Patrick Deutschmann, BSc BSc

More is More: An Analysis of Using Efficient Transformers for Fact-Checking

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Roman Kern, Ass.Prof. Dipl.-Ing. Dr.techn.

Institute for Interactive Systems and Data Science

Head: Stefanie Lindstaedt, Univ.-Prof. Dipl.-Inf. Dr.

Graz University of Technology

Graz, April 2022

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date	Signature

Acknowledgements

This project was carried out in the course of an internship at *Buster.Ai*[1]. First, I want to thank all of my colleagues for allowing me to become part of their team and providing me with a not only productive but also very pleasant environment to conduct my work in. I would especially like to thank Hugo Perrin for his opinions, technical help and kind advice. I would also like to express my gratitude towards my supervisor Professor Roman Kern for his guidance and his thorough and exceptionally timely feedback. Finally, I want to thank my girlfriend Lisa, my family and friends who carried me through this time and showed vital patience with me.

[1] https://buster.ai

Abstract

As the spread of false information has become ever more problematic in recent years, research on automatic fact-checking methods has intensified. Typically, such approaches rely on an explicit knowledge base to verify claims. They use a pipeline that first retrieves relevant documents, then passages therein and, finally, performs entailment, i.e., predicts whether the evidence supports the claim or not. The current state of the art mostly uses a variation of a standard Transformer with full self-attention for the entailment. However, its quadratic memory complexity limits the amount of evidence the model can process. In this thesis, we study the use of various different, more efficient Transformers as entailment models, allowing them to process more evidence. We compare these techniques and balance the advantages and disadvantages. The efficiency improvements allow us to completely remove the passage retrieval step, resulting in significant savings in computational cost for the complete pipeline while achieving 97-99% of the current state-of-the-art performance on the benchmark data set FEVER. Further, our experimental results show that the efficient Transformer Longformer outperforms a RoBERTa baseline for long evidence documents, as it can process more input within the same memory budget. Overall, we find using more evidence beneficial for predictive performance. Using efficient Transformers can reduce the computational costs of fact-checking pipelines and allow them to handle longer evidence documents.

Contents

List of Figures

List of Tables

1. Introduction

False information, commonly referred to as fake news, is becoming an ever more serious issue. In a 2018 Eurobarometer, 37% of respondents claim to come across fake news every day or almost every day [22]. A large majority considers it a problem for their country and democracy in general. The World Economic Forum highlights online misinformation as a major potential source of danger and disruption [23]. At the same time, financial analysts estimate fake news in 2019 alone to have caused $39 billion in stock market losses [55]. Tragically, the phenomenon also soared during the Covid-19 pandemic in the form of myths and rumours. In one dire example, more than 5,800 people have been hospitalised, and 800 have died following a rumour, according to which drinking methanol protects from the coronavirus [34]. Causing economic, social and health fallout, misinformation online is a danger to humankind that cannot be ignored.

Automated fact-checking based on machine learning might be a promising solution to counter it. One way to distinguish the true from the false is to compare claims made online to databases of proven and, per definition, trusted evidence. An illustration of the task is depicted in Figure 1.1. These systems have come a long way: starting as traditional database-driven reference methods, they have evolved to make use of recent advances in Natural Language Processing (NLP). Most modern fact-checking models build on attention-based Transformer models [87], such as BERT [20] and its derivations, to determine whether a claim is supported or refuted by a given piece of evidence. This task is known as *entailment*, *veracity prediction* or *claim verification*. Typically, it is implemented by feeding a joint sequence of the claim and proven evidence into the models. However, the complexity of traditional Transformers is quadratic in the length of the input, significantly limiting the amount of potentially relevant evidence that a model can compare to a claim.

Hence, pipeline architectures, which split the task into three phases, are commonly used: First, the relevant documents (e.g. Wikipedia articles) are retrieved using IR methods, such as BM25 [71]. Then, the relevant passages within the documents are selected with another model. Finally, only the entailment is performed using a Transformer. The downside of this approach is that the model cannot recover if relevant evidence has not been successfully retrieved.

