. We used three types of losses:

- **Contrastive loss** [Hadsell et al., 2006] expects two texts (query and document) and a label. The label is 1 for a positive example and 0 for a negative example. The distance between the query and document embeddings is reduced for a positive example and increased for a negative example. It introduces a margin which is a minimum distance for a negative example (minimum distance between the query and document embedding). The loss penalizes negative examples for being closer that the margin.

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

- **Online Conrastive loss** is similar to Contrastive Loss. The difference is that it only computes the loss for hard positive and hard negative pairs. A hard positive is a positive that is far apart and a hard negative is a negative that is close. It often yields better performances than the Constrative Loss although we will see the opposite in experiment 5.2.1.

- **Triplet loss** takes three inputs: an anchor, a positive, and a negative. In our case, the anchor is the claim, and positive/negative evidence serves as before. It is calculated as:

$$\text{loss} = \max(||\text{anchor} - \text{positive}|| - ||\text{anchor} - \text{negative}|| + \text{margin}, 0), \tag{4.5}$$

  where $||\cdot||$ is the distance measure.

4.5 ICT Pretraining

We also tried to improve the performance of the cross-attention architecture by another pretraining task, the Inverse Cloze Task (Sec. 3.2.7). We tried to adapt the task to the cross-attention architecture.

The pretraining examples were created from the database of articles associated with the dataset. In the case of FEVER CS, it is the Czech Wikipedia, and for the ČTK dataset, it is the collection of news reports and articles. We split the database into **chunks of sentences** the size of a maximum of **288 tokens** when tokenized (as suggested in [Lee et al., 2019]), minus three tokens to make space for the special [CLS] and [SEP] tokens. We split the chunks randomly into train and dev parts. For the FEVER CS dataset, the dev part is around 2% of the chunks which is enough to make over **7500 positive** and **7500 negative** examples. The train part makes over **83.000 examples** each positive and negative. The ČTK database is much bigger and collects over **700.000** positive and negative examples.

The process of creation of the training examples for the ICT task is briefly described in section 3.2.7. We take the chunks in random batches, and from each chunk, we select a random sentence that is masked (removed) with some probability. We used the probability of 0.9. A training example is then formed as described in the cross-attention paradigm in Figure 3.5. The masked sentence represents the query and the chunk without the sentence (or with the sentence in 10% of cases) represents the document. The negative examples are created by combining the query with a random document from the batch.

4.6 Transfer Learning

The database of articles from the ČTK dataset is extensive, containing over **13 million** articles. The training dataset composed of claims and evidence is considerably smaller, containing only around 3000 claims (not all verifiable). This means that the training of a network (cross-attention or two-tower) on the ČTK dataset has only around **700 positive** and **700 negative** training examples available (around 190 for the dev split). The number is lower also because an average ČTK claim has only one evidence set as opposed to the FEVER CS claims.

We decided to compare the results of learning the two-tower network solely on the ČTK data with the results of transfer learning of a network previously trained on the FEVER CS data and finetuned on the ČTK. We also tried to evaluate the results of a network

trained on FEVER CS on the ČTK dataset and compare it. The results are in the next chapter.

The transfer learning approach is not new. For example, Google Research in [Roberts and Raffel, 2020] suggests a T5 model, a Transfer Transformer, to create state-of-the-art results in NLP. They train the model on unlabeled data with a self-supervised task and then finetune it on a smaller labeled dataset.

Chapter 5
Experiments

This chapter describes the setup of our experiments and their results.

5.1 Preselection

The first model used for preselection, the **DrQA model**, calculates the TF-IDF index for all unigrams and bigrams and uses the Weinberger's hashing into 2^{24} bins.

