CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Master’s thesis

Methods of Document Retrieval for Fact Checking

Bc. Martin Rýpar

Faculty of Electrical Engineering
Department of Computer Science
Supervisor: Ing. Jan Drchal PhD.

May, 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Rýpar** Jméno: **Martin** Osobní číslo: **438219**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Datové vědy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Metody document retrieval pro ověřování faktů

Název diplomové práce anglicky:

Methods of Document Retrieval for Fact Checking

Pokyny pro vypracování:

The task is to develop methods of document retrieval to be deployed in the fact-checking scenario.
1) Familiarize yourself with methods of document retrieval aimed for the fact-checking task. Focus on the Czech language and neural network approaches.
2) Work with the Czech Wiki FEVER dataset and/or with other datasets supplied by the supervisor.
3) Select several existing methods focusing on those potentially applicable for the Czech corpora (e.g., tokenization, existing embedding model, etc.).
4) Evaluate and compare the methods.

Seznam doporučené literatury:

[1] Thorne, James, et al. "FEVER: a large-scale dataset for fact extraction and verification." arXiv preprint arXiv:1803.05355 (2018).
[2] Thorne, James, et al. "The fact extraction and verification (fever) shared task." arXiv preprint arXiv:1811.10971 (2018).
[3] Chang, Wei-Cheng, et al. "Pre-training tasks for embedding-based large-scale retrieval." arXiv preprint arXiv:2002.03932 (2020).
[4] Yang, Wei, et al. "End-to-end open-domain question answering with bertserini." arXiv preprint arXiv:1902.01718 (2019).
[5] Binau, Julie, and Henri Schulte. "Danish Fact Verification: An End-to-End Machine Learning System for Automatic Fact-Checking of Danish Textual Claims." (2020).

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Drchal, Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **18.09.2020** Termín odevzdání diplomové práce: _____________

Platnost zadání diplomové práce: **19.02.2022**

Ing. Jan Drchal, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Acknowledgements

First of all, I would like to thank my supervisor Jan Drchal for his dedicated support, guidance and friendly approach. I am very grateful for the opportunity to work with him and all my current and former colleagues from the AI in Journalism project at AIC CTU. They created such a pleasant and motivating environment, for which I would like to express my great thanks to all of them. Last but not least, I would like to express a great thank you to all my family, friends and my girlfriend for their endless support.

Declaration

Citation of this thesis

Rýpar, Martin. *Methods of Document Retrieval for Fact Checking.* Master's thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, 2021.

Abstrakt

Tato práce se zabývá přístupy pro vyhledávání dokumentů. Primárně se zaměřuje na metody hlubokého vyhledávání s využitím jazykových modelů a jejich srovnání s tradičními TF-IDF a BM25 modely. Modely jsou zkoumány v doméně ověřování faktů s cílem je posléze zakomponovat do systému pro ověřování výroků. Naše výsledky potvrzují, že modely založené na jazykových modelech dokáží překonat velmi solidní a robustní tradiční přístupy. Přístup spočívající v dalším předtrénování na úlohách relevantních pro vyhledávání může přinést výrazné zvýšení výkonu, avšak za cenu delšího a pracnějšího tréninku a potřeby velkého množství dat. Model ColBERT implementující nové paradigma pozdní interakce překonal tradiční modely na obou souborech dat.

Klíčová slova vyhledávání dokumentů, ověřování faktů, vyhledávání informací, BERT, TF-IDF, BM25, NLP

Abstract

This paper deals with approaches for large-scale document retrieval. It primarily focuses on deep retrieval methods using language models and their comparison with traditional TF-IDF and BM25 models. The models are investigated in the fact-checking domain with the goal of eventually incorporating them into a system for verifying claims. Our results confirm that language-based contextualized approaches can outperform very solid and robust traditional approaches. The approach of further pre-training on retrieval relevant tasks can yield significant performance gains, but at the cost of longer and more laborious training and the need for large amounts of data. ColBERT model implementing a new late interaction paradigm outperformed traditional models on both datasets.

Keywords document retrieval, fact checking, information retrieval, BERT, TF-IDF, BM25, NLP

Contents

Introduction **1**

1 Fact-checking **3**
- 1.1 Problem description and goals . . . 3
- 1.2 Related work . . . 5

2 Background **6**
- 2.1 Document Retrieval . . . 6
- 2.2 Traditional Approach . . . 7
- 2.3 Language Modelling . . . 12
- 2.4 Hybrid Approach . . . 21
- 2.5 Neural Approach . . . 22

3 Datasets **30**
- 3.1 FEVER CS . . . 30
- 3.2 ČTK . . . 31
- 3.3 Data Quality . . . 32

4 Proposed Solutions **37**
- 4.1 Baseline . . . 37
- 4.2 Neural Models . . . 37
- 4.3 Evaluation . . . 39

5 Experiments **40**
- 5.1 Baselines . . . 40
- 5.2 ColBERT . . . 40
- 5.3 Pretraining mBERT . . . 40
- 5.4 Results . . . 42

Conclusion **44**

Bibliography **46**

A1 Acronyms **52**

A2 Content of enclosed repository **53**

List of Figures

1.1 Demagog fact-checking 3
1.2 Fact-checking Pipeline Scheme 4

2.1 Document Retrieval Scheme 7
2.2 Zipf's law 8
2.3 Bag of words representation 10
2.4 Term frequency of TF-IDF vs. term frequency of BM25 11
2.5 Document length in BM25 11
2.6 Transfer learning 12
2.7 Transformer architecture 14
2.8 Illustrated attention mechanism 15
2.9 Scaled Dot-Product Attention and Multi-Head Attention 15
2.10 Pre-training and fine-tuning procedures for BERT 17
2.11 Language Resource Distribution 18
2.12 Language models comparison by number of parameters 19
2.13 Hybrid Approach Scheme 22
2.14 Query–document matching paradigms in neural IR 23
2.15 SBERT architectures 24
2.16 Pre-training tasks 25
2.17 Inverse Cloze Task 26
2.18 Self-attention patterns 28
2.19 Linformer 29

3.1 FEVER CS data example 31
3.2 ČTK data example 31

5.1 Pretraining on ČTK learning curve 41
5.2 Finetuning on ČTK learning curve 41

List of Tables

3.1 Label distribution in FEVER CS dataset 30
3.2 Label distribution in ČTK dataset 32
3.3 Productivity, utility, coverage and harmonic mean of productivity and coverage on FEVER CS 35
3.4 DCI calculated on translated FEVER CS dataset claims. 35
3.5 DCI calculated on ČTK dataset claims. 35
3.6 Productivity, utility, coverage and harmonic mean of productivity and coverage on ČTK 36

5.1 Precision at *k* on FEVER CS test set. 42
5.2 Recall at *k* on FEVER CS test set. 42
5.3 F1 at *k* on FEVER CS test set. 42
5.4 Mean reciprocal rank at *k* on FEVER CS test set. 42
5.5 Precision at *k* on ČTK test set. 43
5.6 Recall at *k* on ČTK test set. 43
5.7 F1 at *k* on ČTK test set. 43
5.8 Mean reciprocal rank at *k* on ČTK test set. 43

Introduction

The Italian philosopher Gianni Vattimo, who combines the perspectives of a philosopher and a sociologist in his work, came up with the concept of a *trasnparent society.* [Vattimo, 2013] The concept can be understood as another name for contemporary society, which is also referred to as society of mass communication. According to Vattimo, one of the factors that made the transition from modern to so-called postmodern society possible was the emergence of mass media, which continue to play a decisive role in shaping it.

Mass media was expected to make the world clearer, more "transparent" and create a more enlightened society. Nevertheless, it is they that, according to Vattimo, characterize this society as more chaotic and complex. With the vast amount of information that society generates, which can spread very quickly and efficiently thanks to mass media and social networks, it is impossible to achieve anything like a fully informed view of the world. The existence of that volume of information and data also allows the media to choose what information to report and how to frame it. In this way, as Vattimo writes, they will not only present a certain image of the world to their consumers, but also create it. However, is the image of the world thus constructed necessarily the real one, or at least sufficiently representative?

Such a phenomenon can be illustrated on the Czech media scene during the war in Syria and the subsequent migration wave, which was a major source of struggle for the entire European Union at the time. In 2018, the Czech media published over 80,000 articles and reports on refugees and migration. [Prokop, 2020] This was roughly one article for every two refugees and migrants arriving in Europe via the Mediterranean. By comparison, there were only about 20,000 articles and reports on the climate crisis (including articles on the melting of glaciers, carbon dioxide emissions, etc.) or on foreclosures and debt collection, which began to emerge as serious problems for Czech society in this period. [Prokop, 2020]

Fake news also contributes significantly to the aforementioned opacity, chaos and information overload. Fake or otherwise misleading news may be created intentionally or unintentionally, may be part of an organised disinformation campaign or may be mere solitary acts. In any case, this is one of the problems that is increasingly heard in the public space. In the context of fake news, there are very often calls for better education, more media lessons or work with critical thinking. These are all very important elements that should have a place in the educational process or at least be discussed. However, in the era of deep-fakes and the rapid improvement of machine-generated texts [Hao, 2020], it is also necessary to use these technologies and more advanced tools, which are now often mentioned in connection with the notion of automatic fact-checking.

The existence of a huge amount of information and data and the possibility of their very effective distribution to the desired target group are strong preconditions for the effective dissemination of disinformation. At the same time, these are precisely the same prerequisites that make it possible to combat it effectively.

Automated fact-checking cannot be thought of as a silver bullet to solve the problems of misinformation, populism and information overload. But rather as a tool to help people get their bearings and save valuable mental capacity. In that sense, it can also be any tool that helps journalists, media analysts and other professionals process data more efficiently, making their work better and more effective, which hopefully helps us all. The goal of this paper is to contribute to the development of such a fact-checking tool, specifically the part of the tool that is tasked with finding relevant documents from a large collection for a given query.

"The Democrats don't matter. The real opposition is the media. And the way to deal with them is to flood the zone with shit." [Illing, 2020]

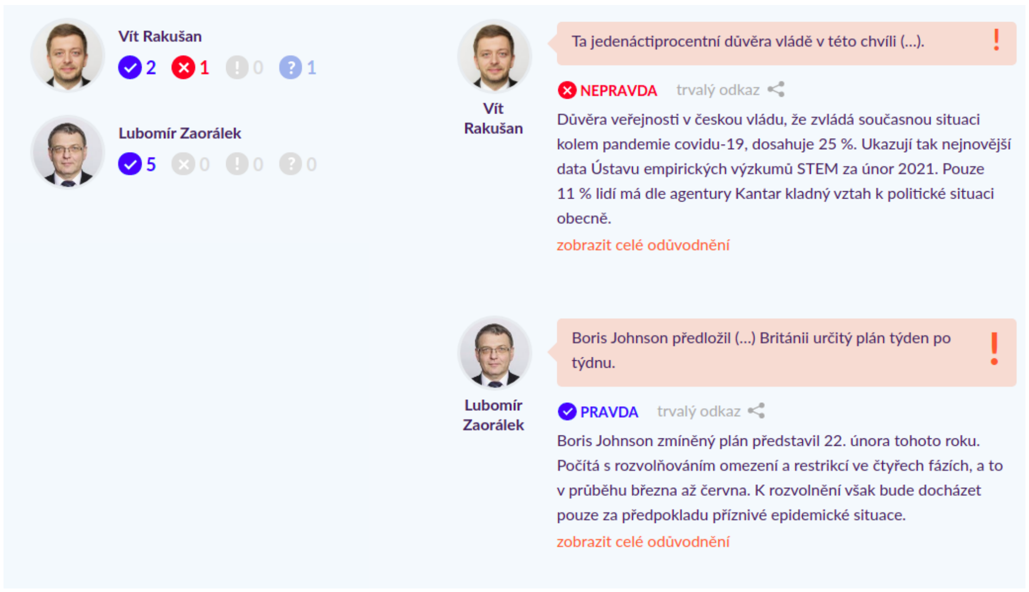
— Steve Bannon, *Advisor to Donald Trump in January-August 2017*

Chapter 1

Fact-checking

1.1 Problem description and goals

As the motivation for this work was already given in the introduction, the following part will describe the problem. Fact-checking is an essential component in the process of news reporting, or at least it should be. Commonly interchangeably used terms *fact-checking* and *verification* are becoming more and more differentiated recently. [Silverman, 2014] According to [Kovach et al., 2007], verification is the essence of journalism, a discipline further described as a scientific-like approach of getting the facts, which also involves verifying the source, time, location and other circumstances. Fact-checking on the other hand is more specific application of verification process focused on evaluating the veracity of a claim in some context.


Vít Rakušan
✓ 2 ✗ 1 ! 0 ? 1

Lubomír Zaorálek
✓ 5 ✗ 0 ! 0 ? 0

Vít Rakušan

Ta jedenáctiprocentní důvěra vládě v této chvíli (...).

✗ NEPRAVDA trvalý odkaz

Důvěra veřejnosti v českou vládu, že zvládá současnou situaci kolem pandemie covidu-19, dosahuje 25 %. Ukazují tak nejnovější data Ústavu empirických výzkumů STEM za únor 2021. Pouze 11 % lidí má dle agentury Kantar kladný vztah k politické situaci obecně.

zobrazit celé odůvodnění

Lubomír Zaorálek

Boris Johnson předložil (...) Británii určitý plán týden po týdnu.

✓ PRAVDA trvalý odkaz

Boris Johnson zmíněný plán představil 22. února tohoto roku. Počítá s rozvolňováním omezení a restrikcí ve čtyřech fázích, a to v průběhu března až června. K rozvolnění však bude docházet pouze za předpokladu příznivé epidemické situace.

zobrazit celé odůvodnění

Figure 1.1: Demagog fact-checking. Source: `https://demagog.cz`

Fact-checking can be illustrated on the Czech *Demagog*[1] project, which is based on *PolitiFact*[2] and *FactCheck.org*[3] projects, and which manually evaluates claims, usually

[1]`https://demagog.cz`
[2]`https://www.politifact.com/`
[3]`https://www.factcheck.org/`

from political debates and interviews. Because human claim verification is laborious, time-consuming, and mentally demanding, its full or partial automation would provide significant benefits and expand the possibilities of its use, such as verifying claim in real-time political debates. However, due to the complexity of this task, in the present it is rather a distant goal. A broader introduction to automated fact-checking is presented in [Thorne et al., 2018a].

We defined the fact-checking task as in the [Thorne et al., 2018b]. Verification of text claims against the knowledge base, where the knowledge base consists of text sources. The fact-checking pipeline according to the above definition is schematized in Figure 1.2 and it can be decomposed into several sub-problems. The first step is to find relevant documents from the collection for the given claim (document retrieval). From these relevant documents, the sentences from which the evidence is formed are then selected. Finally, the veracity of the claim is classified on the basis of formed evidence (Natural Language Inference task) .

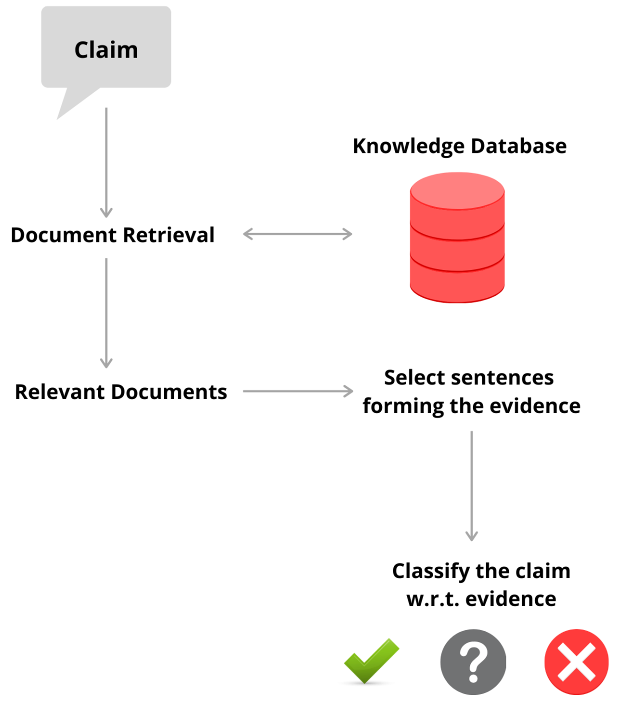

Claim
Knowledge Database
Document Retrieval
Relevant Documents
Select sentences forming the evidence
Classify the claim w.r.t. evidence

Figure 1.2: Fact-checking Pipeline Scheme

The aim of this thesis is to learn about large-scale document retrieval methods, examine them and evaluate them in the context of the above fact-checking pipeline. Therefore, in this thesis I will focus mainly on this part of the pipeline. Furthermore, I will focus more on neural models in order to explore their potential use in an end-to-end fact-checking system.