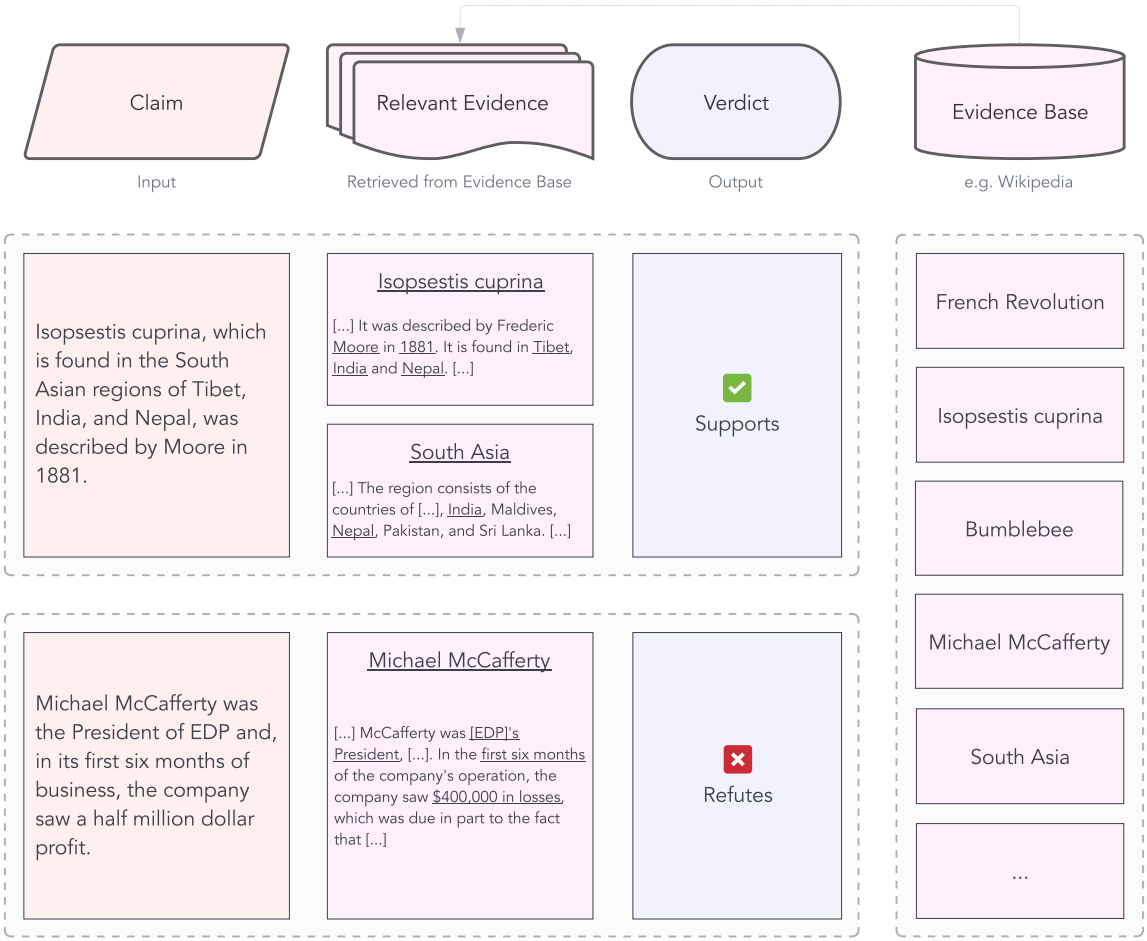


Figure 1.1.: A fact-checking system typically relies on a large evidence base of trusted knowledge. When a claim is put into it, the system predicts whether its evidence supports or refutes it. The classification label is often called a *veracity verdict*. This illustration depicts two claims of the FEVEROUS data set [2]. One is supported, and the other is refuted. The evidence base for this data set is the entirety of articles on English Wikipedia. Most fact-checking systems use an explicit *retrieval* step, where, first, given the claim, the relevant evidence is retrieved from the evidence base. Only then the *entailment*, i.e., the prediction of the veracity label, is performed based on the retrieved evidence.

In this thesis, we integrate the retrieval more with the entailment Transformer by feeding it complete documents, skipping the passage retrieval step. Our hypothesis is that the more potential evidence the entailment Transformer has at its disposal, the better the whole model will perform. However, this requires it to be capable of dealing with the increased input volume. The quadratic complexity of current Transformers renders this computationally infeasible. Fortunately, recent efforts [8, 45, 36, 99] have focused on overcoming this limitation and have achieved promising results in a wide variety of NLP tasks. Following Tay et al. [82], we will call such models *efficient Transformers*. We examine how they can be applied to fact-checking and analyse advantages and disadvantages in terms of scalability and predictive performance.

We set out to answer the following research questions:

- Does the fact that efficient Transformers can handle longer input sequences within a feasible computational budget benefit entailment performance?
- How do the models perform in terms of computational performance compared to traditional approaches?
- How do the different techniques by which efficient Transformers achieve sub-quadratic complexity affect entailment results? Which models work best for the task?
- To which degree can model predictions remain interpretable? For example, how can models, which are fed complete documents, explain how they reached their verdict, i.e., exhibit at which parts of the input they looked?

In the course of this work, we select appropriate data sets to answer these questions and implement a complete fact-checking pipeline with interchangeable pre-processing, retrieval and entailment components. We set up and evaluate various retrieval methods and create synthetic data splits to examine the benefits and drawbacks of different models. Finally, after establishing entailment baselines, we train a variety of efficient Transformers and analyse them with respect to computational and predictive performance.

In short, our contributions are as follows:

1. We confirm that entailment models benefit from using explicit evidence additionally to the implicit knowledge that they gained through pre-training. This finding motivates our study of the effects of using more explicit evidence.
2. We show that using entailment models that can handle longer sequences improves performance in a setting where complete, long documents are used as input.
3. We find that feeding complete documents into efficient entailment models and thereby removing the passage retrieval step achieves comparable

performance on FEVER [83] while simplifying the complete pipeline and reducing the computational cost.

This thesis is structured as follows. After this introduction, we present related work in Chapter 2 and formalise the problem in Chapter 3. Then, in Chapter 4, we explain our method for retrieval and entailment. Chapter 5 reports our evaluation results on the different data sets, discusses them and shows possibilities for future work. Finally, we conclude our findings in Chapter 6.

2. Related Work

In this chapter, we introduce work related to this thesis project. First, we provide an overview of background knowledge that should be helpful to readers unfamiliar with the fields of NLP and fact-checking. Then, we summarise data sets and current state-of-the-art methods as well as efficient Transformer models.