The second, the **Anserini model**, calculates the index with the BM25 method. The parameters we used were calculated by [Rýpar, 2021]. The k_1 parameter for calibrating the document term frequency scaling is 0.9 for the FEVER CS dataset and 0.6 for ČTK. The b parameter for scaling by the document length corresponds to 0.9 for FEVER CS and 0.5 for ČTK. The parameters were found by searching a grid of values with a step of 0.1.

	fever CS				CTK			
Precision	@1	@10	@20	@500	@1	@10	@20	@500
DrQA	0.4242	0.0756	0.0408	0.0021	0.125	0.0307	0.0176	0.0011
Anserini	0.3441	0.0659	0.0370	0.0019	0.1550	0.0335	0.0197	0.0012

Table 5.1: Preselection precision at k

	fever CS				CTK			
Recall	@1	@10	@20	@500	@1	@10	@20	@500
DrQA	0.3914	0.6889	0.7399	0.8753	0.1275	0.3100	0.3550	0.5500
Anserini	0.3165	0.6008	0.6731	0.8588	0.1575	0.3375	0.3975	0.5825

Table 5.2: Preselection recall at k

	fever CS				CTK			
F1	@1	@10	@20	@500	@1	@10	@20	@500
DrQA	0.4072	0.1363	0.0773	0.0042	0.1262	0.0560	0.0336	0.0022
Anserini	0.3298	0.1187	0.0702	0.0039	0.1562	0.0610	0.0376	0.0023

Table 5.3: Preselection F1 at k

	fever CS				CTK			
MRR	@1	@10	@20	@500	@1	@10	@20	@500
DrQA	0.4072	0.5081	0.5114	0.5141	0.1305	0.1831	0.1860	0.1894
Anserini	0.3255	0.4159	0.4211	0.4245	0.1579	0.2095	0.2136	0.2164

Table 5.4: Preselection MRR at k

5.2 Reranking

The following experiment tested the **cross-attention model** (denoted as CA in results) along with the ICT pretraining task. We tried three setups: training only the cross-attention model on reranking the preselected documents (either from DrQA or Anserini), pretraining ICT task on the dataset article databases, and finetuning the pretrained model with the cross-attention model.

For all training, we used the DistlBERT model version `distilbert-base-nli-stsb-mean-tokens` from the sentence transformers with a fully connected linear layer as a pre-classifier and a linear layer with 2 output features at the end as a classifier. In the case of the FEVER CS dataset, the training ran for 10 epochs on a Tesla V100-SXM2-32GB GPU with a batch size of 16, learning rate 1e-5, weight decay 0.01, cross-entropy as the loss function and AdamW optimizer. On the ČTK dataset, it was 15 epochs with the same parameters.

The ICT pretraining ran for 4 and 3 epochs on FEVER CS and ČTK respectively with the same training parameters as the cross-attention model. We ran the cross-attention model with the pretrained models and finetuned them for another 30 and 15 epochs.

	fever CS				CTK			
Precision	@1	@5	@10	@20	@1	@5	@10	@20
DrQA + CA	0.2114	0.0740	0.0450	0.0265	0.0075	0.0025	0.0018	0.0014
Anserini + CA	0.2801	0.0916	0.0526	0.0300	0.0000	0.0025	0.0033	0.0025
DrQA + ICT	0.0392	0.0241	0.0180	0.0132	0.0025	0.0010	0.0013	0.0006
Anserini + ICT	0.0311	0.0189	0.0142	0.0104	0.0075	0.0025	0.0018	0.0013
DrQA + ICT + CA	0.0246	0.0172	0.0130	0.0096	0.0000	0.0010	0.0013	0.0013
Anserini + ICT + CA	0.1804	0.0464	0.0264	0.0161	0.0050	0.0025	0.0018	0.0016

Table 5.5: Cross-Attention precision at k

	fever CS				CTK			
Recall	@1	@5	@10	@20	@1	@5	@10	@20
DrQA + CA	0.1949	0.3407	0.4146	0.4883	0.0100	0.0150	0.0200	0.0300
Anserini + CA	0.2606	0.4233	0.4839	0.5513	0.0025	0.0150	0.0350	0.0525
DrQA + ICT	0.0354	0.1083	0.1622	0.2385	0.0050	0.0075	0.0150	0.0150
Anserini + ICT	0.0278	0.0837	0.1274	0.1877	0.0100	0.0150	0.0200	0.0275
DrQA + ICT + CA	0.0228	0.0765	0.1173	0.1727	0.0025	0.0075	0.0150	0.0275
Anserini + ICT + CA	0.1683	0.2169	0.2466	0.3017	0.0075	0.0150	0.0200	0.0350