1.2 Related work

Automatic fact-checking has recently been gaining more and more attention both in public space and in academic literature. The creation of large-scale Fact Verification and Verification (FEVER) dataset [Thorne et al., 2018b] played a significant role in this as previously published datasets are incomparably smaller. The FEVER dataset defined the task of fact-checking much closer to real use-case by extending the task to an open domain, similarly to question-answering (QA) open-domain task in [Chen et al., 2017]. However, the questions usually contain some kind of clue to help you find the answer. This may not be the case for the claim whose factuality we want to verify, so finding the necessary evidence is more challenging.

With the publication of the FEVER dataset, the authors also called for a submissions in shared task of claim verification using Wikipedia abstracts (first paragraph of each Wikipedia article containing high-level information) as knowledge database. The majority of submitted works followed the pipeline designed in the [Thorne et al., 2018b] and regarding document retrieval the highest-recall solutions extracted noun phrases or named entities from the claim and used them as query in Wikipedia search API [Hanselowski et al., 2018]; [Thorne et al., 2018c]. In 2019, additional tasks were added to improve the resilience of systems. In the first task added, participants were asked to design a system that would generate adversarial examples that would be misclassified. In the second added task, the goal was to use these adversarial examples to create a more resilient system and improve classification performance. However, in terms of document retrieval, there was no significant change compared to the previous year [Thorne et al., 2019].

As far as the Czech language is concerned, we are not aware of any Czech dataset or fact-checking system, except for the dataset presented here [Přibáň et al., 2019].

This thesis is one of several works produced as part of the AI in Journalism project, which was supported by a **Transformation of Journalisms Ethics in the Advent of Artificial Intelligence (TL02000288)**[4] grant from the Technology Agency of the Czech Republic. Other works deal mainly with dataset production and the associated data annotation phase [Ullrich, 2021]; using hybrid (multi-stage) models for document retrieval [Dědková, 2021]; and document retrieval models supporting long inputs [Gažo, 2021].

[4]`https://starfos.tacr.cz/cs/project/TL02000288`

Chapter 2

Background

This chapter briefly introduces the task of document retrieval (DR) and traditional DR models. Then, it provides a description of recently popular methods combining neural approach with the traditional approach and end-to-end neural approach with a short introduction into language modelling and transformer architecture that is currently prevalent in the field of natural language processing (NLP).

2.1 Document Retrieval

Document retrieval task can be defined as the matching of an input query with relevant documents from a document collection, which are typically a very large (tens or more millions of documents). Both the input query and documents are more or less structured textual data, therefore the task is sometimes called text retrieval [Manning, 2008]. The task of finding a relevant documents is usually completed by ranking them by relevance, so the highly relevant documents appear at the top of the list. The term document can be perceived as overloaded here and in fact it can be a collection of documents, article, paragraph, sentence or even single word depending on the particular case. The whole process is illustrated in the below scheme (see Figure 2.1).

For the sake of clarity, I might use the term information retrieval (IR) later in this thesis. The IR is a more general term compared to DR, which is typically categorized as a branch of IR in the taxonomy and classic problem of IR [Mitra Bhaskar, 2018]. Despite the above, the two terms have very similar meaning and unless stated otherwise, I will continue to use them interchangeably in this work.

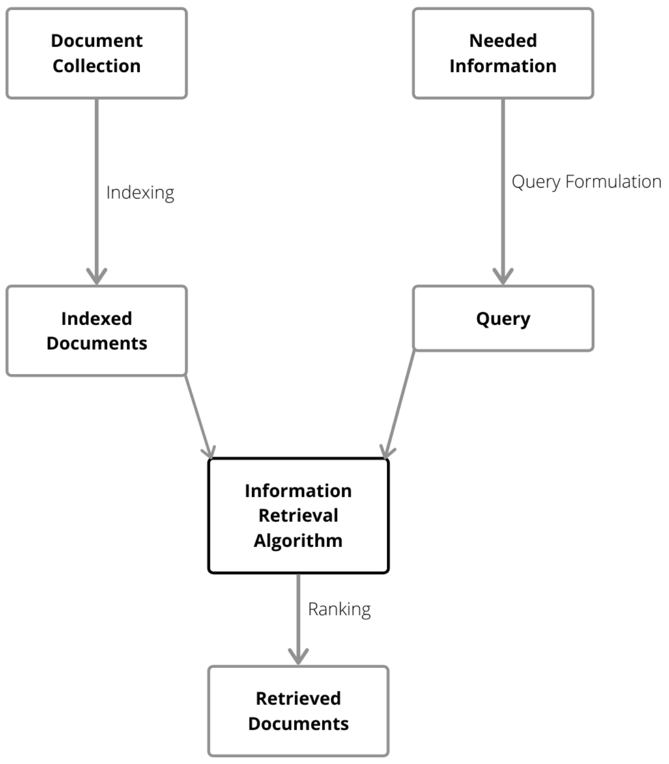


Figure 2.1: Document Retrieval Scheme

In the Text Retrieval Conference (TREC) competition [Craswell et al., 2020], they distinguish retrieval models into three categories:

1. **Traditional** - if it uses only TF-IDF/BM25 like models;
2. **Neural** - if it employs some form of neural network based approach, but does not fall into "Neural using Language Models" category;
3. **Neural using Language Models** - if it uses large scale pre-trained neural language models (LM).

In this work, we have adopted this division. Nevertheless, we continue to deal only with the traditional models and neural models using LM, where the latter is the primary object of our interest.

2.2 Traditional Approach

Traditional approaches proceed from an empirically found law called *Zipf's law*, which is commonly used as a model of the distribution of terms in a collection. This law states that the frequency f_i of the *ith* most common term in a collection is proportional to the inverse of its rank:

$$f_i \approx \frac{1}{i} \tag{2.1}$$

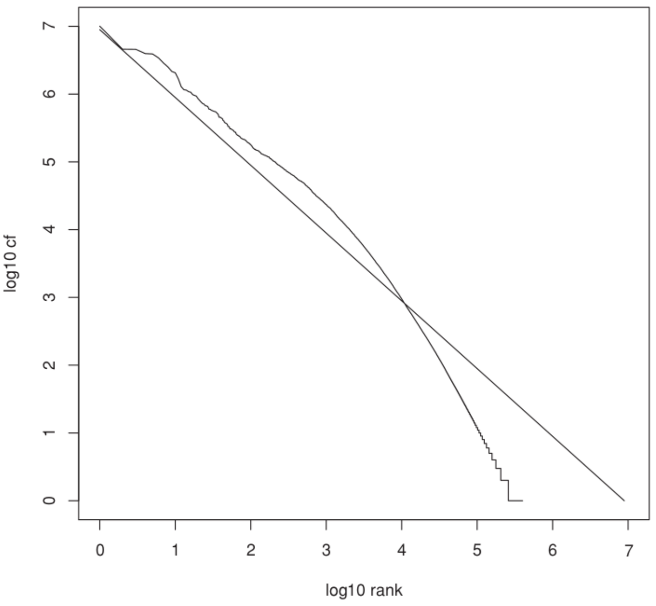


Figure 2.2: Zipf's law for Reuters-RCV1. Frequency is plotted as a function of frequency rank for the terms in the collection. [Manning, 2008]

2.2.1 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a model (or weighting scheme more precisely) created with a knowledge of the Zipf's law. In the field of information retrieval, it can be considered as "evergreen" — very popular and efficient method, which is implemented in a vast number of systems and applications as a search mechanism. TF-IDF is based on an underlying assumption that if terms from a given query are present more often in a document A than in a document B, then there is a closer relation between the query and document A compared to the document B (document A should have a higher score). That is represented by the *term frequency (tf)* part where a weight is assigned to each term t in a document d according to its frequency in the document.

However, the words do not provide the same amount of information. Consider a document containing frequently the word "bird" and compare it with analogous knowledge only with a word "the". Common words that are distributed over numerous documents provide only poor indication of a document's content. This introduces the second assumption represented by the *inverse document frequency (idf)* part which scales the weight given by the *term-frequency.*

$$idf_t = \log \frac{N}{df_t} \tag{2.2}$$

where:

N is the total number of documents in a collection

df_t is a document frequency of a term t defined as the number of documents in the collection containing the term t

Both parts can be combined into a single formula expressing the weight of the term t in a document d.

$$\textit{tf-idf}_{t,d} = \textit{tf}_{t,d} \cdot \textit{idf}_t \tag{2.3}$$

Mechanics of weighting can be summarized into 4 illustrative cases

1. term t is assigned **highest** weight in document d if t occurs many times within a small number of documents;
2. term t is assigned **lower** weight in document d if t occurs fewer times, or occurs in many documents;
3. term t is assigned **lowest** weight in document d if t occurs in all documents;
4. term t is assigned weight **zero** in document d if t does not occur in the document at all. [Manning, 2008]

By computing these weights for all terms per document from a collection, we can get the indexed collection (see Figure 2.1). Each document d will be represented by a vector of $\textit{tf-idf}_d$ weights, where each part of the vector corresponds to *tf-idf* weight of term t from a dictionary[5]. The document vector representation will be usually a very long (a lot of different words in the dictionary) and sparse (typically only a minority of existing words is used in a text of a given topic; also depends on the length of the text) vector.

During the retrieval time when it is necessary to search for relevant documents given a query q, the query is converted into *bag of words (BOW)* vector representation (see Figure 2.3). And since such a representation will have the same dimension as the representations of documents in the collection, score for query-document pair expressing the degree of relevance between them can be obtained by a simple dot-product. It is obvious that longer questions will reach a higher score due to the higher number of words, therefore both the document and query vectors are length-normalized (see equation 2.4).

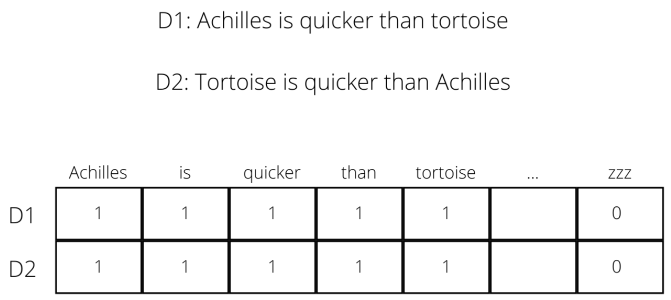
$$sim(q,d) = \frac{V_q \cdot V_d}{|V_q| \cdot |V_d|} \tag{2.4}$$

where:

$|| \cdot ||$ stands for the Euclidean norm [Manning, 2008]

The BOW representation is one of the sparse representations because it usually has the vast majority of values equal to zero. In this case, both the query and the document are represented using BOW, which has some implications. The key simplification of this approach is that it does not include word order information or semantics information. To illustrate that, TF-IDF document representation of the document "Achilles is quicker than tortoise" is equal to the document representation of "Tortoise is quicker than Achilles" (see Figure 2.3). Even though the documents have an opposite meaning, they are considered identical by having the identical document representation. [Manning, 2008] This can be partially suppressed by using n-gram instead of individual words, which adds some semantic information.

[5]the length of the vector is equal to the number of terms in the dictionary



D1: Achilles is quicker than tortoise

D2: Tortoise is quicker than Achilles

	Achilles	is	quicker	than	tortoise	...	zzz
D1	1	1	1	1	1		0
D2	1	1	1	1	1		0

Figure 2.3: Bag of words representation. Dictionary is formed by words Achilles, is, . . . , zzz.

2.2.2 Best Match 25 (BM25)

There is a more complex term weighting scheme called BM25 [Robertson et al., 2009] or Okapi weighting, which is expected to provide better results in practise. This approach proceeds from probabilistic information retrieval theory, but at the same time there is a strong resemblance to TF-IDF. The relevance of the document d to the query q is expressed as

$$BM25_{q,d} = \log idf_t \cdot \frac{(k_1 + 1)tf_{t,d}}{k_1((1-b) + b \cdot (L_d/L_{avg})) + tf_{t,d}} \cdot \frac{(k_3 + 1)tf_{t,q}}{k_3 + tf_{t,q}}. \tag{2.5}$$

The BM25 can be broken up into 3 terms. The first term is the inverse document frequency which approximates the missing user-generated relevancy judgments. In case relevancy judgements are available, the following full form (equation 2.6) can be used

$$\log \frac{(|VR_t| + 0.5)/(|VNR_t| + 0.5)}{(df_t - |VR_t| + 0.5)/(N - df_t - |VR| + |VR_t| + 0.5)} \tag{2.6}$$

where:

$\lvert VR_t\rvert$	stands for the set of relevant documents to term t
$\lvert VNR_t\rvert$	is the set of non-relevant documents to term t according to user feedback [Manning, 2008]

The second term represents the document term frequency and document length scaling. Document term frequency builds on the assumption of the term frequency from TF-IDF. BM25 compared to TF-IDF does not assume that a document A containing 2,000 times relevant term t is $2\times$ more relevant than a document B, which contains only 1,000 times the same term t. But the effect decreases rapidly after reaching a certain level of saturation. This level of saturation can be regulated by the positive tuning parameter k_1.

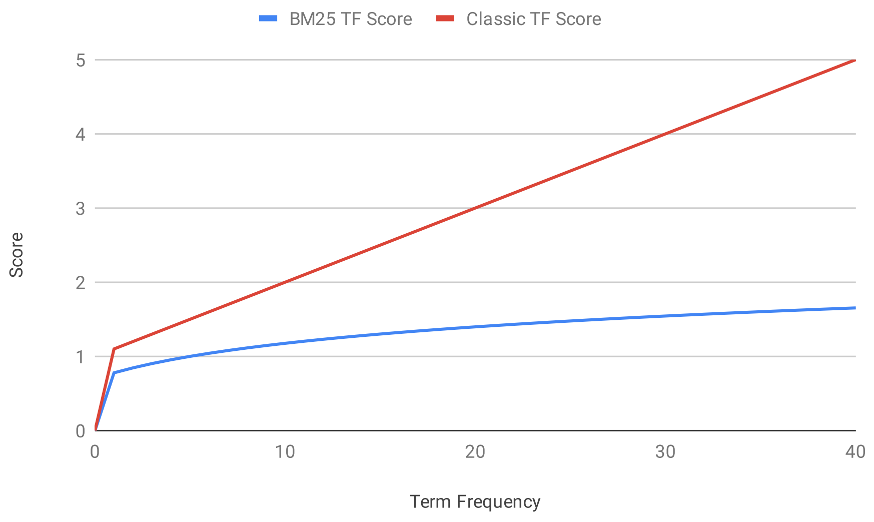


Figure 2.4: Term frequency of TF-IDF vs. term frequency of BM25. Adopted from [Turnbull, 2015]

In the TF-IDF weighting scheme document length is not directly involved, but in the case of BM25 it does not apply. The assumption is that if a term is observed in a short document then it has a bigger impact on the result than it would have in a longer document. Intuitively, assuming a single-word query "corruption" there is a higher chance that a short article is more relevant than an entire book if both documents contain the same number of that word. Size of the effect is calculated as a proportion of document length to an average length of all documents in the collection. The effect on the whole part is calibrated by a tuning parameter b ranging from 0 (ignoring document length) to 1 (full influence of the document length).

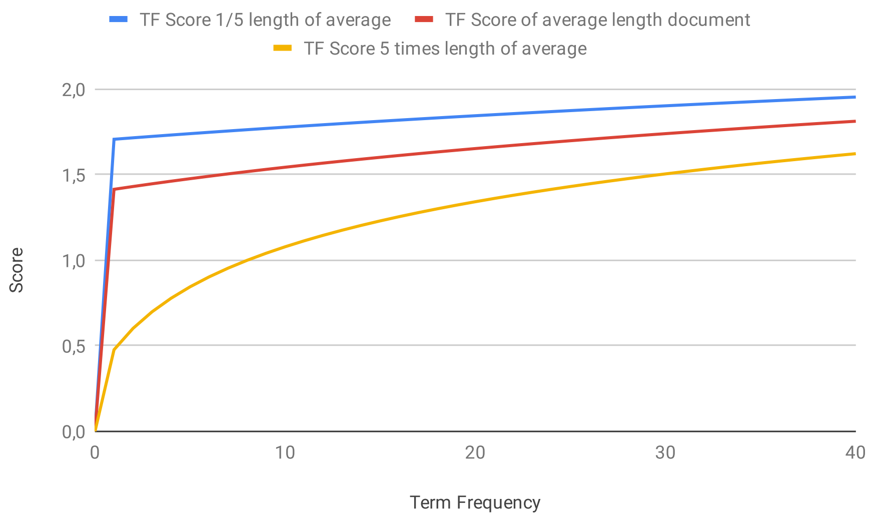


Figure 2.5: Document length in BM25. Adopted from [Turnbull, 2015]

The third term scales the term frequency in the query. It has the analogical form as the document term frequency. Since queries are usually much shorter than documents, a simplified formula can be used. However, if the query is longer, such as a full paragraph, it may be more appropriate to use the full formula as in the second part of the equation 2.5. [Manning, 2008]

Further in the thesis I will refer to the models described in this section as TF-IDF-like models, as they are in principle very similar.

