

Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Detecting Previously Fact-Checked Claims

Bc. Vít Šenfeld

Supervisor: Ing. Jan Drchal, Ph.D.

Field of study: Open Informatics

Subfield: Artificial Intelligence

May 2023

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Šenfeld** Jméno: **Vít** Osobní číslo: **483446**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávající katedra/ústav: **Katedra počítačů**
Studijní program: **Otevřená informatika**
Specializace: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Detekce dříve ověřených tvrzení

Název diplomové práce anglicky:

Detecting Previously Fact-Checked Claims

Pokyny pro vypracování:

The task is researching and experimenting with methods to detect previously fact-checked documents.

- 1) Research state-of-the-art methods for detecting previously fact-checked claims. Focus on connection to Document Retrieval/Evidence Retrieval approaches.
- 2) Find or create datasets for the task, such as data based on the PolitiFact database.
- 3) Select appropriate methods, preferably based on neural language models of Transformer architecture. You may consider a combination with keyword-based approaches like BM25. Design and implement the NLP pipeline.
- 4) Select an appropriate evaluation methodology in alignment with previous works.
- 5) Evaluate implemented methods' performance on the selected datasets.

Seznam doporučené literatury:

- [1] Shaar, S. et al. (2020) 'That is a Known Lie: Detecting Previously Fact-Checked Claims', pp. 3607–3618. doi:10.18653/v1/2020.acl-main.332.
- [2] Shaar, S. et al. (2021a) 'Assisting the Human Fact-Checkers: Detecting All Previously Fact-Checked Claims in a Document', arXiv:2109.07410 [cs] [Preprint]. Available at: <http://arxiv.org/abs/2109.07410> (Accessed: 7 March 2022).
- [3] Vo, Nguyen, and Kyumin Lee. "Where are the facts? searching for fact-checked information to alleviate the spread of fake news." arXiv preprint arXiv:2010.03159 (2020).

Jméno a pracoviště vedoucí(ho) diplomové práce:

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

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

Datum zadání diplomové práce: **17.02.2023**

Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **22.09.2024**

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

I would like to thank my supervisor Ing. Jan Drchal, Ph.D, for the patient guidance, helpful consultations, valuable comments and the opportunity to create this thesis. In addition, I would like to thank my family and all my friends for their great patience and support during my studies.

The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics” is also gratefully acknowledged.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 26. 2023

Abstract

In recent years, with the significant increase in fake news published daily, there is a need for a reaction. This thesis focuses on the support of current fact-checking organizations, which nowadays mostly rely on manual fact-checking in the form of the detection of previously fact-checked claims, a process that could potentially offload a significant portion of their workload.

In the first part, there is a description of the theoretical background of such a task (information retrieval task) and the various models available to perform it – including both traditional and neural models (in the form of recently very popular transformer models) such as BERT, RoBERTa, ELECTRA and various distilled models. The theoretical part is accompanied by a brief introduction of approaches to the usage of such models.

In the second part, experiments are introduced and conducted on publicly available datasets described in corresponding sections. These experiments, run on cross-attention, and two-tower neural models, allow us to compare the performance of various chosen models in terms of computational cost and, most importantly, quality, measured by various metrics introduced in the theoretical part. These same experiments are also performed on the same models after finetuning them using the introduced datasets, where we also experiment with a sampling of negative samples by comparing hard and random negative samples. The comparison was run not only between various neural models but also against a strong baseline which was a traditional (BM-25) model.

Keywords: fact-checking, BM25, transformers, NLP, document retrieval, information retrieval

Supervisor: Ing. Jan Drchal, Ph.D.

Abstrakt

V posledních letech došlo k razantnímu nárůstu počtu publikovaných fake-news. V reakci na tuto skutečnost je cílem této práce podpora stávajících organizací zabývajících se fact-checkingem, které se v současnosti většinou spoléhají na manuální ověřování faktů, a to ve formě detekce již dříve ověřených tvrzení, což by potenciálně mohlo pomoci výrazně snížit jejich pracovní zátěž.

První část práce se zabývá představním teoretické části takové úlohy (úloha získávání informací) a různými modely, které ji mohou řešit. Prozkoumal jsem jak klasické modely, tak v současnosti populární Transformers sítě jako jsou BERT, RoBERTa, ELECTRA a dále různé destilované modely. Teoretická část je doplněná krátkým představením přístupů k používání takových modelů.

V druhé části práce byli představeny a provedeny experimenty na veřejně dostupných datasetech. Tyto experimenty, provedené na cross-attention a two-tower modelech Transformers modelech, nám umožní porovnat výpočetní náročnost a především výkonost, měřenou několika metrikami představenými v teoretické části práce, jednotlivých vybraných modelů. Totožné experimenty jsou pak zopakovány na vybraných modelech po jejich dotrénování za využití představených datasetů. V této části také experimentujeme s různými přístupy k trénování, týkajících se výběru negativních vzorků a to porovnáním hard negative samples a random negative samples. Výše uvedené výsledky jsou pak také porovnány s tradičním (BM-25) modelem.

Klíčová slova: fact-checking, BM-25, transformers, NLP, vyhledávání dokumentů, získávání informací

Překlad názvu: Detekce dříve ověřených tvrzení

Contents

1 Introduction and Motivation	1		
2 Related work	3		
2.1 Information retrieval task	3		
2.2 Traditional IR models	4		
2.2.1 TF-IDF	4		
2.2.2 BM-25	5		
2.3 Neural models	6		
2.3.1 Tokenization	6		
2.3.2 Usage in the document retrieval pipeline	6		
2.3.3 Usage during evaluation of individual document	7		
2.3.4 Two-Tower paradigm	7		
2.3.5 Cross-Attention paradigm	8		
2.3.6 Other query-document interaction paradigms	8		
2.3.7 BERT	9		
2.3.8 DistilBERT	11		
2.3.9 TinyBERT	12		
2.3.10 MiniLM	14		
2.3.11 ELECTRA	15		
2.3.12 RoBERTa	16		
2.3.13 DistilRoBERTa	18		
3 Metrics	19		
3.1 Precision	19		
3.1.1 Precision@k	19		
3.2 Recall	19		
3.2.1 Recall@k	20		
3.3 Mean Reciprocal Rank	20		
3.3.1 MRR@k	20		
3.4 Average Precision @ k	20		
3.5 Mean Average Precision @ k	21		
3.6 Normalized Discounted Cumulative Gain	21		
3.6.1 NDCG@k	21		
4 Datasets	23		
4.1 General use datasets	23		
4.1.1 SQuAD dataset	23		
4.1.2 MS MARCO dataset	23		
4.1.3 Quora Duplicate Questions dataset	24		
4.1.4 NQ dataset	24		
4.1.5 STS Benchmark	24		
4.2 Datasets used for finetuning	24		
4.2.1 CLEF	24		
4.2.2 Where Are the Facts? Searching for Fact-checked Information to Alleviate the Spread of Fake News, EMNLP 2020	25		
5 Methodology	29		
5.1 Introduction of experiments	29		
5.2 Evaluation experiments – traditional models	30		
5.3 Evaluation and finetuning experiments – neural models	30		
5.4 Finetuning experiments	31		
5.5 Pipeline	32		
5.5.1 CLEF dataset preprocessing	32		
5.5.2 EMNLP dataset preprocessing	32		
5.5.3 Baseline	33		
5.5.4 Finetuning neural models	34		
5.5.5 Evaluating neural models	34		
5.6 Implementation	35		
5.6.1 Introducing used tools	35		
5.7 Models used	37		
5.7.1 Pyserini	37		
5.7.2 Elasticsearch	38		
5.7.3 all-MiniLM-L6-v2	38		
5.7.4 multi-qa-DistilBERT-cos-v1	38		
5.7.5 all-DistilRoBERTa-v1	38		
5.7.6 distiluse-base-multilingual-cased-v2	39		
5.7.7 nq-DistilBERT-base-v1	39		
5.7.8 stsb-TinyBERT-L-4	39		
5.7.9 quora-RoBERTa-base	39		
5.7.10 qnli-DistilRoBERTa-base	39		
5.7.11 ms-marco-TinyBERT-L-2-v2	39		
5.7.12 ms-marco-MiniLM-L-6-v2	39		
5.7.13 ms-marco-ELECTRA-base	40		
5.7.14 XLM-RoBERTa-large-squad2	40		
5.8 Choosing models from multiple runs	40		
6 Experiments	41		
6.1 Experimental setting	41		
6.2 Finetuning setting	41		
6.3 Measured performance	42		
6.4 Results	42		
6.4.1 Baseline and base neural models	42		
6.4.2 Finetuned models – CLEF dataset evaluation	50		

6.4.3 Finetuned models – EMNLP	
dataset evaluation	59
6.4.4 Inference and finetuning times	66
6.5 Conclusion	67
7 Conclusion	69
Bibliography	71
A Appendix	75
A.1 Software requirements	75
A.2 Attachments structure	75
A.3 Tables	77

Figures

2.1 Difference between two-tower and cross-attention models(Chang et al. 2020).....	8
2.2 Schematic diagrams illustrating query-document matching paradigms in neural IR (Khattab & Zaharia 2020).....	8
2.3 Overall pre-training and finetuning procedurs for BERT (Devlin et al. 2019).....	10
2.4 The Transformer – model architecture (Vaswani et al. 2017) .	11
2.5 Overview of Deep Self-Attention Distillation (Wang et al. 2020)....	15
2.6 GLUE Score – Pre-train FLOPs performance evaluation of NLP models (Clark et al. 2020)	16
4.1 Attached image.....	27
6.1 MRR@1 for base neural models and baseline evaluated on the CLEF dataset	44
6.2 MRR@3 for base neural models and baseline evaluated on the CLEF dataset	45
6.3 MRR@3 for base neural models and baseline evaluated on the EMNLP Politifact nosplit dataset in both base and image text variant .	46
6.4 MRR@3 for base neural models and baseline evaluated on the EMNLP Politifact base text dataset in both split and nosplit variant ..	47
6.5 MRR@3 for base neural models and baseline evaluated on the EMNLP Politifact image text dataset in both split and nosplit variant ..	48
6.6 MRR@3 for base neural models and baseline evaluated on the EMNLP Snopes base text dataset in both split and nosplit variant.....	48
6.7 MRR@3 for base neural models and baseline evaluated on the EMNLP Snopes image text dataset in both split and nosplit variant ..	49
6.8 MRR@3 for bi-encoders finetuned using the CLEF dataset with both negative samples evaluated on the CLEF dataset	51
6.9 MRR@3 for cross-encoders finetuned using the CLEF dataset with both negative samples evaluated on the CLEF dataset	52
6.10 MRR@1 for bi-encoders finetuned using the CLEF dataset with both negative samples evaluated on the CLEF dataset	52
6.11 MRR@1 for cross-encoders finetuned using the CLEF dataset with both negative samples evaluated on the CLEF dataset	53
6.12 MRR@3 for neural models finetuned using EMNLP Politifact nosplit dataset with best negative samples approaches comparing the influence of image and base text on the CLEF dataset	54
6.13 MRR@3 for neural models finetuned using EMNLP Politifact nosplit dataset with random negative samples approaches comparing the influence of image and base text on the CLEF dataset	55
6.14 MRR@3 for neural models finetuned using EMNLP Politifact nosplit base text dataset with both negative samples approaches on the CLEF dataset	56
6.15 MRR@3 for neural models finetuned using EMNLP Politifact nosplit image text dataset with both negative samples approaches on the CLEF dataset	57
6.16 MRR for bi-encoders finetuned using the EMNLP Politifact nosplit dataset with best negative samples and only base text query evaluated on the CLEF dataset	58

6.17 MRR for bi-encoders finetuned using the EMNLP Politifact nosplit dataset with best negative samples and image text query evaluated on the CLEF dataset	58
6.18 MRR@3 CLEF finetuned using best negative samples evaluated on the EMNLP Politifact split image text dataset	60
6.19 MRR@3 for models finetuned with best negative samples using CLEF dataset evaluated on the EMNLP Politifact nosplit base text dataset	61
6.20 MRR@3 for models finetuned with random negative samples using CLEF dataset evaluated on the EMNLP Politifact nosplit base text dataset	62
6.21 MRR@3 for models finetuned with best negative samples using CLEF dataset evaluated on the EMNLP Politifact nosplit image text dataset	63
6.22 MRR@3 for models finetuned with random negative samples using CLEF dataset evaluated on the EMNLP Politifact nosplit image text dataset	64
6.23 MRR@3 best CLEF finetuned models evaluated on EMNLP comparison	65

Tables

2.1 Tokenization example	6
4.1 CLEF query – verified claim pair example	25
4.2 CLEF dataset size	25
4.3 EMNLP query example	26
4.4 EMNLP dataset size	27
6.1 Shortcuts of model names used within graphs	42
A.1 Traditional models comparison – CLEF dataset	77
A.2 Pyserini – EMNLP dataset results	78
A.3 Base variants of bidirectional models – CLEF dataset results ...	79
A.4 Base variants of cross-encoder models – CLEF dataset results ...	80
A.5 Base bidirectional models – EMNLP Politifact nosplit image text dataset	81
A.6 Base bidirectional models – EMNLP Politifact split image text dataset	81
A.7 Base bidirectional models – EMNLP Snopes split image text dataset	82
A.8 Base cross-encoder models – EMNLP Politifact nosplit image text dataset	82
A.9 Base cross-encoder models – EMNLP Politifact split image text dataset	83
A.10 Base cross-encoder models – EMNLP Snopes nosplit image text dataset	83
A.11 Base cross-encoder models – EMNLP Snopes split image text dataset	84
A.12 CLEF with best negative samples finetuned bidirectional models – CLEF results	85
A.13 CLEF with random negative samples finetuned bidirectional models – CLEF results	86

A.14 CLEF with best negative samples finetuned cross-encoder models – CLEF results	87
A.15 CLEF with random negative samples finetuned cross-encoder models – CLEF results	88
A.16 EMNLP finetuned all-MiniLM-L6-v2 – CLEF results .	89
A.17 EMNLP finetuned all-DistilRoBERTa-v1 – CLEF results	90
A.18 EMNLP finetuned ms-marco-TinyBERT-L-2-v2 – CLEF results	91
A.19 EMNLP finetuned ms-marco-MiniLM-L-6-v2 – CLEF results	92
A.20 EMNLP finetuned all-MiniLM-L6-v2 – EMNLP results	92
A.21 EMNLP finetuned ms-marco-MiniLM-L-6-v2 – EMNLP results	93
A.22 CLEF finetuned all-MiniLM-L6-v2 – EMNLP results	93
A.23 CLEF finetuned stsb-TinyBERT-L-4 – EMNLP results	94
A.24 CLEF finetuned ms-marco-MiniLM-L-6-v2 – EMNLP results	94
A.25 Sum of inference times comparison [s]	95
A.26 Finetuning times [s]	95

Chapter 1

Introduction and Motivation

Today, with an increasing number of fake news published every hour, there is a need for a reaction to them. Traditionally, the fight against fake news was led by a number of organizations that rely on a manual approach – every single claim must be evaluated by an (ideally trained) human.

Now I will provide two examples of fake news evaluated by the organization Politifact. The first one is a Facebook post stating that "A confidential source claims that the Moon is a habitable place and that it is inhabited by more than 250 million humanoid aliens,"¹ – when evaluating this statement, their employee first provides a definition of a habitable planet (according to NASA), then it checks various of the necessary condition against current knowledge about the moon (cited from trustworthy sources such as NASA or ESA). This leads them to refuting the statement (as they prove that it cannot "sustain life for a significant period of time") and proclaim it "Pants on fire" (the worst rating on their "truth-o-meter" – the scale of how true they find the evaluated statement).

While the first example required various reliable sources and was generally quite complex, lots of potentially fake news is much easier to evaluate, as depicted by the evaluation I chose as the second example. The second example is straightforward – the claim is "Saudi team gets \$460K Rolls-Royces for upset win."² It is connected to surprising win of Saudi national team against favoured Argentina national team during the group stage of the 2022 FIFA World Cup in Qatar. It was easily refuted by citing a question from one of the press conferences that took place during the tournament that tackles this exact topic (allowing to rate it "false"). The interesting thing about this evaluation is that it includes multiple examples of the same claim, which is an important fact for this thesis.

One of the advantages of having already functioning organizations as described is that we have already created quite a large dataset of evaluated potential fake news. The existence of the dataset could potentially prove to be very useful, as a lot of fake news tend to repeat itself many times (as an example, take a politician during the preelection campaign that will make

¹Avialable at: <https://www.politifact.com/factchecks/2023/mar/06/facebook-posts/is-the-moon-habitable-for-people-not-with-its-lack/>

²Avialable at: <https://www.politifact.com/factchecks/2022/nov/29/viral-image/saudi-coach-striker-say-no-rolls-royces-beating-ar/>

various claims at many different events. However, many of these claims will be essentially the same, although differently expressed). Also, this could be very useful when dealing with fake news not related to politics but to other topics, such as scientific or historical facts. Another fact that is needed to take into consideration is that manual fact-checking is, in fact, quite time-consuming.

The rise of social networks allowed for the number of potentially fake news to rise significantly; therefore, it is of essence to provide assistance to the system currently discovering false claims. Many possible approaches exist, such as fully automatic fact-checking (where the claim is evaluated based on a knowledge base without human interaction). However, humans tend to trust these solutions less than they trust human evaluation. (Shaar et al. 2020)

Therefore I decided to aim my work at tools that can possibly aid human evaluators. The idea is that by detecting previously fact-checked claims, we could possibly offload much of their work; also, even when the found verified claim is not exactly the same, it could prove helpful for the process. To achieve this, I will evaluate various models for information retrieval tasks.

First, in the theoretical part, I will introduce such models (both traditional and neural network models). Also, I will introduce various metrics by which these can be evaluated and a few datasets we could run their comparison on.

Secondly, in the practical part of my thesis, I will describe particular models that will be used (because in the theoretical part, I will describe only their base idea, here, I will focus on specific used models). Then, I will evaluate these models and compare their results. I will not only rely on already pre-trained models, but I will also finetune (a process which I will also thoroughly describe) them and compare the results of finetuned models with the original ones. While previously mentioned matters mostly to neural network models, I will also have to choose a good enough baseline traditional model as a weak baseline is the problem of many such comparisons (Yang et al. 2019).

Finally, this evaluation will provide much helpful information (not only by comparing the results of various systems but also the speed of their finetuning and their speed during inference, as time is also of the essence) that could potentially be used when implementing a real, deployable, semi-automatic fact-checking system employing document retrieval methods (document retrieval being a branch of information retrieval where the information is stored primarily in textual form).

Chapter 2

Related work

This chapter focuses on introducing the task of information (document) retrieval and later parts also on introducing various models (mostly neural but also traditional ones) used for performing such a task.

2.1 Information retrieval task

As defined before, we need to retrieve information in order to support fact-checking organizations. This leads us to the information retrieval task (IR), which is defined in (Christopher D. Manning, Prabhakar Raghavan 2008) as "finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)".

However, information retrieval is a broad topic, and we need to only focus on one of its branches (although arguably one of the most popular and often interchanged with general information retrieval – a trend that I will mostly follow in this work). This mentioned branch is called (ranked) document retrieval (DR) and is one of the classic IR problems (also performed by well-known web search engines such as Google, Bing, etc.). Its main goal is to retrieve a ranked collection of documents corresponding to a given query – the collection is ranked by the relevance of individual documents to the query, from best to worst retrieved. (Mitra & Craswell 2018)

As mentioned before, this task can be solved by many traditional and neural models, some of which I will describe in the following sections of this chapter (focusing on models that will be used during the experimental part of my work).

The connection to the detection of previously fact-checked is clear: the new claim is the query in the IR (DR) task, while the collection of already verified claims is the collection of documents; our goal is to retrieve the best corresponding documents (as they are the most probable to be already fact-checking the given claim or at least similar one).

2.2 Traditional IR models

The section describes the theory and history of traditional IR models used during the experimental part. Although neural models become very popular nowadays, there is still an important place in the IR task for traditional models, whether as one of the stages in multi-stage retrieval or as separately used models in many situations. They still have some advantages, such as (usually) lower inference times, and the best implementations of them are also solid in terms of results (as I will show in the experimental part), which is no surprise, given they were dominant in this task up until the early 2010s.

2.2.1 TF-IDF

One of the most famous traditional models of IR and the basis of many applications. It is based on the logical assumption that the document (or zone) that mentions the query term more often is more relevant and, thus, should receive a higher score than other documents. Moreover, it also takes into account that some words generally appear more often without offering much relevance (in English, typically words such as "the", "a", "an", etc.).

The base of the model is a scheme called term frequency (TF), denoted $tf_{t,d}$ (with t denoting term and d document) where $tf_{t,d}$. It is based on the bag of words model, which means that exact ordering is ignored. The elementary approach is to set $tf_{t,d}$ equal to the number of occurrences of term t in document d , but any other mapping of this tuple to a real positive value is possible. Term frequency alone is a score that is usable in IR tasks.

However, TF suffers from one crucial problem: all terms are equal in their contribution, even though some words really do not have any relevance when trying to retrieve documents relevant to the query (with examples mentioned above). This is where inverse document frequency (IDF) comes in handy. Basically, it is a mechanism for distinguishing and appropriately weighting terms that occurs too often in the collection of documents to be of relevance.

First, we have to introduce document frequency df_t , which is equal to the number of documents in the collection where the term t is present. Then we could continue and define IDF, where IDF of term t with a collection of documents of size N is

$$idf_t = \log\left(\frac{N}{df_t}\right) \quad (2.1)$$

The intuition behind IDF is easy; the more documents the term is in, the lower the score.

Now let us continue to complete TF-IDF:

$$tf-idf_{t,d} = tf_{t,d} \cdot idf_t \quad (2.2)$$

At this point, we have a weighted score created by term t in document d . For the complete scoring system, we need to evaluate the whole document d based on complete query q :

$$\text{Score}(q, d) = \sum_{t \in q} \text{tf-idf}_{t,d} \quad (2.3)$$

Still, this score has a problem with differences in the lengths of documents in a collection. As long as all the documents are not of the same length, there is a possibility that the score will be higher for longer, less relevant documents, than for shorter more relevant ones because the longer document might just somehow accumulate enough occurrences of query terms.

Therefore there is a need for further expansion of the previous concept. (Christopher D. Manning, Prabhakar Raghavan 2008) does this by utilizing the vector space model combined with TF-IDF (and thus a bag of words representation). Define a vector derived from document d $V(d)$, in which every component is a dictionary term, and it is computed using TF-IDF. Then all the representations of documents (and even queries) will be vectors of the same length. The similarity between two documents can then be computed by dot-product. This is a convenient concept, but it does not really solve the problem with different sizes of documents. Therefore we need it also to be length-normalized – for this, a simple Euclidean norm (denoted as $|\cdot|$) can be utilized. Leading us to the equation:

$$\text{Score}(q, d) = \frac{V(q) \cdot V(d)}{|V(q)| |V(d)|} \quad (2.4)$$

Then we can simply compute such a score for every document in the collection and by sorting the results from highest to lowest obtain results (usually we retrieve top-k scoring documents with k being an arbitrary natural number). (Christopher D. Manning, Prabhakar Raghavan 2008)

2.2.2 BM-25

Originally introduced in (Robertson & Zaragoza 2009), one of the most popular traditional IR models. It also employs TF and IDF concepts that I have introduced in 2.2.1. It is defined as follows:

$$\text{BM25}(q, d) = \sum_{t \in q} \text{idf}(t_q) \cdot \frac{\text{tf}(t_q, d)(k_1 + 1)}{\text{tf}(t_q, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})} \quad (2.5)$$

where avgdl is the average length of documents in the collection D , k_1 and b are tunable parameters (usually tuned on a validation dataset). (Mitra & Craswell 2018) claims that oftentimes $k_1 \in [1.2, 2.0]$ and b is set equal to 0.75. Term idf is here defined as follows:

$$\text{idf}(t_q) = \log\left(\frac{|D| - \text{df}(t) + 0.5}{\text{df}(t) + 0.5}\right) \quad (2.6)$$

Base BM25 only aggregates contributions from individual terms. (Mitra & Craswell 2018) BM-25 is suitable for a variety of use cases as shown for example in (Shaar et al. 2021).

2.3 Neural models

In this section, I will describe various approaches and paradigms tackling the usage of neural models (in document retrieval pipeline, during pre-training and finetuning) in the first subsections, while the later subsections will be about various neural models of which most of them will be later used during the experiments.

2.3.1 Tokenization

A token can be a word, subword or character; our goal is to have a finite vocabulary that we can use to express our input text – i.e., we transform the input text into a sequence of tokens so that the subsequences of textual input match the tokens. Examples of tokenization methods will be introduced in the following sections, for now, let us limit ourselves to an example (Table 2.1) of tokenization using one of the models that will be discussed in this chapter – MiniLM, which utilizes one of the most famous tokenizer – WordPiece (introduced in (Wu et al. 2016)) (Wang et al. 2020).

Text	One of the advantages of having already functioning organizations as described is that we have already created quite a large dataset of evaluated potential fake news.
Tokenization	['one', 'of', 'the', 'advantages', 'of', 'having', 'already', 'functioning', 'organizations', 'as', 'described', 'is', 'that', 'we', 'have', 'already', 'created', 'quite', 'a', 'large', 'data', '###set', 'of', 'evaluated', 'potential', 'fake', 'news', '.']

Table 2.1: Tokenization example

2.3.2 Usage in the document retrieval pipeline

Generally speaking, using neural models is more computationally expensive than using traditional models (especially so when using the cross-attention paradigm – introduced in 2.3.5). Thus we have basically two main options for how to use them.

The first one is to use only them – i.e. evaluate every document in the collection against the claim, even at the higher computational cost (single-stage retrieval).

The second one is to use them only for reranking – i.e. when checking against a large collection, select use them only to rerank top k documents chosen by traditional models. Even for an extensive collection, we will obtain reasonable results at high enough k (can be as high as 500) for traditional models and then use the neural models to rerank the retrieved documents (so that we can use only top l documents, $l \ll k$). This potentially leads to results worse than the first variant, but it is significantly faster when used for a large collection. This approach is called multi-stage document retrieval, and as

shown in (Nogueira et al. 2019), it is possible to have more than two stages that I have described. They also used a traditional BM-25-based model as the first stage, but then they used two different BERT models for reranking (as the second and third stages).

■ 2.3.3 Usage during evaluation of individual document

In this part, I will describe the usage of a neural model when focusing on the processing outside of the model, while the direct usage of the model will be described in the subsections 2.3.4 and 2.3.5. One thing that we need to focus on is how to input the data into the model. As most of the documents and a significant portion of queries are longer than the input accepted by ordinary neural models (for example, the base variant of BERT accepts 512 tokens (Devlin et al. 2019) while one token is on average cca 0.75 of English word).

Two base approaches were introduced in (Sun et al. 2019). The first is to truncate the input (i.e. take only the beginning, the end, or beginning and end combination), and the second one is to divide the text into fractions of acceptable length and feed them to the network one by one.

Another approach could be to employ text summarization on the document and then check only on this shortened version.