Table 5.6: Cross-Attention recall at k

	fever CS				CTK			
F1	@1	@5	@10	@20	@1	@5	@10	@20
DrQA + CA	0.2028	0.1216	0.0811	0.0503	0.0086	0.0043	0.0032	0.0026
Anserini + CA	0.2700	0.1506	0.0949	0.0569	0.0000	0.0043	0.0059	0.0048
DrQA + ICT	0.0372	0.0394	0.0323	0.0251	0.0033	0.0018	0.0023	0.0012
Anserini + ICT	0.0293	0.0308	0.0256	0.0197	0.0086	0.0043	0.0032	0.0024
DrQA + ICT + CA	0.0237	0.0280	0.0235	0.0182	0.0000	0.0018	0.0023	0.0024
Anserini + ICT + CA	0.1742	0.0764	0.0478	0.0307	0.0060	0.0043	0.0032	0.0031

Table 5.7: Cross-Attention F1 at k

	fever CS				CTK			
MRR	@1	@5	@10	@20	@1	@5	@10	@20
DrQA + CA	0.1950	0.2496	0.2596	0.2650	0.0084	0.0109	0.0119	0.0124
Anserini + CA	0.2618	0.3235	0.3323	0.3369	0.0021	0.0062	0.0079	0.0091
DrQA + ICT	0.0396	0.0663	0.0731	0.0779	0.0042	0.0053	0.0061	0.0061
Anserini + ICT	0.0300	0.0500	0.0562	0.0600	0.0084	0.0111	0.0119	0.0123
DrQA + ICT + CA	0.0254	0.0448	0.0513	0.0548	0.0021	0.0036	0.0044	0.0053
Anserini + ICT + CA	0.1779	0.1964	0.2000	0.2033	0.0126	0.0145	0.0150	0.0158

Table 5.8: Cross-Attention MRR at k

5.2.1 Losses

We trained the **two-tower model** on both datasets with the DistilBERT model, however we encountered a problem with the size of the ČTK dataset that we tried to solve with a transfer learning approach (Sec. 4.6 and results in 5.2.3).

Concerning the FEVER CS dataset, in this experiment, we examined three different loss functions. We trained the model for 5 epochs with a batch size of 16. We used the NFC norm for all training. We experimented with different hard negative percentages and ultimately decided on 100% being the best, therefore using it for all experiments with loss functions. We used the DrQA preselection for training on the FEVER CS dataset because it yields better results on this task as well as the following one 5.2.2.

	fever CS			
Precision	@1	@5	@10	@20
Contrastive	0.8369	0.1953	0.0999	0.0509
Online Contrastive	0.5788	0.1624	0.0878	0.0467
Triplet	0.7912	0.1846	0.0954	0.0490

Table 5.9: Two-Tower loss precision at k

	fever CS			
Recall	@1	@5	@10	@20
Contrastive	0.7882	0.8981	0.9110	0.9233
Online Contrastive	0.5396	0.7538	0.8111	0.8572
Triplet	0.7433	0.8548	0.8780	0.8977

Table 5.10: Two-Tower loss recall at k

	fever CS			
F1	@1	@5	@10	@20
Contrastive	0.8118	0.3209	0.1800	0.0964
Online Contrastive	0.5585	0.2673	0.1584	0.0886
Triplet	0.7665	0.3037	0.1721	0.0929

Table 5.11: Two-Tower loss F1 at k

	fever CS			
MRR	@1	@5	@10	@20
Contrastive	0.7968	0.8374	0.8394	0.8403
Online Contrastive	0.5191	0.5995	0.6077	0.6113
Triplet	0.7542	0.7961	0.7989	0.8002

Table 5.12: Two-Tower loss MRR at k

5.2.2 Negative example selection

In this experiment, we try out different percentages of hard negatives (Sec. 4.1.2). The setup is the same as for the previous experiment. We chose the contrastive loss and utilize it in all training of this experiment.