2.1. Background

This section introduces the concepts relevant to this thesis project. It gives an overview of fact-checking in general, details traditional information retrieval approaches and explains the deep learning methods we use.

2.1.1. Fact-Checking

The task of fact-checking is often associated with the domain of journalism. There exist numerous websites where humans manually fact-check claims of various sources and support their results using trusted evidence[1]. They form an indispensable resource for society to distinguish between true and false information.

In this project, we are concerned with partly automating this process using machine learning. For a computer to be able to fact-check a claim, it currently needs to refer to some knowledge base[2]. No model can be expected to unconditionally predict if a statement is correct or incorrect. It can only determine whether its evidence base supports it or not. Hence, most approaches in this field make the important restriction that a claim is supported or refuted according to the evidence at hand.

[1] https://research.ewu.edu/journalism/factcheck, accessed 2022-01-05

[2] Readers familiar with this issue might lament that most fact-checking models possess not only explicit knowledge in the form of a knowledge base but also implicit knowledge, which they have gained through pre-training. In fact, there exist approaches without an explicit knowledge base that reach reasonable performance on similar, knowledge-intensive tasks [70]. When we say that models cannot make predictions without a knowledge base, we thereby mean all the knowledge they have ever been presented with, including the implicit knowledge in their parameters.

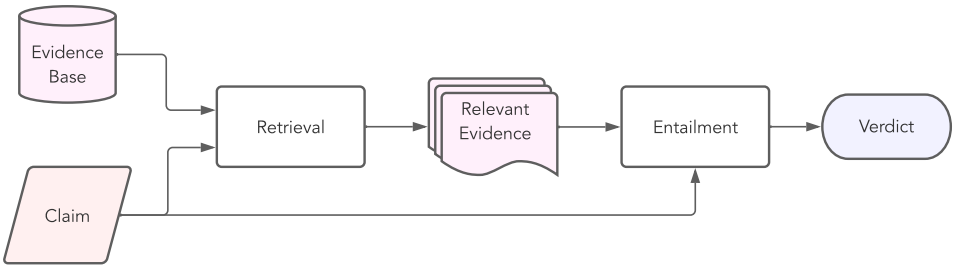


Figure 2.1.: High-level overview of a fact-checking pipeline

Conceptually, the problem can be grouped into two sub-problems: (1) retrieving the relevant evidence from the knowledge base that discusses the claim and (2) determining whether the retrieved evidence supports or refutes the claim. Taken together, the first step, which we call *retrieval*, and the second step, which we call *entailment*, form a so-called *fact-checking pipeline*. A high-level overview of it is depicted in Figure 2.1. The nature of the problem brings about the challenge of cascading errors: If the evidence necessary to support or refute the claim has not been successfully retrieved, it takes away the foundation on which the entailment model could make a decision. Then it must rely on its implicit knowledge or resort to random guessing.

To tackle the fact-checking task, current research employs both traditional Machine Learning (ML) approaches, such as classical IR methods, as well as more novel deep learning models. While the latter typically outperform the former, they are usually more costly to train and operate in practice. We elaborate on the approaches with which we experiment in the following subsections.

Unfortunately, the literature in the field often uses different terms for the same or very similar concepts. We aim to stick with a consistent terminology but give an overview of synonyms and related terms in the Glossary (Appendix C).

2.1.2. Information Retrieval (IR) Methods

For the retrieval component of our pipeline, we partly rely on traditional IR methods. Their task is to find relevant evidence discussing the claim at hand. In particular, we use the techniques TF.IDF [53] and BM25 [71] which are explained in the following sub-sections.

TF.IDF

Term Frequency times Inverse Document Frequency or TF.IDF, for short, is a statistic that estimates the importance of a term to a document [53, 73]. It can be intuitively understood as a relative measure of how often a term occurs in a document, scaled down by its overall commonness.

A formal definition in the notation of Rajaraman and Ullman [67] is the following: For a term i and a document j, it is defined as $\mathrm{TF.IDF}_{ij} = \mathrm{TF}_{ij} \cdot \mathrm{IDF}_{ij}$, where TF_{ij} denotes the *term frequency* and IDF_{ij} the *inverse document frequency*. Given a collection of N documents, where we call f_{ij} the frequency, i.e., number of occurences of term i in document j, we define the *term frequency* $\mathrm{TF}_{ij} = f_{ij} / \max_k f_{kj}$ representing the frequency of the term normalised by the maximum number of occurences of any term. Supposing term i appears in n_i of all N documents, $\mathrm{IDF}_i = \log_2(N/n_i)$. This measure is included to reflect that terms do not necessarily characterise a document better, just because they appear more frequently.

In the application of document retrieval, documents with a high total TF.IDF for all terms in a given query are likely relevant and should be retrieved. While multiple variations of TF.IDF exist, it has no learnable parameters or hyper-parameters and is very fast to run both at indexing and retrieval time.