2.3 Language Modelling

A language model (LM) is a general name for statistical models which are able to predict the probability of a next word given some text input, simply put. More formally, it can be described as a probability distribution over sequences of tokens. Then the probability of a sequence t can be expressed as a product of conditional probabilities

$$p(t) = \prod_{i=1}^{n} p(t_i|t_1, \ldots, t_{i-1}) \tag{2.7}$$

Depending on the definition of the token, LM can be of different types. In case the token is a word, then the LM is called an unigram model and the probability of each word depends only on its own probability in the document or corpus

$$p(t) = \prod_{i=1}^{n} p(t_i) \tag{2.8}$$

Generally, tokens can be modelled as *n-grams* (see equation 2.7), then the mechanics of using them in the model can differ, which leads to other types of LM's.

In information retrieval context, the LM's are used for some time already. The use is based on the idea that a document is relevant to a query if the LM of the particular document is likely to generate the query, which happens in the case if the document contains the query words often. [Manning, 2008] One can notice a clear relation to traditional TF-IDF approaches in the idea.

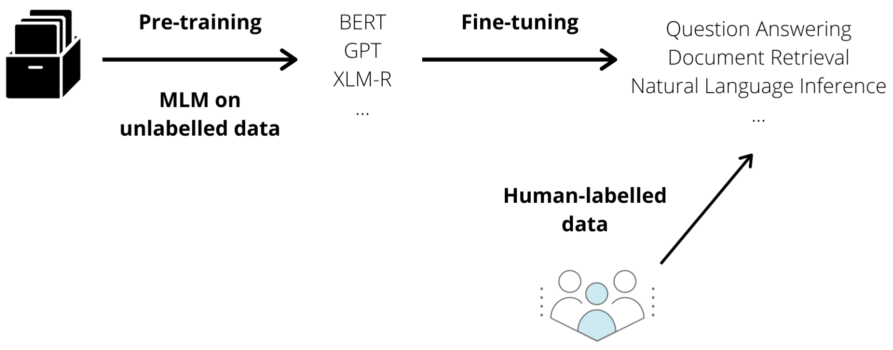


Figure 2.6: Transfer learning. Adapted from [Ruder, 2021]

Even though the LM's can be used "directly" for document retrieval (see query likelihood model in [Manning, 2008] for example), their use went in another direction recently. In that direction, which is characteristic of the whole NLP field, LM's are primarily used for generating a rich contextual representation of texts that are consequently used as input features for other task-specific models, which are further trained (finetuned). [Ruder, 2021] This transfer learning based approach both enables to utilize huge and expensive to train LM's and significantly improve the state-of-the-art on various downstream tasks. This paradigm has also been established in the field of information retrieval, and recent work suggests that even here this approach can outperform traditional baselines. [Craswell et al., 2020]; [Lee et al., 2019]; [Karpukhin et al., 2020]. In the vast majority of cases, LM's are currently implemented using Transformer architecture and trained on a huge amount of data (see section 2.3.1).

2.3.1 One Model to Rule Them All!

Since 2017, the NLP world is dominated by a family of models called Transformers. They were firstly introduced in [Vaswani et al., 2017] and later become widely popular with their variations as BERT [Devlin et al., 2018] or GPT-2 [Radford et al., 2019]. There exist an overwhelming number of more or less different variants, which implementations can be found in Hugging Face's library called Transformers. [Wolf et al., 2020]

Transformer models got very rapidly adopted and applied to a diverse spectrum of tasks. From machine translation, question answering and document retrieval to non-NLP tasks as well. Transformers enable to model sequences — often called sequence-to-sequence models — and due to better parallelization offers faster processing compared to previously very frequently used recurrent neural networks (RNN).

The transformer architecture is composed of an encoder and a decoder unit (see Figure 2.7). Both units contain 6 repeatedly stacked identical layers. Each layer consists of a self-attention layer followed by a normalization layer and a fully connected feed-forward layer with a normalization layer. For both sub-layers there is a residual connection present.

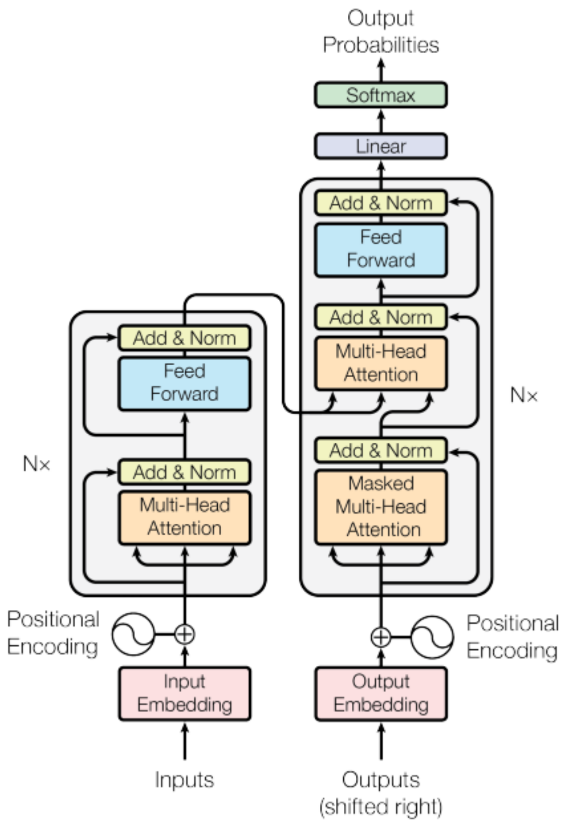


Figure 2.7: Transformer architecture. Encoder on the left and decoder on the right. [Vaswani et al., 2017]

In addition to the encoder, the decoder contains a third sub-layer, which performs multi-head attention over the output from the encoder stack. Self-attention layers of the decoder are modified, so they have access to only already decoded positions (prediction for position i can depend only on the known outputs at positions less than i). This masking is crucial as it keeps the model auto-regressive (with leftward information flow) in all steps. Therefore, every prediction at a position i depends only on the known outputs from positions before i.

2.3.1.1 Attention Mechanism

The very core of the transformer is an attention mechanism. **Self-attention** of a word in an input sentence provides its relation to all the other words in the input sentence as it is depicted in Figure 2.8.

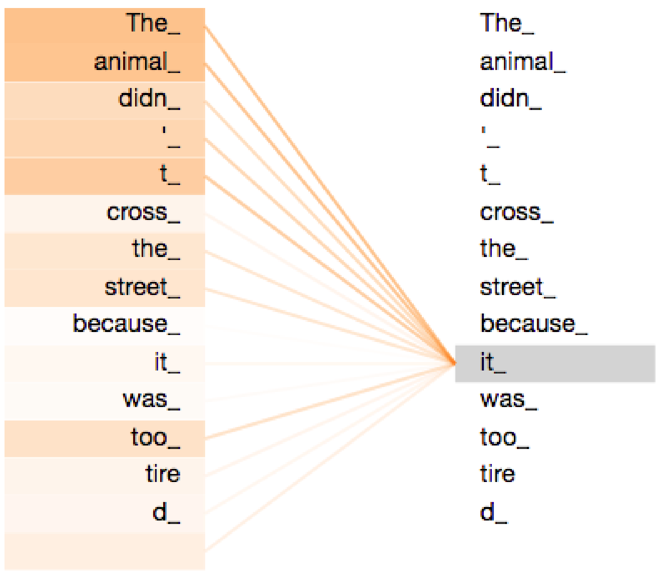


Figure 2.8: Illustrated attention mechanism. [Alammar, 2018]

Self-attention works with 3 matrices: query Q, key K and value V which represent certain abstractions. These matrices are obtained by projecting each word, respectively its word embedding, using trained weight matrices for each particular abstraction. The self-attention of a word is computed by multiplying its corresponding query vector from Q and the key vector from K. This is normalized by dividing it by the square root of the dimension of the key vector and passing it through the softmax. Multiplying this softmax score with each value vector from V and summing them results in the **scaled dot-product attention**.

Instead of performing this scaled dot-product attention once, the **multi-head attention** mechanism calculates the scaled dot-product attention multiple times in parallel. This is done by creating a multiple of different Q, K and V matrices with different learned linear projections. That will produce multiple different output attentions, which are concatenated and projected to the final output dimension.

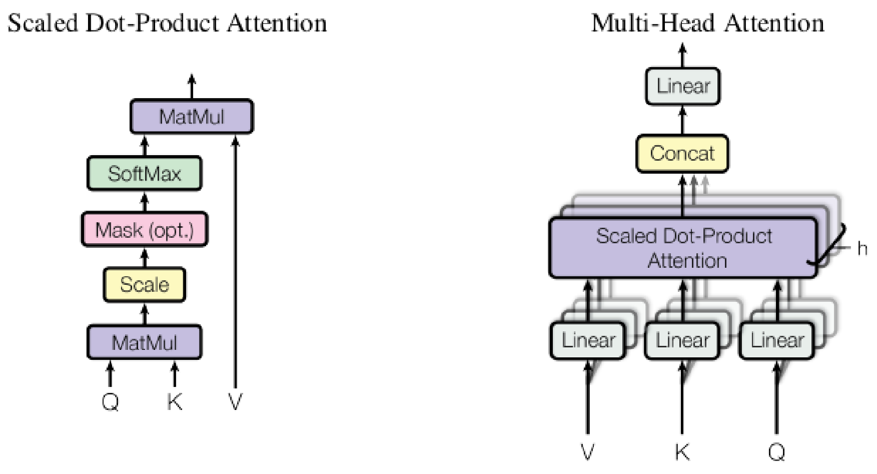


Figure 2.9: Scaled Dot-Product Attention and Multi-Head Attention [Vaswani et al., 2017]

2.3.1.2 Positional Encoding

Since the transformer model does not process words in a sentence sequentially, but in parallel (all at once), it loses information about order. This is solved by the **positional encoding**, which provides a sense of position or order for each word. The positional encoding is a d-dimensional vector with the same dimension as has word embedding and components computed by the following rule

$$\begin{aligned} PE_{pos,2i} &= \sin\left(\frac{pos}{10\,000^{\frac{2i}{d}}}\right) \\ PE_{pos,2i+1} &= \cos\left(\frac{pos}{10\,000^{\frac{2i}{d}}}\right) \end{aligned} \tag{2.9}$$

where:

pos is the desired position in the input sentence

d is the encoding dimension, which must be divisible by 2

i is the dimension. Each dimension of the PE vector corresponds to a sinusoid and the wavelengths form a geometric progression from 2π to $10\,000 \cdot 2\pi$

The positional encoding vector is then summed with a word embedding which explains the choice of dimension of PE vector. [Vaswani et al., 2017]

$$PE_{pos} = \begin{bmatrix} \sin(\omega_1) \\ \cos(\omega_1) \\ \vdots \\ \sin(\omega_{d/2}) \\ \cos(\omega_{d/2}) \end{bmatrix}$$

where $\omega_j = \frac{1}{10\,000^{2j/d}}$.

2.3.2 BERTology

Bidirectional Encoder Representations from Transformers (BERT) presented in [Devlin et al., 2018] was designed to pretrain deep bidirectional representations from unlabeled data. These representations are obtained by implementing bidirectional self-attention that computes both with left and right context in all layers. This fundamental difference from unidirectional (left to right) models makes better use of pre-trained representations especially in approaches involving fine-tuning, they argue in the paper.

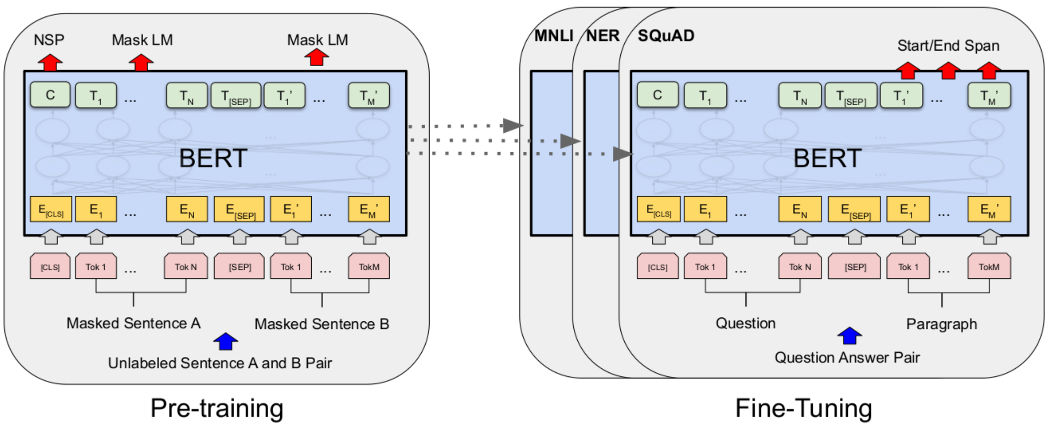


Figure 2.10: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers). [Devlin et al., 2018]

BERT workflow consists of two steps: pre-trained on unlabeled data over different pre-training tasks and fine-tuning on downstream task (e.g. question answering or name entity recognition) using labeled data. So at the end, there was a fine-tuned model variation for each of the downstream task. The pre-training step is performed indirectly by using two pre-training tasks. The first is **masked language model (MLM)**, which randomly masks some of the tokens from the input and model tries to predict the original words in masked places based only on its context. The second is **next sentence prediction (NSP)**, where the task is to classify whether a two given sentences are neighbors or not. [Devlin et al., 2018]

The model architecture is a multi-layer bidirectional Transformer encoder based on the original Transformer implementation [Vaswani et al., 2017]. A benefit of this model is its unified architecture that differs minimally between pre-training and fine-tuning setup.

This model is so popular and widely used that it has led to the creation of a meta-study aptly named Bertology, which provides an overview of the accumulated knowledge about BERT. [Rogers et al., 2020]

2.3.3 As many languages you know, as many times you are a language model

When working with text data, using various NLP techniques and especially expensive to train language models, one can notice a big disproportion. Only a very small number of languages have available data and thus pre-trained language models. Even in the small cluster of languages having the richest language resources, there is a clear gap between English and the rest of them (see Figure 2.11). Despite the data scarcity problem does not have an easy solution yet, there are some interesting research results that can help to reduce the impact of the gap. Training multilingual and cross-lingual models provide some of them.

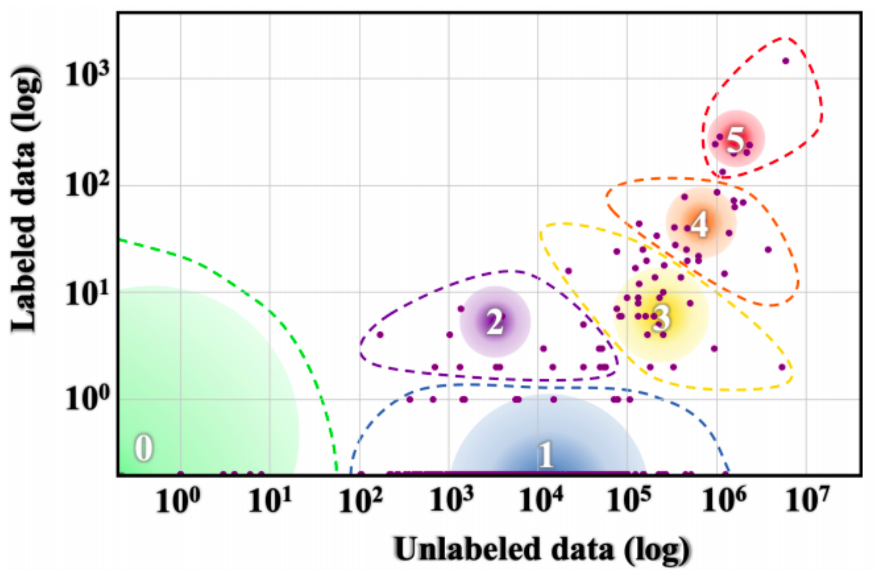


Figure 2.11: Language Resource Distribution: The size of the gradient circle represents the number of languages in the class. The color spectrum VIBGYOR, represents the total speaker population size from low to high. Bounding curves used to demonstrate covered points by that language class. [Joshi et al., 2020]

Pre-training of LM is performed indirectly by using pre-training tasks. In addition to the above **MLM** and **NSP**, there exists other pre-training tasks for example **causal language modelling (CLM)**, which models the probability of a word given previous words in a sentence. Usually, there is no need for labelled data in the pre-training tasks as they have an unsupervised nature. This becomes a great benefit given the fact that only a minority of existing text data is labelled and that LM needs a substantial amount of data for pre-training.