	fever CS			
Precision	@1	@5	@10	@20
100%	0.8369	0.1953	0.0999	0.0509
75%	0.5303	0.1638	0.0898	0.0476
50%	0.3756	0.1300	0.0756	0.0418
0%	0.0255	0.0184	0.0134	0.0098

Table 5.13: Two-Tower HNP precision at k

	fever CS			
Recall	@1	@5	@10	@20
100%	0.7882	0.8981	0.9110	0.9233
75%	0.4973	0.7532	0.8204	0.8663
50%	0.3525	0.6011	0.6938	0.7624
0%	0.0237	0.0861	0.1245	0.1802

Table 5.14: Two-Tower HNP recall at k

	fever CS			
F1	@1	@5	@10	@20
100%	0.8118	0.3209	0.1800	0.0964
75%	0.5133	0.2691	0.1619	0.0903
50%	0.3637	0.2137	0.1364	0.0792
0%	0.0246	0.0304	0.0242	0.0186

Table 5.15: Two-Tower HNP F1 at k

	fever CS			
MRR	@1	@5	@10	@20
100%	0.7968	0.8374	0.8394	0.8403
75%	0.5038	0.6080	0.6170	0.6199
50%	0.3556	0.4509	0.4634	0.4683
0%	0.0237	0.0450	0.0506	0.0541

Table 5.16: Two-Tower HNP MRR at k

5.2.3 Transfer Learning

We chose the best performing loss function and HNP percentage from the FEVER CS experiments (contrastive loss and 100% HNP) and used them on the training on ČTK data. As the data was not extensive, we trained only for 2 epochs and monitored evaluation loss to avoid overfitting. The results were underwhelming and we present only the most important evaluation measure, the recall, for completeness.

	ČTK			
Recall	@1	@5	@10	@20
FEVER CS	0.0025	0.0025	0.0025	0.0010
FEVER CS + ČTK	0.0025	0.0050	0.0125	0.0150
ČTK	0.0025	0.0075	0.0225	0.0375

Table 5.17: Two-Tower recall at k evaluated on ČTK

Chapter 6
Conclusion

The goal of this work was to familiarize ourselves with the document retrieval algorithms employed in the fact-checking scenario focusing on the Czech language. We studied classical methods of document retrieval as well as modern state-of-the-art algorithms.

The modern methods often rely on the basics and improve them by a novel approach. We combined the traditional and proven methods, such as the TF-IDF and BM25 models, with more powerful transformer-based networks. We utilized the classics in the first stage of retrieval and we tried to improve these methods by reranking their results with BERT-like models.

We worked with two Czech datasets, the FEVER CS and ČTK. The FEVER CS provides a larger training dataset including a little over 100.000 claims (compared to around 2000 verifiable claims for ČTK). The ČTK dataset, on the other hand, contains a broader base of articles than the Czech Wikipedia.

We experimented with two different architectures and various setups on these. Firstly, we explored the Cross-Attention network that did not exceed the baseline laid by the DrQA and Anserini models. It decreased on Recall@20 by 25% with DrQA and by 12% with Anserini. We experimented with the ICT pretraining task that was successfully employed in the work of [Rýpar, 2021] in combination with a multilingual BERT model. We tried the pretraining task in the cross-attention setting but the results are far weaker, decreasing the results of the cross-attention model trained alone. We found that the choice of the preselected documents (DrQA vs Anserini) greatly influences the results of the pretrained and finetuned model. The Anserini preselection working better by around 15% on Recall@20 on the FEVER CS dataset.

Secondly, we experimented with the Two-Tower architecture in two areas of interest: the loss functions and the amount of hard negative examples. We discovered that the strongest loss is the Contrastive loss over the expectation of the Online Contrastive loss being stronger. We outperformed the DrQA baseline on the FEVER CS dataset by all three loss function setups. The strongest, Contrastive loss, by 19%, the Online Contrastive by 12%, and the Triplet loss by 16% on Recall@20.