■ 2.3.4 Two-Tower paradigm

Query-document interaction paradigm that gained popularity mostly due to its lower computational cost (for example, when compared to 2.3.5, sometimes referred to as representation-based. Its main idea is to compute the encodings in independent "towers" (models, possibly same, possibly different, but preferably BERT based) for query and document (without any interaction) and then use these encodings to compute similarity score using, for example by employing cosine similarity or simple dot product. (Chang et al. 2020). The following advantage is that with the separated encoding of queries and document collection, we can pre-encode the entire document collection, which will significantly quicken the pipeline.

■ Siamese paradigm

For a Siamese paradigm, the only difference from the Two-Tower paradigm is that the models used for creating document and query embeddings are the same (same type, same weights, etc.). Thus it is a narrower concept. Still, it is a very popular concept. It will be also one of the concepts used within this thesis, as in the experimental part, we will heavily rely on the SentenceTransformers framework introduced in (Reimers & Gurevych 2019).

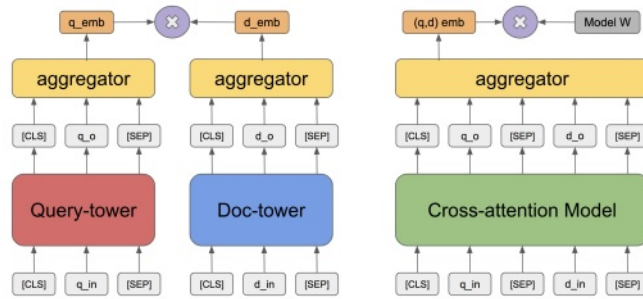


Figure 2.1: Difference between two-tower and cross-attention models (Chang et al. 2020)

2.3.5 Cross-Attention paradigm

Another query-document interaction paradigm (sometimes called all-to-all interaction). It is more computationally expensive than the Two-Tower paradigm (2.3.4), but it also usually achieves better results – there is a tradeoff between speed and quality of results. The idea is that we use only one model, with both query and document concatenated to one input (both parts separated by a special separating token). (Chang et al. 2020).

2.3.6 Other query-document interaction paradigms

Although the two previously mentioned paradigms are arguably the most famous, there are also a few other possibilities – for example, query-document interaction or late interaction. Late interaction is employed in ColBERT (Khattab & Zaharia 2020).

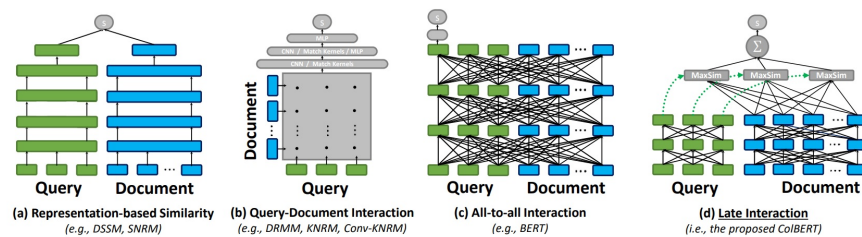


Figure 2.2: Schematic diagrams illustrating query-document matching paradigms in neural IR (Khattab & Zaharia 2020)

Also, please note that the models introduced in the following parts are mostly not inherently dependent on the query-document interaction paradigms. Most of them have various versions with varying interaction paradigms employed. This will be fully visible during the experimental stage, where some used models will have their representatives in both cross-attention and the siamese paradigm model groups tested.

■ 2.3.7 BERT

Language model pre-training has been shown to be effective in improving many NLP tasks, such as natural language inference or paraphrasing. For applying pre-trained language representations to downstream tasks, two strategies exist—feature-based and finetuning. The feature-based approach uses task-specific architectures that include pre-trained representations as additional features – an example of this approach is ELMo. The finetuning approach tends to introduce a minimal number of task-specific parameters and is trained on downstream tasks by finetuning the pre-trained parameters – the most famous example of this approach is OpenAI GPT (Generative Pre-trained Transformer).

Authors of BERT (Devlin et al. 2019) argued that both techniques “restrict the power of the pre-trained representations, especially for the finetuning approaches”. They stated that one of the main limitations is that standard language models are unidirectional and thus limited when choosing architectures used during pre-training. As an example, they stated OpenAI GPT, where left-to-right architecture was employed, which meant that in the self-attention layers, every token could only attend to previous tokens. They claimed these restrictions to be sub-optimal for sentence-level tasks and that it could be harmful to tasks such as question answering, where it is important to work with context from both directions. Aiming to improve the finetuning of BERT (Bidirectional Encoder Representations From Transformers) was proposed.

The BERT framework consists of two steps: pre-training and finetuning.

In the pre-training phase, the model is trained over different tasks on unlabeled data over two different tasks I will now describe:

To alleviate the unidirectionality constraint, BERT makes use of MLM (masked language model). MLM was inspired by the Cloze task. The MLM mask some of the input tokens (chosen randomly), and the objective is to predict the original vocabulary id of the masked word only based on its context. They masked 15 % of all WordPiece tokens in each sequence randomly. The inconvenience of this approach is that the mask tokens are only to be seen during pre-training, thus creating a mismatch with finetuning. To mitigate this, when a token is chosen for replacement, there is only an 80 % chance of the replacement being a mask token; 10 % of the time, it is unchanged and the remaining 10 % is for it being replaced with a random token. An important fact is that the MLM objective enables the fuse of both left and right context, which allows to pre-train deep bidirectional Transformer.

In addition to MLM, they used a “next sentence prediction” task to jointly pre-train text-pair representations, as many downstream tasks focus on understanding the relationship between two sentences. Therefore they pre-trained for a binarized next sentence prediction (conveniently, this task can be easily generated from any monolingual corpus). Each pretraining example consists of sentences A and B. While A precedes B, B is with a probability 0.5 sentence that actually follows A in the original document (labelled IsNext) and with probability 0.5, it is a random sequence from the total pre-training corpus (labelled NotNext). They claim this task to be beneficial to question

answering and natural language inference tasks, although, as I will mention later in 2.3.12, it might not always be for the best.

The dataset used for pre-training consisted of BooksCorpus and English Wikipedia (extracted only text passages), totalling 3.3M words.

As for the finetuning phase: it is quite straightforward, as for many other Transformers. The attention mechanism in the transformer allows finetuning on various downstream tasks involving both single text or text pairs. For each such task, one has to simply plug appropriate specific inputs and outputs into BERT and finetune all the parameters end-to-end. Also, compared to pre-training, it is relatively computationally inexpensive. Important to note that due to the nature of BERT, every finetuned model for a specific downstream task is technically the same one, it comes from the same pre-trained parameters that then only differ because of finetuning, but technically (in the number of parameters, and layers) they are all the same.

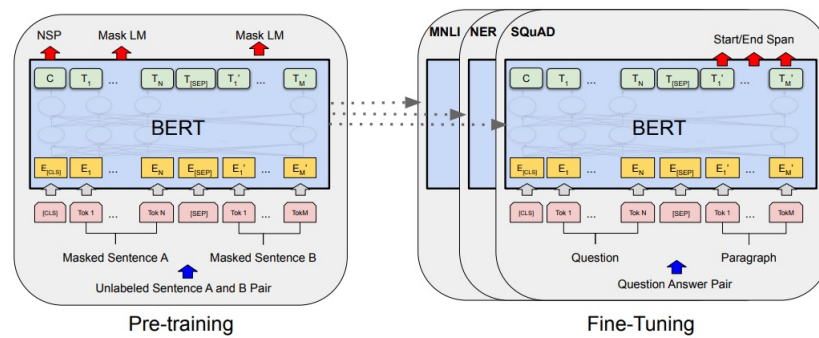


Figure 2.3: Overall pre-training and finetuning procedures for BERT (Devlin et al. 2019)

The architecture of the BERT is a multi-layer bidirectional Transformer encoder. They claim that their implementation is almost identical to the original implementation described in (Vaswani et al. 2017). They have implemented two models – BASE with 110M parameters and LARGE with 340M. As for input/output representations, they focus on being able to unambiguously represent both single sentences and their pairs in one sequence (this is important, for example, for the question-answering task). They utilized WordPiece embeddings first introduced in (Wu et al. 2016) with a vocabulary of 30 000 tokens.

When introduced, BERT achieved state-of-the-art level results for a wide range of tasks and thus soon became one of the most famous models. Nowadays, it is still widely used; various models are based on it (either finetuned using BERT or its improvements, such as RoBERTa), and many of them are included in this work.

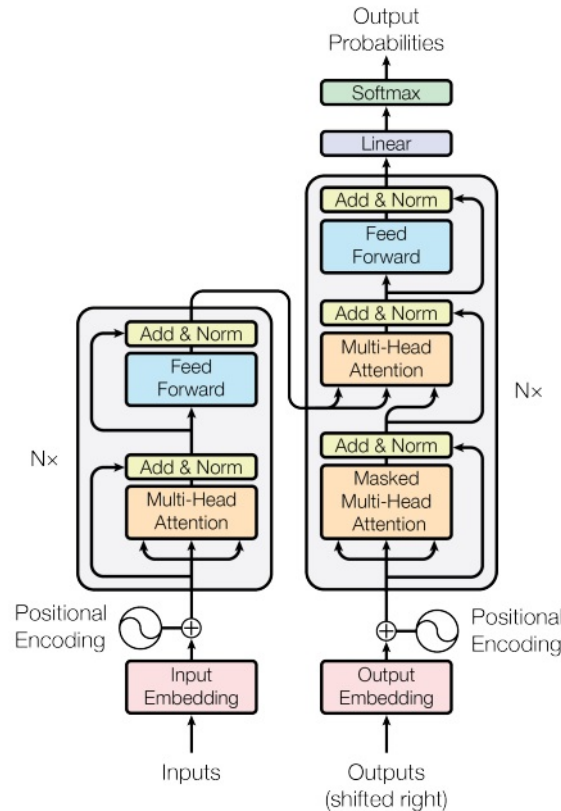


Figure 2.4: The Transformer – model architecture (Vaswani et al. 2017)

2.3.8 DistilBERT

With large-scale pre-trained models becoming basic NLP tools leading to significant improvement, the trend toward bigger and bigger models raises several concerns, such as the environmental costs of running these computationally demanding models and also their inability to run in real-time on-device, which slows down wider adoption – the problem is also lack of memory on these devices. DistilBERT (Sanh et al. 2019) is one of the first models that shows that it is possible to achieve good results on numerous downstream tasks with much smaller models pre-trained with knowledge distillation. Resulting in models being smaller, less demanding and thus can run on many more devices (such as mobile devices). They claim that 40 % smaller Transformer pre-trained via distillation can achieve similar results to the original one while simultaneously being 60 % faster.

Knowledge distillation is, for DistilBERT and many other models, the base building block of their success. It is "a compression technique in which a compact model – the student – is trained to reproduce the behaviour of a larger model – the teacher – or an ensemble of models". This is achieved via the minimization of training loss over soft target probabilities of teacher and student (t_i being probability estimated by the teacher and s_i probability

estimated by the student).

$$L_{ce} = \sum_i t_i \log(s_i) \quad (2.7)$$

They also utilized softmax-temperature:

$$p_i = \frac{\exp \frac{z_i}{T}}{\sum_j \exp \frac{z_j}{T}} \quad (2.8)$$

where z_i is the model score for class i and T is used to control the smoothness of output distribution. During training same T is applied to both student and teacher, while at inference, the temperature is set to 1 to use standard softmax. The final objective of the training is a linear combination of L_{ce} combined with supervised training loss and also cosine embedding loss. By adding cosine embedding loss, they aimed to align directions of hidden states vectors of student and teacher.

The architecture of DistilBERT is generally the same as BERT's, although "highly optimized by modern linear algebra frameworks". They mostly focused on reducing the number of layers of the model.

Finally, they claim that while achieving the previously mentioned reduction in size and increase in speed, DistilBERT still retains 97% of original understanding capabilities.

2.3.9 TinyBERT

With pre-training and then finetuning on downstream tasks have become a new paradigm for NLP and achieved notable success in many tasks (i.e. models such as BERT, XLNet, RoBERTa or ELECTRA). However, these models usually have quite a long inference time and therefore are difficult to be deployed on devices such as mobile phones. However, it was also proven that while they are quite huge in a number of parameters, they possibly contain many redundancies.

Therefore it is feasible to reduce the computational overhead of the pre-trained language models (PLMs) while retaining most of their performance. There are various techniques available such as quantization or weight pruning. In this chapter, I will focus on one of the techniques called knowledge distillation (KD). Its main aim is to transfer knowledge from the large teacher model to a much smaller student network. It was shown by (Jiao et al. 2019) that when using suitable methods, it is possible to distil knowledge from one of the most famous PLM BERT in such a way that we retain over 96 % of performance with both the number of parameters and inference times being less than 15 % of the original values. This was proven with the model TinyBERT (Jiao et al. 2019). Thus confirming the importance of the knowledge distillation method.

Now I will briefly introduce the KD method. As mentioned before, we aim to transfer knowledge from teacher network T to possibly a much smaller student network S . Student network is therefore trained to mimic the behaviour of

the teacher network. Formally KD can be modelled as the minimization of objective function:

$$L_{KD} = \sum_{x \in X} L(f^S(x), f^T(x)) \quad (2.9)$$

where $L(\cdot)$ is the loss function evaluating the difference between teacher and student network on text input x from training dataset X .

TinyBERT utilizes a novel distillation method for Transformer-based models proposed in the corresponding article (basically it performs KD not only in the task-specific training stage but also during the pre-training). They used both student and teacher networks built with the Transformer layer. They introduced the problem as having a student model with M Transformer layers and a teacher model with N such layer. As a first thing, we choose M out of N teacher model layers for the Transformer-layer distillation – mapping $n = g(m)$ (m -th student layer learns from $g(m)$ -th teacher layer. Formally we minimize the difference between corresponding behaviour functions of layers multiplied by a hyperparameter denoting the importance of the individual m -th layer.

Practically the distillation is divided into three parts:

Transformer-layer distillation that includes attention-based distillation and hidden states based distillation. Attention-based part is motivated by recent findings that claim that attention weights that BERT learned can possibly capture rich linguistic knowledge such as syntax and coreference information which are essential for natural language understanding. The student learns to fit matrices of multi-head attention in the teacher network. They also distil knowledge from the output of the transformer layer, basically trying to minimize the mean square error between the teacher’s hidden layer and the student’s hidden layer multiplied by the learnable transformation of the student network’s hidden states into the same space as the teacher states.

Embedding-layer distillation is quite similar to transformer-layer distillation – it also minimizes the mean square error between the teacher embedding layer and the student embedding layer, which is also multiplied by the learnable transformation of the student network embedding layer to teachers one.

Prediction-layer distillation – not only do they want to imitate behaviours of the intermediate level, but they also distil the knowledge to fit the predictions of the teacher model. This was achieved by penalizing soft cross-entropy loss between teacher network’s and student network’s logits.

As mentioned before, the distillation consists of two stages – they distil knowledge during both pre-training and finetuning. In both parts, the original BERT is utilized as a teacher.

The goal of general distillation is to learn the rich knowledge embedded in pre-trained BERT; then we obtain general TinyBERT ready to be finetuned for downstream tasks. During this phase, the reduction of the model is quite visible, and TinyBERT performs notably worse than BERT.

However, as mentioned before, for domain-specific tasks, modern models such as BERT suffer from over-parametrization; therefore, it is possible to

achieve similar results with much smaller models. This allows them to produce competitive (when compared to BERT), finetuned TinyBERTs through task-specific distillation.

■ 2.3.10 MiniLM

With knowledge distillation proven to be a promising way to compress large models to smaller and less computationally demanding ones, MiniLM ((Wang et al. 2020)) is another distillation framework for task-agnostic Transformer based LM distillation (with others being previously mentioned DistilBERT, TinyBERT, etc.). Their approach is called deep self-attention distillation (2.5) and is a combination of various novel improvements to the distillation process of transformers. Their idea was novel in the way they mimic the self-attention modules; instead of performing layer-to-layer distillation as TinyBERT does, they focus on the last Transformer layer of the teacher model. This approach mitigates the difficulties of layer mapping and thus could be the layer number of the student model much more flexible. In addition to attention distributions (scaled dot-product of queries and keys) used in previously existing work, they introduced scaled dot-product between values in the self-attention module as the new deep self-attention knowledge. This allows the conversion of representations of different dimensions into relation matrices with the same dimensions, therefore allowing arbitrary hidden dimensions for the student model. They also introduced teacher assistant to the distillation process.

With the base idea of knowledge distillation (minimization of the difference between teacher and student layers) already introduced in previous sections, here I will focus mostly on the novel parts of the MiniLM approach. As mentioned before, their idea is three-fold – train the student by deeply mimicking the self-attention module of the teacher’s last layer, introduce transferring the relations between values in addition to performing attention distributions transfer in self-attention modules and they introduce teacher assistant, which helps the distillation when the size gap between teacher and student is large.

With the attention mechanism being a highly successful component crucial for pre-trained LMs, the transferring of self-attention distributions has been used in many transformer distillation approaches. MiniLM specifically minimizes the KL-divergence between the self-attention distributions of the teacher and student. As mentioned before, the novel thing in this part is that they use only attention maps of the teacher’s last Transformer layer. This allows more flexibility when choosing the number of layers of the student model (and also removes the necessity to find optimal student-teacher layer mapping). Self-attention value-relation transfer is another novel approach introduced by MiniLM. The value relations are computed using a multi-head scaled dot-product and the training objective is again KL-divergence with training loss being the sum of the attention distribution transfer loss and value-relation transfer loss. This allows the student to deeply mimic the teacher’s self-attention behaviour; moreover, it also allows the student to use more flexible hidden dimensions. The last novel thing introduced for

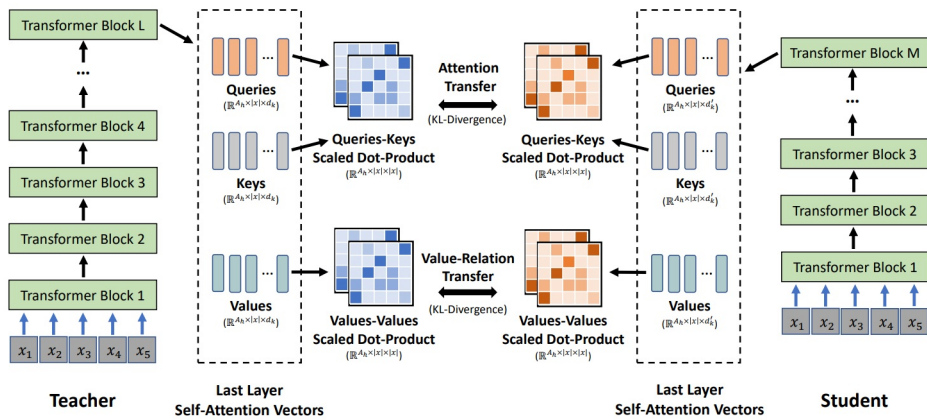


Figure 2.5: Overview of Deep Self-Attention Distillation (Wang et al. 2020)

MiniLM is teacher assistant. Shortly it is the intermediate-size student model, they first distil the teacher into the teacher assistant and then they use the assistant for training the final student model, with the aim being bridging the size gap between the teacher and the final student model.

These three improvements allowed them to achieve results quite close to the original BERT model. (They stated their results as quite close to BERT, mostly slightly better than TinyBERT and significantly better than DistilBERT – DistilBERT being the weakest of compared distilled models). (Wang et al. 2020)

2.3.11 ELECTRA

While the three previously mentioned models – DistilBERT, TinyBERT and MiniLM – focus mostly on reducing inference time, respectively its computational complexity, ELECTRA (Clark et al. 2020) is focused in yet another direction, namely increasing the effectiveness of the learning phase (from there its name "Efficiently Learning an ENcoder that Classifies Token Replacements Accurately" – ELECTRA). As pre-training nowadays requires quite a large amount of computational power, they argue that pre-training computational efficiency should also be considered. As mentioned in previous sections, models such as BERT (or XLNet, etc.) learn by utilizing masked language modelling (MLM), a method that was described more thoroughly in the section tackling BERT. ELECTRA, on the other hand, propose a novel approach to learning called "replaced token detection". The input is corrupted by replacing some tokens with samples from a proposal distribution – they utilized a small MLM that generates words in place of masked ones. Therefore the task is not only to replace the mask token validly but to identify which words were replaced. They call the learning network a discriminator, whereas the traditional MLM is a generator. This applies to the pre-training phase (similarly to other models, ELECTRA is first pre-trained and then later finetuned on specific downstream tasks).

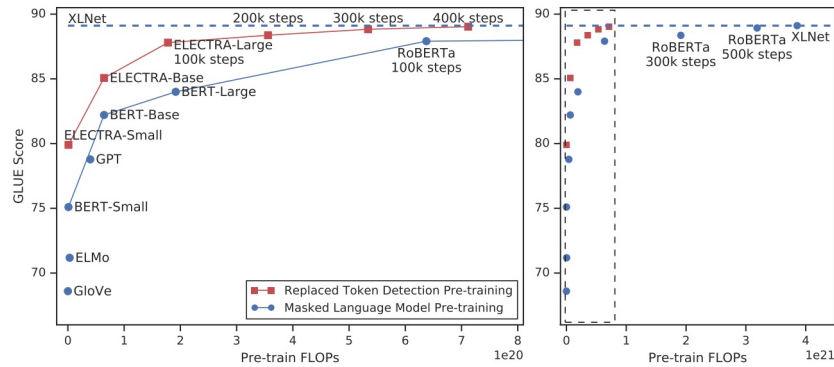


Figure 2.6: GLUE Score – Pre-train FLOPs performance evaluation of NLP models (Clark et al. 2020)

As mentioned before, they utilize two neural networks – generator G and discriminator D. Both primarily consists of encoder mapping sequence of input tokens into contextualized vector representations. Generator G generates the probability of a particular token in place of a mask token using a softmax layer. The discriminator for a given position predicts whether the token is real or comes from the generator distributions utilizing a sigmoid output layer. So the process is as follows MLM selects positions that will be replaced with mask tokens (randomly), and then we replace them with samples that the generator generated thus creating corrupted input. This corrupted input is then used for training the discriminator, which decides which token is original and which is replaced.

They mention its similarity to the training objective of GAN (generative adversarial networks), but state that there are some differences – for example when the generator generates a correct original token, that token is considered real later on. Also, they do not supply the generator with a noise vector.

They created various variants of ELECTRA (that differ by the size of training data and thus its computational complexity). For the same training computational complexity (measured by Train FLOPs) ELECTRA slightly outperformed their trained BERT model. One of their larger models then could go on par with XLNet while still being less demanding to train. While their base variant significantly outperforms XLNet and even the BERT variant of the same training computational complexity.

2.3.12 RoBERTa

Self-training NLP methods such as nowadays famous GPT or BERT brought a significant rise in performance in last years. Therefore it can be sometimes difficult to determine which of their aspects make significant contributions and which do not so much. Also due to varying training settings (size of data, length, etc.) our ability to compare various modelling approaches is limited. The goal of RoBERTa (Liu et al. 2019) is to try to optimize hyperparameters

of tuning, the training set size etc. to achieve as much improvement over standard BERT as possible. With this goal set, they make various simple modifications such as longer training, increasing the batch size, utilising a large amount of training data (and obtaining now large dataset), removing the next sentence prediction objective (thus leaving only the MLM objective alone) and also dynamically changing the masking pattern applied to training data.

As it is derived from the BERT model that was already described in previous sections. Therefore I will only describe their modifications over this "base" model and its training process. One important thing to note is that BERT is generally undertrained, thus allowing expansion of the training process without the threat of overfitting. Another note of importance is that they evaluated each individual change against the "base" variant of BERT that they reimplemented and trained.

The first change was to obtain a larger dataset – in total they acquired 5 datasets containing English-language text of various sizes and domains, totalling over 160 GB. (In comparison, the original training set is only one-tenth of the complete corpora they obtained).

BERT pre-training relies heavily on MLM. Original BERT performed masking only before the pre-training. To avoid using the same mask every epoch, the authors of RoBERTa duplicated each sentence 10 times with different masking (still being a static masking variant). Against this improvement, they compared another variant when masking is regenerated every time the sequence is fed to the model. With dynamic masking being on par or even slightly better and also more efficient, they choose to utilize static masking.

The next change was to remove Next Sentence Prediction from the pre-training procedure (and leaving only MLM) which again produced slight improvement. Also, they modified the input format instead of combining two segments (with probability 0.5 from the same document), they use one document up to the size of 512 tokens and only if the document is shorter than the rest is sampled from the next document (with an extra separating token).

Arguably easiest change was to increase the size of training batches (from a batch size of 256 sequences of original BERT paper to 8K sequences).

Also, they changed text encoding, while still utilizing Byte-Pair Encoding (BPE) – original BERT used character-level BPE vocabulary with the size of 30K (learned after preprocessing), RoBERTa employed larger BPE vocabulary with 50K subword units (without preprocessing). This led to cca 15M additional parameters when used for base BERT.

Combining mentioned modifications leads to a Robustly optimized BERT approach (RoBERTa). They claim that this model, when trained over all 160GB of data and for 500K steps, overperforms other compared models (large variants of both BERT and XLNet) measured with SQuAD, GLUE and RACE benchmarks.

As a conclusion of the section about RoBERTa, it is yet again important to note that the most important thing it shows us is that there is still a great

amount of unused potential in already existing models.

■ 2.3.13 DistilRoBERTa

DistilRoBERTa is a model distilled from RoBERTa in the same way DistilBERT is distilled from BERT (process described in 2.3.8). Therefore as it need not be described again, please refer to the corresponding sections (about DistilBERT (2.3.8) for the distillation part and about RoBERTa (2.3.12) for differences between RoBERTa and BERT).

Chapter 3

Metrics

In the following chapter, I will briefly introduce the metrics that will be later used to evaluate the quality of the results of the document retrieval tasks which will be performed during the experimental part of this thesis.

3.1 Precision

Precision (P) equals the fraction of relevant documents in the retrieved documents from the query.

$$\frac{|\text{relevant retrieved documents}|}{|\text{retrieved documents}|} \quad (3.1)$$

3.1.1 Precision@k

Precision@k is calculated nearly the same way as base precision, with the only difference being that we limit ourselves to top k retrieved documents. Thus resulting in an equation

$$\frac{|\text{relevant retrieved documents}|}{|\text{retrieved documents}|} \quad (3.2)$$

where

$$|\text{retrieved documents}| = k \quad (3.3)$$

Precision@k is oftentimes called R-precision. Although this score is more relevant for situations, where we have multiple relevant documents (which is not exactly our case, as we have mostly one, only for a few cases, there are more than one, but still it is a small number), I will still include it, as at least for lower k it could still prove useful.

3.2 Recall

Recall (R) is the friction of successfully retrieved relevant documents from all the relevant documents.

$$\frac{|\text{relevant retrieved documents}|}{|\text{relevant documents}|} \quad (3.4)$$

It is one of the easiest scores to use (alongside precision) and interpret, but still quite useful.