The process of pre-training multilingual LM is very similar as for pre-training monolingual LM's. The primary and obvious difference is that multilingual ones are trained with data of multiple languages with the goal to learn a coexisting vector-space representations for them. It was shown that low-resource language often benefits from training together with a higher-resource language, especially when it shares a significant fraction of its vocabulary. [Lample et al., 2019] In addition to universal pre-training tasks, there exist also tasks specific to multilingual setup. For example, in [Lample et al., 2019] they presented **translation language modelling (TLM)** task, which utilizes parallel data resulting in strong cross-lingual features provided by the model.

The choice of pre-training tasks can significantly affect the model, its vector space and thus the provided representations, especially in the multilingual setup. The term *cross-linguality* refers to the property of the model to create general text representations across languages. Simply put, given two semantically very similar words in different languages, the model should provide representations which are also very close.

Monolingual models are in most cases superior in performance compared to multilingual models. [Martin et al., 2020]; [Dumitrescu et al., 2020] Nevertheless, XLM-RoBERTa (XLM-R) proposed in [Conneau et al., 2020] is a multilingual model trained on a very large scale regarding the data (2.5 TB) as well as the number of languages (one hundred). It also showed that a multilingual model can be competitive with a monolingual ones on some tasks while providing solid cross-lingual features. The performance is quite surprising as the model was trained only on unsupervised data using MLM without an explicit training signal providing information about the language of a given text such as parallel data provides. Further findings shows that if you train a large enough network on a large enough amount of data, you can get equivalent performance to a monolingual model, while being able to develop model that can do well on multiple languages at once. [Li et al., 2021]

Considering Czech language, XLM-R's Czech reading comprehension was examined in [Macková et al., 2020], where the XLM-R showed competitive performance compared to monolingual models even without training on parallel data. And a very recently released monolingual Czech model Czert [Sido et al., 2021], which in most tasks surpasses the performance of multilingual models, but unfortunately the comparison with XLM-R is missing.

2.3.4 Distillation

In recent years, there has been a trend towards increasing language models, which produced 8.3 billion parameters Megatron LM [Shoeybi et al., 2019] and currently the biggest 175 billion parameters GPT-3 [Brown et al., 2020]. They provide significant improvements in various NLP tasks — the latter was not even made public in its full scale due to safety-related concerns — but for a very high price. [Schwartz et al., 2020] The substantial size of these models also makes them slower and less convenient due to a high computational and memory requirements. This can significantly reduce the possibility of their deployment, particularly in document retrieval, where the run time is expected to be near real-time.

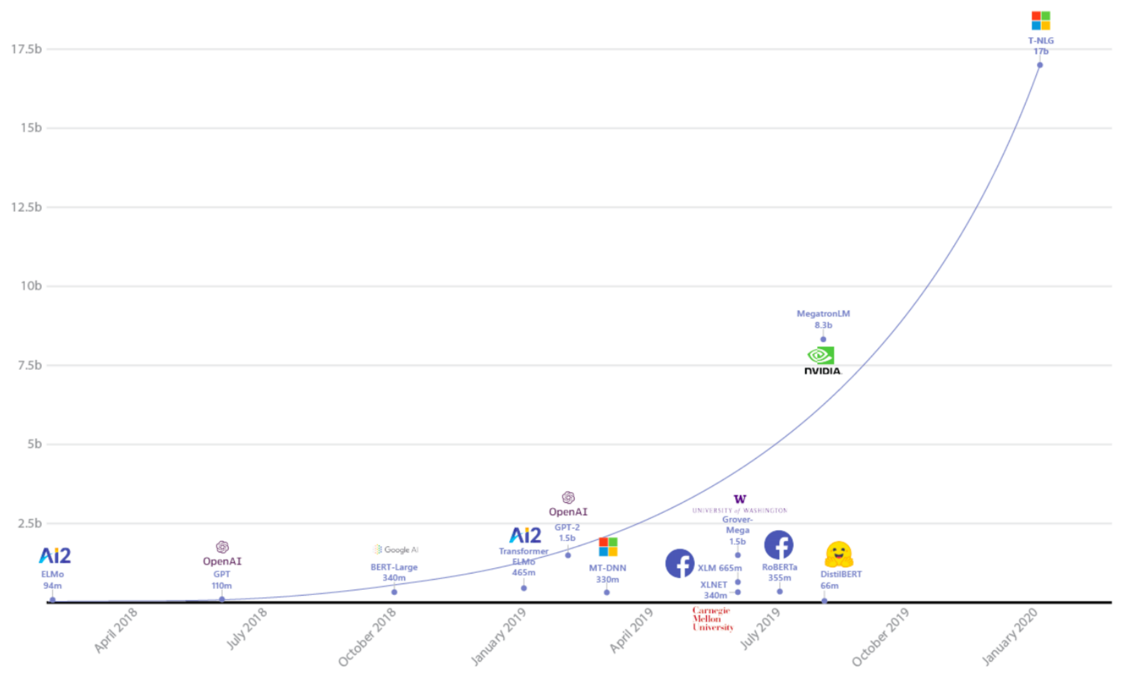


Figure 2.12: Models compared by a number of parameters (currently the biggest model GPT-3 with a significant margin is not included). [Rosset, 2020]

Recent results showed that it is possible to reach a very similar performance on many downstream tasks using much smaller LM pre-trained with *knowledge distillation technique*. [Sanh et al., 2019] Knowledge distillation is a compression technique presented in [Buciluundefined et al., 2006] and later applied to neural networks in [Hinton et al., 2015]. This technique is based on an elegant idea where the smaller (distilled) model (student) is trained to reproduce the behavior of a larger model (teacher) or an ensemble of models.

In the case of BERT, the distilled version is created by lowering the number of layers, removing token-type embeddings and poolers, while the general architecture is preserved. An important step is the initialization, which affects the speed of convergence of the student model. Due to the common dimensionality, the student model is initialized by the teacher model weights, which provides fast convergence. [Sanh et al., 2019]

Another interesting applications of distillation are proposed in [Hofstätter et al., 2020], where they use cross-architecture knowledge distillation to improve the effectiveness of the neural passage ranking models with efficient query latency; or in [Reimers et al., 2020], in which distillation is used for transforming monolingual sentence embeddings into multilingual by aligning vector spaces between languages.

2.3.5 Tokenization

In NLP, tokenization is a process of splitting a text into smaller units called tokens. A token can be a word, subword, or character. The key motivation is to have a finite set of symbols (vocabulary) which parts can be combined to get the desired result. Simple to describe, yet in practice it is a more complicated problem.

There is a tradeoff between the size of the vocabulary affecting the computational complexity and performance of the model. Having tokens on the level of characters will result in a small memory footprint as there is a relatively small number of existing characters, but it will be very tricky to learn a general representation of a single character. That can result in lower performance of the model. On the other hand, having tokens on the level of words will increase the memory footprint and thus also the computational complexity (consider different endings for a single word stem), but it will be much easier to learn the representation of a whole word, which will have a positive effect on the performance of the model.

Good tradeoff provides subword tokenizers, which keep a reasonable vocabulary size while learning a meaningful context-independent representation is enabled. One of such tokenizers is the algorithm called **WordPiece**, which is used for BERT transformer model. WordPiece initializes the vocabulary with every character present in the training text and it gradually learns a given number of merge rules (see Algorithm 1).

Algorithm 1 WordPiece Algorithm [Schuster et al., 2012]

```
Initialize: Initialize the vocabulary with all the characters present in the training text.
Step1: Build a language model on the training data using the vocabulary from the previous step.
Step2: Generate a new subword unit (wordpiece) by combining two units out of the current vocabulary to increment the vocabulary by one. Choose the new subword unit out of all the possible ones that increases the likelihood on the training data the most when added to the model.
Step3: Go to Step2 until a predefined limit of subword units is reached or the likelihood increase falls below a certain threshold.
```

Similarly to WordPiece, **Byte Pair Encoding (BPE)** tokenizer also gradually learns a given number of merge rules, but in a different way. [Sennrich et al., 2016] BPE tokenizer assumes the data are already splitted into words. Having words represented as a sequence of characters, the vocabulary is initialized in the same way as for the WordPiece algorithm.

BPE then iteratively counts all subword pairs and replace each occurrence of the most frequent pair (e.g. “A”, “L”) with a new symbol “AL”. Frequent subwords or even whole words are eventually merged into a single token. The final vocabulary size is equal to the number of distinct characters in the training data (the size of initial vocabulary) plus the number of merge rules, which is the only hyperparameter of the algorithm. One can see the key difference is in choosing a merge rule step. BPE chooses simply the most frequent subword pair in the data, compared to the pair maximizing the likelihood chosen by WordPiece.

Tokenizers stated above assume either the training text is already splitted into words or it can be relatively easily done by some pre-tokenizer that assumes words are separated by whitespace. This approach becomes a problematic at the moment we try to tokenize a language, which does not use whitespace to separate words. A possible solution is to use pre-tokenizer created for that particular language. More general solution is to use **SentecePiece** tokenizer which enables to train subword models directly from raw sentences by treating whitespace as one of the tokens inside the vocabulary. SentecePiece implements an optimized BPE [Sennrich et al., 2016] with $\mathcal{O}(n \log n)$ due to using priority queue and unigram language model [Kudo, 2018]. Besides that, SentecePiece provides further functionality — e.g. manages a vocabulary to id mapping to directly convert the input text into an id sequence; it enables customizable character normalization; it makes the reproduction of preprocessing steps easy by embedding all the rules and parameters into self-contained model by design — which makes it end-to-end system that does not depend on any language-specific processing. Due to that, SentecePiece is a very convenient tokenizer for multilingual models.

Those tokenizers mentioned above are only a subset of the existing ones. A nice overview of subword tokenizers as well as their implementation provides HuggingFace library. [Wolf et al., 2020]

2.4 Hybrid Approach

Another approach to document retrieval is the *hybrid approach* that combines traditional methods and language models. With the advent of transformer models, there occurred many works experimenting with this approach, for example [Nogueira et al., 2019a] or other works implementing integrated systems such as Bertserini [Yang et al., 2019b] or Birch [Yilmaz et al., 2019].

The underlying idea behind this approach is to divide the task of DR into two parts as was presented in [Chen et al., 2017]. In the first part, usually called *retriever*, is performed a rough pre-selection of documents using typically faster and more efficient traditional methods, which allow to realize retrieval in sublinear time using the inverted index. In the second part, computationally more demanding neural model is used to re-rank the relatively small number of already pre-selected documents. That part is usually called the *reader* (see Figure 2.13).

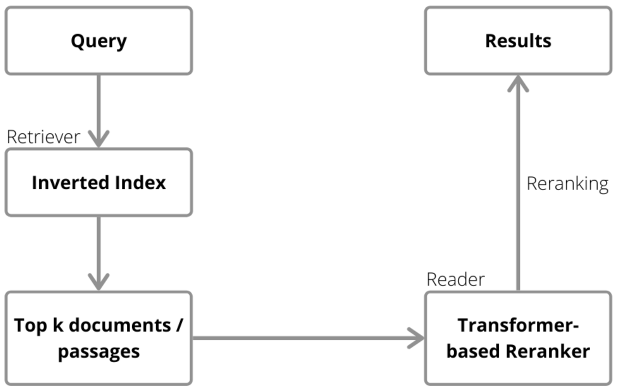


Figure 2.13: Hybrid Approach Scheme

In general, specific document retrieval pipeline depends on the task and the data we are working with. Therefore, the pipeline can be of arbitrary complexity and it can have more steps than only retrieval and re-ranking. In [Diggelmann et al., 2020] their retrieval pipeline utilizes natural granularity of textual data and consists of three steps:

1. document retrieval using BM25 operating on full-length Wikipedia articles, returning top d documents;

2. sentence retrieval using pre-trained LM for generating sentence embeddings and returning top s sentences;

3. sentence re-ranking using the same LM for computing relevance scores of claim-sentence pairs and providing re-ranked top s sentences

The BM25/TF-IDF retrievers are very efficient and proven by a wide range of applications in industry as they provide reasonable trade-off between latency and accuracy. Their popularity is also evident from the tables with the submitted solutions of the MS MARCO passage and document ranking tasks[6].

However, they have a clear disadvantage as the quality of the retrieved results depends on the performance of the retriever, which lacks the semantic and contextual understanding, as was described in section 2.2.1. Although there exists techniques, that try to mitigate the term mismatch error by enriching documents with potential query terms, presented in [Nogueira et al., 2019c]; [Nogueira et al., 2019b] or replace the BM25's term frequency with LM-estimated term importance [Dai et al., 2019], their effects are limited. End-to-end neural retrieval that provides deeply-contextualized semantic representations of queries and documents bridging the widespread problem of vocabulary mismatch can offer a better solution.

2.5 Neural Approach

With the successful application of neural models in combination with traditional models, the research community focused on the end-to-end neural retrieval, also referred to in the

[6]`https://microsoft.github.io/msmarco/`

literature as dense retrieval. Their efforts crystallized into several different paradigms (see Figure 2.14).

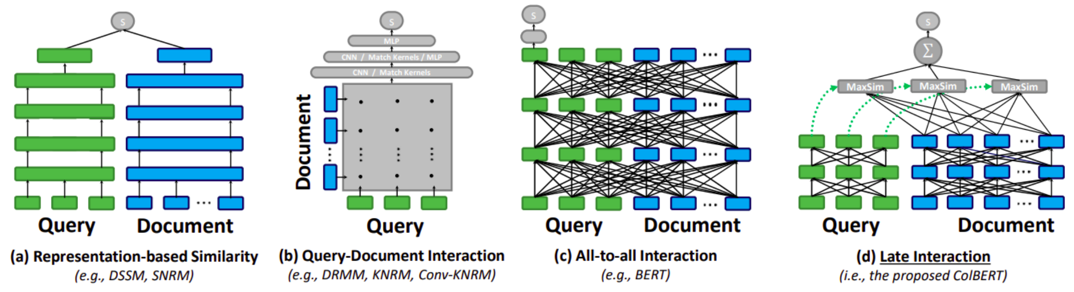

s
Query
Document
(a) Representation-based Similarity
(e.g., DSSM, SNRM)
MLP
CNN / Match Kernels / MLP
CNN / Match Kernels
Document
Query
(b) Query-Document Interaction
(e.g., DRMM, KNRM, Conv-KNRM)
Query
Document
(c) All-to-all Interaction
(e.g., BERT)
Σ
MaxSim
MaxSim
MaxSim
Query
Document
(d) Late Interaction
(i.e., the proposed ColBERT)

Figure 2.14: Schematic diagrams illustrating query–document matching paradigms in neural IR. [Khattab et al., 2020]

2.5.1 Cross-Attention Paradigm

This paradigm enables to model interactions between words within as well as across a query and document at the same time, see Figure (c) in 2.14. This is realized by concatenating the query with the document separated by some special token and inputting them into the transformer model. A linear layer or multilayer perceptron can be appended to the transformer predicting the relevancy score between the query and document or binary relevant/non-relevant output signalizing whether the document is relevant to the query.

This interaction-based paradigm tends to be more effective compared to two-tower paradigm from section 2.5.2 as it might be very hard to represent a single document by a single low-dimensional vector, especially when the document is long and contains a mixture of different topics. [Mitra Bhaskar, 2018]

On the other hand, it would be impossible for the model to work with reasonable latency, especially in the case of large collections of documents (tens of millions of documents) and quadratic computational complexity (attention mechanism). Therefore, its application in early document retrieval stage is not particularly suitable.

2.5.2 Two-Tower Paradigm

Two-tower paradigm, how is it called in [Chang et al., 2020] (further it can be found under siamese-network or representation-based approach names in the literature), is illustrated in Figure 2.14 (a). In this paradigm, the embeddings for a query and for documents are independently computed and then estimates of the relevance between the query and each document are calculated using some similarity / distance metric or dot-product. The embeddings can be computed either by the same or different language models. This separation of query branch and document branch enables to use dense retrieval in end-to-end fashion as the collection of documents can be pre-computed into embeddings offline, which comes as a great latency-related benefit during runtime.