We examined the effect of hard negative example percentage (HNP) and discovered that the strongest model is the one trained with 100% of hard negatives and no soft negatives. The trained models outperformed the baseline down to 50% HNP from where they began underperforming. The final strongest combination is a two-tower model trained with contrastive loss and 100% HNP, exceeding the baseline by 19% on the FEVER CS dataset.

We conducted the same experiments on the ČTK dataset without success. We connect the results with the size of the training dataset which makes it more difficult to train models. Also, the larger amount of articles to search the evidence in makes the task more challenging. We tried to overcome this problem with transfer learning from the FEVER

CS dataset but the datasets are too dissimilar. We noticed a bias in focusing on short texts (eg. article titles) in the retrieved documents and tried to eliminate it by simply forbidding to retrieve titles, although this heuristic helped only marginally and is not included in the experiments.

In future work, we will examine the dataset more thoroughly to discover possible biases and irregularities. An interesting direction could be to try the Two-Tower model with different networks for each tower as well as experiment with various BERT models in the Cross-Attention setting since the current models quickly become obsolete in the fast-evolving field of NLP.

Bibliography

[Alammar, 2018] Alammar, J. (2018). The illustrated transformer. [online; accessed 08-08-2021].

[Aly et al., 2021] Aly, R., Guo, Z., Schlichtkrull, M., Thorne, J., Vlachos, A., Christodoulopoulos, C., Cocarascu, O., and Mittal, A. (2021). Feverous: Fact extraction and verification over unstructured and structured information.

[Bahdanau et al., 2016] Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

[Baines et al., 2020] Baines, D., Elliott, R., et al. (2020). Defining misinformation, disinformation and malinformation: An urgent need for clarity during the covid-19 infodemic. *Discussion Papers*, 20.

[Binau and Schulte, 2020] Binau, J. and Schulte, H. (2020). Danish fact verification: An end-to-end machine learning system for automatic fact-checking of danish textual claims.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[Buciluǎ et al., 2006] Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

[Cer et al., 2018] Cer, D., Yang, Y., yi Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder.

[Chang et al., 2020] Chang, W.-C., Yu, F. X., Chang, Y.-W., Yang, Y., and Kumar, S. (2020). Pre-training tasks for embedding-based large-scale retrieval.

[Chen et al., 2017] Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions.

[Cho et al., 2014a] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches.

[Cho et al., 2014b] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

[Chung et al., 2015] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2015). Gated feedback recurrent neural networks.

[Das et al., 2016] Das, A., Yenala, H., Chinnakotla, M., and Shrivastava, M. (2016). Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 378–387, Berlin, Germany. Association for Computational Linguistics.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Fleiss, 1971] Fleiss, J. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378—382.

[Gažo, 2021] Gažo, A. (2021). Algorithms for document retrieval in czech language supporting long inputs.

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

[Hochreiter, 1998] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

[Iyyer et al., 2014] Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R., and Daumé III, H. (2014). A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 633–644.

[Johnson et al., 2017] Johnson, J., Douze, M., and Jégou, H. (2017). Billion-scale similarity search with gpus.

[Jones et al., 2000] Jones, K. S., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments. In *Information Processing and Management*, pages 779–840.

[Khattab and Zaharia, 2020] Khattab, O. and Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over bert.

[Lee et al., 2019] Lee, K., Chang, M.-W., and Toutanova, K. (2019). Latent retrieval for weakly supervised open domain question answering.

[Liddy, 2005] Liddy, E. D. (2005). *Automatic Document Retrieval. In Encyclopedia of Language and Linguistics.* 2nd Edition. Elsevier Press.

[Lin et al., 2021] Lin, J., Ma, X., Lin, S.-C., Yang, J.-H., Pradeep, R., and Nogueira, R. (2021). Pyserini: An easy-to-use python toolkit to support replicable ir research with sparse and dense representations.

[Luong et al., 2015] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation.

[MacAvaney et al., 2020] MacAvaney, S., Nardini, F. M., Perego, R., Tonellotto, N., Goharian, N., and Frieder, O. (2020). Efficient document re-ranking for transformers by precomputing term representations. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, USA.

[Matveeva et al., 2006] Matveeva, I., Burges, C., Burkard, T., Laucius, A., and Wong, L. (2006). High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 437–444.