3.2.1 Recall@k

Recall@k is computed nearly the same way as base recall, the only difference is, that we limit the returned list of documents to top k results.

3.3 Mean Reciprocal Rank

For a single query, MRR is equal to $RR = \text{rank}_i^{-1}$, where rank_i is the rank of the highest-ranked relevant document retrieved for the query i . For a sample of queries Q , it holds that

$$\text{MRR} = \frac{1}{|Q|} \cdot \sum_{n=1}^{|Q|} \frac{1}{\text{rank}_i} \quad (3.5)$$

3.3.1 MRR@k

It is calculated similarly to MRR, but when it holds that $\text{rank}_i > k$ then $RR@k$ is equal to zero.

$$\text{MRR@k} = \frac{1}{|Q|} \cdot \sum_{n=1}^{|Q|} \frac{1}{\text{rank@k}_i} \quad (3.6)$$

$$\text{rank@k} = \begin{cases} \text{rank}_i & \text{if } \text{rank}_i \leq k \\ 0 & \text{otherwise.} \end{cases} \quad (3.7)$$

This score is particularly useful for our case, as we ideally need to find at least one relevant document (already completed an evaluation of a potential fake claim), and we can suppose, that the researchers from fake news fighting organizations will go through potentially relevant documents in top-down manner sorted by rank, therefore we want him to arrive at earliest possible to first relevant claim.

3.4 Average Precision @ k

Again @k means we are taking into account only k top-ranked documents.

$$\text{AP@k} = \frac{1}{|\text{relevant documents}|} \sum_{l=1}^k (\text{precision@l} \cdot \text{relevant@l}) \quad (3.8)$$

where relevant @ l is equal to 1 if the l -th retrieved document is relevant and 0 otherwise.

3.5 Mean Average Precision @ k

With AP defined, we could define mean AP @ k:

$$\text{MAP@k} = \frac{1}{|Q|} \sum_{n=1}^{|Q|} \text{AP}_j \quad (3.9)$$

where again Q is a sample of queries and AP_j is AP corresponding to j -th query. MAP is based on the following metrics: recall, precision, confusion matrix (matrix consisting of four attributes – true positives, true negatives, false positives, false negatives) and intersection over the union.

3.6 Normalized Discounted Cumulative Gain

NDCG is a measure of ranking quality determined by comparing the relevance of the returned items against the hypothetical ideal return. The actual return is evaluated by DCG (discounted cumulative gain)

$$\text{DCG} = \sum_{i=1}^{|\text{return}|} \frac{\text{rel}_i}{\log_2(i+1)} \quad (3.10)$$

where rel_i is the graded relevance of the result at position i . The ideal return is evaluated by IDCG, which is a DCG but calculated for the ideal return. With these definitions, we can continue to the calculation of NDCG being:

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}} \quad (3.11)$$

3.6.1 NDCG@k

It is a NDCG with the return truncated to the first k documents.

Chapter 4

Datasets

Here, used datasets will be introduced. In the first section, datasets used for training the used models (i.e. datasets that I am not using but are important to the work) are introduced, I will focus on datasets that are used alone for some of the chosen models or are significantly famous. In the second part, I will introduce datasets that I used for local finetuning and evaluation.

4.1 General use datasets

Here I will introduce datasets that I personally did not use for experiments in this work but are of importance because they are used for training the models I used.

4.1.1 SQuAD dataset

The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset of over 100 000 questions created by paid crowd workers. Their first obtained the top 10 000 articles from English Wikipedia, then sampled 536 articles randomly from this set. From each of these articles, they extracted individual paragraphs (with a length of at least 500 characters and without any tables, images etc.), resulting in a total of 23 215 extracted paragraphs (80 % of them are part of the training set, 10 % of development set and the same amount is part of test set). Then for each paragraph, an assigned crowd worker was asked to create up to 5 questions about the content (with highlighted answers in the paragraph). Also, they have obtained at least 2 additional answers for every answer from the development and test set. The dataset tests various knowledge (such as synonyms, world knowledge, syntactic variations and multiple-sentence reasoning). (Rajpurkar et al. 2016)

4.1.2 MS MARCO dataset

The dataset introduced in (Nguyen et al. 2016) (MACHINE Reading COMprehension dataset). It contains 1 010 916 anonymized questions from Bing's search query logs with human-generated answers and 182 669 completely human rewritten generated answers. It also contains 8 841 823 passages from

3 563 535 web documents Bing retrieved, providing necessary information for natural language answers. It is more than ten times larger than SQuAD. (Nguyen et al. 2016) As it contains queries and possibly relevant passages of them, it is, in fact, quite suitable for this thesis.

■ 4.1.3 Quora Duplicate Questions dataset

This dataset uses data from the popular question-answering website Quora. It contains possibly duplicate questions pair and information whether they are duplicates or not (although this labelling contains some noise). It contains over 400 000 lines of such question pairs. ¹

■ 4.1.4 NQ dataset

Natural Question corpus is another question-answering dataset created from real anonymized, aggregated queries submitted to the Google search engine. It consists of 307 373 examples with single annotation and 7830 with 5-way annotation for the training set and 7 842 5-way annotated as test data. The answers (short and long variants) are created by human annotators and based on Wikipedia pages from the top 5 search results. (Kwiatkowski et al. 2019)

■ 4.1.5 STS Benchmark

A collection of English datasets for STS task (semantic text similarity). It contains 8628 sentence pairs split into various genres – news, caption, forum and for each of these genres, there is a prepared train-dev-test split (in total, 5749 pairs for training, 1500 in dev split and 1379 for testing).

■ 4.2 Datasets used for finetuning

In this section, I will focus on datasets that were used for finetuning during the experimental part of this thesis. Both of these datasets are English only.

■ 4.2.1 CLEF

The dataset attached to CLEF2020-CheckThat! lab task 2: claim retrieval ². Formatting is inspired by TREC (Text REtrieval Conference) information retrieval campaign. It comes in the format of short verified claims (title and verified claim – vclaim) and queries (tweets), already split into test and training datasets. Note that vclaim is a shortcut for verified claim.

As an example, there is a query and verified claim example in table 4.1 (Barrón-Cedeño et al. 2020). For information about the size of the dataset refer to table 4.2.

¹Avialable at: <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>

²Avialable at: <https://github.com/sshaar/clef2020-factchecking-task2>

query		im screaming. google featured a hoax article that claims Minecraft is being shut down in 2020 pic.twitter.com/ECRqyfc8mI — Makena Kelly (@kellymakena) January 2, 2020
verified claim	title vclaim	Is Minecraft Shutting Down? The popular video game Minecraft is shutting down in 2020.

Table 4.1: CLEF query – verified claim pair example

verified claims	10375
test set queries	197
train set queries	800

Table 4.2: CLEF dataset size

4.2.2 Where Are the Facts? Searching for Fact-checked Information to Alleviate the Spread of Fake News, EMNLP 2020

The dataset attached to paper (Vo & Lee 2020), which was part of the conference on Empirical Methods in Natural Language Processing in the year 2020 (EMNLP2020). In other parts of this thesis, I will abbreviate the name of the dataset as EMNLP or EMNLP2020. It was introduced with its own Multimodal Attention Network, but I will omit it due to it also focusing on visual features (I will, for some experiments, also use the text in the images – if there is any – but not the image as a whole).

This dataset is technically consisting of two datasets, as I will explain later, but I will mostly treat them as one in terms of my approach to them (with one notable exception I will explain later), however, the results will be separated to correspond with the intention of the original paper.

Both of the datasets are based on major fact-checking organisations (PolitiFact and Snopes). After preprocessing 19 341 original tweets (18 961 of them unique) and 2845 original FC-articles were used. Then they hired native U.S. English speakers to label each document-tweet pair 1 if the article fact-checks the tweet and 0 otherwise – the final label is based on a majority vote. As the Kappa value was 0.56 the agreement between labellers was moderate – it was caused by many article-tweet pairs where the article and tweet are topically similar, but the article does not, in fact, really fact check the tweet. Also, there were many false negative pairs (Snopes pair is positive and same Politifact – as there is an intersection between datasets – is negative and vice versa) – to avoid confusion of the model, the original complete dataset was split into the two mentioned parts.(Vo & Lee 2020)

It resulted in 11 202 positive pairs in the Snopes dataset and 2 037 positive pairs for Politifact – for more information see table 4.3. On average, each poster created 1,35 tweets in the database. The topic covered is politics for Politifacts. For Snopes, politics is the biggest part (more than 40 %), but

not the only one, as it also covers other topics such as fauxtography, junk news, etc.

An example of a query in this dataset is introduced in table 4.3 with an attached image 4.1. Interestingly, this example shows that the recognition of the image text in this dataset was not perfect, as it leave out part of the text.

OriginalTweetContent_Raw	@ReedDFarman @JAD204 @bobatl What i do know is we never went to the moon..Van Allen radiation belt thousands of reals of moon footage missing and we cant go back in 2017 https://t.co/Mes5klOafQ
OriginalTweetContent_Processed	@user @user @user What i do know is we never went to the moon .. Van Allen radiation belt thousands of reals of moon footage missing and we cant go back in 2017 url
text_for_the_image	Neil Armstrongs Space Boots

Table 4.3: EMNLP query example

variant processed part/subdataset	Preprocessed	Final version	
		Snopes	Politifact
positive pairs		11 202	2 037
tweets	19 341 (18 961 unique)	11 167	2 026
articles	2 845	1 703	467

Table 4.4: EMNLP dataset size



Figure 4.1: Attached image

Chapter 5

Methodology

In this chapter, first, I will briefly explain the experiments and then describe the complete pipeline of the experimental part of my work and the implementation will be briefly introduced. In the last part, I will introduce the models used for conducting the experiments.

5.1 Introduction of experiments

One of the goals of this thesis is to evaluate selected NLP models' performance on obtained datasets to compare their suitability to the task.

To achieve this goal, first I need to obtain the results from the selected traditional baseline (Pyserni) from every variant of the dataset. Then I have to rerun the same experiments with the base variant of the neural models; there, I have to add more division of the EMNLP dataset, as also "split" and "nosplit" approaches become important.

With the results from the baseline and the base variants evaluated, I have to finetune the neural models using the obtained dataset. This means I will create a finetuned variant of the model for every possible variant of the dataset used, totalling to 2 CLEF finetuned variants and 16 EMNLP variants which every model is finetuned on.

At last, these finetuned variants are evaluated similarly to the way the baseline models were. Except I have to reduce the cross-validation of every finetuned model on every dataset, as it will lead to an excessive number of results. Thus I have decided to evaluate every finetuned model on the CLEF dataset, as it is due to its unanimity ideal choice for such a comparison. Another reason to choose the CLEF dataset is that it is significantly smaller than some parts of the EMNLP dataset and therefore is faster to evaluate.

Also, every CLEF finetuned model is evaluated on every EMNLP dataset variant, however, the EMNLP finetuned models are only evaluated on the test set belonging to the exact same version of the dataset they were finetuned on. However, as our approach to negative samples does not affect the datasets for the evaluation, this still means that two versions of every model are evaluated on one variant of the EMNLP dataset – this allows for another comparison directly within the EMNLP dataset (even more when comparing also CLEF finetuned and baseline models).

Moreover, I have decided for the smaller dataset (CLEF) to run every finetuning multiple times (10) and evaluate every of the created models, this will allow us to compare how stable the finetuning process is, and also to measure the times better (as we will evaluate basically the same model – only differing in weight – multiple times, thus allowing us to discover potential outliers), generally it will provide more statistically significant results.

5.2 Evaluation experiments – traditional models

The various base settings we could evaluate our experiments on are described in the following list:

1. CLEF dataset
2. EMNLP dataset
 - a. Politifact subdataset
 - (i) base text
 - (ii) image text
 - b. Snopes subdataset
 - (i) base text
 - (ii) image text

For the EMNLP dataset, the base and image text variants correspond to the ways of obtaining the query as introduced in 4.2.2 – i.e. the base text is when we use only the textual part of the social media post that is our query, while image text means, that also the eventual text in the attached image was used.

For the CLEF dataset, also another traditional model than Pyserini was tested – namely Elasticsearch, as it was the baseline the original paper used. (Barrón-Cedeño et al. 2020)

These 5 variants are the base settings that we evaluate traditional models on.

The most important experiment here, besides obtaining baseline results for our later comparison, is to show the importance of using the eventual text from images possibly attached to our evaluated claims.

5.3 Evaluation and finetuning experiments – neural models

This section focuses on the settings we will evaluate the neural models on that are listed in the following list:

1. CLEF dataset
2. EMNLP dataset

a. Politifact subdataset

- (i) base text
 - (A) split
 - (B) nosplit
- (ii) image text
 - (A) split
 - (B) nosplit

b. Snopes subdataset

- (i) base text
 - (A) split
 - (B) nosplit
- (ii) image text
 - (A) split
 - (B) nosplit

Clearly, we could see, that while for the CLEF dataset, the situation is still the same, there was introduced a new division of evaluation for the EMNLP dataset, namely split and nosplit.

The reason is that the neural models are heavily limited by the length of the input text and the EMNLP dataset unlike the CLEF one uses full articles from fact-checking organizations in the document collection. There are various approaches how to handle longer input text (Sun et al. 2019), however, we will limit ourselves to two main approaches.

The nosplit approach means that we simply input the text as is – i.e. the framework we use truncates it to an acceptable length and passes it to the neural network, meaning we reduce ourselves to the beginning of such documents.

The split approach ((Sun et al. 2019) calls this variant hierarchical) is based on the splitting of the documents – and even queries when needed – to sequences short enough to be mostly fully processable by the model – i.e. shorter than their maximum input length.

These additional evaluation settings will allow us to determine which of the two approaches is better suited for our task when performing the quality of the results and also the computational complexity.

It will also allow us to determine whether the rise in the number of samples used for finetuning when implementing the split approach is beneficial or not.

5.4 Finetuning experiments

While the previously explained experiments were bounded to both the evaluation and the finetuning part, in this section, we will introduce the experiment conducted during the finetuning (although naturally evaluated during the evaluation part).

As I have to obtain negative samples for the finetuning process, I have decided to use two different strategies – random sampling and static hard negatives sampling (Tabassum et al. 2022), which I will further on call the best negative samples. I have decided to sample 3 negative samples for each query. The random negative sampling is clear; for each query in the train and validation set, we randomly choose the negative samples from the collection of documents in such a way that the chosen samples will not cause any leakage (i. e. are not positive samples in any other set) and are not the positive sample for the query itself. The best negative samples strategy employs the result from our traditional model baseline. We choose the best-ranked available (i.e. under the same condition as for the random strategy).

Our goal is to evaluate, which approach is more beneficial to the neural models used within this thesis.

5.5 Pipeline

Here I will briefly describe the pipeline of my experiments (i.e. the steps and their order to obtain the presented results) and the experiments themselves.

5.5.1 CLEF dataset preprocessing

The test part contains 197 queries, and the training part originally consisted of 800 queries. To prevent any leakage between datasets, I disqualified any training query with the same gold label vclaim. Then I split 100 queries as a validation set (I have to use such a specific number and abandon the idea of a validation set of equal size to the test set because of an insufficient number of queries left to also support functional finetuning). This led to 197 queries big test set, 100 queries for the validation set and 506 training set queries.

As for the verified claims, I have also obtained negative samples for the training and validation datasets. To minimize leakage, no verified claim that is a gold label vclaim for query from one dataset was utilized as a negative sample for another dataset. The final formatting tackling verified claims is that I have concatenated the title and claim part together.

5.5.2 EMNLP dataset preprocessing

As each of the two EMNLP "subdatasets" was monolithic, I have to split it to create train, validation and test parts of the datasets. I have decided to use 70:15:15 split for train:validation:test. As there were much more queries than verified claims, I had to split it by splitting these verified claims first and then connecting them to queries which used them as a positive sample. This allowed me to prevent leakage between these parts of the datasets; on the other hand, it prevented it from following the ratio of split exactly (as there were great differences in the number of connected queries for each verified claim), which was more visible for the "smaller half" of the dataset (the Politifact one).

Also, it was necessary to handle queries with multiple positive samples (although their number was low – 10 for the Politifact part and 52 for the Snopes one) – to prevent leakage, all their positive sample articles must be in the same part of the split. I have decided to assign them all to the training set, as due to the article:query ratio, their assignment at random could affect the split greatly. This leads to the test set containing only queries with one positive sample.

For the Politifact part, it resulted in having 1500:276:261 split of training:test:validation queries, while the Snopes part was split 8470:1405:1327.

As for the CLEF dataset, also here I have to sample negative samples, a process which will be later explained in 6; even though there were negative samples provided – their number was too large (as I decided to go with 3 negative samples per test query and 1 per validation query, while they oftentimes provided close to 100 of them).

Now, after a general introduction, I will specify how the dataset is used in this thesis. As mentioned before, the dataset is divided between Politifact and Snopes part. Then I introduced the difference between taking only the text of the tweet as a query and including also the text contained in the image, which is our second division. Then there is a third division based on dealing with long texts, as mentioned before, according to (Sun et al. 2019) we could either use truncation methods (I called this variant "nosplit") or hierarchical method (split the text into sequences of suitable size and feed them to the model one by one against the query – "split"). The last division (although this applies merely when used for finetuning) is based on the approach to generating negative samples – one variant is random (take k negative samples – any document that is not a positive sample – from the entire collection – so that there won't be leakage as defined before) or best samples (take top k samples ordered by selected model – in our case Pyserini's BM-25 – that are not a positive sample).

As for the individual queries and verified claims. For queries, I have used the processed variant of tweet content for the base variant, potentially combined with the text from the attached images for the image text variant. The verified claims consist of the concatenation of the title and text of the article and also the summarised claim provided in the dataset. These queries and verified claims are then split for the split variant of evaluation as described in the corresponding section.

■ 5.5.3 Baseline

For baseline (5.7.1), the pipeline is quite straightforward. First, I had to convert the datasets to a format that Pyserini accepts (see its documentation). Then for every variant of data – note that here I do not distinguish between "split" and "nosplit" EMNLP variants, I have to create indexes first. With created indexes, I could run the queries against these indexes. Both of these parts mean I have to run a series of terminal commands (for examples of them, see attachments or visit Pyserini's documentation). After the previous parts

were completed, it means I now have ready collections of ranked retrieved documents for every query. Now, I have to simply evaluate it using the evaluation script described before in section 5.6.1.

The evaluation of the baseline is crucial for this thesis, as it will allow us to evaluate the performance of other models used – therefore, I have to evaluate the baseline on both the datasets and all their variants.

■ 5.5.4 Finetuning neural models

For both datasets, the finetuning is quite easy. With the split of the datasets, I described in the relevant chapter (4) and also both negative and positive samples prepared, I have to simply call a Python script (depending on the dataset differing in the dataset used) with the right parameters. The used hyperparameters will be described in 6.2. For the CLEF dataset and "nosplit" variant of the EMNLP dataset (thus the truncation using parts), this is very straightforward because inputted data simply consists of pairs with the sample (positive or negative) and the query attached to a corresponding query.

The situation is worse for the "split" variant of the EMNLP dataset, as there I have to run every sequence from a query against every sequence of the sample, leading to significantly higher computational cost.

Also, there represented both best negative samples and random negative samples approach for all these variants used for finetuning.

■ 5.5.5 Evaluating neural models

Here the situation is different from the baseline, as I have to rely on my own scripts to evaluate the neural models, whereas for the baseline, I simply run the commands, that Pyserini has built-in.

■ Truncated data

Again, for the truncation using experiments, simple one-stage retrieval was used (thus the entire collection is evaluated by the model). I have to evaluate every query from the test set against every document of the collection. Then created output consisting of concatenated ranked results of such evaluation. Then I again simply run the evaluation script to obtain the scores.

■ Split data

For the variant that (Sun et al. 2019) calls hierarchical (I refer to it as "split"), the situation is different. Due to the extremely high demand for the computation power used, I have to use multistage retrieval. Similarly to (Nogueira et al. 2019), the first stage consists of BM-25 (Pyserini is used in my case) retrieval, and then I take the top 250 results (note that for real usage due to still high time consumption, it might be better to use even lower number), that are against evaluated using every chosen model. The difference here is that I retrieve the results of every sequence of the query against every

sequence of the document. Similarly to (Sun et al. 2019) I used max-pooling to retrieve the results, in my case, followed by a simple sum. I choose the maximal results of every query sequence – article sequence combination for every query sequence and then sum these maxima together to retrieve the resulting evaluation of the document. For my experimental setup, I have used statical pre-retrieved results of BM-25 (obtained during the evaluation of baseline) to make the evaluation less demanding when it is not necessary. After I retrieve and order the results, I have again to run the evaluation script to obtain the scores.

The creation of split data is simple. I have split it into individual sentences (or groups of them if they are short) in such a way they do not preferably exceed the set limit of length. However, if an individual sentence is longer, it will after all be truncated. The soft limit for one sequence is 150 characters, due to the cross-encoders' need to input both query and document together – therefore accepting two concatenated sequences – and also to have some overdraft reserves. The models I used accepts 128 tokens – cca 96 words, with English word being on average cca 5 characters long, it means we could accept about 480 characters – and again for the cross-encoders without any overlap, we need to accept 300 character, which makes the reserve of approximately 180 characters reasonable to handle the most of the overlapping sequences.

■ 5.6 Implementation

In this section, I will describe the implementations that I used to conduct the experiments that I will introduce later on in chapter 6. It consists of many evaluation scripts tackling various evaluation settings (for clarity, I tend not to use one length script with many if else clauses to specify various input and processing settings, but rather to split it into smaller scripts). Also, I needed to create a few scripts for converting datasets into reasonably formatted input files; Jupyter Notebook was used for choosing the best variant of finetuned models when finetuning the same model with the same dataset and the same settings multiple times. Another important part was reworking the evaluation module and also creating scripts that process the results of the evaluation module (i.e. creating graphs, converting results to more suitable formatting etc.)

■ 5.6.1 Introducing used tools

This section will introduce all the necessary tools for running all the provided scripts (although I will here name individually every crucial part, the total list of used libraries could be found in the attachments (A.1) – containing a list of cluster modules used and the requirements of the virtual environment used there and also the requirements for conda environment for the parts run locally)

■ SentenceTransformers

The experimental part of my work relies heavily on SentenceTransformers¹ Python framework (and thus also on the Huggingface² framework and PyTorch³ library). An important note is that while some model accepts longer input sequence, the standard maximal input length is 128 tokens (cca 96 words on average, longer sequences are truncated) which is a limit I choose to preserve to have equal settings for all the models.

■ Evaluator

For evaluation, I have used a Python script⁴ delivered within (Barrón-Cedeño et al. 2020). Their script relies on an open-source Python library TREC TOOLS (specifically designed for evaluating TREC-like campaigns, which exactly suits their use of it and the formatting of their dataset). I only have to make small changes, such as the expansion of the scoring provided by TREC TOOLS and also adding the recall metric, which was not supported. Another change was to allow for calling the evaluation from directly within other scripts – i.e. change it into a Python module rather than being a Python script.

■ Convertors

I have created numerous Jupyter notebooks converting both datasets to a more convenient format. For the CLEF dataset, it contained mostly only concatenating parts of the input (or splitting for the variant with evaluating sequences of the documents and queries one by one); also, switch them to be saved as a JSON for more convenient future use.

The conversion of the EMNLP dataset was much more expressive although, with the same goals and means as mentioned above, this was due to the dataset being much more complex (also, multiple variants have to be created due to dealing with text from the attached images (to every information about the attached image, there was also a transcription of the text, if it contains any).

■ Final used scripts

For both finetuning and evaluation, I have created various scripts. Although I aimed at making the scripts as multi-use as possible, still I have to create various versions based on whether we used a two-tower model or a cross-attention one, which dataset was used (CLEF x EMNLP) both for evaluation and the finetuning and also for the EMNLP dataset there are different variants for different approaches to the data (truncation or splitting into sequences).

¹www.sbert.net

²huggingface.co

³pytorch.org

⁴available at: github.com/sshaar/clef2020-factchecking-task2

For all the scripts, SentenceTransformers was their base building block. (All variations were either introduced in 4.2 or will be introduced in 5.5.)

In every script, there is a time-measuring part that across different variants always covered the same part (meaning that it measured equivalent parts of evaluation or finetuning, although differently approached).

■ 5.7 Models used

In this section, I will introduce specific models that were used during the Experimental part of this work. While the ideological and theoretical background of each model was introduced in a corresponding subsection of 2, here I will introduce each specific model (as there may be many variants of the "same" model differing in pretraining data used, potential finetuning and sometimes even in the number of layers and thus the total number of parameters). Note that sbert.net (website for (Reimers & Gurevych 2019)) calls two-tower (siamese) models bi-encoder, whereas cross-attention models cross-encoder, which is a notation I will often follow.

■ 5.7.1 Pyserini

With the rise of pre-trained transformer-based models, there is a need for stable, replicable baseline and first-stage retrieval. Pyserini – a toolkit built as python-based inference to Anserini (developed by the same group) aims to fulfil all these roles and support "the complete research life cycle". It has the following features that are key to our needs:

- It can be used as a standalone module to generate batch retrieval runs but also can be integrated as a library for applications with interactive retrieval
- It supports sparse (BM25 using bag-of-words representation), dense (nearest-neighbour on encoded representations) and also hybrid retrieval.

(Lin et al. 2021)

I have decided to use its sparse retrieval part (BM25 – based on theoretical background introduced in 2.2.2) for many various purposes during this work. I have used Pyserini-generated results to select "best negative samples" when needed. I have also decided to use Pyserini's results (and its evaluation) as a baseline for comparison. As stated in (Yang et al. 2019) for serious comparison, there is a need for a strong baseline – a thing that various papers in the years before did not fulfil. However in recent years with the increasing popularity of Pyserini, when it becomes basically a universal baseline) and its relatively strong results (when for example compared to 5.7.2) I have decided for using it as a baseline because it solves exactly this issue better than most other traditional models I know – the performance was also confirmed by (Ullrich et al. 2023)

■ 5.7.2 Elasticsearch

Although Pyserini is a good representation of a traditional IR model, I have also decided to include Elasticsearch in the list. This is due to it being stated as a go-to baseline model in the documentation coming with the CLEF dataset⁵. Elasticsearch⁶ is a free and open search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured. It is based on Apache Lucene and written in Java, however, clients are available in many languages (such as Python, Ruby, etc.).

During experimentations, I will only use Elasticsearch as a baseline for the CLEF dataset alongside Pyserini, while the EMNLP dataset will be evaluated only by Pyserini (due to Pyserini being more standard for NLP research usage).

■ 5.7.3 all-MiniLM-L6-v2

This bi-encoder model⁷ is a 6-layer version of MiniLM model (2.3.10). Finetuned during 100 000 steps using a batch size of 1024 with sequence length limited to 128 tokens. The learning rate is set to 2e-5 and the used optimizer is AdamW. Totally finetuned on more than 1 billion sentences from various datasets such as Reddit comments, S2ORC – citation pairs, WikiAnswers – duplicate question pairs, already presented MS MARCO (4.1.2), SQuAD2.0 (4.1.1), Quora Duplicate Question (4.1.3) and NQ dataset (4.1.4). It has 384 dimensions.