Line of works motivated by the finding that sentence-level embeddings perform better when using transfer learning for downstream tasks than word-level embeddings [Cer et al., 2018] use this paradigm for training Sentence-BERT (SBERT) [Reimers et al., 2019] model or TwinBERT [Lu et al., 2020], which are dealing with the semantic search task. Their approach is to add a pooling layer to the BERT-like transformer, that generates fixed size embeddings. The pooling layer encodes sentence tokens from the input, over which it

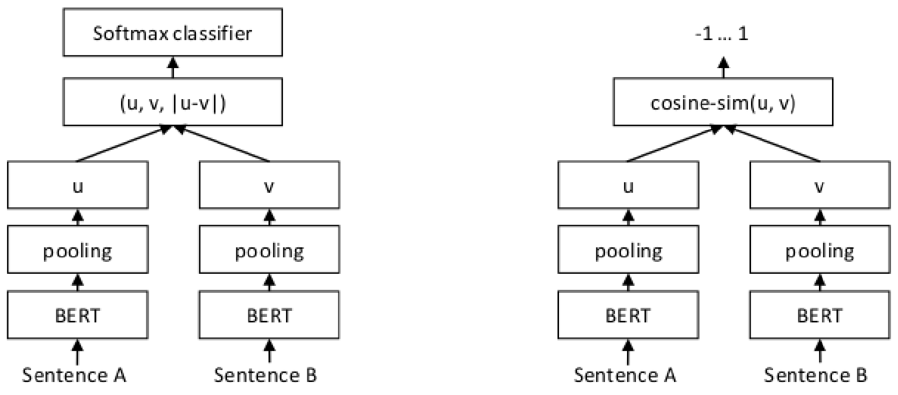


Figure 2.15: SBERT architecture variants. **Left**: classification objective function (softmax) and concatenation head (concatenating embeddings of both sentences and their element-wise difference resulting in $\mathbb{R}^{3n}$, where n is the dimension of the sentence embedding). **Right**: regression objective function (MSE) and cosine similarity as head. [Reimers et al., 2019]

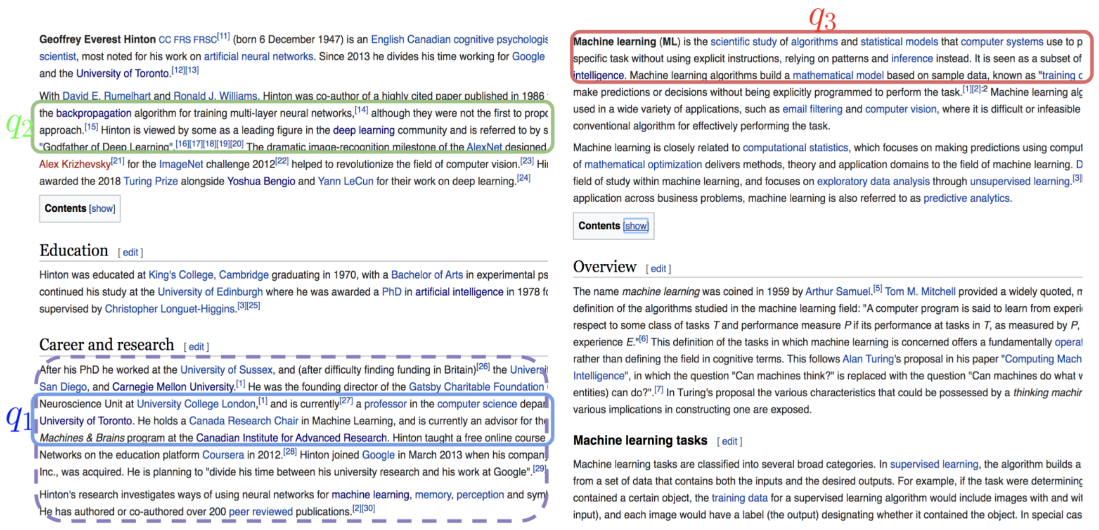
computes *mean* or *maximum* operation resulting in the sentence embedding. The network head and thus the final output together with the loss function depend on the available training data, examples of such architectures are shown in the Figure 2.15.

Besides these examples, you can also use the triplet objective function, where the triples *(query sentence, positive sentence, negative sentence)* are given. The network is tuned by the triplet loss (see equation 2.10) such that the distance between the query and positive sentence is smaller than between the query and negative sentence. This approach provides a certain flexibility as the relevance between the query and a sentence does not necessarily be semantic, but for example thematic. [Ein Dor et al., 2018]

$$\max(||s_q - s_p|| - ||s_q - s_n|| + \epsilon, 0) \tag{2.10}$$

where:

$\|\| \cdot \|\|$	is Euclidean norm
ϵ	is margin.



Geoffrey Everest Hinton CC FRS FRSC[11] (born 6 December 1947) is an English Canadian cognitive psychologis
scientist, most noted for his work on artificial neural networks. Since 2013 he divides his time working for Google
and the University of Toronto.[12][13]

With David E. Rumelhart and Ronald J. Williams, Hinton was co-author of a highly cited paper published in 1986
the backpropagation algorithm for training multi-layer neural networks,[14] although they were not the first to propo
approach.[15] Hinton is viewed by some as a leading figure in the deep learning community and is referred to by s
"Godfather of Deep Learning".[16][17][18][19][20] The dramatic image-recognition milestone of the AlexNet designed
Alex Krizhevsky[21] for the ImageNet challenge 2012[22] helped to revolutionize the field of computer vision.[23] Hi
awarded the 2018 Turing Prize alongside Yoshua Bengio and Yann LeCun for their work on deep learning.[24]

Contents [show]

Education [edit]

Hinton was educated at King's College, Cambridge graduating in 1970, with a Bachelor of Arts in experimental ps
continued his study at the University of Edinburgh where he was awarded a PhD in artificial intelligence in 1978 fo
supervised by Christopher Longuet-Higgins.[3][25]

Career and research [edit]

After his PhD he worked at the University of Sussex, and (after difficulty finding funding in Britain)[26] the Universi
San Diego, and Carnegie Mellon University.[1] He was the founding director of the Gatsby Charitable Foundation
Neuroscience Unit at University College London,[1] and is currently[27] a professor in the computer science depar
University of Toronto. He holds a Canada Research Chair in Machine Learning, and is currently an advisor for the
Machines & Brains program at the Canadian Institute for Advanced Research. Hinton taught a free online course
Networks on the education platform Coursera in 2012.[28] Hinton joined Google in March 2013 when his compan
Inc., was acquired. He is planning to "divide his time between his university research and his work at Google".[29]

Hinton's research investigates ways of using neural networks for machine learning, memory, perception and sym
He has authored or co-authored over 200 peer reviewed publications.[2][30]

Machine learning (**ML**) is the scientific study of algorithms and statistical models that computer systems use to p
specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of
intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training c
make predictions or decisions without being explicitly programmed to perform the task.[1][2]:2 Machine learning alg
used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible
conventional algorithm for effectively performing the task.

Machine learning is closely related to computational statistics, which focuses on making predictions using comput
of mathematical optimization delivers methods, theory and application domains to the field of machine learning. D
field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning.[3]
application across business problems, machine learning is also referred to as predictive analytics.

Contents [show]

Overview [edit]

The name *machine learning* was coined in 1959 by Arthur Samuel.[5] Tom M. Mitchell provided a widely quoted, m
definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experi
respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*,
experience *E*."[6] This definition of the tasks in which machine learning is concerned offers a fundamentally operat
rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Mach
Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what v
entities) can do?".[7] In Turing's proposal the various characteristics that could be possessed by a *thinking machir*
various implications in constructing one are exposed.

Machine learning tasks [edit]

Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a
from a set of data that contains both the inputs and the desired outputs. For example, if the task were determining
contained a certain object, the training data for a supervised learning algorithm would include images with and wi
input), and each image would have a label (the output) designating whether it contained the object. In special cas

Figure 2.16: Example of the ICT, BFS, WLP pre-training tasks. Each randomly chosen sentence is denoted as a query and its corresponding paragraph (positive example) is denoted as *d*. ICT is defined within a paragraph (*q1*); BFS is defined globally within an article (*q2*) and WLP is defined by hyperlinking a two Wikipedia articles through some entity (*q3*). [Chang et al., 2020]

LM's are usually pre-trained on MLM task or some similar task (see sections 2.3.3 and 2.3.2), which helps LM's to get general language comprehension, but this may not necessarily be sufficient for retrieval task. There are some works emphasizing the importance of further LM pre-training prior to application to downstream tasks to improve retrieval capabilities. In [Chang et al., 2020], they propose to further pre-train the LM on these paragraph-level retrieval relevant tasks:

1. **Inverse Cloze Task (ICT)** It was originally proposed in [Lee et al., 2019], where the task is supposed to capture the semantic context of a sentence. ICT randomly extracts a sentence from a passage *p* and then tries to predict from what passage it comes (see Figure 2.17).

2. **Body First Selection (BFS)** It is supposed to capture semantic relationship outside of the local passage. BFS chooses a random sentence from the first summary paragraph of Wikipedia page (since it contains information central to the topic) and tries to classify a passage from the same document.

3. **Wiki Link Prediction (WLP)** This task is proposed to capture inter-page semantic relation. It chooses a random sentence from the first summary paragraph of Wikipedia page and tries to classify a corresponding section from a linked page.

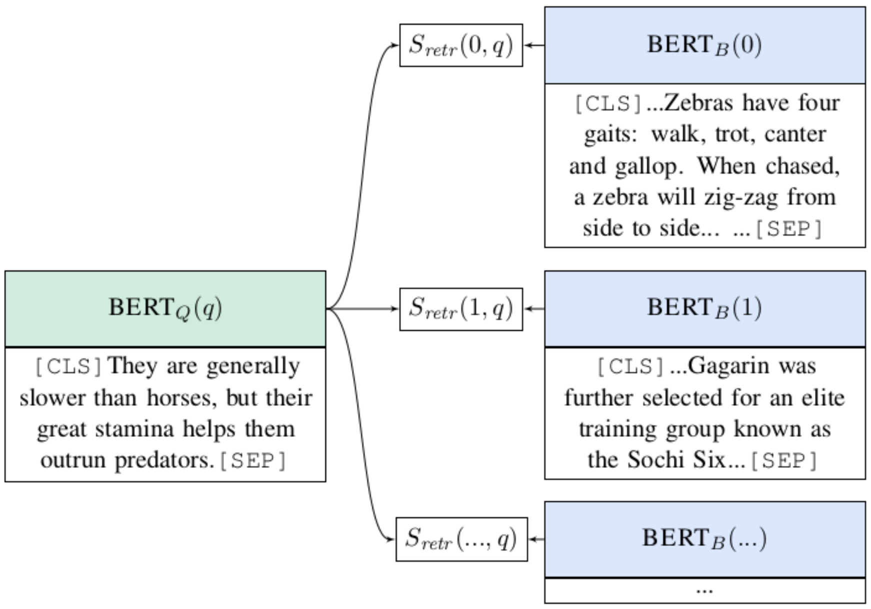

$S_{retr}(0, q)$
BERT$_B(0)$
[CLS] ...Zebras have four gaits: walk, trot, canter and gallop. When chased, a zebra will zig-zag from side to side... ...[SEP]
BERT$_Q(q)$
[CLS] They are generally slower than horses, but their great stamina helps them outrun predators. [SEP]
$S_{retr}(1, q)$
BERT$_B(1)$
[CLS] ...Gagarin was further selected for an elite training group known as the Sochi Six... [SEP]
$S_{retr}(..., q)$
BERT$_B(...)$
...

Figure 2.17: ICT example used for retrieval pre-training. A random sentence (pseudo-query) and its context (pseudo evidence text) are derived from the text snippet: *"...Zebras have four gaits: walk, trot, canter and gallop.* **They are generally slower than horses, but their great stamina helps them outrun predators.** *When chased, a zebra will zig-zag from side to side..." The objective is to select the true context among candidates in the batch.* [Lee et al., 2019]

In this way, the further pre-trained transfomer model serves as method for converting queries and documents into common embeddings space. In the inference phase, relevant documents for a query are retrieved using nearest neighbors search (or its approximation for higher efficiency). In addition to the ability to capture deeper semantic relationships between query and documents, another advantage over traditional sparse-representation based approaches is the ability to optimize models for a given task. [Chang et al., 2020] For completeness, there are also optimized and high-performing applications without involving any further pre-training step. [Ding et al., 2020]

2.5.3 Late Interaction Paradigm

Late interaction paradigm (see Figure 2.14 (d)) was presented in [Khattab et al., 2020] and it tries to combine the benefits of the two previous paradigms. It enables fine-grained interaction of query and documents like the cross-attention setup, while the document representations can be precomputed offline.

It encodes each term of a query using BERT-based encoder into a bag of embeddings E_q. Similarly, it is done for each term of each document, which provides bag of fixed-sized embeddings E_d. Term refers to WordPiece token [Schuster et al., 2012]. Using those two bags of embeddings, the relevance score between q and d is computed as the sum of the maximum similarity between query term embeddings and document term embeddings. Particularly, for each term embedding from E_q is calculated similarity with all term embeddings from E_d, and only the highest (max) similarity is kept. By summing those maximum similarities for each query term, we get the desired relevance score between

q and d.

Instead of the maximum similarity operation, it is possible to use average similarity, or others. However, in the ColBERT paper [Khattab et al., 2020] they strongly argue for the use of the maximum similarity operator due to its pruning-friendly nature, which is leveraged later in the retrieval process.

ColBERT model enables to re-rank already retrieved top-k documents or perform end-to-end top-k retrieval itself. Since the late interaction mechanism is specifically designed to enable end-to-end retrieval from a large collection with the goal to improve recall, it is the expected modus operandi. The retrieval utilizes a fast large-scale vector-similarity search from the FAISS [Johnson et al., 2019] library, which makes it possible to conduct the search between the query embedding and all document embeddings across the full collection efficiently.

When processing queries, the retrieval procedure consists of two parts to retrieve top-k most relevant documents from the collection. First, for each query term embedding from E_q is retrieved top-k' matches for that vector over all document embeddings. Each of those matched document term embeddings is then mapped to its document origin, which results in $N_q \times k'$ document IDs (N_q = number of tokens in the query), while only K $\leq N_q \times k'$ of which are unique. These K documents probably contain one or more very similar embeddings to some query term embeddings. Efficiency of this step is ensured by the IVPF index from the FAISS library, which divides the indexing space into P partitions based on k-means clustering algorithm, so the document embeddings are mapped to their nearest clusters subsequently. This makes it possible to avoid a direct exhaustive search over all documents in the collection, by scoring only the documents from the nearest C clusters.

In the second step, only a relatively small number of documents K retrieved in the first step are exhaustively scored. Each document is represented by a matrix of embeddings (#tokens $\times$ embedding dimension), which results in 3-dimensional tensor. The score of each document for a query q is then calculated using the max-sim operator and summation over all query terms, a more detailed description can be found in the [Khattab et al., 2020]. At the end, the documents are ranked according to their score. The obvious advantage over the cross-attention approach is that it is not necessary to calculate a relatively expensive attention for query q and each document d ($\mathrm{N_q}$ + $\mathrm{N_d}$ tokens), but only for the query q ($\mathrm{N_q}$ tokens).

2.5.4 FAISS

Since both neural approaches (see 2.5.2 and 2.5.3), we experimented with, use methods from the FAISS library [Johnson et al., 2019], we will give a few words about it. FAISS is a library for efficient similarity search and clustering of dense vectors, developed by Facebook AI Research. It provides algorithms for search in sets of vectors of any size. It is highly optimized, which makes it fast and allows working with vectors that do not fit in RAM. It also contains supporting code for evaluation and parameter tuning. The library is written in C++ with optional GPU support provided via CUDA. It also provides complete wrappers for Python/numpy.

FAISS contains several methods for similarity search and very efficient implementation of k-means clustering, PCA and product quantization. It works with instances represented as vectors, that can be compared with L2 (Euclidean) distance, dot product or cosine similarity[7]. The library offers several indexing structures ranging from direct ones

[7]which is a dot product on normalized vectors

that provide exact search to ones that involve some trade-off between search time, search quality, memory used per index, training time or need for external data for unsupervised training.

To give you an idea, some of the potential adjustments to the index are as follows. Faster search is possible by segmenting the database into Voronoi cells. At the search time, only the database vectors contained in the cell the query falls in and a few neighboring ones are compared against the query vector. Lowering memory footprint of an index is enabled by PCA or product quantization that will reduce the dimension to a configurable number. This will make possible to scale up even to very large datasets (billions of vectors).

2.5.5 Long, Longer, Longest

Along with the significant increase in the use of BERT-like transformers, its limits also became more apparent. One of them is a limited number of tokens at the input of the model as BERT operates with a maximum sequence length equal to 512 tokens. This is caused by an expensive self-attention mechanism. While powerful in performance, self-attention memory and computational requirements grow quadratically with sequence length, making it very costly or even infeasible to process long sequences using current hardware. Over the last two years, a number of works have appeared that address these limits and modify the attention mechanism. As this area of research opens up the possibility of working with long texts at the level of entire documents, we present a summary of it, as it can also have an effect on DR.

The Longformer paper [Beltagy et al., 2020] suggests to replace expensive self-attention with a combination of cheaper attention patterns (see Figure 2.18). Specifically it uses local sliding window attention, dilated attention and global attention, which makes the Longformer to scale linearly with the input sequence length. They also propose a clever initialization scheme, that copies RoBERTa weights and absolute position embeddings into a Longformer version of RoBERTa with 8 times higher capacity supporting sequences of up to 4096 tokens in length. Using that simple yet effective idea makes the model pretraining to very quickly converge with a small number of gradient updates.