BM25

Another widespread ranking function used in document retrieval is BM25[3] [71]. It can be considered an extension of TF.IDF that, additionally to term frequency and document frequency, also accounts for document length and term frequency saturation. As it is designed to operate on complete queries, we provide the following common definition: Given a document D and a query Q, the BM25 score is defined as

$$\mathrm{score}(D, Q) = \sum_{i=1}^{n} \mathrm{IDF}\left(q_i\right) \cdot \frac{f\left(q_i, D\right) \cdot \left(k_1 + 1\right)}{f\left(q_i, D\right) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\mathrm{avgdl}}\right)} \tag{2.1}$$

where $f(q_i, D)$ is the term frequency of q_i, $|D|$ is the length of the documents in words and avgdl is the mean document length. k_1 and b are hyper-parameters that can be adjusted per application.[4]

[3] https://kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm-25/, accessed 2021-12-21

[4] https://en.wikipedia.org/wiki/Okapi_BM25, accessed 2021-12-21

The differences to TF.IDF make BM25 barely more expensive to compute while exhibiting a few more desirable properties. We, therefore, use it in all of our non-deep (*sparse*) experiments.

Limitations

TF.IDF and BM25 rely on the exact matching of terms between claims and documents. This works well in the case of named entities, but it cannot handle situations where semantic understanding is required, such as with synonyms. For example, for the claim *"Prince Philip died in 2021."*, these methods will relatively easily be able to retrieve the Wikipedia article *Prince Philip, Duke of Edinburgh*. However, given *"In California, it's prohibited to drink alcohol in parks."*, they might struggle to retrieve the article *Public intoxication*, as for doing so, they would need to know about the connection between *alcohol* and *intoxication* as well as the fact that *in parks* refers to the *public*. In order to address these shortcomings, retrieval methods based on deep learning have been developed, which we will introduce in Section 2.2.2.

2.1.3. Deep Learning Models

Machine learning models based on deep learning have achieved remarkable results in recent years as adoption and research has dramatically increased [74]. They have outperformed more traditional approaches across various domains, especially in those that require handling large amounts of unstructured information, such as computer vision [74].

As for verifying claims, models have to be able to understand human language, the sub-field relevant for this project is Natural Language Processing (NLP). Also there, many tasks are solved using deep neural networks. One of the most important model architectures to this day is the *Transformer* building upon the concept of *attention*, initially introduced in 2017 [87]. While in most recent history, different models [24] were developed that achieve comparable performance, the overwhelming majority of NLP models to date builds on a variation of a Transformer, as do the ones we use in this project.

In the following, we first briefly introduce the basics of deep learning. Then, we explain the Transformer architecture and its most important building block, *attention*.

Basics

This sub-section contains a brief and simplified introduction of some basic concepts in deep learning. Training a neural network is an intricate process with many caveats. This short introduction merely aims to convey the conceptual idea. For a much more complete explanation, please refer to Goodfellow et al. [25], on which the following paragraphs are based.

The term *deep learning* is typically used in connection with *artificial neural networks*. They consist of multiple so-called *layers* of *neurons* which sequentially process information[5]. Generally, the output of the previous layer is the input to the next layer. The input to the first layer is a numeric representation of the model inputs, and the outputs of the final layer are the final predictions. One of the most common basic layers is a *linear layer*, which performs a matrix multiplication of its inputs with its *weight matrix*. Weights are trainable parameters that are learned during the training phase. Layers are typically followed by non-linear activation functions, such as *Sigmoid*, *Tanh* or *ReLU* [25].

There exist multiple settings in which neural networks can be trained. Two of the most common ones are *supervised* and *unsupervised* learning. In the case of supervised learning, a labelled data set exists, which consists of pairs of inputs and corresponding *labels* or *targets*. For example, in fact-checking, the inputs might be claims, and the labels might be whether or not the claims are correct. In an unsupervised setting, there are no labels. One example would be language modelling, where networks are tasked to generate text. For the purposes of this explanation, we will focus on a supervised setting.

Given the labels and the model predictions, a *loss* is computed. The loss is high for bad predictions and low for good predictions. The function that computes it is chosen depending on the task. For regression, a common loss function is *Mean Squared Error*, while for classification, *Cross-Entropy Loss* has more desirable properties. Hence, training a neural network roughly means solving the optimisation problem of finding model parameters that result in minimal loss. Today, most neural networks are trained using *backpropagation* and *gradient descent*. In short, the idea is that the errors are backpropagated from the loss through the network. Then, the gradients of the parameters with respect to the loss are computed. Finally, an *optimiser* (such as *Adam* [41]) performs a gradient step that slightly adjusts the network's parameters. The degree to which the parameters are changed is called *learning rate*. This process is repeated multiple times with the whole training data set.

[5]Networks with only sequential processing are called *feed-forward* neural networks or Multi-Layer Perceptrons (MLPs). If they contain recurrent components, they are called Recurrent Neural Networks (RNNs). A prominent example for RNNs are Long Short-Term Memory (LSTM) models [31].

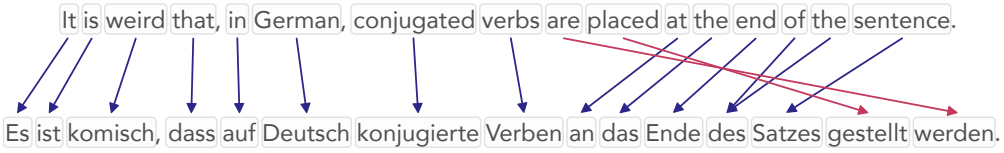


Figure 2.2.: Example translation from English to German where tokens are far away from each other.