■ 5.7.4 multi-qa-DistilBERT-cos-v1

Bi-encoder version⁸ of DistilBERT model (2.3.8) with 768 dimensions. Finetuned with about 215M question-answer pairs. Examples of the datasets are similar to above – WikiAnswers, MS MARCO, NQ dataset, SQuAD2.0, NQ dataset and many more.

■ 5.7.5 all-DistilRoBERTa-v1

Bi-encoder model⁹ with 768 dimensions finetuned from base DistilRoBERTa (2.3.13) during 920 000 steps using a batch size of 512 with sequence length 128, AdamW optimizer and learning rate 2e-5. They used over 1 billion sentences, and the list of datasets is similar to 5.7.3.

⁵<https://github.com/sshaar/clef2020-factchecking-task2>

⁶<https://www.elastic.co/what-is/Elasticsearch>

⁷Available at: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

⁸Available at: <https://huggingface.co/sentence-transformers/multi-qa-DistilBERT-cos-v1>

⁹Available at: <https://huggingface.co/sentence-transformers/all-DistilRoBERTa-v1>

■ 5.7.6 distiluse-base-multilingual-cased-v2

Is a multilingual variant of DistilBERT meant to be used as a siamese network trained by SBERT. This bi-encoder model¹⁰ has 512 dimensions. It is one of the original models coming from (Reimers & Gurevych 2020) being a trained variant of DistilBERT (2.3.8). It has 512 dimensions and is trained using many various datasets such as GlobalVoices (a parallel corpus of news stories), TED2020 (translated subtitles for about 4000 TED talks in over 100 languages), OpenSubtitles2018 (translated movie subtitles from opensubtitles.org) and many more.

■ 5.7.7 nq-DistilBERT-base-v1

Bi-encoder DistilBERT model¹¹ (2.3.8) trained on NQ dataset (4.1.4)

■ 5.7.8 stsb-TinyBERT-L-4

Pretrained cross-encoder model¹² based on TinyBERT (2.3.9) with 4 layers. For training STSB dataset was used (4.1.5). This means that it is not directly pre-trained for our task, I have included this model to represent the TinyBERT model with 4 layers, therefore relevant scores should be expected mostly when evaluating finetuned variants.

■ 5.7.9 quora-RoBERTa-base

Cross-encoder RoBERTa⁸(2.3.12) trained using Quora duplicate questions dataset (4.1.3),

■ 5.7.10 qnli-DistilRoBERTa-base

DistilRoBERTa cross-encoder model⁸ trained using SQuAD dataset (4.1.1).

■ 5.7.11 ms-marco-TinyBERT-L-2-v2

Another cross-encoder model¹³ representing TinyBERT models family (2.3.9), this time with lower number of layers (2) but pretrained on datasets matching our task – MS MARCO dataset (4.1.2).

■ 5.7.12 ms-marco-MiniLM-L-6-v2

Cross-encoder MiniLM (2.3.10) model⁹ with 6 layers trained using MS MARCO dataset (4.1.2).

¹⁰Avialable at: <https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2>

¹¹Avialable at: <https://www.sbert.net/docs/pretrained-models/nq-v1.html>

¹²Avialable at: https://www.sbert.net/docs/pretrained_cross-encoders.html

¹³Avialable at: <https://www.sbert.net/docs/pretrained-models/ce-msmarco.html>

■ 5.7.13 ms-marco-ELECTRA-base

Cross-encoder base ELECTRA model⁹ trained using MS MS MARCO dataset (4.1.2).

■ 5.7.14 XLM-RoBERTa-large-squad2

Multilingual RoBERTa (2.3.12) model¹⁴, I have decided to use it as a cross-encoder when using it through SentenceTransformers framework. Finetuned using SQuAD2.0 (4.1.1) with a batch size of 32 and training rate 1e-5 during 3 epochs. This model was added as the only multilingual cross-encoder (and the only multilingual model that is not a distilled model). Also even though we finetune using English datasets, it still could prove useful in multilingual settings, as shown in the zero-shot category description in BERT documentation¹⁵ (as RoBERTa is technically a variant of BERT). Note, that this model is not in the base state quite ready to be used (although it is possible) through the SentenceTransformers framework (the warning states, that some weights were not initialized, thus leading to weak results), however, this is not the case, once I finetune it.

■ 5.8 Choosing models from multiple runs

For some of the experiments, I have decided to finetune the same model multiple times to create n different variants. However, due to high computational costs, it is not possible for every part. Thus I have decided to evaluate only the best of the n variants there. For evaluating the variants, I have created a simple scoring system. The main idea was, that we want to save the time of fact-checking organizations, therefore we need to select the model that when selecting top-k documents (for lower k) returns "good" results (it is really not important, that a model is better with large k, as the fact-checker will read only first few documents). This idea is appropriately represented by recall@k score for a lower k (I have decided to use $k \in \{1, 3, 5\}$). Also, I want to prioritize results for lower k (as every researcher will check the first best result, but fewer of them will check the third or fifth. The result is the following score:

$$\text{model_variant_usefulness} = \frac{3 \cdot \text{recall}@1 + 2 \cdot \text{recall}@3 + \text{recall}@5}{6} \quad (5.1)$$

which is basically a weighted normalized recall score @ $k \in \{1, 3, 5\}$.

¹⁴Avialable at: <https://huggingface.co/deepset/XLM-RoBERTa-large-squad2>

¹⁵Source: <https://github.com/google-research/bert/blob/master/multilingual.md>

Chapter 6

Experiments

In this chapter, I will describe the settings of the experiments and the hardware used. Later I will describe the results and point out various interesting comparisons so that I will fulfil the goal of helping to select suitable models for usage within the pipeline of detecting previously fact-checked claims.

6.1 Experimental setting

For the conduction of the experiments, I have used the RCI¹ cluster. The used hardware was as follows: 32 GB RAM, 4 cores of either AMD EPYC 7543 or AMD EPYC 7763 CPU (depending on the assignment of the task in the cluster internally, which was for this particular setting out of my control – however, most of the heavy lifting is done by the GPU – which applies almost completely to the time measured section) and one Tesla A100 40GB GPU.

6.2 Finetuning setting

I have decided to use every model with the same finetuning hyperparameters (with one exception), as it is a fair way to finetune them because tuning hyperparameters for every used model would consume too much computing power and also, it is not the goal of this thesis. I used a learning rate of $1e-05$ during 4 epochs and AdamW optimizer (which is somewhat classic, as could be seen in previous sections). The only differing hyperparameter was the batch size, where I usually used a batch size of 32; however, for finetuning *XLM-RoBERTa* on EMNLP split variants, I have to scale down to a batch size of 16 – alternative might be to use gradient accumulation, that is however left for potential future work, as it was not tested during experimental part of this thesis. These settings are one of the used during distillation of (Jiao et al. 2019); it was also used for some of the tasks in the original BERT paper (Devlin et al. 2019). The reason why I have to use smaller batches during some of the variants of *XLM-RoBERTa* finetuning is simple: for the Snopes part of the "split" EMNLP dataset, the GPU memory available was insufficient (and to have consistency within experiments, I decided to have

¹rci.cvut.cz

decided to use it for entire experiment with EMNLP "split" dataset variant for this model).

6.3 Measured performance

1. Result quality scoring and their comparison for various models
2. Inference times comparison
3. Finetuning times comparison
4. Influence of approaches to the selection of negative samples
5. Influence of approach to queries and articles longer than accepted input of the model
6. Influence of adding recognized text from images attached to the queries on the quality of the results

6.4 Results

In this section, the results of the experiments will be described, and I will then conduct a brief comparison of a few interesting cases (brief in the comparison against the aggregated data).

	model name	shortcut name
Bi-Encoders	all-MiniLM-L6-v2	b-MiniLM
	'multi-qa-DistilBERT-cos-v1	b-qa-distilbert
	all-DistilRoBERTa-v1	b-distilroberta
	distiluse-base-multilingual-cased-v2	b-distiluse
	nq-DistilBERT-base-v1	b-nq-distilbert
Cross-Encoders	stsb-TinyBERT-L-4	c-stsb-TinyBERT
	quora-RoBERTa-base	c-roberta
	qnli-DistilRoBERTa-base	c-distilroberta
	ms-marco-TinyBERT-L-2-v2	c-msm-TinyBERT
	ms-marco-MiniLM-L-6-v2	c-MiniLM
	ms-marco-ELECTRA-base	c-electra
	XLM-RoBERTa-large-squad2	c-xlm-roberta
Traditional	Pyserini	Pyserini

Table 6.1: Shortcuts of model names used within graphs

6.4.1 Baseline and base neural models

This section will be used for the introduction of the results of baseline and base variants of models evaluated on both datasets.

■ Baseline – CLEF dataset results

In this section, we will evaluate the results of the traditional (baseline) models and the base variants of neural models (with no finetuning on our side).

First, I shall evaluate the results (Table A.1) of the traditional models achieved when running on the CLEF dataset test set. We could clearly see that *Pyserini* outperforms *Elasticsearch* in every measured score (except for recall@all, but this is due to *Pyserini* truncating the output, whereas *Elasticsearch* returns a list of every document in the collection ranked). This confirms that my decision to use only *Pyserini* as a traditional model baseline was reasonable – as I would again refer to (Yang et al. 2019), there is a need for a strong baseline. From our possibilities, *Pyserini* is clearly the better one. However, It is necessary to remark that both the models achieved solid results in most of the metrics, as will be later confirmed.

■ Baseline – EMNLP dataset results

With the baseline model justified, we could move to its results on the EMNLP dataset (Table A.2). We could see that for both parts of the dataset (Politifact and Snopes), the score for queries containing only tweet text is rather low. In contrast, the score rises when incorporating the text from attached images (if there is any). The scores seem justified compared to baseline scores from (Vo & Lee 2020). Also, in agreement with their results, the scores from Politifact are generally higher than the Snopes one for the first variant, whereas the Snopes results are better for the second variant.

Now, since I have already made an argument that the scores for lower @k are more important to us, for most of the following tables, I will limit the results only to them – as I have already presented the results I have available, I can focus on the important ones. Moreover, It will have minimal influence (at least for k equal to all returned documents, as, for example, the recall@all is only really influenced by the approach to the collection returned and not the ranking quality, while the precision is a nonsensical score in this setting, as It must always return 0.000).

Also, I have decided to leave out information on the MAP score from the tables regarding the evaluation using the EMNLP dataset to make the results more clear. I was able to do that, due to the MAP and MRR scores achieving equal results for every model and depth presented for the following tables. This was due to the scores being equivalent for datasets with only one gold label article per query as for such a case there holds that

$$\sum_{l=1}^k \text{precision}@k \cdot \text{relevant}@l = \frac{1}{\text{rank}_i} \quad (6.1)$$

Clearly, for such a case, there is only one relevant document and the precision for this document is equal to the reciprocal. Therefore for the MAP results, we can simply refer to the MRR scores and leave out this doubled information.

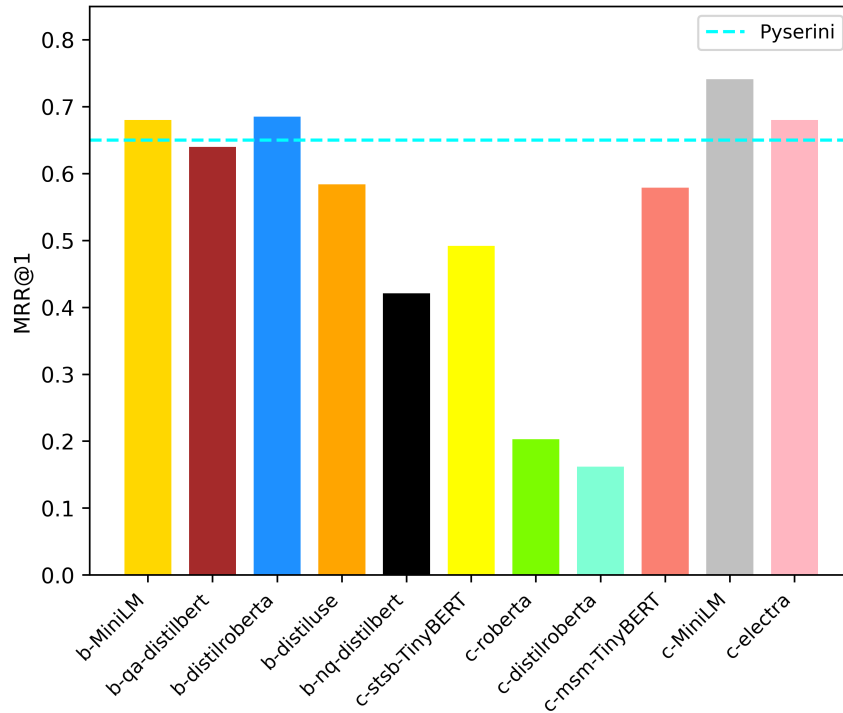


Figure 6.1: MRR@1 for base neural models and baseline evaluated on the CLEF dataset

■ Base models – CLEF data

Now, It is necessary to present the results of the base models. It will allow us to evaluate both the suitability of the individual models for the task, and also it will be a baseline for evaluation of the influence of the finetuning in the following sections.

The graphs 6.1 and, 6.2 presents the results of the MRR@ metrics for $k \in 1, 3$. First, let us compare the bi-encoders and the cross-encoders separately

For the bi-encoders, there are two models that outperform the baseline – *all-MiniLM* and *all-DistilRoBERTa*, while the rest of them are worse, with the *nq-DistilBERT* being clearly the weakest in both settings. The same conclusion is also supported by the table A.3.

For the cross-encoders, only the *ms-marco-MiniLM* outperforms the baseline in both settings, while the *ms-marco-ELECTRA* fails to do so for the MRR@3. Both *TinyBERTs* were at least able to outperform the worst-performing bi-encoder. The remaining models achieved poor results. The *XLNet-RoBERTa* results were omitted from the graphs due to it being unusable without finetuning – as SentenceTransformers framework has generated some of its weight at random as per the 5.7.14 – this is supported by the results in table A.4, where you can also explore more detailed results of the other models.

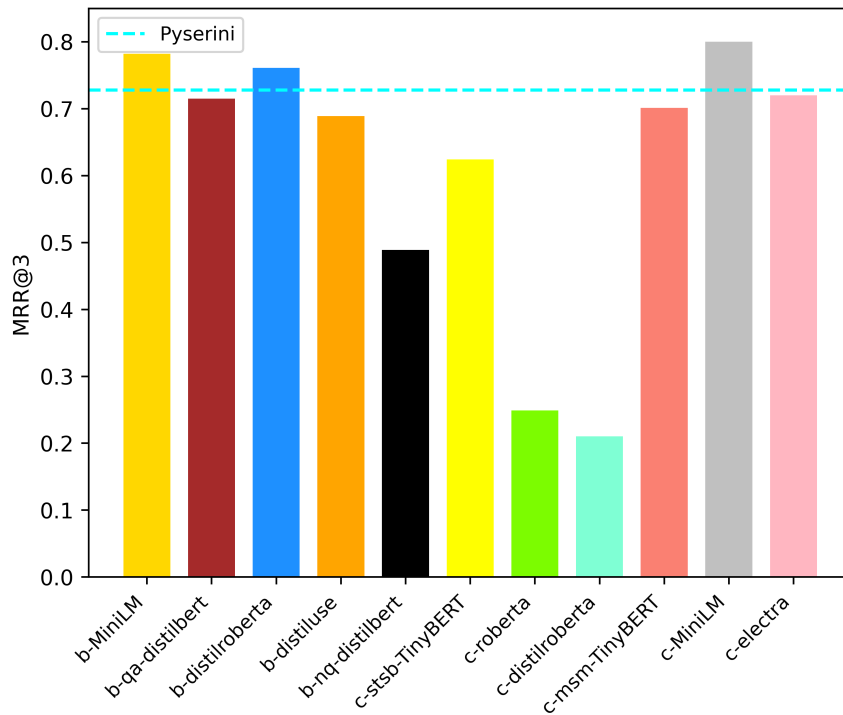


Figure 6.2: MRR@3 for base neural models and baseline evaluated on the CLEF dataset

Overall, we could see that the cross-encoders were clearly less suited for this task in their default. However also, great potential in them is visible, as *ms-marco-MiniLM* outperforms every other model mentioned for the lower @k metrics.

■ Base models – EMNLP dataset

Generally, for the evaluation using the EMNLP datasets, I will describe only parts of the results, as the total number of produced results, tables, and comparisons is too high to describe every part of it thoroughly in the scope of this thesis – therefore I will focus myself on the results of the MRR@3 score for the four different variants of this dataset. For more detailed results, please refer to the tables in the appendix – however, even there to maintain readability, I focus only on lower @k matrices and omitted the precision, as its @1 value is always equal to MRR@1, therefore it will add less information than the rest of the metrics, also I have again left out the MAP score for reasons stated in previous sections.

I will focus mostly on the influence of the split and nosplit approach, however, let us begin with a reaffirmation of a conclusion from the baseline evaluation of this dataset. The graph 6.3 confirms that the image text inclusion in the query is beneficial for the overall results. Also, it presents, that every model overperforming baseline for the image text model was able

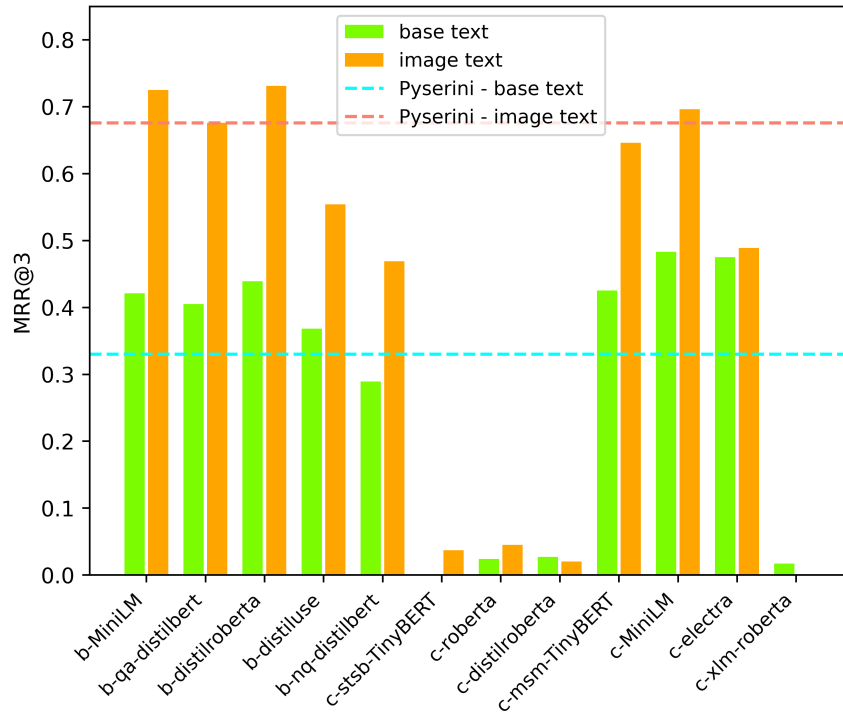


Figure 6.3: MRR@3 for base neural models and baseline evaluated on the EMNLP Politifact nosplit dataset in both base and image text variant

to also outperform it in the base text setting, however, vice versa it does not hold. It might be due to the image text providing more or better keywords overall than the base text, which will greatly benefit the traditional model.

Now, let us move to the comparisons of the influence of the split and nosplit variants of the dataset.

The graph 6.4 presents the MRR@3 scores for individual models when evaluated using the Politifact base text scenario. We can notice that the bi-encoders generally perform much better for the nosplit scenario – they overall achieved great results, and except for one model, were able to overperform the baseline. As for the cross-encoders, it clearly indicates which models are suited for the task in their base version – supporting the conclusion from the previous section. Interestingly, while for these suited models, we also achieved better results using the nosplit approach, for the rest, it was quite the opposite. The reason might be, that this approach is closer to the task the models were trained for.

For the Politifact image text scenario – graph 6.5, the situation is a little bit different as all the cross-encoders perform better for the split approach. This might be due to the query becoming longer, which allows for less of the document processed – this is not the case for bi-encoders, as they encode the query and document separately and thus allow them for the same length – this issue is generally solved by the split approach, where we arrange for the query sequence and document sequence pair to be of acceptable length

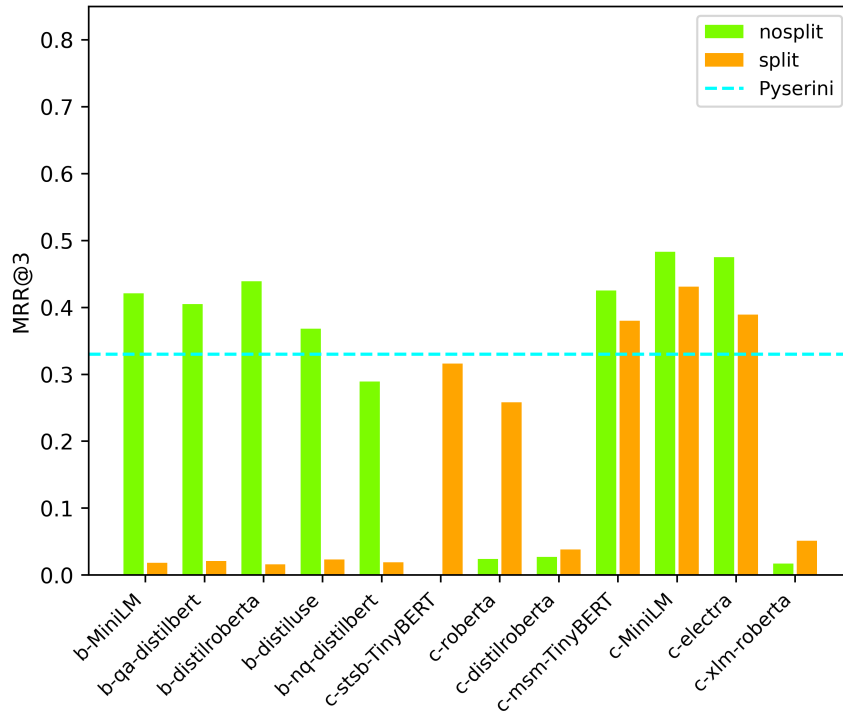


Figure 6.4: MRR@3 for base neural models and baseline evaluated on the EMNLP Politifact base text dataset in both split and nosplit variant

when together. This shows that the split approach could potentially be very useful. We could also see, that it holds for other scores than MRR@3 if we refer to corresponding tables A.6 and A.7 for bi-encoders and A.8 with A.9 for cross-encoders.

The graphs 6.6 and 6.7 present us same scenarios for the Snopes part of the dataset. There are a few notable differences. The less important one is that for the Snopes dataset, the split approach was superior for some cross-encoders even in the base text settings. The second one is more notable – most of the models were not able to equalize the results of the baseline for the Snopes image text variant. We have already concluded, that the Snopes dataset is tougher for the models than the Politifact ones and these results indicate, that it holds even more so for the neural models. As it is illustrated by tables A.10 A.11, this holds not only for the MRR@3 but also for other metrics with different depths.

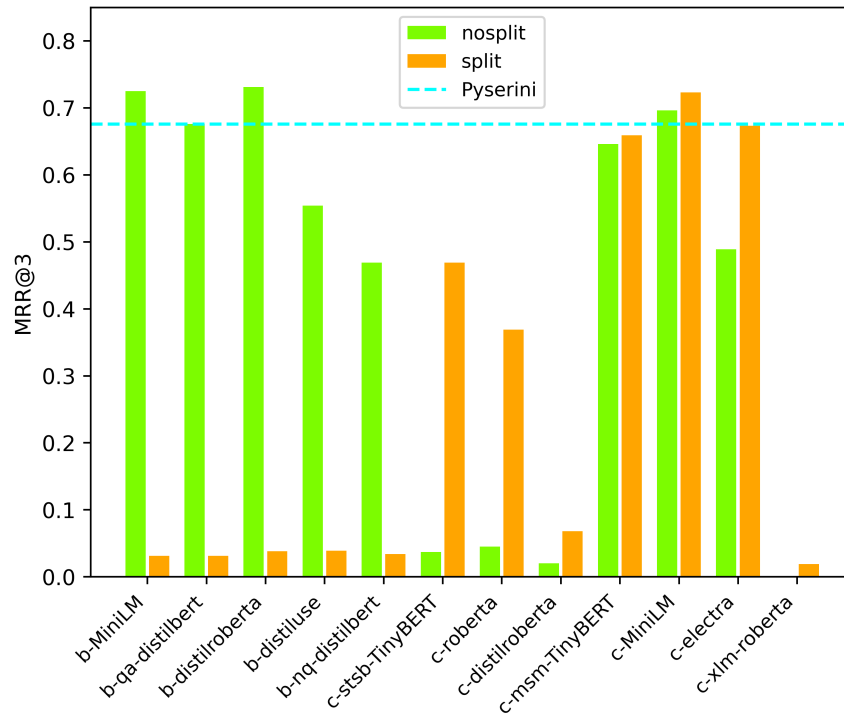


Figure 6.5: MRR@3 for base neural models and baseline evaluated on the EMNLP Politifact image text dataset in both split and nosplit variant

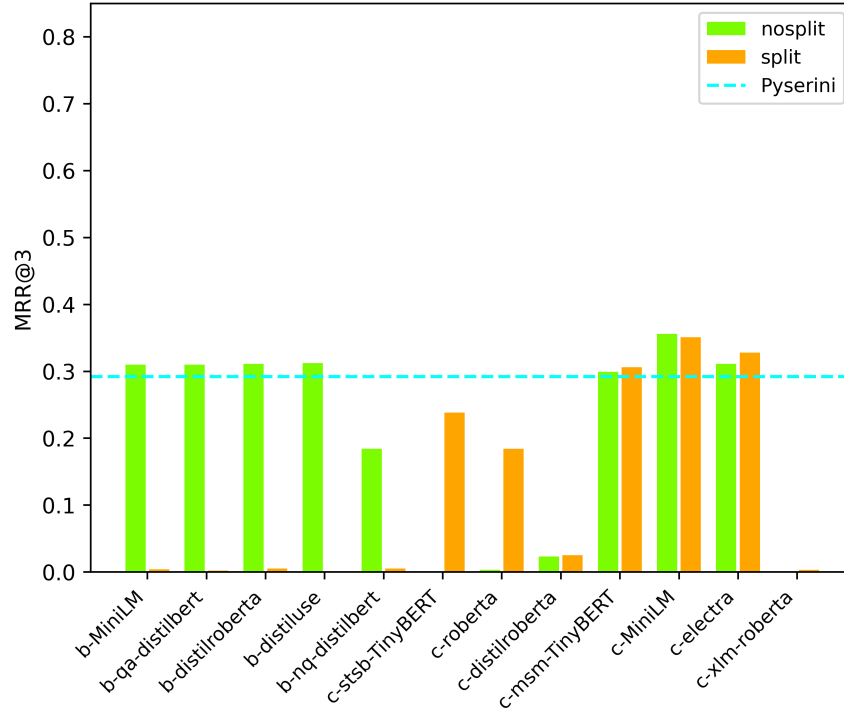


Figure 6.6: MRR@3 for base neural models and baseline evaluated on the EMNLP Snopes base text dataset in both split and nosplit variant

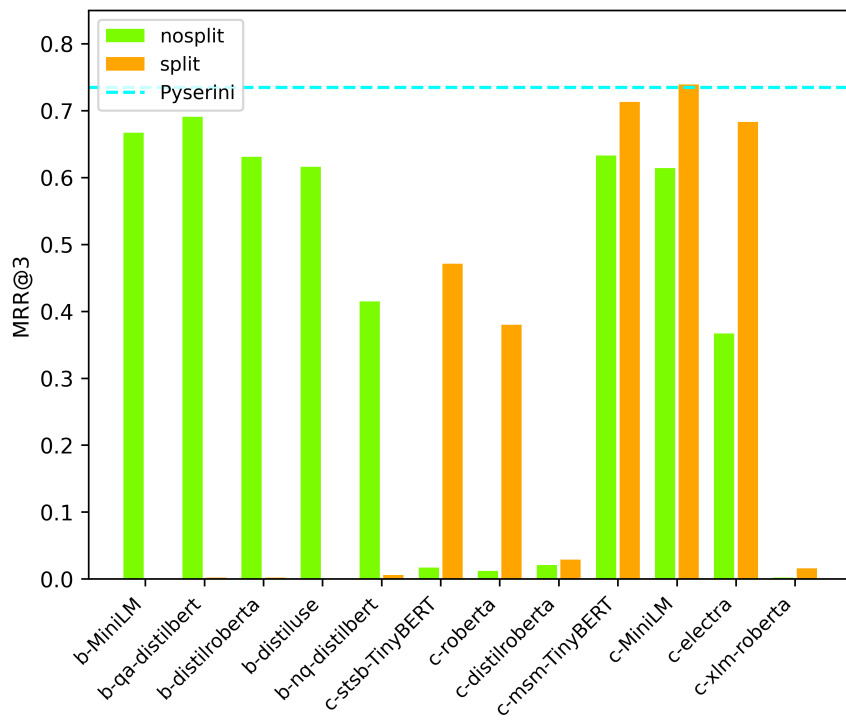


Figure 6.7: MRR@3 for base neural models and baseline evaluated on the EMNLP Snopes image text dataset in both split and nosplit variant

6.4.2 Finetuned models – CLEF dataset evaluation

This section will generally describe the results of finetuned models on the CLEF dataset. As mentioned before, we have two variants for each model based on the approach to negative samples – random and best.