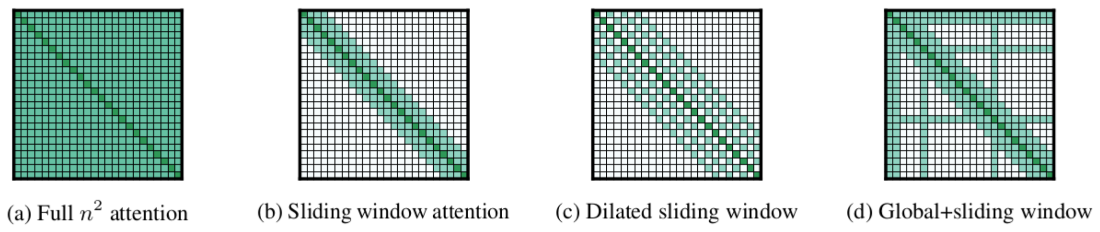

(a) Full n^2 attention
(b) Sliding window attention
(c) Dilated sliding window
(d) Global+sliding window

Figure 2.18: Comparing the full self-attention pattern and the configuration of attention patterns used in Longformer. [Beltagy et al., 2020]

Another group of methods seeks to make the attention mechanism more efficient by approximating the full attention matrix with a lower rank matrix with size independent of the input length. The argument for this approach is based on the key observation that the self-attention is low rank. [Wang et al., 2020] In other words, attention weights are dominated by a mere minority of key entries rather than being diffuse over the whole sequence. By performing spectral analysis, they showed that 90% of the variation is explained by only the first 128 of 512 eigenvalues obtained by SVD.

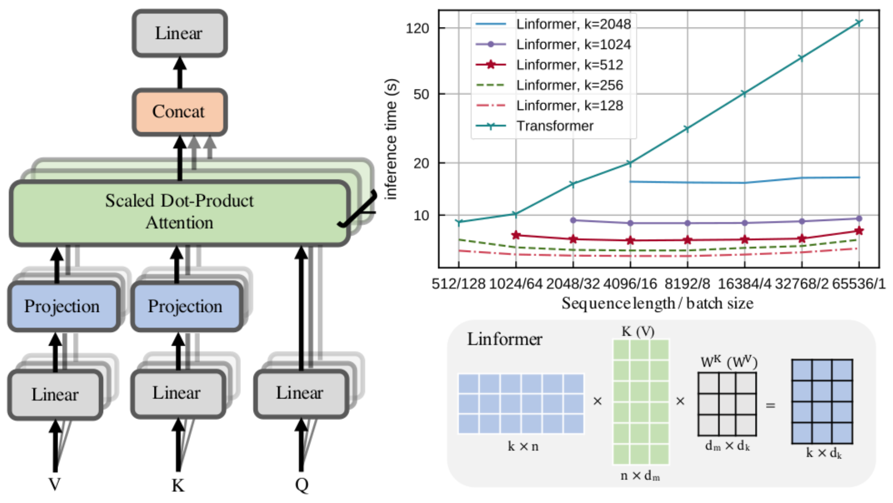


Figure 2.19: Left and bottom-right show architecture and example of the proposed multi-head linear self-attention. Top right shows inference time vs. sequence length for various Linformer models. [Wang et al., 2020]

Linformer model [Wang et al., 2020] adds two linear projection matrices to the original self-attention (see the left part of the Figure 2.19). First, they project the original $(n \times d)$-dimensional key (KW_W) and value (VW_V) layers into $(k \times d)$-dimensional projected key and value layers. Then those two matrices are multiplied together to produce a $(n \times k)$-dimensional agreement matrix. Finally, multiplication of this agreement matrix (n, k) with the down-projected value matrix (k, d) will results in $(n \times d)$-dimensional just like in the original self-attention. By choosing a very small projected dimension k, such that $k \ll n$, makes the memory and space requirements significantly reduced. Computationally-wise, both computational and space complexity is linear with the respect to the sequence length n (see the right side of the Figure 2.19).

In addition to the above mentioned methods, there are a number of others using more or less different approaches, for example in the *Routing transformer* [Roy et al., 2021] they decided to tackle this problem by k-means clustering. A nice overview capturing the development in this “making transformers more efficient” research direction and summarizing majority of the relevant work is provided in this survey [Madison, 2020].

CHAPTER 3

Datasets

This chapter describes the datasets used in the experiments and work related to the measuring of the quality of constructed claims.

3.1 FEVER CS

The original FEVER dataset is presented in [Thorne et al., 2018b] containing $\approx$ 185,000 claims based on $\approx$ 50,000 popular Wikipedia articles. Each claim is annotated as either *Supports, Refutes* or *Not Enough Info.* In case the claim is verifiable — annotated as *Supports* or *Refutes* — there is also provided evidence on which is the annotation based. Evidence consists of single or multiple documents and even particular sentences which contain evidence.

dataset	**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 3.1: Label distribution in FEVER CS dataset (with forced label uniformity in the validation sets to remove advantage for heavily biased predictors)

The dataset was created in two stages. Firstly, the claims were generated using only the first paragraph (abstract[8]) of a randomly sampled Wikipedia article. The annotators were asked to create claims about some of the article's entities. In order to create more complex claims, the annotators had the option to use hyperlinked articles to include information from them. In the second stage, the annotators were asked to label the claim using one of the three mentioned labels. In case they choose either *Supports* or *Refutes* label they need to select the evidence paragraph for their decision.

[8]this paragraph contains general information relevant to the whole article

ID: 24173
Verifiable: VERIFIABLE
Claim: Mlčení jehňátek je náboženství.
Evidence: Mlčení jehňátek (v originále The Silence of the Lambs) je americký thriller, který režíroval Jonathan Demme. Hlavními herci filmu jsou Jodie Fosterová, Anthony Hopkins, Scott Glenn, Anthony Heald, Ted Levine a Frankie Faison. Film měl premiéru ve Spojených státech 14. února 1991. Scénář je napsán podle stejnojmenného románu Thomase Harrise. Film získal celkově pět Oscarů - nejlepší film, nejlepší režie, nejlepší herec v hlavní roli, nejlepší herečka v hlavní roli a nejlepší adaptovaný scénář.
Verdict: Refuted

Figure 3.1: FEVER CS data example

Original English claims were translated into Czech using Google Cloud Translate API. Since Wikipedia has Czech mutation, we used Czech Wikipedia dump[9], processed it using WikiExtractor library [Attardi, 2019] and kept only the abstract paragraphs. After this processing, the article database had about 453,500 articles. We used the training/development split available on FEVER website[10]. A more detailed description of the creation of the FEVER CS dataset is given here. [Ullrich, 2021]

3.2 ČTK

Inspired by [Thorne et al., 2018b] and [Binau et al., 2020], we started creating a Czech version of Fact-Extraction and Verification dataset[11]. We used vast database of articles provided by the Czech News Agency[12] instead of the Wikipedia as the knowledge database. The Czech News Agency produces and also provides news articles, which are taken over by the media, public institutions and private companies. Because they are news texts, these articles have a different structure compared to Wikipedia articles as they use different language style, provide a much broader context and a different order of paragraphs - the first paragraph does not contain a summary as an article on Wikipedia that contains an abstract, which was used in the FEVER.

ID: 142
Verifiable: VERIFIABLE
Claim: Astrid Lindgrenová neměla žádné děti.
Evidence: Švédská spisovatelka Astrid Lindgrenová (1907-2002) už jako šestnáctiletá začala pracovat jako elévka v redakci regionálních novin. Záhy se však stala svobodnou matkou a nalezla si ve Stockholmu místo sekretářky. Protože měla málo peněz, svěřila syna Larse do péče pěstounů v Dánsku. Roku 1931 se provdala za úředníka Sturea Lindgrena, od té doby se mohla věnovat výchově syna a později i dcery Karin.
Verdict: Refuted

Figure 3.2: ČTK data example

[9]available from `https://dumps.wikimedia.org/`
[10]`https://fever.ai/resources.html`
[11]data collection platform available at `https://fcheck.fel.cvut.cz`
[12]Česká Tisková Kancelář (ČTK)

The collection process was not fundamentally different from the collection in the FEVER dataset. Because ČTK articles were not linked by hyper-references of name entities such as Wikipedia, a “dictionary” was provided to annotators enabling them to use evidence from there as well. This “dictionary” was consisted of the most relevant articles found using the TF-IDF and pretrained two-tower dense retriever model described in section 4.2.2. In cooperation with journalists from the Faculty of Social Sciences of Charles University, we generated $\approx 3\,000$ at least once annotated claims. Hence, the following analysis and results are based on the current snapshot from 05.05.2021. A more detailed description of data and data collection is given in [Ullrich, 2021].

dataset	**supported**	**refuted**	**NEI**
train	1,282	556	555
dev	100	100	100
test	200	200	200

Table 3.2: Label distribution in ČTK dataset (with forced label uniformity in the validation sets to remove advantage for heavily biased predictors)

Articles containing sports results and financial market results, which consisted mainly of tables of numbers, were first filtered from the ČTK article database. This reduced the number of articles to about 2,507,500. We worked with the texts at the paragraph level, so the articles had to be divided into paragraphs then. This step resulted in about 13,619,500 different paragraphs.

3.3 Data Quality

In the claim generation phase of both FEVER and ČTK datasets, the annotators are asked to create variations of an initial claim by rephrasing, substituting part of the claim, negating or making it more general/specific. These mutations may have a different truth label than the original claim or even be non-verifiable with the given knowledge database, which will produce more claims of all annotation labels. During trials in [Thorne et al., 2018b], they found that a majority of annotators had difficulty with creating non-trivial negation mutations beyond adding “not” to the original.

In the following work [Derczynski et al., 2020], they investigated the impact of these trivial negations on the quality of the dataset. To examine the claims in the context of quality of the dataset, they proposed two quality metrics: *dataset-weighted cue information (DCI)* and *cue productivity and coverage*. These metrics should help to reveal potential surface-level linguistic patterns that “leak” class information and cause bias in the data.

Dataset-weighted cue information (DCI) is a simple measure based on information theory, which can indicate how much a pattern contributes to a classification. By calculating entropy (equation 3.1)

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{3.1}$$

of class-balanced distribution N_k of cue frequencies for cue k (equation 3.2)

$$N_k = \{\frac{|A_{cue=k} \cap A_{class=y}|}{|A_{cue=k}|}\ y \in Y\} \tag{3.2}$$

where:

Y	denotes the set of possible labels $\{$*supports, refutes, not enough info*$\}$
A	is the set of all claims
$A_{cue=k}$	is the set of claims containing cue k
$A_{class=y}$	is the set of claims annotated with label y

We get an information-based factor λ_h expressing the information gain as

$$\lambda_h = 1 - H(N) \tag{3.3}$$

This is further corrected for rareness of cues by involving frequency-based scaling factor λ_f, which plays a similar role to the IDF-term in the TF-IDF model.

$$\lambda_f = (\frac{|A_{cue=k}|}{|A|})^{1/s} \tag{3.4}$$

Where s is a scaling factor corresponding to the estimated exponent of the feature's power law frequency distribution. In [Derczynski et al., 2020] they suggest using $s = 3$ for English and we did the same for Czech.
Finally, multiplying those two factors and taking their squared root will result in the DCI metric

$$DCI = \sqrt{\lambda_h \cdot \lambda_f} \tag{3.5}$$

They propose to use skip-grams as cues (patterns) thanks to the fact that they capture a sufficient amount of information and also ignore usually rare named entities and rather focus on the surrounding language. The skip-grams are generalization of n-grams, in which the words not need to be consecutive, but may leave a gaps which are skipped over. For example in a short sentence *Cash walks the line*:

bigrams: Cash walks, walks the, the line;

1-skip-bigrams: Cash walks, Cash the, walks the, walks line, the line;

2-skip-bigrams: Cash walks, Cash the, Cash line, walks the, walks line, the line.

Cue productivity and coverage metrics proposed in [Niven et al., 2019] are used with the slightly modified methodology as the structure of both datasets differ. Potential cues (patterns) are extracted from the data in the form of unigrams and bigrams. The *productivity* of a cue (π_k) is calculated as the frequency of the most common label across the claims that contain the cue divided by the total number of claims which contain the cue irrespective of their label. Based on the definition of productivity, it can range $[\frac{1}{|Y|}, 1]$.

$$\pi_k = \frac{\max_{y \in Y} |A_{cue=k} \cap A_{class=y}|}{|A_{cue=k}|} \tag{3.6}$$

There is also a proposed metric suitable for comparison between datasets, called utility that normalizes productivity by a number of distinct labels, which may differ across datasets

$$\rho_k = \pi_k - \frac{1}{|Y|} \tag{3.7}$$

The *coverage* of a cue is defined as

$$\xi_k = \frac{|A_{cue=k}|}{|A|} \tag{3.8}$$

It should be emphasized that the productivity metric assumes a balanced dataset with respect to labels. Otherwise, preference would be given to the most frequent label. In [Derczynski et al., 2020], they propose the creation of a balanced data set by subsampling the majority class, which they achieve by creating ten random subsamples. Then the resulting metrics are obtained by averaging these subsample metrics.

We computed those metrics according to the above described methodology for both FEVER CS and ČTK datasets. In addition to unigram and bigram cues for the productivity and coverage metrics, we also tried to use lower-granular worpiece tokens as cues. Regarding the DCI, we used wordpieces, unigrams and 4-skip-bigrams as cues. Then we also calculated the harmonic mean of productivity and coverage, which allows us to reflect the overall effect of the cue on the dataset, because there exist a large number of cues with maximal possible productivity but minimal effect (for example, a given cue occurs in the dataset in only one claim, which is consistently labeled will result in $\pi_k = 1$).

The results confirmed that the original FEVER dataset does indeed contain some cues that may indicate bias. This was also reflected in the translated FEVER CS, where the words "není" and "pouze" showed high productivity of 0.57 and 0.55 and ended in the first 20 cues sorted by *harmonic mean*[13]. However, their impact on the quality of the entire dataset is limited as their coverage is not high, which is illustrated by their absence in the top-5 most influential cues (see table 3.3).

According to the *harmonic mean*, when using wordpiece tokens, the most influential are "##'", which is accent at the end of the word token, and "UNK", which is a special token that includes any token not found in the dictionary (see table 3.4). Despite the fact that they provide very little information to the model, they hold a dominant position in the results due to their high occurrence.

The results on the ČTK dataset are significantly affected by the fact that the number of claims is quite low. This causes that even specific cues based on the thematic cluster formed around the original statement have a relatively higher impact on the dataset (for example, "Bühler Motor" in the table 3.5). Although ČTK also contains some constructs with a higher productivity, for example "Thomas Alva" (0.69) or "není" (0.7), their influence on the whole dataset is minimal to negligible.

Although the analysis did not confirm any significant bias in the claims, there is still a need to monitor these metrics in the future as more claims are made. This part is not directly related to document retrieval, but it is very useful for the last step of the fact-verification pipeline (see Figure 1.2). In this step, a claim is classified either as true, false, or unverifiable in the context of the evidence provided. And awareness of the linguistic patterns that can leak information about the label can help explain the behavior of the algorithm and better evaluate it. This problem is modeled using a task called Natural Language Inference (NLI) and is further addressed in the work of [Ullrich, 2021].

[13] *harmonic mean* refers to harmonic mean of productivity and coverage

Cue	Productivity	Utility	Coverage	Harmonic Mean
		Wordpieces		
##’	0.3378	0.0045	0.6659	0.4482
UNK	0.3389	0.0056	0.6321	0.4412
je	0.3455	0.0121	0.2682	0.3020
v	0.3495	0.0162	0.2650	0.3015
##a	0.3485	0.0152	0.1219	0.1806
		Unigrams		
je	0.3469	0.0136	0.2653	0.3007
v	0.3462	0.0128	0.2115	0.2625
byl	0.3731	0.0397	0.1131	0.1736
se	0.3690	0.0356	0.0986	0.1556
na	0.3604	0.0270	0.0832	0.1351
		Bigrams		
v roce	0.4591	0.1258	0.0564	0.1004
se narodil	0.4496	0.1162	0.0193	0.0370
ve filmu	0.5163	0.1830	0.0115	0.0224
narodil v	0.4748	0.1415	0.0104	0.0203
je v	0.3443	0.0110	0.0103	0.0200

Table 3.3: Productivity, utility, coverage and harmonic mean of productivity and coverage calculated on translated FEVER CS dataset claims sorted by the harmonic mean.

Cue	DCI
Wordpieces	
##’	0.6914
UNK	0.6846
je	0.5975
v	0.5913
byl	0.5281
Unigrams	
je	0.5967
v	0.5686
byl	0.5281
se	0.4991
na	0.4912
4-skip-bigrams	
v roce	0.4455
se v	0.4172
byl v	0.4100
je v	0.3980
se narodil	0.3762

Table 3.4: DCI calculated on translated FEVER CS dataset claims.