In NLP, before the text can be fed into a neural network, it needs to be transformed into a numerical representation with which the network can perform computations. This process is done by a *tokeniser* that splits the input text into multiple tokens. Various tokenisers have been introduced in the past, such as *Byte Pair Encoding (BPE)* [77], *WordPiece* [75] and *SentencePiece* [44]. For various tasks, special tokens are used, for example, a separator token (`SEP`) to divide separate parts of the input. Tokens are assigned numbers which are then, in the neural network, projected into a high-dimensional vector space using *embeddings*. Finally, these embeddings are processed by the rest of the model.

Attention

The concept of attention was introduced by Bahdanau et al. [6] to address a problem that occurs when using RNNs for sequence-to-sequence (seq2seq) tasks[6], for example, translating a sentence into another language. As Bahdanau et al. [6] observe, the way these networks are structured makes it hard for them to find the link between tokens far away from each other. For example, it would be hard to learn if a word that occurs early in the source sentence should be placed at the end of the target sentence as illustrated in Figure 2.2. The attention mechanism solves this problem by giving networks a way to pay *attention* to different tokens, no matter where they are located in the sequence. Conceptually this is comparable to what humans do when solving problems: Not giving equal importance to all context but focusing (or paying *attention*) to the relevant parts.

Consider the example of translation, where we aim to translate a source sentence $\mathbf{x} = [x_1, x_2, ..., x_n]$ of n tokens into a target sentence $\mathbf{y} = [y_1, y_2, ..., y_m]$ of m tokens[7]. The attention mechanism can be used to create a context representation $\mathbf{c}_t$ for token $t \in [1, m]$ that *attends* to all source token representations $\mathbf{h}_i$ with $i \in [1, n]$. The degree of the influence every source token has on the resulting context is defined by the attention weights $\alpha_{t,i}$, i.e., $\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i}\mathbf{h}_i$. Thereby, $\mathbf{c}_t$ is now a representation that contains influences from relevant source tokens.

[6] see https://paperswithcode.com/method/seq2seq, accessed 2021-12-22

[7] Example and notation from https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html, accessed 2021-12-21

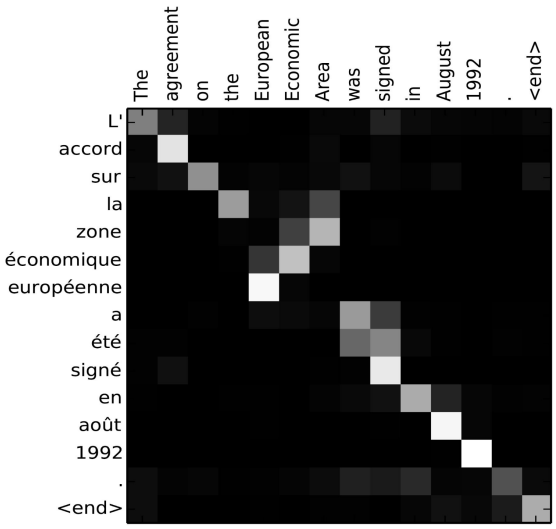


Figure 2.3.: Figure from [6] visualising the attention weights in a translation task between English and French. While for most words, the positions match, we see that the word order of *"European Economic Area"* is inverted compared to *"zone économique européenne"*.

Figure 2.3 visualises the attention weights $\alpha_{t,i}$. A crucial aspect here is how they are computed. Various approaches have been proposed[8], a very common one being *Scaled Dot-Product attention* where

$$\alpha_{t,i} = \text{softmax}\left(\frac{\mathbf{s}_t^T \mathbf{h}_i}{\sqrt{n}}\right) \tag{2.2}$$

with $\mathbf{s}_t$ being a hidden representation of the already generated token and

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}. \tag{2.3}$$

As, in practice, the calculation is performed simultaneously for a batch of samples, a common notation introduced by Vaswani et al. [87] is as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2.4}$$

where Q is a matrix of queries, K a matrix of keys, V a matrix of values and d_k the dimensionality of Q and K. For our introductory example from Bahdanau

[8] for an overview, see https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#summary, accessed 2021-12-22

et al. [6], $d_k = n$, $Q = [\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^b]^T$ and $K = V = [\mathbf{h}^1, \mathbf{h}^2, ..., \mathbf{h}^b]^T$ where we call b the batch size.

This formulation also gives rise to replacing Q, K and V with other values. Our example, where $Q \neq K = V$, is often called *cross-attention*. Another useful version where $Q = K = V$ is called *self-attention*. It is a common building block for many of today's architectures [87, 100]. Generally, multiple attention *heads* are used, which are simply attention weight transformations that are independent of each other. They allow these so-called *multi-head attention* modules to learn different representation subspaces [87].

An important observation in the context of this project is that in the case of self-attention, the matrix multiplication QK^T will result in a large matrix of size $N \times N$ where we call N the length of the input sequence. Therefore, the canonical form of self-attention has a computational memory complexity of $\mathcal{O}(N^2)$, severely limiting the length of input sequences that can be processed. Recently, Rabe and Staats [63] introduced a simple algorithm of memory complexity $\mathcal{O}(1)$ that, however, still has a time complexity of $\mathcal{O}(N^2)$. In this project, we will analyse mechanisms that aim to tackle this computational challenge.