CLEF finetuned

In this section, the results of CLEF finetune models on the CLEF dataset will be discussed. The results attached to it are, similarly to the base models evaluation, quite compact. Therefore, I will provide more detailed evaluation results for this section. Another fact that stands out is that due to the size of the CLEF dataset, I was able to rerun the experiments multiple times (10); thus, in this section, the results will be the mean accompanied by standard deviation, allowing us better insight into the results.

Table 6.8 exhibits that all the bi-encoders improved when evaluated on the CLEF dataset after finetuning on the CLEF dataset with either approach to negative samples. However, we could clearly observe, that the random negative samples are more beneficial to the performance of bi-encoder models. This is supported also by the other metrics results in tables A.12 and A.13, which also indicates, they lead to significantly lower standard deviation for some models. Also, they were able to surpass the baseline score, and all of them were able, to overperform their base variant, at least for the random negative sample variant. The best bi-encoders in this scenario were *all-MiniLM* and *all-DistilRoBERTa-v1*, while *nq-DistilBERT-base-v1* was arguably the weakest.

The situation is somewhat different for cross-encoders (measured by MRR@3). As shown by graph 6.9, cross-encoders benefit more from using the best negative samples in this scenario as every one of them achieved better results for the best negative samples approach than for the random negative samples approach. This is illustrated also by the fact that all the best negative samples variants overperformed the baseline; only 3 of the random negative samples variants were able to do so. All of the models, when finetuned with the best negative samples, achieved better results than their base variants.

The best-performing cross-encoders in this scenario are *ms-marco-MiniLM-L-6-v2*, *ms-marco-ELECTRA-base* and *XLM-RoBERTa-large-squad2* – which is hardly a surprise, as *ms-marco-MiniLM* achieved good results even for the base variant, and the *XLM-RoBERTa* is the largest model in the comparison. Still, I want to emphasize the *XLM-RoBERTa* results, as it was tuned from practically non-existent performance to the second-best model in this comparison, which is impressive.

For a more detailed comparison, values from tables A.14 and A.15 could be utilized – they allow us to extend the statements made above to most of the metrics and most of the depths available there.

These tables also indicate that mostly the standard deviation is higher for lower @k than for higher, which is also supported by the graphs 6.10 and 6.11 where the used metrics is MRR@1 when compared to the graphs with

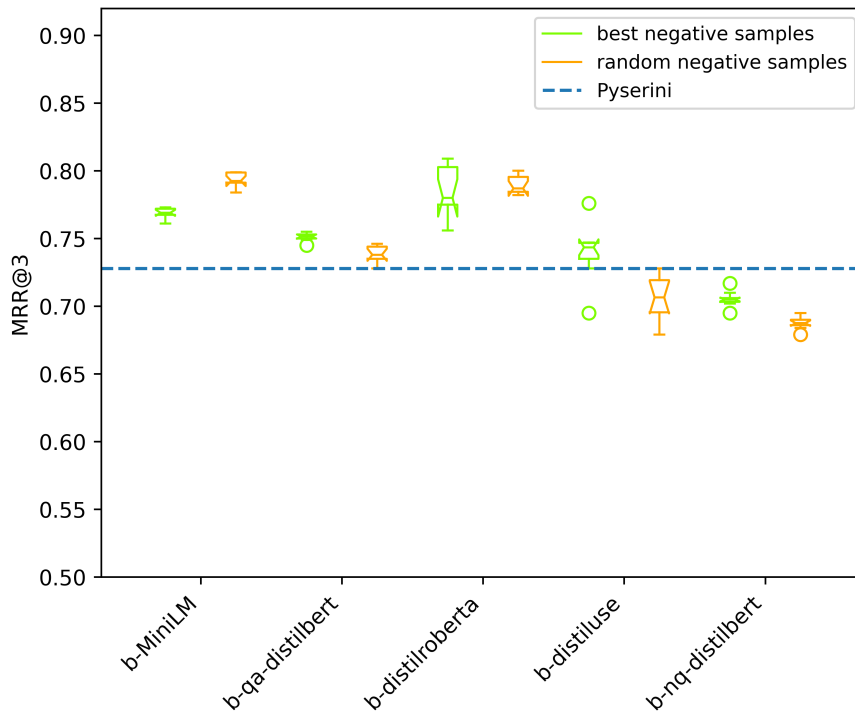


Figure 6.8: MRR@3 for bi-encoders finetuned using the CLEF dataset with both negative samples evaluated on the CLEF dataset

MRR@3 results.

Finally, the last conclusion these tables allow us to draw is that while the best negative samples exhibit slight improvement over their base version, however only for lower @k, for the higher @k the results are slightly worse while in the random negative sample scenario, models generally perform better, even at higher @k, when compared to their base variants. For the cross-encoders, the corresponding tables indicate, that they generally improve over their base variant, no matter the negative samples approach.

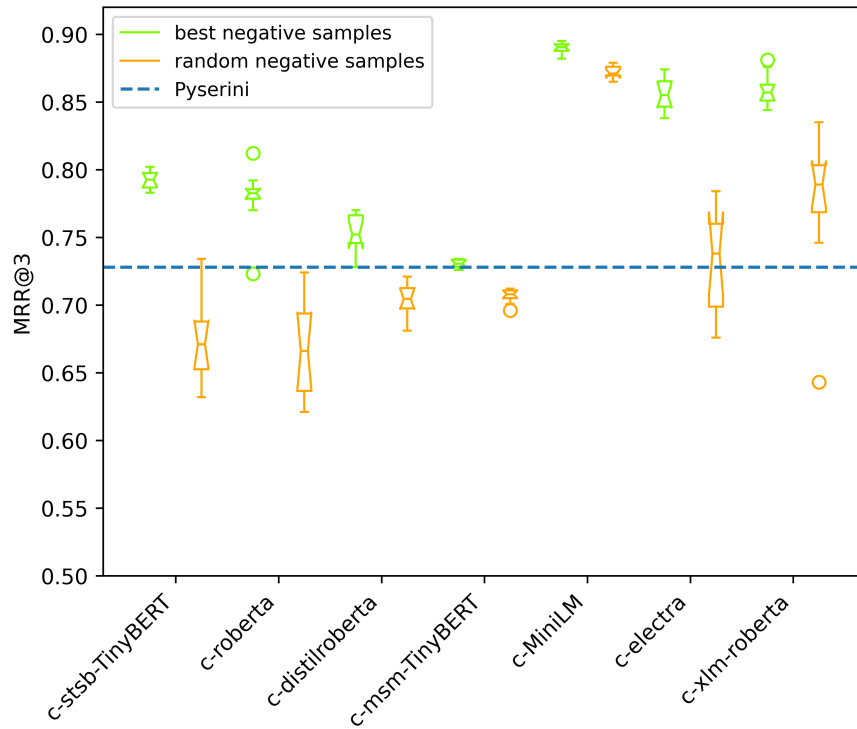


Figure 6.9: MRR@3 for cross-encoders finetuned using the CLEF dataset with both negative samples evaluated on the CLEF dataset

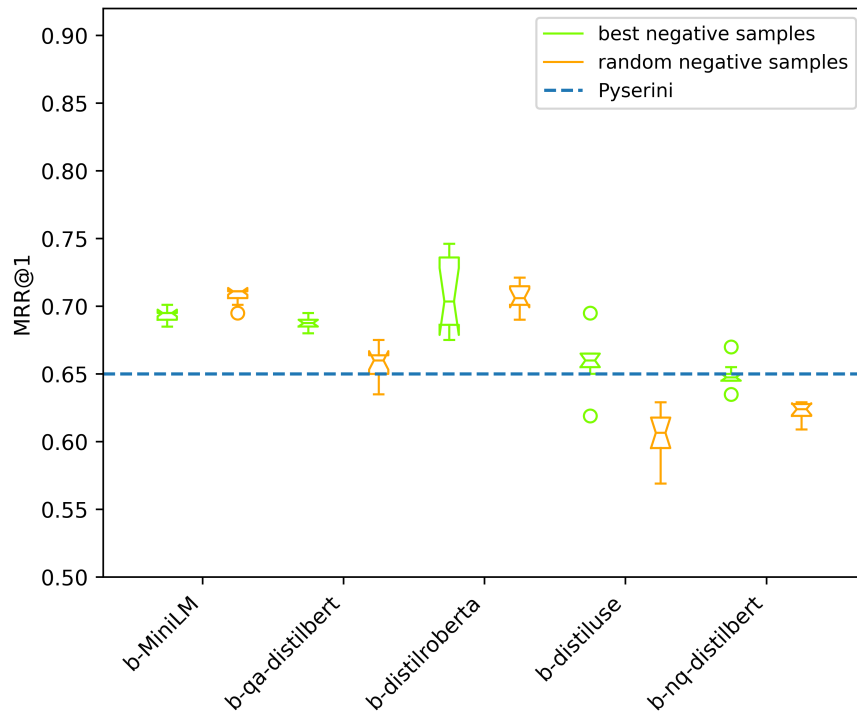


Figure 6.10: MRR@1 for bi-encoders finetuned using the CLEF dataset with both negative samples evaluated on the CLEF dataset

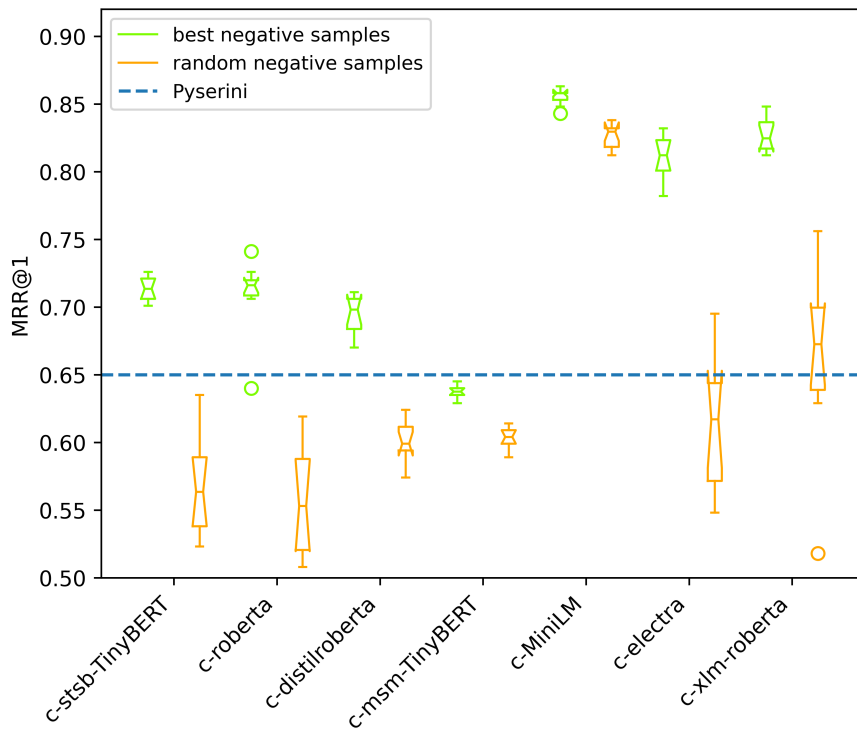


Figure 6.11: MRR@1 for cross-encoders finetuned using the CLEF dataset with both negative samples evaluated on the CLEF dataset

EMNLP finetuned

In this section, I will introduce EMNLP finetuned models on the CLEF dataset. I want to state right away that for most of them; their weak performance did not meet my expectation (I will discuss possible reasons later). However, this section will still be beneficial due to it reintroducing the EMNLP dataset for the following section.

First, I will focus on the evaluation of base and image text influence on the finetuning process and also the influence of the negative samples approach. Then, we will briefly compare results depending on the @k settings of the MRR. The last part will be on-model comparisons of various approaches, for some of the interesting models we identify in the previous part – for this part, I will use solely tables. Note that the structure of each table is easy. All scores are evaluated on the CLEF dataset; every other description specifies its finetuning settings – i.e. whether the Politifact or Snopes subdataset is used, the approach to negative samples, long sequences and also to image text.

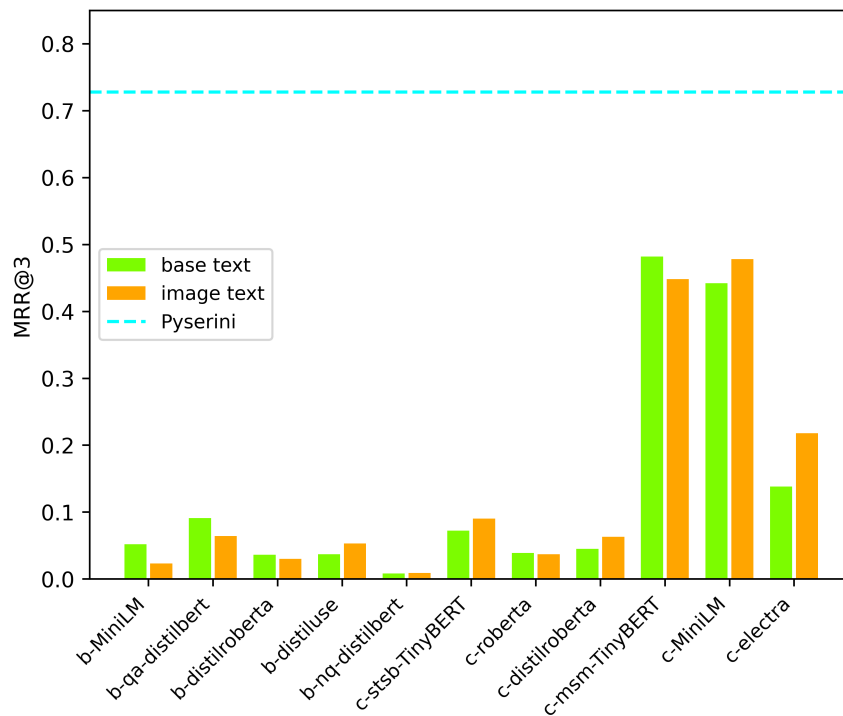


Figure 6.12: MRR@3 for neural models finetuned using EMNLP Politifact nosplit dataset with best negative samples approaches comparing the influence of image and base text on the CLEF dataset

The graphs 6.12 (best negative samples) and 6.13 (random negative samples) present us a comparison of MRR@3 scores when evaluated on the CLEF dataset after finetuning on the EMNLP with either base or image text. They exhibit, that models are stable in performing better on one of the settings no

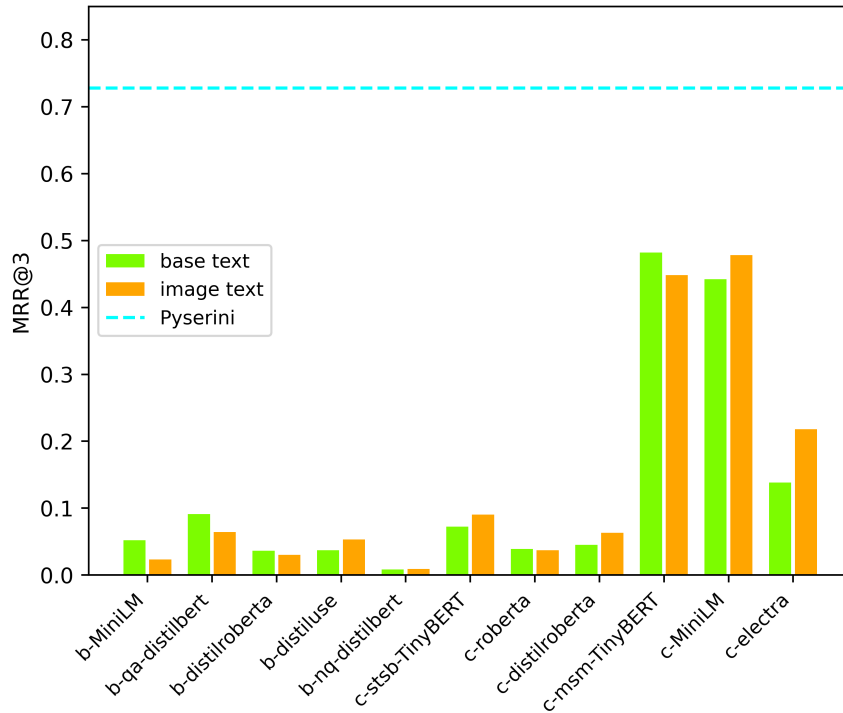


Figure 6.13: MRR@3 for neural models finetuned using EMNLP Politifact nosplit dataset with random negative samples approaches comparing the influence of image and base text on the CLEF dataset

matter the negative samples approach. Also, they present clear dominance of the cross-encoders in this setting as all of the 3 best performing models are cross-encoders – *ms-marco-TinyBERT*, *ms-marco-MiniLM* and *ms-marco-ELECTRA*. On the other hand, I have again left out the results of the *XLM-RoBERTa*, as it performed so poor there was no sense in keeping it in the comparison – we will confirm these results later using another comparison.

Now, let us evaluate the influence of the negative samples approach when combined with base text 6.14 and 6.15. For the base text variant, although there are a few models where the best negative samples are the better choice, it is not always the case; however, it is the approach which performed better for the best-performing models. A similar conclusion can be made using the image text variant, except that there is the random approach clearly beneficial to *ms-marco-ELECTRA*, which is the third-best performing model. However, generally, the finetuning using the EMNLP dataset was not as successful as the CLEF one, we should not overestimate the value of such results when comparing these approaches.

All the previously mentioned graphs present the results of the EMNLP finetuned models to be significantly worse than the baseline results.

As for the comparison for different @k values, the graphs 6.16 and 6.17 indicate, that they are quite stable – meaning that there is no model that is notably better for a specific @k than expected, they even preserves their

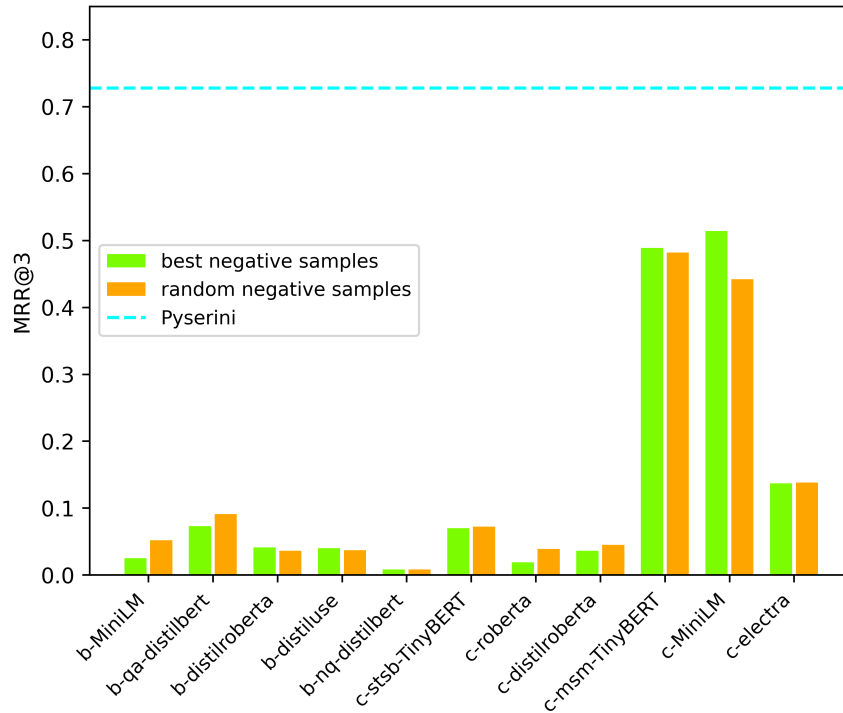


Figure 6.14: MRR@3 for neural models finetuned using EMNLP Politifact nosplit base text dataset with both negative samples approaches on the CLEF dataset

ranking when order by performance for every @k. Also the graph 6.17 supports my decision to leave out the *XLM-RoBERTa* from previous comparisons.

Now let us move to the on-model comparison part. Here, I will also evaluate the influence of the split and nosplit approach and compare the results to the baseline.

The first evaluated model is the so-far good performing bi-encoder *all-MiniLM* (Table A.16). It indicates that finetuning using the EMNLP dataset – at least with our finetuning setting – is not a success story equivalent to the finetuning using the CLEF dataset. The second evaluated bi-encoder is *all-DistilRoBERTa* which was also, in the previous settings, quite a strong model. The table A.17 indicates that it also did not succeed in any finetuning settings. The only interesting fact about these two bi-encoders is that both resulted in their respective smallest degradation in different finetuning settings. This was probably caused due to the differences in the models. An important conclusion is that the split variant results were significantly weaker for the Politifact subdataset finetuning, and that both Snopes variants (split and nosplit) were also significantly weaker than Politifact nosplit version. This leads us to the conclusion, that one of the problems here was overfitting, as all the mentioned contains a significantly larger number of training samples than the Politifact nosplit subdataset.

Now let us individually assess the results of two chosen best-performing

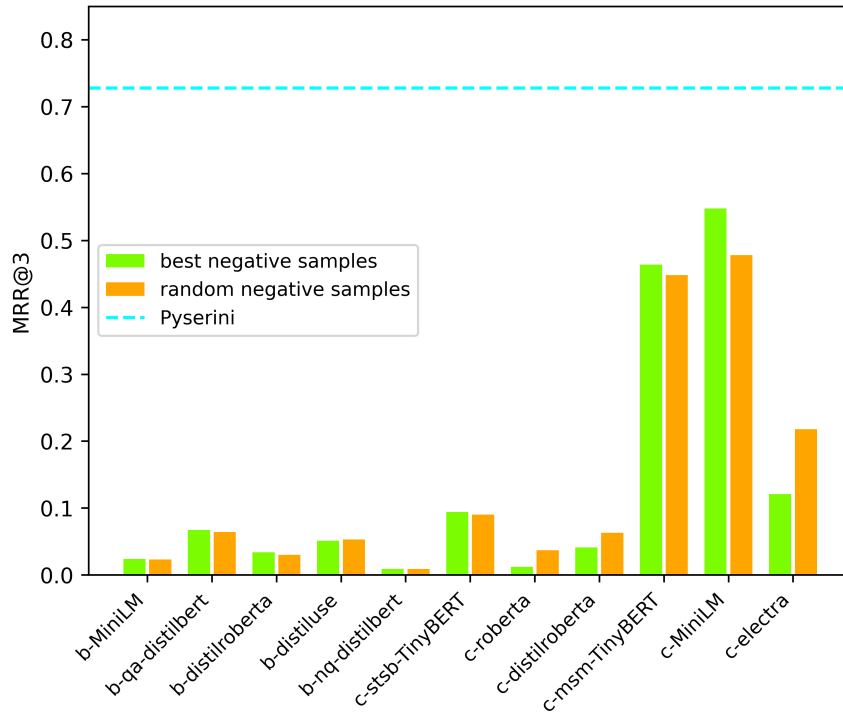


Figure 6.15: MRR@3 for neural models finetuned using EMNLP Politifact nosplit image text dataset with both negative samples approaches on the CLEF dataset

cross-encoders:

Table A.18 presents us with the results of various finetuned variants of the *ms-marco-TinyBERT* model. In comparison to bi-encoder, this model is retaining some of its performance even after finetuning, even though it is weaker than the base variant. Although, this is not mostly the case for the split variant of finetuning. Another interesting fact is that it achieved far better results for the Snopes nosplit subdataset than the previously mentioned bi-encoders. This support my conclusion about overfitting, as the split versions contain by an order of magnitude greater number of samples – even the Politifact split is significantly larger than Snopes nosplit although when the same variant is used, Snopes subdataset is generally larger.

The last introduced model – cross-encoder *ms-marco-MiniLM* (A.19) is, in general, similar in behaviour to the *ms-marco-TinyBERT* model introduced previously for this scenario. An interesting and novel fact is that the best-finetuned variant slightly overperforms the base variant, proving the potential of the EMNLP dataset for finetuning, when choosing the right finetuning setting.