Cue	DCI
Wordpieces	
v	0.6270
z	0.5668
##y	0.5376
##u	0.5361
na	0.5307
Unigrams	
v	0.5936
se	0.5282
na	0.5243
a	0.5014
je	0.5012
4-skip-bigrams	
v roce	0.4420
se v	0.4181
v v	0.3972
na v	0.3945
Bühler Motor	0.3860

Table 3.5: DCI calculated on ČTK dataset claims.

Cue	Productivity	Utility	Coverage	Harmonic Mean
		Wordpieces		
v	0.3505	0.0171	0.3822	0.3656
z	0.3456	0.0123	0.2009	0.2541
##y	0.3508	0.0175	0.1487	0.2088
##u	0.3552	0.0218	0.1466	0.2075
se	0.3521	0.0188	0.1414	0.2017
		Unigrams		
v	0.3489	0.0156	0.2725	0.3060
se	0.3504	0.0171	0.1401	0.2002
na	0.3485	0.0151	0.1248	0.1838
je	0.3545	0.0212	0.0999	0.1558
V	0.3924	0.0590	0.0968	0.1552
		Bigrams		
v roce	0.4013	0.0679	0.0398	0.0723
V roce	0.3649	0.0316	0.0166	0.0317
se v	0.3755	0.0422	0.0150	0.0288
v Praze	0.4300	0.0966	0.0139	0.0270
více než	0.3804	0.0471	0.0122	0.0236

Table 3.6: Productivity, utility, coverage and harmonic mean of productivity and coverage calculated on ČTK dataset claims sorted by the harmonic mean.

Chapter 4

Proposed Solutions

This chapter describes proposed solutions, motivation for their choice and methods of their evaluation.

4.1 Baseline

We originally chose DrQA [Chen et al., 2017] model, more precisely its document retriever part, as our document retrieval baseline. DrQA was designed for "machine reading at scale" — combination of large-scale open-domain question answering and machine comprehension of text. The system was originally used for answering factoid questions while using Wikipedia as the knowledge database, which is relatively close to the task being solved in fact-checking. The DR part itself is based on TF-IDF weighting of BOW vectors while optimized by using hashing.

Later, inspired by the criticism of choosing weak baselines presented in [Yang et al., 2019a], we decided to validate our TF-IDF baseline against the proposed Anserini toolkit. More specifically, we used Python toolkit — Pyserini [Lin et al., 2021] — to do so. Anserini itself is implemented in Java and built on the indexing and search features providing library — Lucene, which in combination with a significant optimization makes it very effective. We used BM25 model from Anserini library and compared the results with DrQA using the FEVER CS dataset (see section 3.1) and ČTK (see section 3.2). Compared to DrQA, anserini's BM25 performance was slightly worse on the FEVER CS test set but higher on the ČTK test set (see the results in the section 5.4).

4.2 Neural Models

The aim of the work was to investigate neural models in the initial phase of DR and whether they can outperform very solid traditional baselines, as some recent work suggests. Aware of future use in the fact-checking pipeline, we were primarily interested in recall and mean reciprocal rank (MRR) metrics (see section 4.3), which capture the ability of models to identify relevant documents, which is central in the initial phase of DR. So our second goal was to come up with a model that would maximize performance on these two metrics. We did not pay much attention to hybrid neural models precisely because they incorporate traditional models to retrieve documents in the initial phase.

4.2.1 ColBERT

We replicated this recently published model that should provide the benefits of both cross-attention and two-tower paradigm under one roof (see 2.5.3) using the implementation provided by the authors[14]. We made only minimal changes such as changing the backbone model to multilingual mBERT and adjusting the special tokens.

The model was trained using triples *query; positive paragraph; negative paragraph* with the objective to correctly classify paragraphs using a cross-entropy loss function. We constructed the training triples such that the claim created by a human annotator was taken as a query, a paragraph containing evidence as a positive and a random paragraph from a randomly selected document was used as a negative. This was done for both ČTK and FEVER CS datasets.

For the ČTK dataset, the number of created claims is significantly lower than for FEVER CS. Therefore, we further created more ČTK training triples with a similar process only instead of sampling the negative paragraph from a random document, we sampled it from the same document containing the positive paragraph (evidence). These negative paragraphs are called hard negatives. The number of training triples was still quite low, so we generated also synthetic triples as follows. We extracted a random sentence from a random paragraph, which we used as a query. The remaining paragraph after extraction was used as a positive paragraph. The negative paragraph was selected as a random paragraph from a random document. As a result, we generated about 600,000 triples ($\approx 500,000$ synthetic and $\approx 100,000$ using human-created claims) for the ČTK dataset.

4.2.2 Pre-training mBERT

A significant portion of time and work went to pre-training multilingual mBERT model in two-tower paradigm. Motivated by findings in [Chang et al., 2020], we tried to apply this approach to large-scale DR, which is closer to a realistic situation than a relatively small SQuAD dataset used in the above mentioned work.

In principle, we used the same setup as in [Chang et al., 2020], the multilingual mBERT [Devlin et al., 2018] with added FC linear layer which consolidated the output into embedding of the required dimension 512. This model was pre-trained unsupervised on ICT and BFS tasks. In the case of the FEVER CS dataset, we pre-trained the model on full-length Wikipedia articles. In the case of the ČTK dataset, the model was pre-trained on the entire collection of documents (articles) provided by the Czech News Agency. This was followed by a supervised finetuning phase, where real claim was used as a query, passage containing evidence for the given claim as positive passage.

ICT pre-training examples were specifically formed by dividing the article into passages with a maximum length of 288 tokens (hyper-parameter taken from [Lee et al., 2019]). Increasing to 512 tokens (the maximum capacity of the BERT model) did not bring any noticeable improvement. From each passage a sentence was randomly selected that was consequently extracted in 90% of cases and in the rest of the cases remained in the passage. The chosen sentence was considered a query and a passage from which it came as positive (relevant). Pre-training examples for BFS were created similarly, with the only difference that the positive passage was not the one containing the query but a randomly selected passage from the same document (article).

In the finetuning phase, we used the constructed claims and their evidence as relevant (positive) passages. There exists claims, although relatively small amount, that were created by combining several passages from different articles (multi-hop claims). Without

[14]available from `https://github.com/stanford-futuredata/ColBERT`

prejudice to the generality of the solution, we split the combined evidence, so query is always in a relation with only a single evidence passage. This way we slightly increase the amount of data by cloning the query for each part of its evidence.

Then we used this finetuned model to generate paragraph embeddings of the whole collection, so they are prepared for retrieval. In the retrieval phase, we used the FAISS library [Johnson et al., 2019] and constructed *PCA384 Flat* index for ČTK data and *Flat* index for FEVER CS data. In the case of the *PCA384 Flat* index, the original output of the pre-trained model with dimension 512 is transformed into a 384-dimensional vector using PCA. This was done to lower the memory footprint. *Flat* index uses the full 512-dimensional vector. When given a query, the FAISS index then retrieves top-k most relevant documents, implemented as k-means clustering.

4.3 Evaluation

We used standard precision, recall, F1 score and mean reciprocal rank (MRR) metrics to evaluate all models. MRR corresponds to the harmonic mean of the positions of first relevant document for each query (see equation 4.4). So it partially reflects also the ability of a model to provide relevant documents in the front position which was in combination with recall our key performance indicator.

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

$$\text{recall} = \frac{|(\text{relevant documents}) \cap (\text{retrieved documents})|}{|(\text{relevant documents})|} \tag{4.2}$$

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

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

where:

Q stands for a sample of queries

$rank_i$ refers to the rank position of the first relevant document retrieved for the i-ith query

In retrieval tasks, the output is typically a set of most relevant documents or list of documents sorted by a relevance score. The metrics are dependent on the number of returned documents and therefore it is appropriate to monitor their development depending on the number of retrieved documents k. In that case, the metrics are marked with the suffix @k informing about the number of returned documents as well.

Chapter 5

Experiments

This chapter summarizes the setup of performed experiments and their results.

5.1 Baselines

DrQA: We calculated the TF-IDF index using DrQA implementation for all unigrams and bigrams with 2^{24} buckets for hashing.
BM25: We computed the index and then finetuned the *k1* and *b* hyper-parameters using grid search on defined grid *k1* $\in [0.6, 1.2]$, *b* $\in [0.5, 0.9]$ both with step 0.1. On a sample of 10,000 training claims, we selected the best performing parameter values:

dataset	*k1*	*b*
FEVER CS	0.9	0.9
ČTK	0.6	0.5

5.2 ColBERT

We tried two setups here, 64-dimensional term representation with with document trimming to a maximum of 180 tokens and richer 128-dimensional term representation with document trimming to a maximum of 220 tokens on FEVER CS. For the ČTK dataset we counted only the larger version, as it proved to be more powerful. The query is always truncated to a maximum of 32 tokens. Training was done on triples (see section 4.2.1) using two NVIDIA Tesla V100 GPU's with batch size 64, learning rate 3e-6, masked punctuation tokens, mixed precision and L2 similarity metric.

5.3 Pretraining mBERT

We used the cased multilingual mBERT from the HuggingFace library [Wolf et al., 2020] as the backbone model.
FEVER CS: The model was pretrained on ICT and BFS tasks on the Wikipedia articles for 20 epochs. The training was done on 4 GPUS's NVIDIA Tesla V100 with batch size 128, Adam optimizer with 1e-5 learning rate and no weight decay. In the finetuning phase was used the model from the pretraining phase that showed best performance on FEVER CS development set. Then it was finetuned for next 20 epochs with the same

setup except for learning rate, which was reduced to 5e-06 similarly to [Chang et al., 2020]. As a result, we saved the model with the highest performance on the development set.
ČTK: We utilized the model already pretrained for 20 epochs on the Wikipedia (see above) and further pretrain it on the ČTK collection for 10 epochs with the same setup as stated above. In the finetuning phase, the model was finetuned using real claims and their annotated evidences for 20 epochs with 5e-06 learning rate and the rest of the setup remain unchanged.

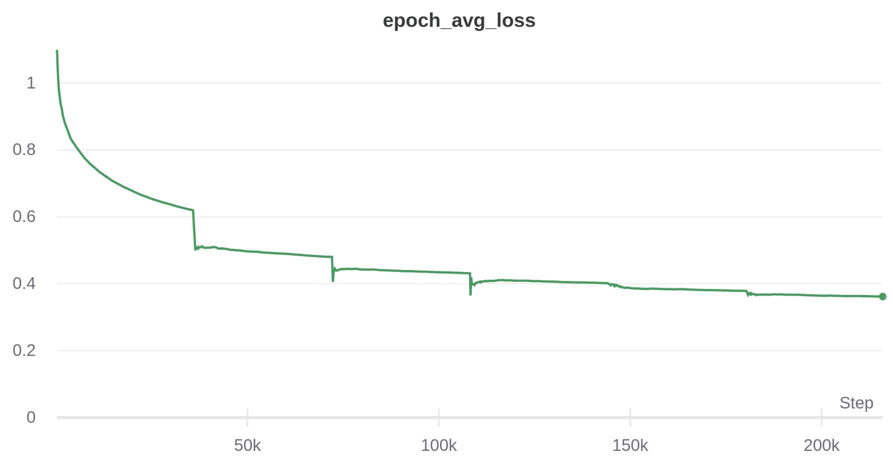


Figure 5.1: Pretraining on ČTK learning curve

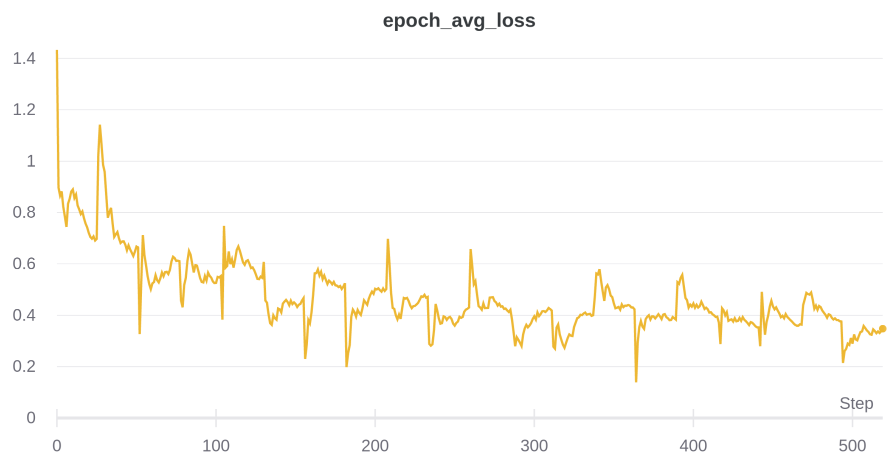


Figure 5.2: Finetuning on ČTK learning curve

5.4 Results

model	P@1	P@5	P@10	P@20
DRQA	42.42	13.66	7.56	4.08
Anserini BM25 finetuned	41.24	13.12	7.37	4.02
mBERT BFS+ICT	**65.87**	**19.13**	**10.16**	**5.28**
ColBERT 128 (FEVER CS)	56.33	15.45	8.20	4.29
ColBERT 128 (ČTK + FEVER CS)	45.93	13.94	7.69	4.15
ColBERT 64 (ČTK + FEVER CS)	44.21	13.20	7.28	3.93

Table 5.1: Precision at k on FEVER CS test set.

model	R@1	R@5	R@10	R@20
DRQA	39.14	62.39	68.89	73.99
Anserini BM25 finetuned	38.12	60.41	67.43	73.18
mBERT BFS+ICT	**61.27**	**87.43**	**91.75**	**94.48**
ColBERT 128 (FEVER CS)	52.39	71.48	75.38	78.40
ColBERT 128 (ČTK + FEVER CS)	42.57	64.31	70.42	75.46
ColBERT 64 (ČTK + FEVER CS)	41.24	60.98	66.88	72.01

Table 5.2: Recall at k on FEVER CS test set.

model	F1@1	F1@5	F1@10	F1@20
DRQA	40.72	22.42	13.63	7.73
Anserini BM25 finetuned	39.62	21.55	13.29	7.63
mBERT BFS+ICT	**63.49**	**31.40**	**18.29**	**10.01**
ColBERT 128 (FEVER CS)	54.29	25.41	14.80	8.14
ColBERT 128 (ČTK + FEVER CS)	44.19	22.91	13.86	7.86
ColBERT 64 (ČTK + FEVER CS)	42.67	21.71	13.13	7.46

Table 5.3: F1 at k on FEVER CS test set.

model	MRR@1	MRR@5	MRR@10	MRR@20
DRQA	40.72	49.94	50.81	51.14
Anserini BM25 finetuned	39.39	48.09	49.02	49.44
mBERT BFS+ICT	**57.57**	**69.32**	**70.05**	**70.26**
ColBERT 128 (FEVER CS)	55.10	62.67	63.17	63.36
ColBERT 128 (ČTK + FEVER CS)	44.48	53.18	53.92	54.28
ColBERT 64 (ČTK + FEVER CS)	43.63	51.92	52.68	52.99

Table 5.4: Mean reciprocal rank at k on FEVER CS test set.

model	P@1	P@5	P@10	P@20
DRQA	12.50	5.05	3.07	1.76
Anserini BM25 finetuned	15.50	5.80	3.35	1.97
ColBERT 128 (FEVER CS + ČTK)	**19.25**	**7.00**	**3.97**	**2.29**
mBERT BFS+ICT (FEVER CS + ČTK)	0.75	0.95	0.80	0.60

Table 5.5: Precision at k on ČTK test set.

model	R@1	R@5	R@10	R@20
DRQA	12.75	25.50	31.00	35.50
Anserini BM25 finetuned	15.75	29.25	33.75	39.75
ColBERT 128 (FEVER CS + ČTK)	**19.50**	**35.25**	**40.00**	**46.00**
mBERT BFS+ICT (FEVER CS + ČTK)	1.00	5.00	8.25	12.25

Table 5.6: Recall at k on ČTK test set.

model	F1@1	F1@5	F1@10	F1@20
DRQA	12.62	8.43	5.60	3.36
Anserini BM25 finetuned	15.62	9.68	6.10	3.76
ColBERT 128 (FEVER CS + ČTK)	**19.37**	**11.68**	**7.23**	**4.36**
mBERT BFS+ICT (FEVER CS + ČTK)	0.86	1.60	1.46	1.14

Table 5.7: F1 at k on ČTK test set.

model	MRR@1	MRR@5	MRR@10	MRR@20
DRQA	13.05	17.72	18.31	18.60
Anserini BM25 finetuned	15.79	20.40	20.95	21.36
ColBERT 128 (FEVER CS + ČTK)	**18.32**	**24.21**	**24.85**	**25.26**
mBERT BFS+ICT (FEVER CS + ČTK)	1.47	2.78	3.12	3.45

Table 5.8: Mean reciprocal rank at k on ČTK test set.