Transformers

The Transformer is a model architecture introduced by Vaswani et al. [87]. It was the first approach that relied only on attention mechanisms for seq2seq tasks, entirely without recurrent or convolutional components. While many variations and improvements have been proposed[9], the core idea remains the same and powers most modern NLP models.

Figure 2.4 depicts the high-level architecture. Like many seq2seq models, the Transformer has an encoder-decoder structure. Given an input sequence $\mathbf{x} = [x_1, ..., x_n]$, the encoder generates a representation $\mathbf{z} = [z_1, ..., z_n]$ from which the decoder then computes the output sequence $\mathbf{y} = [y_1, ..., y_n]$[10]. Additionally to the encoder outputs, the decoder consumes the previously generated tokens.

The encoder consists of N identical layers, each consisting of a multi-head self-attention module, a feed-forward neural network, layer normalisations [5] and residual connections [28]. Similarly, the decoder consists of N layers that are mostly the same as the encoder ones. However, it contains another multi-head-attention module that performs cross-attention between the representations

[9] for an overview, see https://huggingface.co/docs/transformers/v4.14.1/en/model_summary, accessed 2021-12-22

[10] notation taken from http://nlp.seas.harvard.edu/2018/04/03/attention.html, accessed 2021-12-22

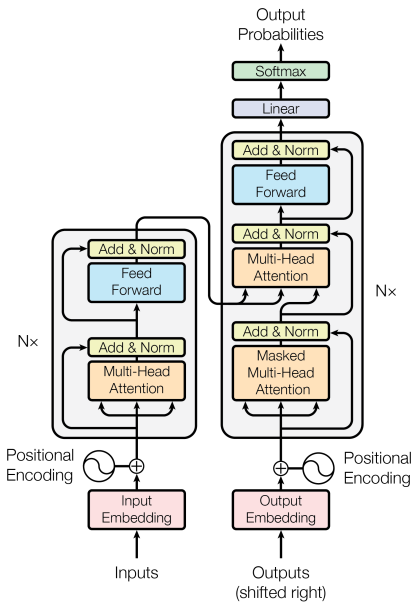


Figure 2.4.: Architecture of the Transformer from Vaswani et al. [87]

generated by the encoder ($\mathbf{z}$) and its intermediary representation from the first self-attention sub-layer.

For text processing, the input sequences are tokenised and then embedded using learnable embeddings. In contrast to convolutional or recurrent neural networks, Transformers have no inductive bias for the principle of locality. In other words: without additional facilities, a Transformer would treat sequences like bags of words. Hence, the authors add positional encodings that give the model information about the position of inputs tokens.

An essential concept commonly used with Transformers is pre-training. The idea is that the model is *pre-trained* on a general task that teaches it about language and the world. Only then, it is *fine-tuned* to a downstream task such as entailment. Pre-training brings about two advantages: First, pre-training data is relatively cheap to obtain, as we will see in the following. Second, it prepares Transformers for a setting called *transfer learning*, meaning that the pre-trained weights can be re-used. This makes the training of large models considerably cheaper, as they need not be re-trained from scratch for every new task or data set. It is common that researchers publish their model weights so that others can not only reproduce their results but also fine-tune the models to different use cases without complete re-training.

A fundamental approach on which many of today's models are based is BERT [20]. Its authors first introduced two important pre-training tasks: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). In MLM, a large corpus of text is provided as input, and certain tokens (or words) are

masked, i.e., hidden from the model. The model is then trained to predict the masked tokens. For NSP, the model gets a sentence and two choices for the sentence that follows, sentence *A* and *B*. *A* is the true sentence, and *B* is a random sentence sampled from the data set. The model's task is to determine whether sentence *A* or *B* is the real sentence following the input. These approaches both have the advantage that samples can be generated easily from arbitrary texts, while data labelling for supervised settings can be expensive. While previously, these pre-training tasks were called unsupervised [20], some argue that they actually represent a form of self-supervised learning[11].

2.2. State of the Art

In this section, we introduce current state-of-the-art research in the field of fact-checking. We start by elaborating on commonly used evaluation data sets and presenting current methods. Then, we describe the current field of research in the area of efficient Transformers, which we aim to apply to the task of fact-checking.

2.2.1. Data Sets

Methods for fact-checking in previous work vary significantly, depending on which data set they were optimised for. Amongst the most common data sets is the FEVER *(Fact Extraction and VERification)* series [83, 84, 2] in which artificially created claims are verified against Wikipedia pages. The objective is for a model to retrieve the relevant pages and passages therein and then assign the label *Supported*, *Refuted* or *Not Enough Info*. Version 2.0 of the challenge [84] requires a model to be more resistant to adversarial samples, whereas the 2021 version, FEVEROUS [2], includes structured data in the form of Wikipedia tables and lists. The original challenge forms part of the KILT benchmark [62]. While in FEVER claims are artificially created, MultiFC [4] is a data set of real-world claims that need to be checked against news outlets. PolitiHop [57] explicitly introduces the additional challenge of requiring multi-hop checking, where only a set of connected evidence pieces leads to the final verdict, unlike FEVER [83] and MultiFC [4], which can mostly be answered with a single sentence. Another fairly recently introduced data set is FaVIQ [59]. It consists of a large number of claims that were generated from real questions and uses the Wikipedia dump from KILT [62] as its evidence base. There are also smaller-scale and more specialised data sets focusing on scientific facts (SciFACT [88]), climate change