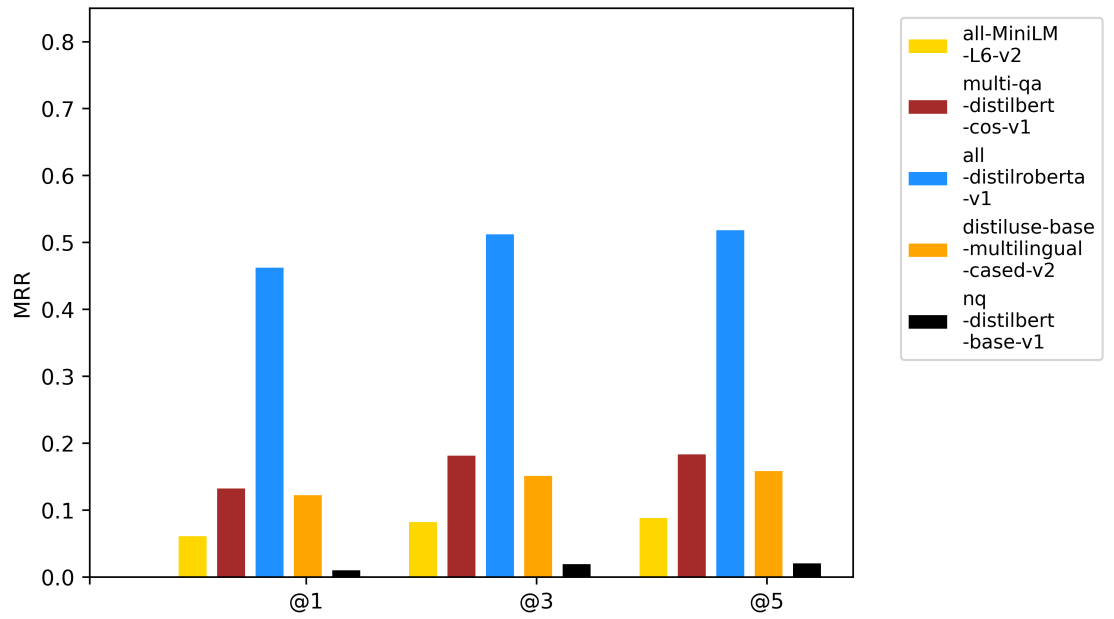


Figure 6.16: MRR for bi-encoders finetuned using the EMNLP Politifact nosplit dataset with best negative samples and only base text query evaluated on the CLEF dataset

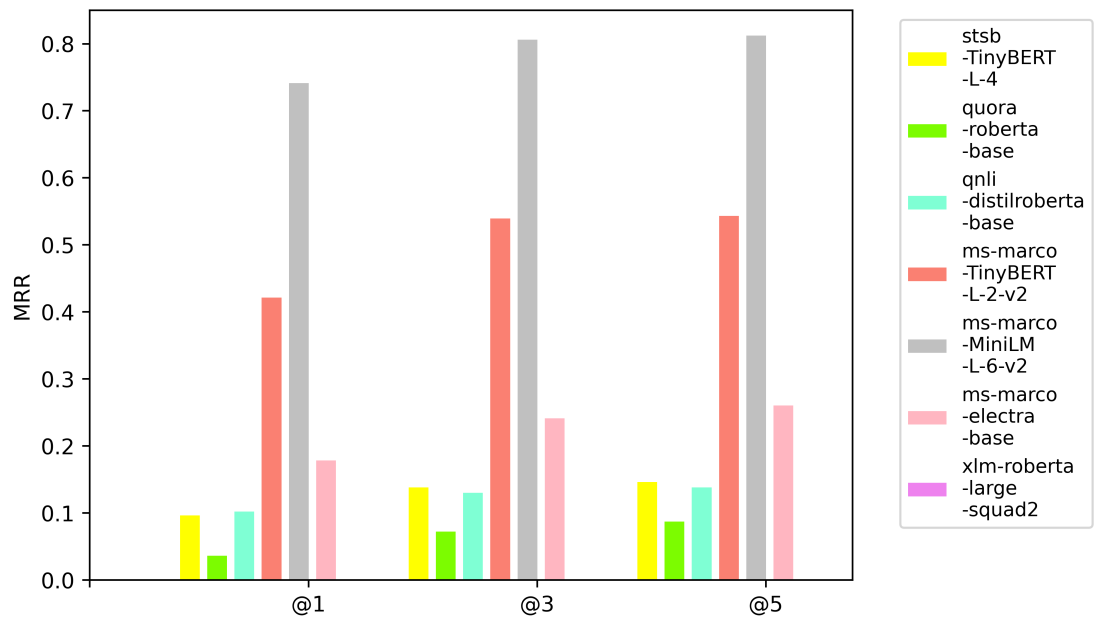


Figure 6.17: MRR for bi-encoders finetuned using the EMNLP Politifact nosplit dataset with best negative samples and image text query evaluated on the CLEF dataset

■ Conclusion

The CLEF dataset evaluation allows us to draw several conclusions.

First, the finetuning process can greatly aid the performance of our task in certain scenarios, however, it is not automatically guaranteed as per the results of the EMNLP finetuned models.

Another fact is that the best-performing cross-encoders outperform every introduced bi-encoder. It is an expectable result due to their nature. Interestingly, the cross-encoders performed better when finetuning in the best negative samples scenario, whereas the bi-encoders were the opposite way.

The CLEF finetuned part also allowed us to identify the best-performing model. For the bi-encoders they were clearly *all-MiniLM* and *all-DistilRoBERTa*. For the cross-encoders family, many neural models achieved great results compared to the bi-encoders and the baseline, however, the best performing one was certainly *ms-marco-MiniLM*.

Also, for both bi-encoders and cross-encoders, the multilingual models were able to achieve strong results. The *XLM-RoBERTa* was actually the second-best performing cross-encoder. As mentioned in the 5.7.14 section, even though we finetuned and evaluated in the English language setting, it will still prove useful even for other languages, thus creating great potential for future use of *XLM-RoBERTa*.

Generally, it also showed that for a specific task, the distilled models could perform on par with the standard ones.

■ 6.4.3 Finetuned models – EMNLP dataset evaluation

In this section, I will evaluate the results of finetuned models on the EMNLP variants.

■ EMNLP finetuned

As mentioned before, the EMNLP finetuned models achieved weak results and in this section, I will briefly support this statement by introducing the results of the evaluation on the test set corresponding to the dataset they were finetuned on – i.e. model variant finetuned on the Politifact – image text – best negative samples scenario is evaluated again on the Politifact – image text scenario. Therefore the only directly comparable results are these evaluated for the same variant of the dataset (regardless of negative samples approach). Due to the weak performances, I will compare the performances in this section only using the recall metrics, as it is sufficient in this case – with recall mostly close to zero, no other metrics could report considerably different results.

As the example of the performance of bi-encoder model in this situation, I choose to introduce the results of *all-MiniLM* (A.20) due to its good performance in other scenarios. The conclusion here is simple; the models perform very poorly – even the variant that stood out for the CLEF dataset evaluation does not stand out of line. It might be surprising, as generally, we

could expect better results for the dataset the model was finetuned on than for some other. As even though we used separated train and set parts of the dataset with no leakage, they still are similar in format. At least the EMNLP dataset consists of full articles, while the CLEF dataset is somewhat reduced.

The second and last model evaluated in this section is *ms-marco-MiniLM*. The table (A.21) shows us that it is quite similar to the introduced bi-encoder results; therefore it only confirms my conclusions for this section and extends their validity also for the cross-encoders model family.

■ CLEF finetuned

In this section, I will evaluate the results of CLEF finetuned on the EMNLP dataset. It will allow us to affirm some facts about the approach to the evaluation, which were already visible from the "base" variants evaluation; however, I want to minimize such a part to reduce the size and focus on the eventual novel information provided by this evaluation. In the beginning, it is important to note that the variants that performed best on the CLEF dataset were chosen to be used here (as I ran the finetuning and evaluation in clef finetuned – clef evaluated scenario 10 times, I chose the best from the set of the 10 variants).

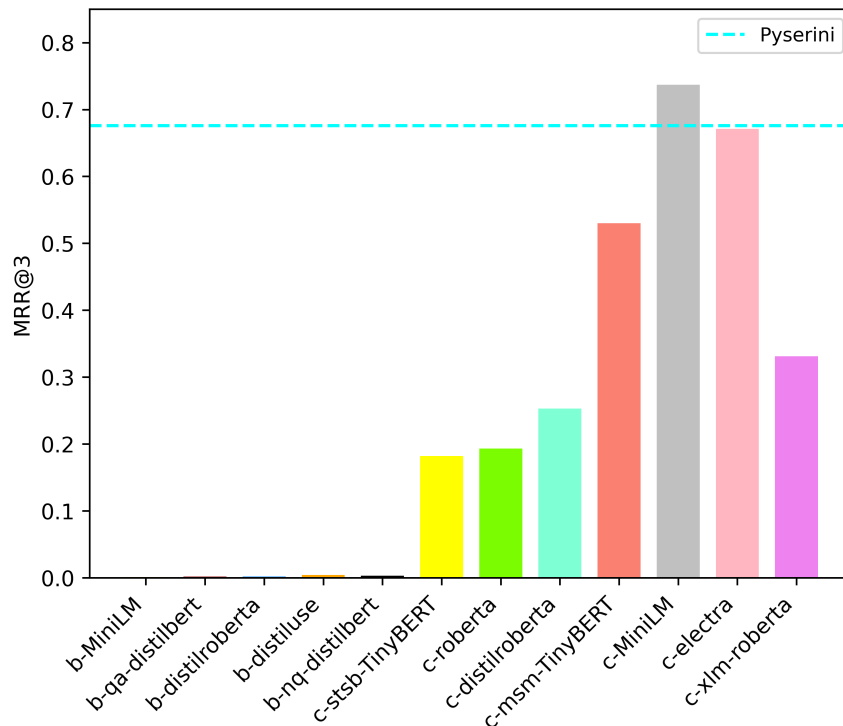


Figure 6.18: MRR@3 CLEF finetuned using best negative samples evaluated on the EMNLP Politifact split image text dataset

First, the graph 6.18 presents MRR@3 results of the CLEF finetuned with best negative samples models on the EMNLP split image text dataset. It indi-

cates that the bi-encoders are really not suited for the split scenario even after finetuning (as for their base variants, we obtained similar results). However, even for the cross-encoders, the CLEF finetuning was not beneficial. Although some of the models (*qnli-DistilRoBERTa* and *XLM-RoBERTa*) improved, other models achieved weaker results (*ms-marco-TinyBERT*, for example), while the best-performing *ms-marco-MiniLM* was really not influenced at all for MRR@3.

Now, let us focus on the MRR@3 results for different Politifact nosplit settings, whether finetuning ones – i.e. approach to negative samples, or evaluation – image or base text.

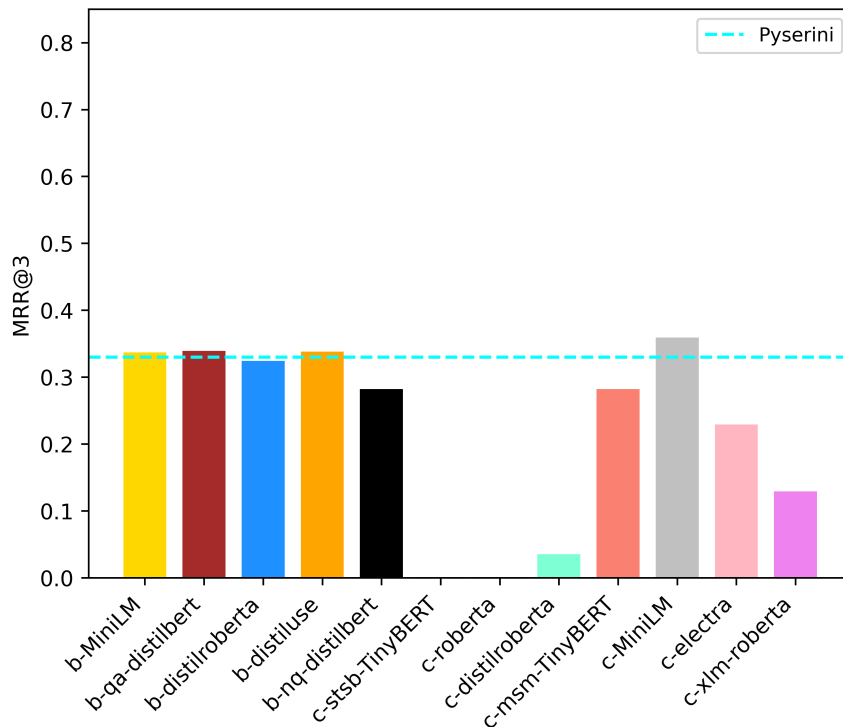


Figure 6.19: MRR@3 for models finetuned with best negative samples using CLEF dataset evaluated on the EMNLP Politifact nosplit base text dataset

First, graph 6.19 presents the results for the best negative samples tuned models when evaluating using the base text. Although few models there overperformed the baseline, we could observe they are weaker than the base variants. Also, this graph affirms that this setting overall is better suited for the bi-encoder models.

The graph 6.20 indicates, that the same conclusion also applies to the random negative variant. Surprisingly, there is not really a visible difference between the approaches to the negative samples.

As for the image text variants – graphs 6.21 and 6.22, they exhibit significant improvement for *multi-qa-DistilBERT* model as well as for *distiluse-base-multilingual*, *nq-DistilBERT-base*, and *XLM-RoBERTa* (naturally, as it was the weakest base model in the experiments due to reasons described in 5.7.14)

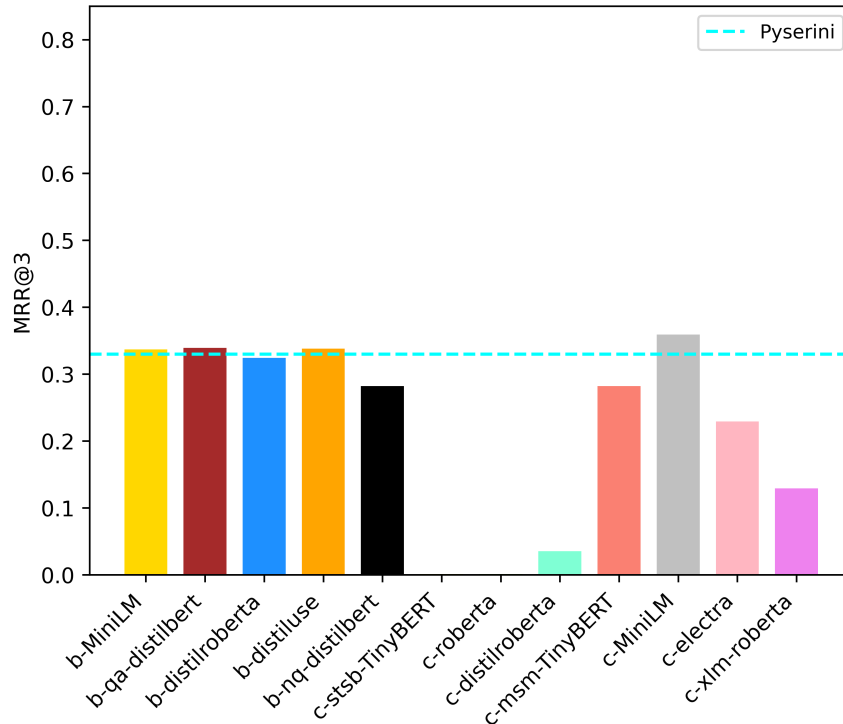


Figure 6.20: MRR@3 for models finetuned with random negative samples using CLEF dataset evaluated on the EMNLP Politifact nosplit base text dataset

when compared to their baseline, otherwise, they are really similar to the base model variants results.

The previously mentioned graphs helped us to identify the generally best-performing models in this setting – *all-MiniLM* and multi-qa as representatives of bi-encoders, *ms-marco-MiniLM* and *ms-marco-ELECTRA* for cross-encoders, and obviously the baseline traditional model Pyserini.

Their further comparison for every potential setting (in terms of subdataset, approach to split data and base or image text and also whether they were finetuned using random or best negatives samples) are presented by the graph 6.23. It confirms our previous conclusion about bi-encoders not being suited for the split approach. Other than that, it showed the models are very similar in results. It also supports the fact that the *Pyserini BM-25* model is not really suited for the base text scenario.

Now I will perform some on-model evaluation of the results to revisit some of the best-performing models using corresponding tables.

First, let me introduce the structure of the tables used in this section. We have denoted the subdataset, and then whether we use image text and also the approach to the long sequences, these settings are connected to the evaluation dataset used. The last parameter is negative samples – it connects to the finetuning setting of the models, namely whether, when finetuning on the CLEF dataset, the best negative samples were used or the random ones.

The first model is no other than the bi-encoder *all-MiniLM* (Table A.22).

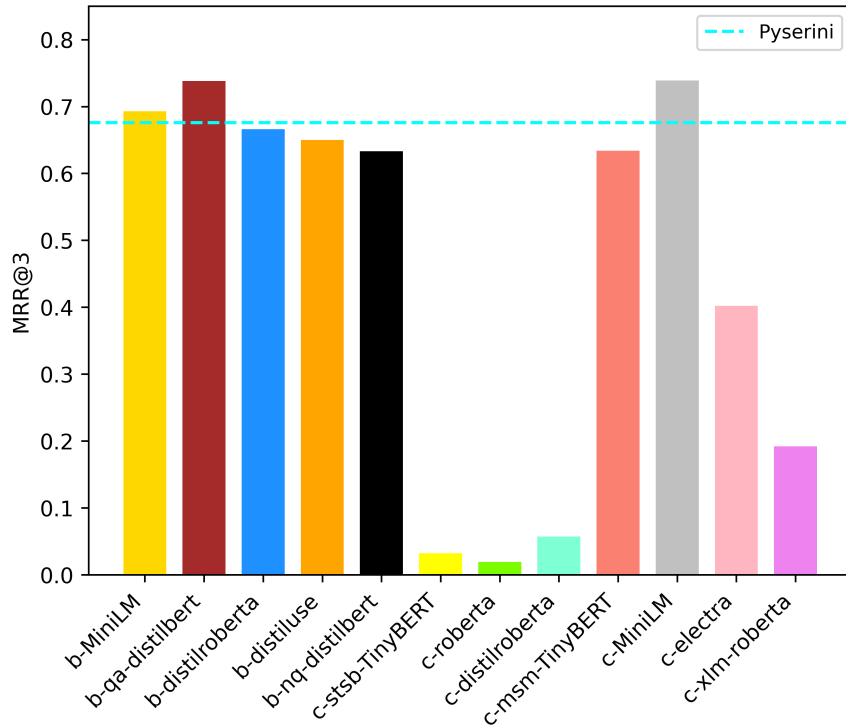


Figure 6.21: MRR@3 for models finetuned with best negative samples using CLEF dataset evaluated on the EMNLP Politifact nosplit image text dataset

For the Politifact nosplit base variant, both negative and positive sample variants performed slightly better than its base variant, while for the image text variant, only the model trained using random samples slightly improved. For the Snopes subdataset, we also achieved improvements for both random and best negative samples. The split variant evaluation for all settings stayed weak.

The next model discussed, the first cross-encoder in this section – *stsb-TinyBERT* (A.23) – was chosen not due to its overall performance but due to its peculiar results. Interestingly, it performs significantly better on the split variants than on the nosplit ones, behaviour not exhibited by any other introduced model in this section. The same way also behaved the base model (however, for this finetuned variant, it shows that the results are worse) – still, it is interesting that it was preserved even after the process of finetuning, which was arguably successful when measured by the evaluation using the CLEF dataset.

Arguably the best-performing cross-encoder in this section is *ms-marco-MiniLM* – table A.24. While it performed somewhat worse than its base variant for the split parts, it improved for the nosplit parts (results in tables A.10 and A.11).

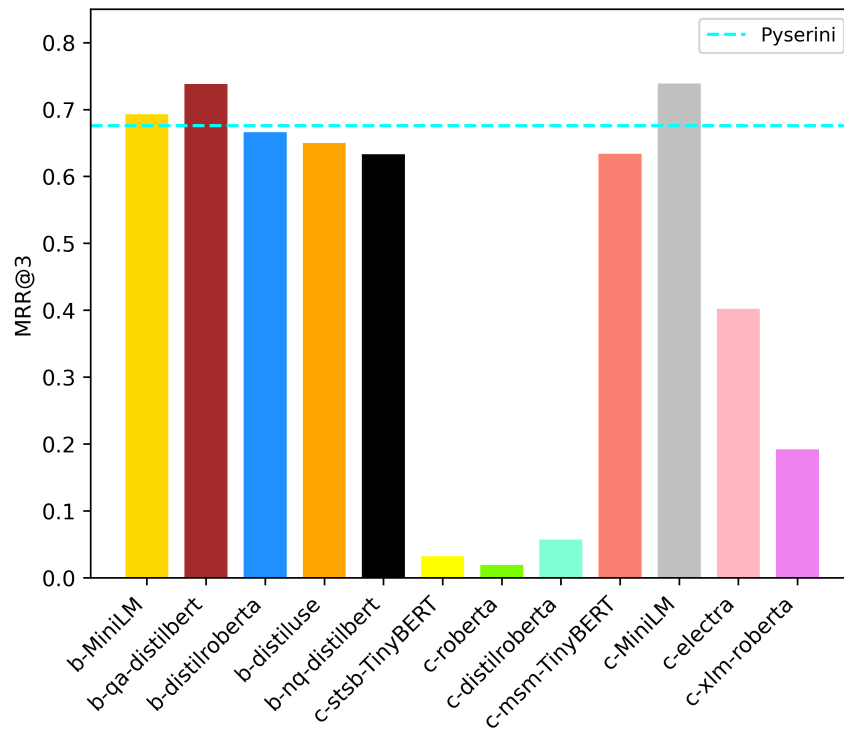


Figure 6.22: MRR@3 for models finetuned with random negative samples using CLEF dataset evaluated on the EMNLP Politifact nosplit image text dataset

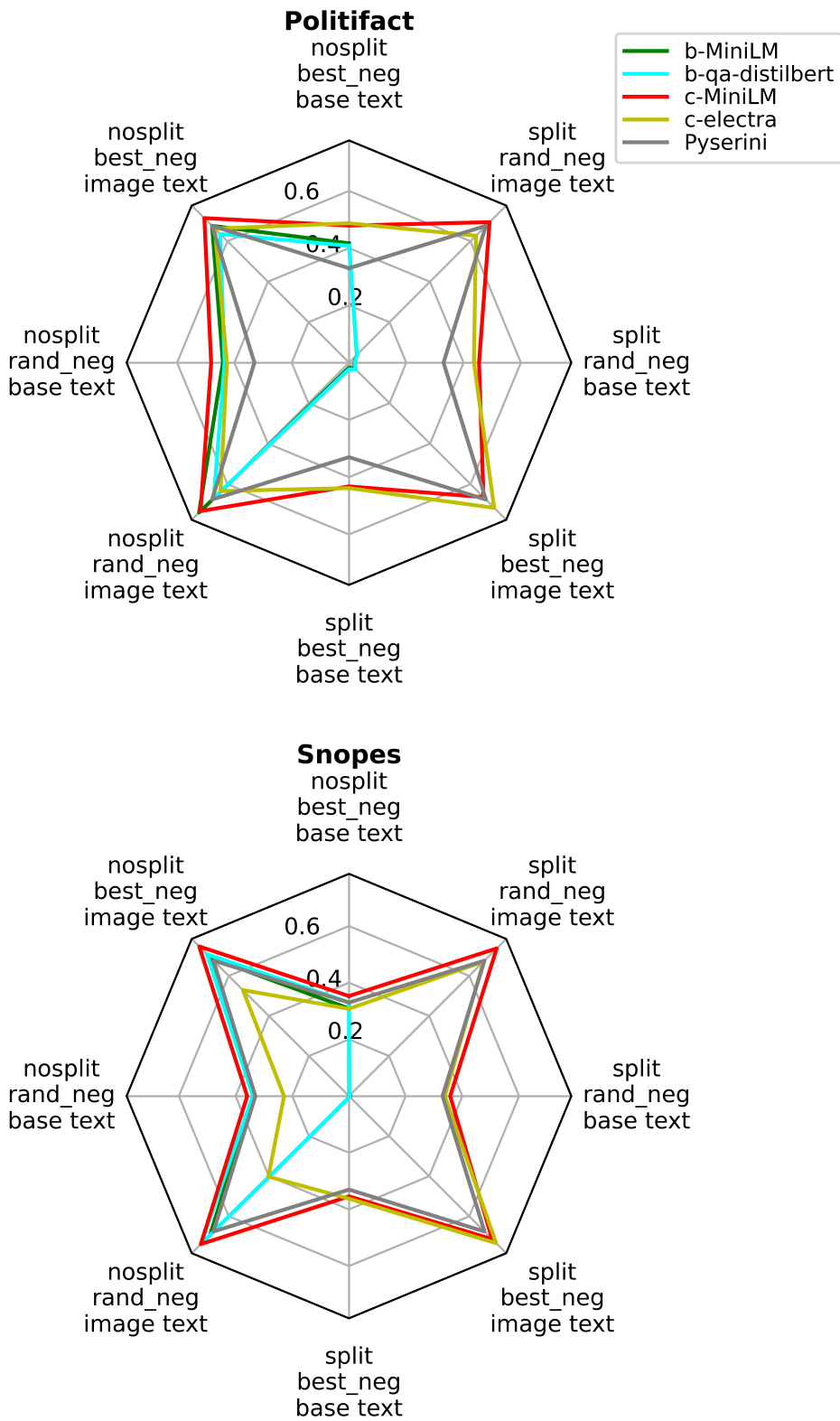


Figure 6.23: MRR@3 best CLEF finetuned models evaluated on EMNLP comparison

■ Conclusion

This section clearly affirmed a few conclusions made already in previous sections – such as the usage of image text in addition to the base text of the query (social media post) is quite crucial. Also, it outlined the potential of the split approach, although it clearly showed the nosplit one to be superior in our current settings.

In addition, it also demonstrated that models finetuned for a specific format of a document – query pairs might not transfer their performance to other formats well.

Also, this section clearly reaffirmed the weak results of most of the EMNLP finetuned models.

■ 6.4.4 Inference and finetuning times

With the results already introduced, the last part we want to examine is the computational complexity of using the models.

First, let's us examine the results of the sum of inference times of individual query-document pairs for the CLEF dataset and the Politifact subdataset of the EMNLP dataset. Note that due to the models preserving all the factors influencing inference times even after finetuning, these results will also apply to the finetuned variants of corresponding models.

The table A.25 exhibits several notable things. First, the bi-encoders are notably faster for the same scenario than the cross-encoders, even though I did not use their biggest advantage, which is the possibility of preprocessing the document collection. Clearly, when inference time is of importance, the bi-encoders are a logical choice – supported by the great performance of models such as *all-MiniLM*.

As for the cross-encoders, clearly, the most demanding one is *XLM-RoBERTa-xl*, which is expectable due to it being the largest model in the comparison. Note that while the bi-encoders were all distilled models, for the cross-encoders family, there is a mixture of distilled and standard variants of models. It allows us to compare directly *DistilRoBERTa* model with the base *RoBERTa* model and *RoBERTa-xl* model, where the *RoBERTa* is roughly 1.6 times slower, while the *RoBERTa-xl* is 3 times slower than the base *RoBERTa*. It showed the *TinyBERT*, regardless of the number of layers, to be the fastest in our comparison for cross-encoders.