Conclusion

Over the course of several months, we studied a wide range of articles and literature, conducted dozens of experiments with several models, and contributed to the production of the initial version of Czech Fact Extraction and Verification dataset using Czech News Agency articles (ČTK). As the datasets have undergone gradual development and modifications due to successive data collection and refinement, it is not possible to subject all experiments to a uniform comparison table. Therefore, we only present "condensed" results on the latest versions of both datasets (see chapter 3).

In this thesis, we have investigated large-scale document retrieval methods in the fact-checking domain. We primarily focused on end-to-end neural models and their comparison with traditional TF-IDF and BM25 models. Traditional models proved to be very strong and robust baselines that are quite hard to outperform by neural methods, especially in the case of limited data. Nevertheless, both of our proposed solutions can outperform traditional methods.

The solution based on multilingual BERT pre-trained on BFS and ICT tasks significantly outperformed traditional baselines on FEVER CS dataset by more than 20% Recall@20 and 19% MRR@20. The results on the ČTK dataset are considerably weaker. We need to examine them more thoroughly, but we already believe that this is due to several factors. First, the lack of training data (ČTK 2K vs. FEVER CS 100K), of which this approach requires a substantial amount. Secondly, the several times larger articles database (ČTK 13.5M vs. FEVER CS 0.5M) that makes retrieval on the ČTK more challenging. And thirdly, news texts often contain paragraphs with little or no variation in content[15], but because these paragraphs are in different articles, they have different identifiers and are not captured by the evaluation, even though they might be relevant.

ColBERT outperformed traditional baselines by more than 4% Recal@20 and 12% MRR@20 on FEVER CS and by 6% Recall@20 and almost 4% MRR@20 on ČTK. ColBERT pre-trained on the ČTK + FEVER CS triples (see section 4.2.1) decreased the performance compared to ColBERT pretrained only on the FEVER CS triples. This is probably due to the introduction of additional noise into the model. The memory footprint of the ColBERT index can easily be reduced without much performance penalty. We halved the dimension of token representation, which decreased the performance by 3% Recall@20 and only by 1% MRR@20, which confirms the findings of the ColBERT authors [Khattab et al., 2020].

This thesis also focuses marginally on data collection, more precisely on the analysis of generated claims. The claims are analyzed in terms of potential bias in the form of

[15] often paragraphs that provide a broader context for the report

surface-level linguistic structures that may have a negative impact at later stages of fact-checking. The existence of such structures has not been demonstrated to an extent that would have a noticeable effect on the overall quality of the data.
Future work: In the future work, we would like to focus on experiments with a recently published purely Czech language model CZERT [Sido et al., 2021] and a more robust multilingual LM XLM-R [Conneau et al., 2020]. It would also be interesting to explore the rapidly expanding field of efficient Transformers that can handle long inputs.

Bibliography

Alammar, Jay (June 2018). *The Illustrated Transformer.* [online; accessed 14-12-2020]. URL: `https://jalammar.github.io/illustrated-transformer`.

Attardi, Giuseppe (2019). *WikiExtractor.* `https://github.com/attardi/wikiextractor`.

Beltagy, Iz, Matthew E. Peters, and Arman Cohan (2020). *Longformer: The Long-Document Transformer.* arXiv: `2004.05150 [cs.CL]`.

Binau, Julie and Henri Schulte (2020). “Danish Fact Verification: An End-to-End Machine Learning System for Automatic Fact-Checking of Danish Textual Claims”. In:

Brown, Tom B. et al. (2020). *Language Models are Few-Shot Learners.* arXiv: `2005.14165 [cs.CL]`.

Buciluundefined, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil (2006). “Model Compression”. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD ’06. Philadelphia, PA, USA: Association for Computing Machinery, pp. 535–541. URL: `https://doi.org/10.1145/1150402.1150464`.

Cer, Daniel et al. (2018). *Universal Sentence Encoder.* arXiv: `1803.11175 [cs.CL]`.

Chang, Wei-Cheng et al. (2020). “Pre-training tasks for embedding-based large-scale retrieval”. In: *arXiv preprint arXiv:2002.03932.*

Chen, Danqi et al. (2017). “Reading Wikipedia to Answer Open-Domain Questions”. In: *Association for Computational Linguistics (ACL).*

Conneau, Alexis et al. (2020). “Unsupervised Cross-lingual Representation Learning at Scale”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* URL: `http://dx.doi.org/10.18653/v1/2020.acl-main.747`.

Craswell, Nick et al. (2020). *Overview of the TREC 2019 deep learning track.* arXiv: `2003.07820 [cs.IR]`.

Dai, Zhuyun and Jamie Callan (2019). *Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval.* arXiv: `1910.10687 [cs.IR]`.

Dědková, Barbora (2021). “Multi-stage Methods for Document Retrieval in the Czech Language”. `https://www.overleaf.com/read/gkymfyhfkrkq`. MA thesis. Czech Technical University in Prague.

Derczynski, Leon, Julie Binau, and Henri Schulte (July 2020). “Maintaining Quality in FEVER Annotation”. In: *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*. Online: Association for Computational Linguistics, pp. 42–46. URL: https://www.aclweb.org/anthology/2020.fever-1.6.

Devlin, Jacob et al. (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805*.

Diggelmann, Thomas et al. (2020). *CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims*. arXiv: 2012.00614 [cs.CL].

Ding, Yingqi Qu Yuchen et al. (2020). *RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering*. arXiv: 2010.08191 [cs.CL].

Dumitrescu, Stefan, Andrei-Marius Avram, and Sampo Pyysalo (2020). “The birth of Romanian BERT”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. URL: http://dx.doi.org/10.18653/v1/2020.findings-emnlp.387.

Ein Dor, Liat et al. (July 2018). “Learning Thematic Similarity Metric from Article Sections Using Triplet Networks”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 49–54. URL: https://www.aclweb.org/anthology/P18-2009.

Gažo, Alexander (2021). “Algorithms for Document Retrieval in Czech Language Supporting Long Inputs”. https://gitlab.fel.cvut.cz/gazoalex/master-thesis/-/tree/master. MA thesis. Czech Technical University in Prague.

Hanselowski, Andreas et al. (Nov. 2018). “UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification”. In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Brussels, Belgium: Association for Computational Linguistics, pp. 103–108. URL: https://www.aclweb.org/anthology/W18-5516.

Hao, Karen (Dec. 2020). *A college kid created a fake, AI-generated blog. It reached 1 on Hacker News*. URL: https://www.technologyreview.com/2020/08/14/1006780/ai-gpt-3-fake-blog-reached-top-of-hacker-news/.

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2015). *Distilling the Knowledge in a Neural Network*. arXiv: 1503.02531 [stat.ML].

Hofstätter, Sebastian et al. (2020). *Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation*. arXiv: 2010.02666 [cs.IR].

Illing, Sean (Jan. 2020). *”Flood the zone with shit”: How misinformation overwhelmed our democracy*. [online; accessed 14-12-2021]. URL: https://www.vox.com/policy-and-politics/2020/1/16/20991816/impeachment-trial-trump-bannon-misinformation.

Johnson, Jeff, Matthijs Douze, and Herve Jegou (2019). “Billion-scale similarity search with GPUs”. In: *IEEE Transactions on Big Data*, pp. 1–1. URL: http://dx.doi.org/10.1109/tbdata.2019.2921572.

Joshi, Pratik et al. (2020). “The State and Fate of Linguistic Diversity and Inclusion in the NLP World”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. URL: http://dx.doi.org/10.18653/v1/2020.acl-main.560.

Karpukhin, Vladimir et al. (2020). “Dense Passage Retrieval for Open-Domain Question Answering”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).* URL: `http://dx.doi.org/10.18653/v1/2020.emnlp-main.550`.

Khattab, Omar and Matei Zaharia (July 2020). “ColBERT”. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* Implementation: `https://github.com/stanford-futuredata/ColBERT`. URL: `http://dx.doi.org/10.1145/3397271.3401075`.

Kovach, B. and T. Rosenstiel (2007). *The Elements of Journalism: What Newspeople Should Know and the Public Should Expect.* Three Rivers Press. URL: `https://books.google.cz/books?id=iMA1WhtiRBkC`.

Kudo, Taku (2018). “Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* URL: `http://dx.doi.org/10.18653/v1/P18-1007`.

Lample, Guillaume and Alexis Conneau (2019). *Cross-lingual Language Model Pretraining.* arXiv: `1901.07291 [cs.CL]`.

Lee, Kenton, Ming-Wei Chang, and Kristina Toutanova (2019). “Latent Retrieval for Weakly Supervised Open Domain Question Answering”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* URL: `http://dx.doi.org/10.18653/v1/P19-1612`.

Li, Bo et al. (2021). *Scaling End-to-End Models for Large-Scale Multilingual ASR.* arXiv: `2104.14830 [cs.CL]`.

Lin, Jimmy et al. (2021). *Pyserini: An Easy-to-Use Python Toolkit to Support Replicable IR Research with Sparse and Dense Representations.* arXiv: `2102.10073 [cs.IR]`. URL: `https://github.com/castorini/anserini`.

Lu, Wenhao, Jian Jiao, and Ruofei Zhang (2020). *TwinBERT: Distilling Knowledge to Twin-Structured BERT Models for Efficient Retrieval.* arXiv: `2002.06275 [cs.IR]`.

Macková, Kateřina and Milan Straka (2020). “Reading Comprehension in Czech via Machine Translation and Cross-Lingual Transfer”. In: *Lecture Notes in Computer Science*, pp. 171–179. URL: `http://dx.doi.org/10.1007/978-3-030-58323-1_18`.

Madison, May (2020). *A Survey of Long-Term Context in Transformers.* [online; accessed 15-04-2021]. URL: `https://www.pragmatic.ml/a-survey-of-methods-for-incorporating-long-term-context`.

Manning, Christopher (2008). *Introduction to information retrieval.* New York: Cambridge University Press.

Martin, Louis et al. (2020). “CamemBERT: a Tasty French Language Model”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* URL: `http://dx.doi.org/10.18653/v1/2020.acl-main.645`.

Mitra Bhaskar Craswell Nick, et al. (2018). *An introduction to neural information retrieval.*

Niven, Timothy and Hung-Yu Kao (2019). “Probing Neural Network Comprehension of Natural Language Arguments”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* URL: `http://dx.doi.org/10.18653/v1/P19-1459`.

Nogueira, Rodrigo and Kyunghyun Cho (2019a). *Passage Re-ranking with BERT*. arXiv: 1901.04085 [cs.IR].

Nogueira, Rodrigo, Jimmy Lin, and AI Epistemic (2019b). "From doc2query to docTTTTTquery". In:

Nogueira, Rodrigo et al. (2019c). *Document Expansion by Query Prediction*. arXiv: 1904.08375 [cs.IR].

Přibáň, Pavel, Tomáš Hercig, and Josef Steinberger (Sept. 2019). "Machine Learning Approach to Fact-Checking in West Slavic Languages". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd., pp. 973–979. URL: https://www.aclweb.org/anthology/R19-1113.

Prokop, Daniel (2020). *Slepé skvrny: o chudobě, vzdělávání, populismu a dalších výzvách české společnosti*. Host.

Radford, Alec et al. (2019). "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8, p. 9.

Reimers, Nils and Iryna Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. URL: http://dx.doi.org/10.18653/v1/D19-1410.

— (2020). "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. URL: http://dx.doi.org/10.18653/v1/2020.emnlp-main.365.

Robertson, Stephen and Hugo Zaragoza (2009). *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Rogers, Anna, Olga Kovaleva, and Anna Rumshisky (2020). "A Primer in BERTology: What We Know About How BERT Works". In: *Transactions of the Association for Computational Linguistics* 8, pp. 842–866. URL: http://dx.doi.org/10.1162/tacl_a_00349.

Rosset, Corby (Feb. 2020). *Turing-NLG: A 17-billion-parameter language model by Microsoft*. URL: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/.

Roy, Aurko et al. (Feb. 2021). "Efficient Content-Based Sparse Attention with Routing Transformers". In: *Transactions of the Association for Computational Linguistics* 9, pp. 53–68. URL: http://dx.doi.org/10.1162/tacl_a_00353.

Ruder, Sebastian (2021). *Recent Advances in Language Model Fine-tuning*. http://ruder.io/recent-advances-lm-fine-tuning.

Sanh, Victor et al. (2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *ArXiv* abs/1910.01108.

Schuster, M. and K. Nakajima (2012). "Japanese and Korean voice search". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152.

Schwartz, Roy et al. (Nov. 2020). “Green AI”. In: *Communications of the ACM* 63.12, pp. 54–63. URL: `http://dx.doi.org/10.1145/3381831`.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* URL: `http://dx.doi.org/10.18653/v1/P16-1162`.

Shoeybi, Mohammad et al. (2019). *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.* arXiv: `1909.08053 [cs.CL]`.

Sido, Jakub et al. (2021). *Czert – Czech BERT-like Model for Language Representation.* arXiv: `2103.13031 [cs.CL]`.

Silverman, Craig (2014). *Verification and Fact Checking.* [online; accessed 04-09-2021]. URL: `https://datajournalism.com/read/handbook/verification-1`.

Thorne, James and Andreas Vlachos (2018a). “Automated Fact Checking: Task Formulations, Methods and Future Directions”. In: *Proceedings of the 27th International Conference on Computational Linguistics.* Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 3346–3359. URL: `https://www.aclweb.org/anthology/C18-1283`.

Thorne, James et al. (2018b). “FEVER: a large-scale dataset for fact extraction and verification”. In: *arXiv preprint arXiv:1803.05355.*

Thorne, James et al. (2018c). “The fact extraction and verification (fever) shared task”. In: *arXiv preprint arXiv:1811.10971.*

— (Nov. 2019). “The FEVER2.0 Shared Task”. In: *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER).* Hong Kong, China: Association for Computational Linguistics, pp. 1–6. URL: `https://www.aclweb.org/anthology/D19-6601`.

Turnbull, Doug (Oct. 2015). *BM25 The Next Generation of Lucene Relevance.* [online; accessed 04-02-2021]. URL: `https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation`.

Ullrich, Herbert (2021). “Dataset for Automated Fact Checking in Czech Language”. `https://www.overleaf.com/read/nfxjywqwthgx`. MA thesis. Czech Technical University in Prague.

Vaswani, Ashish et al. (2017). “Attention is all you need”. In: *Advances in neural information processing systems*, pp. 5998–6008.

Vattimo, Gianni (2013). *Transparentní společnost.* Rubato.

Wang, Sinong et al. (2020). *Linformer: Self-Attention with Linear Complexity.* arXiv: `2006.04768 [cs.LG]`.

Wolf, Thomas et al. (Oct. 2020). “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* Online: Association for Computational Linguistics, pp. 38–45. URL: `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Yang, Wei et al. (July 2019a). “Critically Examining the “Neural Hype””. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* URL: `http://dx.doi.org/10.1145/3331184.3331340`.

Yang, Wei et al. (2019b). “End-to-End Open-Domain Question Answering with Bertserini”. In: *Proceedings of the 2019 Conference of the North.* URL: `http://dx.doi.org/10.18653/v1/N19-4013`.

Yilmaz, Zeynep Akkalyoncu et al. (2019). “Applying BERT to document retrieval with birch”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pp. 19–24.

Acronyms

BERT Bidirectional Encoder Representations from Transformers

BM25 Best Match 25

BOW Bag of words

BPE Byte Pair Encoding

ČTK Česká Tisková Kancelář (Czech News Agency)

DR Document Retrieval

FAISS Facebook AI Similarity Search

FC Fully connected

FEVER Fact Extraction and Verification

IR Information Retrieval

LM Language Model

MLM Masked Language Model

NLI Natural Language Inference

NLP Natural Language Processing

PE Positional Encoding

QA Question Answering

SVD Singular value decomposition

TF-IDF Term Frequency - Inverse Document Frequency

TREC Text Retrieval Conference

Content of enclosed repository

The repository is stored here `https://gitlab.fel.cvut.cz/factchecking/experimental-martin` and can be made accessible on request. Later, it will be eventually added also here `https://github.com/ryparmar/master-thesis`.

```
repository.zip ...................................................... repository
 ├─ src ............................................... implementation sources
 │   ├─ anserini ............................ Anserini implementation sources
 │   ├─ drqa ..................................... DrQA implementation sources
 │   ├─ ColBERT ............................... ColBERT implementation sources
 │   ├─ claims-quality ............... data quality metrics implementation sources
 │   └─ pretraining .................. ICT, BFS pretraining implementation sources
 └─ text .............................................. the thesis text directory
     └─ thesis.pdf ............................... the thesis text in PDF format
```