[11] https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence, accessed 2022-02-08

Name	Domain	Claims	Evidence source	Claim source
FEVER [83]	general	185k	Wikipedia (intro)	artificial
FEVEROUS [2]	general	87k	Wikipedia (full)	artificial
FaVIQ [59]	general	188k	Wikipedia (full)	gen. from real questions
SciFact [88]	science	1.4k	Scient. abstracts	artificial
Snopes Corpus [27]	general	6k	Snopes	from Snopes
Climate-FEVER [21]	climate	7k	Wikipedia (full)	real
CREAK [56]	com. sense	13k	Wikipedia (full)	artificial
COVID-Fact [72]	covid	4k	r/COVID19	real, auto. annotated
PolitiHop [57]	politics	0.5k	Politifact	real-world
MultiFC [4]	politics	44k	various news sites	real-world

Table 2.1.: Overview of various fact-checking data sets

(Climate-FEVER [21]) and COVID (COVID-Fact [72]). CREAK [56] is intended to provide a challenge for common sense reasoning while the Snopes Corpus [27] uses the fact-checking website Snopes[12] as its evidence base. We give an overview of multiple relevant data sets including sizes and the evidence sources in Table 2.1.

2.2.2. Fact-Checking Methods

The problem of fact-checking is typically divided into the two sub-problems of (1) retrieving the relevant evidence and (2) determining whether it supports or refutes the claim. Related work that tackles Question Answering (QA) tasks [97] often calls the first component *retriever* and the second *reader*. In fact-checking, the *reader* is often called the *entailment* component or *verdict predictor*. Many approaches split the retrieval task in two: *document retrieval* and *passage retrieval*, where a passage can but must not be a sentence. In this setting, the document retriever retrieves the relevant documents, and the passage retriever only selects the relevant passages therein [7]. This is done to reduce the amount of information the *entailment* model must process. For fact-checking, both sub-tasks have to be tackled. Therefore, we first explain state-of-the-art retrieval methods and then move to entailment.

Retrieval Historically, retrieval was done using traditional approaches such as TF.IDF and BM25. These methods are keyword-based and find overlaps between terms in the query (claim) and the documents. A common approach based on TF.IDF is DrQA [17] that is used for baselines both in FEVER [83] and FaVIQ [59] due to its comparatively good performance and low computational requirements. More recently, these approaches have largely been outperformed by so-called *dense* methods, which is why the aforementioned, today,

[12] https://www.snopes.com, accessed 2022-03-08

are also called *sparse*. Dense methods project documents and queries into a shared embedding space generated by a Transformer model. The documents closest to the claim in the embedding space are returned as retrieval results. A common approach for finding these neighbours is Maximum Inner Product Search (MIPS) [54]. The advantage of dense approaches over sparse ones is that the term overlap between query and document need not be exact but can also be semantic, allowing better handling of synonyms. The first approach to show this was Dense Passage Retrieval (DPR) [39], which outperformed sparse methods. However, it is also much more computationally expensive at indexing time as embeddings need to be pre-computed for all documents. While in DPR, retrieval is trained independently of the downstream task, REALM [26] includes it in an end-to-end fashion: The retrieval model which generates the document embeddings is also fine-tuned when training for, say, a fact-checking task. This results in an even higher computational burden, limiting the scope to which REALM can be applied. RAG [48] tackled this problem by only fine-tuning the query embedding model, leaving the document embedding model untouched. FiD [35] uses the expectation-maximisation-algorithm as an approximation. Two final noteworthy extensions to DPR are Multi-hop Dense Retrieval (MDR) [97] and Reinforced Adaptive Retrieval Embedding (RARE) [12] which yield better performance for tasks, where multiple pieces of evidence need to be retrieved sequentially.

Entailment The task of entailment is a standard task in NLP. It is often called Natural Language Inference (NLI). As it is a classification task, results are typically reported in Label Accuracy (LA)[13]. Hence, it is not surprising that there exist many benchmark data sets for it, such as SNLI [13] and MultiNLI [95]. They were integrated into the standard NLP benchmark GLUE [89], which also contains QNLI, which is derived from the QA data set SQuAD [68] and WNLI based on the *Winograd schema challenge* [46]. SuperGLUE [90], an extension of GLUE to make it more challenging, features another data set for Recognising Textual Entailment (RTE). The leaderboards of these tasks[14] are dominated by large Transformer models, many of which are based on the BERT architecture [20]: RoBERTa [49] (355M parameters), T5 [66] (11B parameters), DeBERTa [29] (1.5B parameters), ERNIE [101] (10B parameters) and GPT-3 [14] (175B parameters).