For the finetuning times, table A.26 shows that it does not matter whether the model is a bi-encoder or a cross-encoder, the crucial distinction is the size of the model. Again, for the CLEF dataset results, we benefited from running each experiment multiple times in the finetuning part, allowing us also to calculate the standard deviation, which was relatively low, meaning that the runtimes of the finetuning process are for our experimental settings quite consistent.

6.5 Conclusion

In this section, I will briefly discuss the main results of the performed experiments.

We have confirmed the importance of a strong baseline model, as it was often able to compete with the more advanced neural models – again, this was not a surprising result, as we could see in (Ullrich et al. 2023) where the Pysnerini’s performance was also quite strong for some settings. With the combination of the relatively low computational cost of such a model, it most definitely has many suitable use cases even nowadays, with the rise of the neural models’ performance in this task.

Also, models that performed well on the information retrieval task in their base variants were identified – both MiniLM models presented being an example accompanied by other models.


We have also demonstrated the importance of finetuning; however, it also showed not to be a guaranteed way to success.

Again, I want to discuss the multilingual models – we have introduced 3 possible choices (two neural models and the baseline). As Pysnerini is naturally language-independent, it is always a wise choice. Another strong possibility is *XLM-RoBERTa*, where as mentioned in 5.7.14, even though we finetuned on English datasets, it will transfer its performance quite well also to another language setting. However, it is quite a demanding model both computationally and in terms of successful finetuning.

For the experiments with the negative samples approaches – we have found out that the static hard negative samples approach (we called it the best negative samples approach) could overperform the soft negative samples approach (the random negative samples approach), at least for some of the models – specifically for the cross-encoders. However, for the bi-encoders, the soft negative samples approach was better.

The EMNLP dataset evaluation affirms, that we should use any possible additive information to the base fake news claim (in our case, exhibited by extracting text from the images) as it will greatly aid the performance of all the models. Additionally, It showed the potential of the split (hierarchical) approach to longer texts, however, it also has been shown to be somewhat more dependent on the model used and dataset evaluated than for the truncation variant.

We have also evaluated the computational cost, where we could clearly see that the bi-encoders are significantly faster when inference times are measured (however finetuning process sees no evidence – it only depends on the size of the model). This difference is so significant that for many use cases, it might be wise to employ a bi-encoder even where there is a better-performing cross-encoder available.



Chapter 7

Conclusion

This thesis researches the detection of previously fact-checked claims employing a document retrieval task. First, in the theoretical part, we have introduced the task itself, then the traditional models used for such a task and later also various neural models and their usage. After a brief description of metrics used to evaluate the performances of the models introduced in the theoretical part, I have introduced the datasets used for evaluation and finetuning. One of them consisted only of query – title and claim tuple pairs, while the other one comes with a large collection of full documents created by notable fact-checking organizations, which allowed us to measure the performance of the chosen models in depth under various different scenarios.

For the experiments, I have briefly described the concrete variants of the models used, and then the results were evaluated. We have clearly demonstrated the importance of choosing a suitable and strong baseline, as one of the traditional models representative – Pyserini’s BM25 clearly outperformed the Elasticsearch model – and choosing the latter model as a baseline would have greatly corrupted the evaluation of other models. We were also able to identify neural models that perform strongly even without further finetuning; some of them even outperformed the baseline. And most importantly, we showed the usefulness of finetuning when various models were able to outperform their base variants and, thus also, the baseline traditional model.

During the evaluation of finetuned models, we have briefly described the difference between using the best negative samples (hard negative samples – (Tabassum et al. 2022)) and random negative samples. The conclusion was that while in the siamese model scenarios, the random ones achieved better results, for cross-attention models, it is the opposite.

Also, We have demonstrated that when available additional information, such as text in the attached image, is available, it is important to use it in the query.

In future work, we could focus ourselves on testing models omitted from the experimental part of this thesis, such as ColBERT. Also, there is a possibility to experiment even more with multi-stage retrieval and dealing with the long input sequences – i.e. try different variants of manual truncation (for example, the start and end of the document combinations as (Sun et al. 2019) did)

or use such models that accept longer input sequences. Another option is to focus on the finetuning process more so that we try to utilize better the parts of the acquired datasets where the finetuning achieved weak results for certain otherwise decently performing models. And even for the parts where finetuned models performed well, we could still try to optimize it even more by tuning the hyperparameters. Generally, the training process of the models is critical to focus on because, as shown by (Liu et al. 2019), some models are significantly undertrained.

I have fulfilled the goals of this thesis as I first researched the methods used for detecting previously fact-checked claims while connecting this task to the document retrieval task. I have obtained datasets usable for this task and selected appropriate models. Then I evaluated individual models under different scenarios, allowing us to distinguish models that perform well for such a task.



Bibliography

- Barrón-Cedeño, A., Elsayed, T., Nakov, P., Da San Martino, G., Hasanain, M., Suwaileh, R., Haouari, F., Babulkov, N., Hamdan, B., Nikolov, A., Shaar, S. & Ali, Z. (2020), Overview of CheckThat! 2020 — automatic identification and verification of claims in social media, *in* ‘Proceedings of the 11th International Conference of the CLEF Association: Experimental IR Meets Multilinguality, Multimodality, and Interaction’, CLEF ’2020, Thessaloniki, Greece.
- Chang, W.-C., Yu, F. X., Chang, Y.-W., Yang, Y. & Kumar, S. (2020), ‘Pre-training tasks for embedding-based large-scale retrieval’.
URL: <https://arxiv.org/abs/2002.03932>
- Christopher D. Manning, Prabhakar Raghavan, H. S. (2008), *Introduction to Information Retrieval*. ISBN: 0521865719.
URL: <http://nlp.stanford.edu/IR-book/>
- Clark, K., Luong, M., Le, Q. V. & Manning, C. D. (2020), ‘ELECTRA: pre-training text encoders as discriminators rather than generators’, *CoRR abs/2003.10555*.
URL: <https://arxiv.org/abs/2003.10555>
- Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019), BERT: Pre-training of deep bidirectional transformers for language understanding, *in* ‘Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)’, Association for Computational Linguistics, Minneapolis, Minnesota, pp. 4171–4186.
URL: <https://aclanthology.org/N19-1423>
- Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F. & Liu, Q. (2019), ‘Tinybert: Distilling BERT for natural language understanding’, *CoRR abs/1909.10351*.
URL: <http://arxiv.org/abs/1909.10351>
- Khattab, O. & Zaharia, M. (2020), Colbert: Efficient and effective passage search via contextualized late interaction over bert, *in* ‘Proceedings of the 43rd International ACM SIGIR Conference on Research and Development

- for Computational Linguistics.
URL: <https://arxiv.org/abs/2004.09813>
- Robertson, S. & Zaragoza, H. (2009), ‘The probabilistic relevance framework: Bm25 and beyond’, *Foundations and Trends in Information Retrieval* **3**, 333–389.
- Sanh, V., Debut, L., Chaumond, J. & Wolf, T. (2019), ‘Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter’, *CoRR* **abs/1910.01108**.
URL: <http://arxiv.org/abs/1910.01108>
- Shaar, S., Alam, F., Martino, G. D. S. & Nakov, P. (2021), ‘Assisting the human fact-checkers: Detecting all previously fact-checked claims in a document’, *CoRR* **abs/2109.07410**.
URL: <https://arxiv.org/abs/2109.07410>
- Shaar, S., Martino, G. D. S., Babulkov, N. & Nakov, P. (2020), ‘That is a known lie: Detecting previously fact-checked claims’.
URL: <https://arxiv.org/abs/2005.06058>
- Sun, C., Qiu, X., Xu, Y. & Huang, X. (2019), ‘How to fine-tune BERT for text classification?’, *CoRR* **abs/1905.05583**.
URL: <http://arxiv.org/abs/1905.05583>
- Tabassum, A., Wahed, M., Eldardiry, H. & Lourentzou, I. (2022), ‘Hard negative sampling strategies for contrastive representation learning’.
- Ullrich, H., Drchal, J., Rýpar, M., Vincourová, H. & Moravec, V. (2023), ‘Csfever and ctkfacts: acquiring czech data for fact verification’, *Language Resources and Evaluation* .
URL: <https://doi.org/10.1007/s10579-023-09654-3>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017), ‘Attention is all you need’.
URL: <https://arxiv.org/abs/1706.03762>
- Vo, N. & Lee, K. (2020), Where are the facts? searching for fact-checked information to alleviate the spread of fake news, in ‘Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)’, Association for Computational Linguistics, Online, pp. 7717–7731.
URL: <https://aclanthology.org/2020.emnlp-main.621>
- Wang, W., Wei, F., Dong, L., Bao, H., Yang, N. & Zhou, M. (2020), ‘Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers’, *CoRR* **abs/2002.10957**.
URL: <https://arxiv.org/abs/2002.10957>



Appendix A

Appendix

This part contains the basic information needed about the attachment of this thesis and other important facts about the project. – With this term paper comes no attachments.



A.1 Software requirements

Requirements files allowing to recreate used Python environments are part of the attachments.



A.2 Attachments structure

Here I will briefly introduce the structure of attachments and where are the most important files. For a complete overview, please refer to `structure_overview.pdf`. Note that due to the nature of the scripts, it is necessary to provide the exact same directory structure as is described there. Also, please note that in order to reduce the size of the attachments, I removed a few files necessary for the functioning of various scripts – these files needed to be regenerated (typically Pyserini output files that are necessary for multi-stage retrieval etc.), and an exhaustive summary of them is again part of `structure_overview`.

```
Senfeld_thesis_attachment
├── data_preprocessing
│   ├── clef
│   └── emnlp
├── structure_overview.pdf - a complete overview of attachment
│   └── structure
├── requirements_Pyserini.txt
├── requirements_sbert.txt
├── bash_scripts - bash scripts used for finetuning and evaluation
├── graphs - generated graphs used within this thesis and the Jupyter
│   └── notebooks used for their creation
├── latex_tables - generated .txt files with latex files used within
│   └── this thesis and the Jupyter notebooks used for their creation
├── clef2020_Pyserini
│   ├── _score - where the results are
│   ├── _in
│   ├── _queries
│   └── _how_to_run_it_clef_Pyserini.txt
├── emnlp2020_Pyserini
│   ├── _scores - where the results are
│   ├── _in
│   ├── _queries
│   └── _how_to_run_it_Pyserini.txt
├── sentenceBert_testing
│   ├── _evalbase - scripts (7) used for evaluation of the base neural
│   │   └── models
│   ├── _scores - results of the base models
│   ├── _inputs
│   └── _training - folder for experiments with finetuning
│       ├── _evalfinetuned - scripts (15) used for evaluation of the
│       │   └── finetuned neural models
│       ├── _finetune - scripts (6) used for finetuning the neural
│       │   └── models
│       ├── _input - finetuning specific input files
│       ├── _models - where the models will be stored
│       ├── _scores_bidirectional
│       └── _scores_crossencoders
```

A.3 Tables

metric		MAP				
model/@k	1	3	5	10	20	all
Elasticsearch	0.485	0.610	0.617	0.623	0.625	0.627
Pyserini	0.647	0.728	0.732	0.739	0.740	0.741
metric		P				
model/@k	1	3	5	10	20	all
Elasticsearch	0.487	0.250	0.156	0.082	0.043	0.000
Pyserini	0.650	0.274	0.169	0.089	0.045	0.000
metric		MRR				
model/@k	1	3	5	10	20	all
Elasticsearch	0.487	0.613	0.619	0.624	0.627	0.629
Pyserini	0.650	0.728	0.733	0.739	0.741	0.742
metric		NDCG				
model/@k	1	3	5	10	20	all
Elasticsearch	0.640	0.709	0.718	0.730	0.739	0.760
Pyserini	0.695	0.768	0.776	0.791	0.797	0.809
metric		R				
model/@k	1	3	5	10	20	all
Elasticsearch	0.637	0.759	0.782	0.817	0.853	1.000
Pyserini	0.693	0.817	0.838	0.883	0.904	0.980

Table A.1: Traditional models comparison – CLEF dataset

Dataset part		Politifact		Snopes	
metric/variant		text only	image text	text only	image text
MAP	1	0.283	0.576	0.260	0.700
	3	0.330	0.676	0.292	0.735
	5	0.343	0.687	0.299	0.741
	10	0.351	0.693	0.304	0.745
	20	0.355	0.696	0.306	0.747
	all	0.361	0.697	0.309	0.748
	P	1	0.283	0.576	0.260
3		0.130	0.264	0.111	0.259
5		0.090	0.168	0.072	0.161
10		0.051	0.089	0.040	0.083
20		0.025	0.046	0.021	0.043
all		0.000	0.000	0.000	0.000
MRR		1	0.283	0.576	0.260
	3	0.330	0.676	0.292	0.735
	5	0.343	0.687	0.299	0.741
	10	0.351	0.693	0.304	0.745
	20	0.355	0.696	0.306	0.747
	all	0.361	0.697	0.309	0.748
	NDCG	1	0.283	0.576	0.263
3		0.345	0.707	0.304	0.752
5		0.369	0.726	0.315	0.763
10		0.389	0.741	0.328	0.769
20		0.404	0.749	0.335	0.775
all		0.465	0.763	0.395	0.794
R		1	0.283	0.576	0.263
	3	0.391	0.793	0.333	0.786
	5	0.449	0.841	0.361	0.811
	10	0.511	0.888	0.399	0.832
	20	0.569	0.920	0.428	0.854
	all	0.946	0.996	0.872	0.981

Table A.2: Pyserini – EMNLP dataset results

metric/model		all-MiniLM -L6-v2	multi-qa -distilbert -cos-v1	all -distilroberta -v1
MAP	1	0.678	0.637	0.683
	3	0.782	0.714	0.761
	5	0.787	0.722	0.767
	10	0.791	0.729	0.772
P	1	0.680	0.640	0.685
	3	0.298	0.267	0.284
	5	0.184	0.168	0.177
	10	0.095	0.088	0.091
MRR	1	0.680	0.640	0.685
	3	0.782	0.715	0.761
	5	0.787	0.723	0.767
	10	0.791	0.730	0.772
NDCG	1	0.690	0.640	0.685
	3	0.813	0.736	0.783
	5	0.823	0.750	0.796
	10	0.833	0.766	0.804
R	1	0.688	0.637	0.683
	3	0.888	0.797	0.848
	5	0.914	0.832	0.878
	10	0.944	0.878	0.904
metric/model		distiluse-base -multilingual -cased-v2	nq -distilbert -base-v1	
MAP	1	0.581	0.419	
	3	0.686	0.486	
	5	0.690	0.494	
	10	0.697	0.496	
P	1	0.584	0.421	
	3	0.271	0.190	
	5	0.165	0.120	
	10	0.089	0.062	
MRR	1	0.584	0.421	
	3	0.689	0.489	
	5	0.691	0.496	
	10	0.699	0.499	
NDCG	1	0.584	0.426	
	3	0.719	0.510	
	5	0.725	0.523	
	10	0.744	0.529	
R	1	0.581	0.424	
	3	0.810	0.566	
	5	0.822	0.596	
	10	0.883	0.617	

Table A.3: Base variants of bidirectional models – CLEF dataset results

metric/model		stsb	quora	qnli	ms-marco
		-TinyBERT -L-4	-roberta -base	-distilroberta -base	-TinyBERT -L-2-v2
MAP	1	0.492	0.203	0.160	0.576
	3	0.624	0.249	0.207	0.701
	5	0.637	0.255	0.217	0.702
	10	0.642	0.261	0.224	0.705
P	1	0.492	0.203	0.162	0.579
	3	0.261	0.103	0.088	0.283
	5	0.169	0.068	0.061	0.171
	10	0.088	0.038	0.036	0.088
MRR	1	0.492	0.203	0.162	0.579
	3	0.624	0.249	0.210	0.701
	5	0.638	0.256	0.220	0.702
	10	0.643	0.261	0.226	0.706
NDCG	1	0.503	0.213	0.162	0.579
	3	0.669	0.268	0.222	0.739
	5	0.693	0.280	0.239	0.741
	10	0.706	0.292	0.255	0.750
R	1	0.503	0.213	0.160	0.576
	3	0.782	0.310	0.261	0.843
	5	0.840	0.338	0.302	0.848
	10	0.881	0.373	0.353	0.878
metric/model		ms-marco	ms-marco	xlm-roberta	
		-MiniLM -L-6-v2	-electra -base	-large -squad2	
MAP	1	0.739	0.678	0.000	
	3	0.798	0.717	0.000	
	5	0.804	0.721	0.000	
	10	0.806	0.723	0.000	
P	1	0.741	0.680	0.000	
	3	0.289	0.255	0.000	
	5	0.179	0.156	0.000	
	10	0.091	0.080	0.000	
MRR	1	0.741	0.680	0.000	
	3	0.800	0.720	0.000	
	5	0.806	0.724	0.000	
	10	0.808	0.726	0.000	
NDCG	1	0.746	0.685	0.000	
	3	0.818	0.733	0.000	
	5	0.829	0.739	0.000	
	10	0.835	0.745	0.000	
R	1	0.744	0.683	0.000	
	3	0.865	0.764	0.000	
	5	0.891	0.779	0.000	
	10	0.909	0.799	0.000	

Table A.4: Base variants of cross-encoder models – CLEF dataset results

metric/model		all-MiniLM -L6-v2	multi-qa -distilbert -cos-v1	all -distilroberta -v1
MRR	1	0.645	0.554	0.656
	5	0.737	0.687	0.741
NDCG	1	0.645	0.554	0.656
	5	0.772	0.733	0.772
R	1	0.645	0.554	0.656
	5	0.877	0.870	0.862
metric/model		distiluse-base -multilingual -cased-v2	nq -distilbert -base-v1	
MRR	1	0.471	0.380	
	5	0.573	0.480	
NDCG	1	0.471	0.380	
	5	0.617	0.519	
R	1	0.471	0.380	
	5	0.746	0.634	

Table A.5: Base bidirectional models – EMNLP Politifact nosplit image text dataset

metric/model		all-MiniLM -L6-v2	multi-qa -distilbert -cos-v1	all -distilroberta -v1
MRR	1	0.018	0.022	0.029
	5	0.033	0.033	0.040
NDCG	1	0.018	0.022	0.029
	5	0.038	0.038	0.044
R	1	0.018	0.022	0.029
	5	0.054	0.051	0.058
metric/model		distiluse-base -multilingual -cased-v2	nq -distilbert -base-v1	
MRR	1	0.033	0.022	
	5	0.041	0.036	
NDCG	1	0.033	0.022	
	5	0.044	0.042	
R	1	0.033	0.022	
	5	0.054	0.058	

Table A.6: Base bidirectional models – EMNLP Politifact split image text dataset

metric/model		all-MiniLM -L6-v2	multi-qa -distilbert -cos-v1	all -distilroberta -v1
MRR	1	0.594	0.641	0.567
	5	0.676	0.701	0.641
NDCG	1	0.602	0.661	0.572
	5	0.713	0.732	0.672
R	1	0.602	0.661	0.572
	5	0.804	0.793	0.752

metric/model		distiluse-base -multilingual -cased-v2	nq -distilbert -base-v1
MRR	1	0.532	0.371
	5	0.624	0.424
NDCG	1	0.544	0.383
	5	0.661	0.452
R	1	0.544	0.383
	5	0.747	0.512

Table A.7: Base bidirectional models – EMNLP Snopes split image text dataset

metric/model		stsb -TinyBERT -L-4	quora -roberta -base	qnli -distilroberta -base	ms-marco -TinyBERT -L-2-v2
MRR	1	0.029	0.033	0.014	0.569
	5	0.042	0.048	0.025	0.658
NDCG	1	0.029	0.033	0.014	0.569
	5	0.048	0.056	0.031	0.693
R	1	0.029	0.033	0.014	0.569
	5	0.069	0.080	0.047	0.797

metric/model		ms-marco -MiniLM -L-6-v2	ms-marco -electra -base	xlm-roberta -large -squad2
MRR	1	0.634	0.446	0.000
	5	0.710	0.503	0.001
NDCG	1	0.634	0.446	0.000
	5	0.742	0.528	0.002
R	1	0.634	0.446	0.000
	5	0.837	0.605	0.004

Table A.8: Base cross-encoder models – EMNLP Politifact nosplit image text dataset

metric/model		stsb -TinyBERT -L-4	quora -roberta -base	qnli -distilroberta -base	ms-marco -TinyBERT -L-2-v2
MRR	1	0.388	0.326	0.051	0.594
	5	0.485	0.381	0.074	0.673
NDCG	1	0.388	0.326	0.051	0.598
	5	0.523	0.405	0.086	0.705
R	1	0.388	0.326	0.051	0.598
	5	0.638	0.478	0.120	0.797

metric/model		ms-marco -MiniLM -L-6-v2	ms-marco -electra -base	xlm-roberta -large -squad2
MRR	1	0.649	0.601	0.007
	5	0.730	0.686	0.024
NDCG	1	0.649	0.605	0.007
	5	0.758	0.719	0.031
R	1	0.649	0.605	0.007
	5	0.841	0.812	0.054

Table A.9: Base cross-encoder models – EMNLP Politifact split image text dataset

metric/model		stsb -TinyBERT -L-4	quora -roberta -base	qnli -distilroberta -base	ms-marco -TinyBERT -L-2-v2
MRR	1	0.013	0.010	0.011	0.577
	5	0.020	0.014	0.023	0.642
NDCG	1	0.013	0.010	0.011	0.591
	5	0.024	0.017	0.028	0.674
R	1	0.013	0.010	0.011	0.591
	5	0.036	0.024	0.045	0.743

metric/model		ms-marco -MiniLM -L-6-v2	ms-marco -electra -base	xlm-roberta -large -squad2
MRR	1	0.564	0.336	0.001
	5	0.623	0.374	0.003
NDCG	1	0.575	0.342	0.001
	5	0.652	0.393	0.003
R	1	0.575	0.342	0.001
	5	0.718	0.439	0.005

Table A.10: Base cross-encoder models – EMNLP Snopes nosplit image text dataset

metric/model		stsb -TinyBERT -L-4	quora -roberta -base	qnli -distilroberta -base	ms-marco -TinyBERT -L-2-v2
MRR	1	0.394	0.342	0.019	0.660
	5	0.494	0.390	0.033	0.720
NDCG	1	0.396	0.342	0.021	0.683
	5	0.540	0.410	0.040	0.751
R	1	0.396	0.342	0.021	0.683
	5	0.670	0.469	0.058	0.809
metric/model		ms-marco -MiniLM -L-6-v2	ms-marco -electra -base	xlm-roberta -large -squad2	
MRR	1	0.690	0.642	0.006	
	5	0.745	0.693	0.023	
NDCG	1	0.712	0.654	0.006	
	5	0.772	0.718	0.031	
R	1	0.712	0.654	0.006	
	5	0.821	0.771	0.057	

Table A.11: Base cross-encoder models – EMNLP Snopes split image text dataset

metric/model		all-MiniLM -L6-v2	multi-qa -distilbert -cos-v1	all -distilroberta -v1
MAP	1	0.691 ± 0.004	0.685 ± 0.005	0.708 ± 0.026
	3	0.768 ± 0.003	0.751 ± 0.003	0.786 ± 0.017
	5	0.776 ± 0.003	0.755 ± 0.004	0.793 ± 0.018
	10	0.779 ± 0.003	0.760 ± 0.003	0.798 ± 0.017
P	1	0.693 ± 0.004	0.688 ± 0.005	0.711 ± 0.026
	3	0.288 ± 0.003	0.277 ± 0.002	0.291 ± 0.002
	5	0.180 ± 0.001	0.170 ± 0.001	0.181 ± 0.002
	10	0.092 ± 0.001	0.088 ± 0.000	0.094 ± 0.001
MRR	1	0.693 ± 0.004	0.688 ± 0.005	0.711 ± 0.026
	3	0.769 ± 0.003	0.751 ± 0.003	0.786 ± 0.017
	5	0.777 ± 0.003	0.755 ± 0.004	0.793 ± 0.018
	10	0.780 ± 0.003	0.760 ± 0.003	0.798 ± 0.017
NDCG	1	0.699 ± 0.004	0.688 ± 0.005	0.710 ± 0.027
	3	0.794 ± 0.005	0.770 ± 0.003	0.808 ± 0.015
	5	0.808 ± 0.004	0.778 ± 0.004	0.820 ± 0.016
	10	0.814 ± 0.004	0.789 ± 0.003	0.832 ± 0.014
R	1	0.696 ± 0.004	0.685 ± 0.005	0.708 ± 0.027
	3	0.860 ± 0.010	0.826 ± 0.006	0.871 ± 0.008
	5	0.893 ± 0.007	0.845 ± 0.005	0.900 ± 0.012
	10	0.912 ± 0.006	0.877 ± 0.004	0.935 ± 0.004
metric/model		distiluse-base -multilingual -cased-v2	nq -distilbert -base-v1	
MAP	1	0.656 ± 0.018	0.646 ± 0.009	
	3	0.738 ± 0.019	0.703 ± 0.005	
	5	0.747 ± 0.018	0.708 ± 0.005	
	10	0.751 ± 0.017	0.714 ± 0.005	
P	1	0.659 ± 0.018	0.649 ± 0.009	
	3	0.279 ± 0.007	0.257 ± 0.002	
	5	0.176 ± 0.003	0.159 ± 0.001	
	10	0.091 ± 0.001	0.083 ± 0.001	
MRR	1	0.659 ± 0.018	0.649 ± 0.009	
	3	0.740 ± 0.019	0.705 ± 0.005	
	5	0.748 ± 0.018	0.711 ± 0.005	
	10	0.752 ± 0.017	0.716 ± 0.005	
NDCG	1	0.660 ± 0.017	0.649 ± 0.009	
	3	0.766 ± 0.020	0.720 ± 0.005	
	5	0.781 ± 0.016	0.730 ± 0.004	
	10	0.790 ± 0.014	0.744 ± 0.004	
R	1	0.658 ± 0.017	0.646 ± 0.009	
	3	0.837 ± 0.024	0.768 ± 0.005	
	5	0.873 ± 0.015	0.792 ± 0.005	
	10	0.901 ± 0.009	0.833 ± 0.006	