[Monz, 2003] Monz, C. (2003). From document retrieval to question answering.

[Nørregaard and Derczynski, 2021] Nørregaard, J. and Derczynski, L. (2021). Danfever: claim verification dataset for danish. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 422–428.

[Pannu et al., 2014] Pannu, M., James, A., and Bird, R. (2014). A comparison of information retrieval models.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

[Roberts and Raffel, 2020] Roberts, A. and Raffel, C. (2020). Exploring transfer learning with t5: the text-to-text transfer transformer. `https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html`. [Online; accessed 10-August-2021].

[Robertson and Zaragoza, 2009] Robertson, S. and Zaragoza, H. (2009). *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[Rýpar, 2021] Rýpar, M. (2021). Methods of document retrieval for fact-checking. `https://www.overleaf.com/read/thbvcjvvvfjp`. [Online; accessed 21-May-2021].

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

[Sanh et al., 2020] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2020). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

[Shahi and Nandini, 2020] Shahi, G. K. and Nandini, D. (2020). Fakecovid–a multilingual cross-domain fact check news dataset for covid-19. *arXiv preprint arXiv:2006.11343*.

[Tang et al., 2015] Tang, D., Qin, B., and Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. pages 1422–1432.

[Taylor, 1953] Taylor, W. L. (1953). "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

[Thorne and Vlachos, 2018] Thorne, J. and Vlachos, A. (2018). Automated fact checking: Task formulations, methods and future directions. *CoRR*, abs/1806.07687.

[Thorne et al., 2018a] Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018a). Fever: a large-scale dataset for fact extraction and verification.

[Thorne et al., 2018b] Thorne, J., Vlachos, A., Cocarascu, O., Christodoulopoulos, C., and Mittal, A. (2018b). The fact extraction and verification (fever) shared task.

[Thorne et al., 2018c] Thorne, J., Vlachos, A., Cocarascu, O., Christodoulopoulos, C., and Mittal, A. (2018c). The FEVER2.0 shared task. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*.

[Ullrich, 2021] Ullrich, H. (2021). Dataset for automated fact checking in czech language.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[Voorhees et al., 2005] Voorhees, E. M., Harman, D. K., et al. (2005). *TREC: Experiment and evaluation in information retrieval*, volume 63. MIT press Cambridge, MA.

[Weinberger et al., 2010] Weinberger, K., Dasgupta, A., Attenberg, J., Langford, J., and Smola, A. (2010). Feature hashing for large scale multitask learning.

[WHO, 2021] WHO (2021). World health organization: Infodemic. `https://www.who.int/health-topics/infodemic`. Accessed: 2021-04-07.

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

[Yang et al., 2017] Yang, P., Fang, H., and Lin, J. (2017). Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1253–1256.

[Yang et al., 2018] Yang, P., Fang, H., and Lin, J. (2018). Anserini: Reproducible ranking baselines using lucene. *Journal of Data and Information Quality (JDIQ)*, 10(4):1–20.

[Zhu et al., 2015] Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.

Appendix A
Acronyms

BERT Bidirectional Encoder Representations from Transformers

BFS Body First Selection

BM25 Best Match 25

CA Cross-Attention

ČTK Česká Tisková Kancelář

DR Document Retrieval

FAISS Facebook AI Similarity Search

FEVER Fact Extraction and Verification

GPT Generative Pre-trained Transformer

GRU Gated Recurrent Unit

ICT Inverse Cloze Task

LM Language Model

LSTM Long Short Term Memory

MLM Masked Language Model

NLI Natural Language Inference

NLP Natural Language Processing

NSP Next Sentence Prediction

RNN Recurrent Neural Network

TF-IDF Term Frequency - Inverse Document Frequency

VSM Vector Space Model

WLP Wiki Link Prediction

Appendix B
Project Structure

B.1 Directory Structure

- `experimental-bara-master.zip` ..... repository with src codes
- `embed` ..... embedding utils
- `evaluatio` ..... DR evaluation
- `slurm` ..... slurm scripts
- `utils` ..... utils