Table A.12: CLEF with best negative samples finetuned bidirectional models – CLEF results

metric/model		all-MiniLM -L6-v2	multi-qa -distilbert -cos-v1	all -distilroberta -v1
MAP	1	0.705 \pm 0.005	0.654 \pm 0.013	0.705 \pm 0.010
	3	0.794 \pm 0.005	0.738 \pm 0.006	0.789 \pm 0.006
	5	0.801 \pm 0.004	0.749 \pm 0.006	0.797 \pm 0.007
	10	0.803 \pm 0.004	0.752 \pm 0.006	0.801 \pm 0.007
P	1	0.707 \pm 0.005	0.657 \pm 0.013	0.707 \pm 0.010
	3	0.300 \pm 0.001	0.279 \pm 0.003	0.296 \pm 0.001
	5	0.186 \pm 0.000	0.177 \pm 0.001	0.184 \pm 0.001
	10	0.094 \pm 0.000	0.091 \pm 0.001	0.095 \pm 0.001
MRR	1	0.707 \pm 0.005	0.657 \pm 0.013	0.707 \pm 0.010
	3	0.794 \pm 0.005	0.738 \pm 0.006	0.790 \pm 0.006
	5	0.801 \pm 0.004	0.749 \pm 0.006	0.797 \pm 0.007
	10	0.803 \pm 0.004	0.753 \pm 0.006	0.802 \pm 0.007
NDCG	1	0.708 \pm 0.005	0.659 \pm 0.013	0.708 \pm 0.009
	3	0.820 \pm 0.004	0.763 \pm 0.005	0.815 \pm 0.004
	5	0.834 \pm 0.003	0.782 \pm 0.005	0.828 \pm 0.006
	10	0.839 \pm 0.003	0.791 \pm 0.005	0.838 \pm 0.006
R	1	0.705 \pm 0.005	0.656 \pm 0.013	0.705 \pm 0.009
	3	0.894 \pm 0.004	0.833 \pm 0.008	0.886 \pm 0.004
	5	0.926 \pm 0.002	0.878 \pm 0.005	0.916 \pm 0.006
	10	0.940 \pm 0.003	0.905 \pm 0.006	0.948 \pm 0.006
metric/model		distiluse-base -multilingual -cased-v2	nq -distilbert -base-v1	
MAP	1	0.603 \pm 0.017	0.620 \pm 0.006	
	3	0.706 \pm 0.016	0.685 \pm 0.004	
	5	0.715 \pm 0.013	0.691 \pm 0.004	
	10	0.718 \pm 0.013	0.698 \pm 0.004	
P	1	0.605 \pm 0.017	0.622 \pm 0.006	
	3	0.276 \pm 0.006	0.255 \pm 0.003	
	5	0.174 \pm 0.002	0.158 \pm 0.001	
	10	0.089 \pm 0.001	0.084 \pm 0.001	
MRR	1	0.605 \pm 0.017	0.622 \pm 0.006	
	3	0.707 \pm 0.016	0.688 \pm 0.004	
	5	0.715 \pm 0.013	0.693 \pm 0.004	
	10	0.719 \pm 0.013	0.699 \pm 0.004	
NDCG	1	0.608 \pm 0.017	0.622 \pm 0.006	
	3	0.738 \pm 0.015	0.706 \pm 0.004	
	5	0.754 \pm 0.011	0.716 \pm 0.004	
	10	0.761 \pm 0.012	0.733 \pm 0.005	
R	1	0.606 \pm 0.017	0.620 \pm 0.006	
	3	0.824 \pm 0.017	0.763 \pm 0.007	
	5	0.862 \pm 0.011	0.789 \pm 0.006	
	10	0.886 \pm 0.012	0.842 \pm 0.009	

Table A.13: CLEF with random negative samples finetuned bidirectional models
– CLEF results

metric/model		stsb	quora	qnli	ms-marco
		-TinyBERT -L-4	-roberta -base	-distilroberta -base	-TinyBERT -L-2-v2
MAP	1	0.713 ± 0.009	0.708 ± 0.025	0.693 ± 0.013	0.635 ± 0.005
	3	0.791 ± 0.006	0.779 ± 0.021	0.751 ± 0.013	0.728 ± 0.003
	5	0.797 ± 0.006	0.785 ± 0.022	0.757 ± 0.011	0.732 ± 0.002
	10	0.801 ± 0.006	0.789 ± 0.021	0.762 ± 0.011	0.736 ± 0.002
P	1	0.713 ± 0.009	0.710 ± 0.025	0.696 ± 0.014	0.638 ± 0.005
	3	0.295 ± 0.001	0.288 ± 0.007	0.273 ± 0.005	0.279 ± 0.001
	5	0.182 ± 0.001	0.178 ± 0.004	0.170 ± 0.003	0.172 ± 0.000
	10	0.094 ± 0.000	0.092 ± 0.001	0.088 ± 0.001	0.089 ± 0.001
MRR	1	0.713 ± 0.009	0.710 ± 0.025	0.696 ± 0.014	0.638 ± 0.005
	3	0.792 ± 0.006	0.779 ± 0.021	0.753 ± 0.013	0.730 ± 0.003
	5	0.797 ± 0.006	0.785 ± 0.022	0.760 ± 0.011	0.734 ± 0.003
	10	0.801 ± 0.006	0.789 ± 0.021	0.765 ± 0.011	0.738 ± 0.003
NDCG	1	0.723 ± 0.009	0.725 ± 0.025	0.701 ± 0.014	0.643 ± 0.004
	3	0.821 ± 0.006	0.805 ± 0.021	0.771 ± 0.013	0.759 ± 0.002
	5	0.831 ± 0.005	0.816 ± 0.021	0.783 ± 0.011	0.767 ± 0.001
	10	0.839 ± 0.004	0.826 ± 0.019	0.795 ± 0.010	0.776 ± 0.002
R	1	0.723 ± 0.009	0.723 ± 0.026	0.698 ± 0.014	0.641 ± 0.004
	3	0.887 ± 0.005	0.859 ± 0.021	0.818 ± 0.015	0.835 ± 0.002
	5	0.912 ± 0.004	0.884 ± 0.022	0.847 ± 0.013	0.855 ± 0.002
	10	0.938 ± 0.003	0.916 ± 0.016	0.882 ± 0.010	0.884 ± 0.005
metric/model		ms-marco	ms-marco	xlm-roberta	
		-MiniLM -L-6-v2	-electra -base	-large	-squad2
MAP	1	0.852 ± 0.006	0.809 ± 0.015	0.824 ± 0.011	
	3	0.890 ± 0.004	0.853 ± 0.011	0.859 ± 0.011	
	5	0.893 ± 0.004	0.859 ± 0.011	0.863 ± 0.011	
	10	0.896 ± 0.004	0.862 ± 0.010	0.866 ± 0.010	
P	1	0.855 ± 0.006	0.811 ± 0.015	0.827 ± 0.011	
	3	0.310 ± 0.000	0.302 ± 0.003	0.301 ± 0.006	
	5	0.189 ± 0.000	0.186 ± 0.001	0.184 ± 0.004	
	10	0.096 ± 0.000	0.095 ± 0.000	0.094 ± 0.001	
MRR	1	0.855 ± 0.006	0.811 ± 0.015	0.827 ± 0.011	
	3	0.890 ± 0.004	0.856 ± 0.011	0.859 ± 0.011	
	5	0.894 ± 0.004	0.862 ± 0.011	0.863 ± 0.011	
	10	0.896 ± 0.003	0.864 ± 0.010	0.866 ± 0.010	
NDCG	1	0.865 ± 0.006	0.821 ± 0.015	0.836 ± 0.012	
	3	0.903 ± 0.003	0.870 ± 0.010	0.874 ± 0.013	
	5	0.910 ± 0.003	0.881 ± 0.009	0.880 ± 0.013	
	10	0.916 ± 0.003	0.888 ± 0.008	0.887 ± 0.012	
R	1	0.862 ± 0.006	0.819 ± 0.015	0.834 ± 0.012	
	3	0.926 ± 0.002	0.903 ± 0.008	0.898 ± 0.018	
	5	0.941 ± 0.002	0.927 ± 0.007	0.913 ± 0.018	
	10	0.960 ± 0.002	0.949 ± 0.004	0.936 ± 0.013	

Table A.14: CLEF with best negative samples finetuned cross-encoder models – CLEF results

metric/model		stsb	quora	qnli	ms-marco
		-TinyBERT -L-4	-roberta -base	-distilroberta -base	-TinyBERT -L-2-v2
MAP	1	0.566 ± 0.036	0.555 ± 0.037	0.599 ± 0.014	0.601 ± 0.007
	3	0.672 ± 0.031	0.666 ± 0.033	0.701 ± 0.012	0.706 ± 0.005
	5	0.684 ± 0.029	0.682 ± 0.031	0.711 ± 0.011	0.711 ± 0.005
	10	0.694 ± 0.029	0.688 ± 0.030	0.719 ± 0.011	0.716 ± 0.005
P	1	0.566 ± 0.036	0.556 ± 0.037	0.601 ± 0.014	0.604 ± 0.007
	3	0.267 ± 0.008	0.268 ± 0.009	0.275 ± 0.004	0.276 ± 0.001
	5	0.173 ± 0.003	0.174 ± 0.004	0.174 ± 0.002	0.171 ± 0.001
	10	0.092 ± 0.001	0.092 ± 0.002	0.093 ± 0.001	0.088 ± 0.000
MRR	1	0.566 ± 0.036	0.556 ± 0.037	0.601 ± 0.014	0.604 ± 0.007
	3	0.673 ± 0.031	0.667 ± 0.033	0.704 ± 0.012	0.707 ± 0.005
	5	0.686 ± 0.029	0.682 ± 0.030	0.713 ± 0.011	0.712 ± 0.005
	10	0.694 ± 0.029	0.688 ± 0.030	0.721 ± 0.010	0.716 ± 0.005
NDCG	1	0.576 ± 0.036	0.606 ± 0.040	0.661 ± 0.015	0.609 ± 0.007
	3	0.709 ± 0.029	0.726 ± 0.033	0.758 ± 0.012	0.739 ± 0.004
	5	0.733 ± 0.026	0.749 ± 0.029	0.775 ± 0.012	0.749 ± 0.004
	10	0.754 ± 0.025	0.764 ± 0.027	0.795 ± 0.011	0.759 ± 0.003
R	1	0.576 ± 0.036	0.604 ± 0.040	0.658 ± 0.015	0.606 ± 0.007
	3	0.800 ± 0.024	0.809 ± 0.030	0.826 ± 0.012	0.822 ± 0.004
	5	0.859 ± 0.018	0.865 ± 0.019	0.867 ± 0.011	0.847 ± 0.004
	10	0.921 ± 0.014	0.910 ± 0.016	0.926 ± 0.010	0.877 ± 0.002
metric/model		ms-marco	ms-marco	xlm-roberta	
		-MiniLM -L-6-v2	-electra -base	-large -squad2	
MAP	1	0.824 ± 0.008	0.613 ± 0.052	0.666 ± 0.063	
	3	0.871 ± 0.004	0.730 ± 0.038	0.777 ± 0.052	
	5	0.878 ± 0.005	0.739 ± 0.034	0.784 ± 0.050	
	10	0.880 ± 0.005	0.744 ± 0.033	0.789 ± 0.047	
P	1	0.826 ± 0.008	0.616 ± 0.052	0.667 ± 0.064	
	3	0.309 ± 0.001	0.288 ± 0.010	0.301 ± 0.013	
	5	0.191 ± 0.001	0.182 ± 0.004	0.187 ± 0.006	
	10	0.097 ± 0.000	0.094 ± 0.001	0.097 ± 0.001	
MRR	1	0.826 ± 0.008	0.616 ± 0.052	0.667 ± 0.064	
	3	0.872 ± 0.004	0.732 ± 0.037	0.778 ± 0.052	
	5	0.879 ± 0.005	0.742 ± 0.034	0.785 ± 0.050	
	10	0.881 ± 0.005	0.746 ± 0.033	0.789 ± 0.047	
NDCG	1	0.837 ± 0.008	0.688 ± 0.050	0.767 ± 0.078	
	3	0.888 ± 0.004	0.793 ± 0.035	0.850 ± 0.055	
	5	0.900 ± 0.005	0.810 ± 0.030	0.862 ± 0.049	
	10	0.906 ± 0.003	0.820 ± 0.026	0.873 ± 0.043	
R	1	0.834 ± 0.008	0.685 ± 0.050	0.765 ± 0.077	
	3	0.922 ± 0.004	0.863 ± 0.028	0.905 ± 0.039	
	5	0.948 ± 0.005	0.905 ± 0.018	0.934 ± 0.027	
	10	0.970 ± 0.000	0.934 ± 0.006	0.967 ± 0.009	

Table A.15: CLEF with random negative samples finetuned cross-encoder models

– CLEF results

Politifact									
		nosplit				split			
neg. samples metric/q. text		best neg		random neg		best neg		random neg	
		base	img	base	img	base	img	base	img
MAP	1	0.061	0.132	0.652	0.142	0.015	0.000	0.015	0.000
	5	0.088	0.176	0.751	0.187	0.023	0.008	0.023	0.000
MRR	1	0.061	0.132	0.655	0.142	0.015	0.000	0.015	0.000
	5	0.088	0.176	0.751	0.187	0.023	0.008	0.023	0.000
NDCG	1	0.061	0.137	0.660	0.147	0.015	0.010	0.015	0.000
	5	0.099	0.196	0.786	0.206	0.027	0.020	0.027	0.007
R	1	0.061	0.137	0.657	0.147	0.015	0.010	0.015	0.000
	5	0.132	0.249	0.878	0.254	0.041	0.025	0.041	0.015

Snopes									
		nosplit				split			
neg. samples metric/q. text		best neg		random neg		best neg		random neg	
		base	img	base	img	base	img	base	img
MAP	1	0.015	0.010	0.030	0.010	0.010	0.000	0.005	0.005
	5	0.028	0.026	0.053	0.026	0.015	0.004	0.014	0.007
MRR	1	0.015	0.010	0.030	0.010	0.010	0.000	0.005	0.005
	5	0.028	0.026	0.053	0.026	0.015	0.004	0.014	0.007
NDCG	1	0.015	0.010	0.030	0.010	0.010	0.005	0.010	0.010
	5	0.033	0.032	0.061	0.033	0.020	0.010	0.023	0.013
R	1	0.015	0.010	0.030	0.010	0.010	0.005	0.010	0.010
	5	0.051	0.051	0.084	0.051	0.030	0.015	0.036	0.015

Table A.16: EMNLP finetuned all-MiniLM-L6-v2 – CLEF results

Politifact									
		nosplit				split			
neg. samples metric/q. text		best neg		random neg		best neg		random neg	
		base	img	base	img	base	img	base	img
MAP	1	0.459	0.051	0.036	0.041	0.030	0.015	0.025	0.015
	5	0.518	0.071	0.046	0.056	0.052	0.025	0.048	0.032
MRR	1	0.462	0.051	0.036	0.041	0.030	0.015	0.025	0.015
	5	0.518	0.071	0.046	0.056	0.052	0.025	0.048	0.032
NDCG	1	0.462	0.051	0.036	0.041	0.025	0.030	0.025	0.036
	5	0.540	0.080	0.051	0.061	0.069	0.047	0.067	0.053
R	1	0.459	0.051	0.036	0.041	0.025	0.030	0.025	0.036
	5	0.604	0.107	0.066	0.076	0.099	0.066	0.099	0.066

Snopes									
		nosplit				split			
neg. samples metric/q. text		best neg		random neg		best neg		random neg	
		base	img	base	img	base	img	base	img
MAP	1	0.025	0.020	0.020	0.020	0.000	0.015	0.000	0.010
	5	0.043	0.035	0.037	0.034	0.002	0.020	0.003	0.014
MRR	1	0.025	0.020	0.020	0.020	0.000	0.015	0.000	0.010
	5	0.043	0.035	0.037	0.034	0.002	0.020	0.003	0.014
NDCG	1	0.025	0.020	0.020	0.020	0.010	0.015	0.015	0.025
	5	0.049	0.040	0.043	0.041	0.020	0.024	0.027	0.031
R	1	0.025	0.020	0.020	0.020	0.010	0.015	0.015	0.025
	5	0.066	0.056	0.061	0.061	0.030	0.030	0.036	0.036

Table A.17: EMNLP finetuned all-DistilRoBERTa-v1 – CLEF results

Politifact									
		nosplit				split			
neg. samples metric/q. text		best neg		random neg		best neg		random neg	
		base	img	base	img	base	img	base	img
MAP	1	0.449	0.421	0.470	0.442	0.378	0.000	0.368	0.000
	5	0.566	0.542	0.594	0.564	0.481	0.008	0.470	0.000
MRR	1	0.452	0.421	0.472	0.442	0.381	0.000	0.371	0.000
	5	0.568	0.543	0.596	0.565	0.484	0.008	0.473	0.000
NDCG	1	0.457	0.426	0.477	0.447	0.381	0.005	0.371	0.000
	5	0.608	0.585	0.638	0.606	0.520	0.013	0.508	0.000
R	1	0.454	0.426	0.475	0.447	0.378	0.005	0.368	0.000
	5	0.718	0.698	0.754	0.718	0.627	0.020	0.612	0.000

Snopes									
		nosplit				split			
neg. samples metric/q. text		best neg		random neg		best neg		random neg	
		base	img	base	img	base	img	base	img
MAP	1	0.401	0.376	0.391	0.360	0.000	0.000	0.000	0.000
	5	0.507	0.474	0.490	0.459	0.000	0.000	0.000	0.000
MRR	1	0.401	0.376	0.391	0.360	0.000	0.000	0.000	0.000
	5	0.508	0.474	0.491	0.459	0.000	0.000	0.000	0.000
NDCG	1	0.406	0.381	0.396	0.365	0.000	0.000	0.000	0.000
	5	0.551	0.510	0.528	0.496	0.000	0.000	0.000	0.000
R	1	0.406	0.381	0.396	0.365	0.000	0.000	0.000	0.000
	5	0.673	0.604	0.627	0.594	0.000	0.000	0.000	0.000

Table A.18: EMNLP finetuned ms-marco-TinyBERT-L-2-v2 – CLEF results

Politifact									
nosplit					split				
neg. samples	best neg		random neg		best neg		random neg		
metric/q. text	base	img	base	img	base	img	base	img	
MAP	1	0.596	0.739	0.688	0.632	0.020	0.000	0.020	0.000
	5	0.677	0.809	0.750	0.708	0.029	0.000	0.033	0.000
MRR	1	0.599	0.741	0.690	0.635	0.020	0.000	0.020	0.000
	5	0.680	0.812	0.753	0.711	0.029	0.000	0.033	0.000
NDCG	1	0.604	0.746	0.695	0.640	0.020	0.000	0.020	0.000
	5	0.709	0.836	0.777	0.741	0.033	0.000	0.038	0.000
R	1	0.602	0.744	0.693	0.637	0.020	0.000	0.020	0.000
	5	0.794	0.901	0.845	0.825	0.046	0.000	0.051	0.000

Snopes									
nosplit					split				
neg. samples	best neg		random neg		best neg		random neg		
metric/q. text	base	img	base	img	base	img	base	img	
MAP	1	0.434	0.475	0.343	0.383	0.000	0.000	0.030	0.000
	5	0.529	0.564	0.454	0.492	0.003	0.001	0.051	0.000
MRR	1	0.437	0.477	0.345	0.386	0.000	0.000	0.030	0.000
	5	0.532	0.566	0.457	0.495	0.003	0.001	0.051	0.000
NDCG	1	0.442	0.482	0.350	0.391	0.000	0.000	0.030	0.000
	5	0.571	0.604	0.498	0.537	0.003	0.002	0.058	0.000
R	1	0.439	0.480	0.348	0.388	0.000	0.000	0.030	0.000
	5	0.683	0.713	0.617	0.662	0.005	0.005	0.081	0.000

Table A.19: EMNLP finetuned ms-marco-MiniLM-L-6-v2 – CLEF results

Politifact									
nosplit					split				
query text	base		img		base		img		
metric/neg. s.	best	rand	best	rand	best	rand	best	rand	
R	1	0.007	0.000	0.007	0.007	0.011	0.011	0.011	0.007
	5	0.014	0.022	0.025	0.025	0.025	0.029	0.033	0.040

Snopes									
nosplit					split				
query text	base		img		base		img		
metric/neg. s.	best	rand	best	rand	best	rand	best	rand	
R	1	0.000	0.000	0.000	0.000	0.001	0.001	0.004	0.004
	5	0.001	0.001	0.001	0.001	0.004	0.006	0.016	0.013

Table A.20: EMNLP finetuned all-MiniLM-L6-v2 – EMNLP results

Politifact									
		nosplit				split			
query text		base		img		base		img	
metric/neg. s.		best	rand	best	rand	best	rand	best	rand
R	1	0.000	0.000	0.000	0.000	0.043	0.047	0.011	0.011
	5	0.004	0.007	0.000	0.007	0.120	0.120	0.051	0.054
Snopes									
		nosplit				split			
query text		base		img		base		img	
metric/neg. s.		best	rand	best	rand	best	rand	best	rand
R	1	0.000	0.000	0.000	0.000	0.008	0.014	0.009	0.012
	5	0.001	0.001	0.000	0.000	0.021	0.051	0.021	0.022

Table A.21: EMNLP finetuned ms-marco-MiniLM-L-6-v2 – EMNLP results

Politifact									
		nosplit				split			
query text		base		img		base		img	
metric/neg. s.		best	rand	best	rand	best	rand	best	rand
MRR	1	0.380	0.380	0.627	0.663	0.014	0.018	0.022	0.029
	5	0.427	0.447	0.688	0.748	0.018	0.022	0.035	0.041
NDCG	1	0.380	0.380	0.627	0.663	0.014	0.018	0.022	0.029
	5	0.447	0.469	0.714	0.780	0.021	0.026	0.041	0.046
R	1	0.380	0.380	0.627	0.663	0.014	0.018	0.022	0.029
	5	0.507	0.536	0.793	0.873	0.029	0.036	0.058	0.062
Snopes									
		nosplit				split			
query text		base		img		base		img	
metric/neg. s.		best	rand	best	rand	best	rand	best	rand
MRR	1	0.275	0.305	0.621	0.608	0.004	0.003	0.001	0.000
	5	0.316	0.344	0.698	0.699	0.007	0.006	0.003	0.002
NDCG	1	0.282	0.311	0.641	0.621	0.004	0.003	0.001	0.000
	5	0.335	0.364	0.734	0.737	0.008	0.007	0.005	0.003
R	1	0.282	0.311	0.641	0.621	0.004	0.003	0.001	0.000
	5	0.381	0.411	0.806	0.822	0.012	0.011	0.009	0.006

Table A.22: CLEF finetuned all-MiniLM-L6-v2 – EMNLP results

Politifact									
		nosplit				split			
query text		base		img		base		img	
metric/neg. s.		best	rand	best	rand	best	rand	best	rand
MRR	1	0.004	0.004	0.036	0.022	0.268	0.272	0.384	0.156
	5	0.007	0.007	0.054	0.046	0.329	0.337	0.515	0.257
NDCG	1	0.004	0.007	0.036	0.022	0.268	0.272	0.384	0.156
	5	0.009	0.010	0.064	0.058	0.352	0.364	0.567	0.307
R	1	0.004	0.007	0.036	0.022	0.268	0.272	0.384	0.156
	5	0.014	0.014	0.094	0.094	0.420	0.442	0.721	0.457

Snopes									
		nosplit				split			
query text		base		img		base		img	
metric/neg. s.		best	rand	best	rand	best	rand	best	rand
MRR	1	0.001	0.001	0.021	0.024	0.178	0.205	0.270	0.131
	5	0.002	0.002	0.036	0.035	0.238	0.241	0.452	0.198
NDCG	1	0.001	0.001	0.023	0.024	0.179	0.207	0.282	0.133
	5	0.003	0.003	0.043	0.041	0.262	0.258	0.524	0.230
R	1	0.001	0.001	0.023	0.024	0.179	0.207	0.282	0.133
	5	0.004	0.007	0.062	0.058	0.330	0.306	0.715	0.321

Table A.23: CLEF finetuned stsb-TinyBERT-L-4 – EMNLP results

Politifact									
		nosplit				split			
query text		base		img		base		img	
metric/neg. s.		best	rand	best	rand	best	rand	best	rand
MRR	1	0.431	0.435	0.652	0.678	0.391	0.413	0.591	0.601
	5	0.483	0.489	0.724	0.747	0.440	0.465	0.681	0.705
NDCG	1	0.431	0.435	0.652	0.678	0.395	0.417	0.591	0.601
	5	0.503	0.510	0.751	0.776	0.463	0.488	0.717	0.741
R	1	0.431	0.435	0.652	0.678	0.395	0.417	0.591	0.601
	5	0.562	0.572	0.830	0.862	0.525	0.554	0.826	0.848

Snopes									
		nosplit				split			
query text		base		img		base		img	
metric/neg. s.		best	rand	best	rand	best	rand	best	rand
MRR	1	0.320	0.322	0.701	0.685	0.319	0.317	0.656	0.677
	5	0.359	0.364	0.753	0.745	0.361	0.364	0.720	0.743
NDCG	1	0.326	0.329	0.722	0.705	0.322	0.322	0.678	0.699
	5	0.377	0.384	0.780	0.774	0.378	0.385	0.752	0.775
R	1	0.326	0.329	0.722	0.705	0.322	0.322	0.678	0.699
	5	0.421	0.431	0.827	0.828	0.426	0.439	0.812	0.838

Table A.24: CLEF finetuned ms-marco-MiniLM-L-6-v2 – EMNLP results

dataset approach model/text source	clef	pol			
		split		nosplit	
		base	img	base	img
nqDistilBERTbase	7.69	35.25	49.52	1.64	1.67
distiluse-base-multilingual	8.20	23.40	33.01	1.15	1.18
allDistilRoBERTa	7.80	35.78	53.66	1.69	1.73
multiqaDistilBERT	8.08	40.58	52.05	1.67	1.75
allMiniLM	8.53	18.94	26.87	2.70	2.88
xlmRoBERTaxl	3441.29	6310.52	23115.71	930.58	964.66
msmarcoELECTRAbase	1149.17	2321.68	8975.42	343.96	361.83
msmacoMiniLM	665.56	1125.98	5402.36	125.45	139.60
msmarcoTinyBERTL2	498.19	725.36	3834.87	106.44	102.16
qnliDistilRoBERTabase	722.12	1335.32	5711.64	184.73	201.03
quoraRoBERTabase	1170.38	2246.84	8895.23	344.34	362.35
stsbTinyBERTL4	533.82	887.20	4687.49	125.66	114.32

Table A.25: Sum of inference times comparison [s]

dataset approach model/text source	clef	pol			
		split		nosplit	
		base	img	base	img
nqDistilBERTbase	16.93 ± 0.50	564.37	2068.48	39.48	57.69
distiluse-base-multilingual	18.68 ± 0.72	623.54	2117.30	18.97	26.44
allDistilRoBERTa	17.43 ± 0.50	575.83	2192.90	39.62	54.82
multiqaDistilBERT	17.12 ± 0.34	569.84	2113.97	40.00	56.14
allMiniLM	12.22 ± 0.36	552.85	1793.67	20.41	24.42
xlmRoBERTaxl	99.63 ± 4.21	3102.22	10714.62	188.84	180.64
msmarcoELECTRAbase	31.05 ± 0.32	808.25	2938.12	60.41	64.01
msmacoMiniLM	11.80 ± 0.16	357.88	1216.14	24.16	24.12
msmarcoTinyBERTL2	5.14 ± 0.11	182.67	558.31	8.32	10.16
qnliDistilRoBERTabase	18.70 ± 0.28	443.88	1685.98	33.93	34.30
quoraRoBERTabase	31.50 ± 0.92	787.19	3058.15	60.66	62.23
stsbTinyBERTL4	8.78 ± 0.50	273.56	914.48	19.69	20.76

Table A.26: Finetuning times [s]