While the standard tasks of retrieval and entailment are building blocks for fact-checking, there is a whole body of work evaluating different approaches jointly and specifically tackling the fact-checking problem. In their FEVER survey paper, Bekoulis et al. [7] group methods into those that (a) follow the

[13] for details, see Section 5.2

[14] https://gluebenchmark.com/leaderboard and https://super.gluebenchmark.com/leaderboard, accessed 2022-01-05

	Validation set		Test set	
Model	**LA**	**FEVER**	**LA**	**FEVER**
Baseline [83]	51.37	31.27	48.84	27.45
KGAT [50]	**78.29**	**76.11**	74.07	70.38
e-FEVER [78]	77.48	74.98	76.60	74.27
BART [47]	-	-	64.0*	**86.74****
RAG [48]	-	-	72.5*	86.31**
MLA [43]	76.92	72.83	77.05	73.72

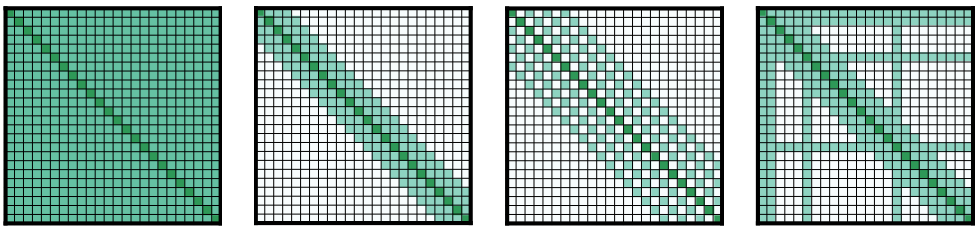
Table 2.2.: Results on the FEVER data set [83]. LA measures the entailment classification accuracy, while the FEVER score also takes into account retrieval performance. *from Lewis et al. [48] **from Bekoulis et al. [7]

traditional pipeline approach and (b) joint models. The pipeline approach refers to models where retrieval and entailment are strictly separated, whereas joint models integrate them, like the aforementioned RAG [48]. According to Bekoulis et al. [7], the best-performing pipeline model for FEVER is e-FEVER [78], which extends the work of Stammbach and Neumann [79] by using GPT-3 [14] for entailment. However, on the validation set, it is outperformed by the graph-based approach KGAT [50]. Another strong contender is MLA [43] with its custom attention module, which achieves top label accuracy on the FEVER test set. All methods above are specifically tailored to and fine-tuned for the fact-checking task. When it comes to FEVER score, Bekoulis et al. [7] report Lewis et al. [47] and Lewis et al. [48] when optimised for the KILT benchmark [62] as the best joint models. We give an overview of all mentioned approaches in Table 2.2.

2.2.3. Efficient Transformers

In the spirit of joint models, our approach aims to alter the traditional pipeline by integrating the passage retrieval and entailment steps using an efficient Transformer. Tay et al. [82] provide a survey of several such methods: While early work used local attention mechanisms at the cost of expressivity (Image Transformer [60]), some newer approaches apply more sophisticated non-dense attention, where not all tokens attend to all others. ETC [1], Big Bird [99] and Longformer [8] do so using fixed patterns, such as neighbours or random tokens, while Reformer [42] learns them.

Another line of work focuses on recurrent processing by caching previous layer activations to attend to them later. The first to introduce this concept was Transformer-XL [19], followed by Compressive Transformers [64], which use compression to be able to keep the caches for longer. Block-Recurrent Transformers [33], on the other hand, extend the recurrence from a token to a



(a) Full self-attention (b) Sliding window (c) Dilated sliding window (d) Global and sliding window

Figure 2.5.: Illustration of Longformer attention patterns from [8]

block level. The resulting architecture is similar to LSTMs [31], but where the individual cells are Transformer layers.

Orthogonal research approximates or simplifies the matrix multiplications used in the QKV attention. Performer [18], Linear Transformer [40] and Linformer [91] use low rank or kernel approaches to approximate them, while Fastformer [96] applies an entirely different attention formulation.

The recently proposed Perceiver method [37] and its extension Perceiver IO [36] keep the self-attention mechanism the same and use cross-attention to compress large inputs to a smaller latent representation, on which they then operate.

In an entirely different spirit, which is in line with MLP-Mixer [85] from vision, FNet [45] completely does away with self-attention. It replaces it with Fourier transformations, which shuffle tokens sufficiently so that the subsequent feed-forward layers can access and learn dependencies between them.

This project aims to compare multiple of these efficient Transformer methods for fact-checking and analyse the advantages and disadvantages that come with them. In the following, we describe some of the methods that are particularly interesting for our application in detail.

Longformer

The taxonomy of Tay et al. [82] groups Longformer [8] in the family of methods that use fixed patterns and employ a form of memory. Longformer reduces the runtime complexity of traditional Transformers from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ using a different attention module that is a drop-in replacement for standard self-attention. While in self-attention, every token can attend to every other token (see Figure 2.5a), in Longformer, a token can only attend to another token in the following cases:

1. **Sliding window**: Tokens can attend to their neighbours with a fixed window size to account for the local context, as depicted in Figure 2.5b.

2. **Dilated sliding window**: In order to increase the receptive field, this case allows attention to neighbouring tokens in a dilated local window, analogously to dilated Convolutional Neural Networks (CNNs) (see Figure 2.5c).
3. **Global tokens**: These tokens can attend to and be attended to by every other token (see Figure 2.5d). Using a so-called *global attention mask*, this can be configured separately for each sample.

The authors use different configurations of these patterns for different model layers. Therefore, of course, the receptive field of non-global tokens is reduced, which can prove detrimental to predictive performance. However, as it allows for processing longer sequences, this disadvantage can be balanced out for long documents. At the time of publication, Longformer achieved new state-of-the-art results on the character-level language modelling tasks `text8` and `enwik8` [52] and QA tasks such as HotpotQA [98]. However, at the time of this writing, Big Bird [99] outperforms Longformer on the latter. For QA tasks, all tokens that are part of the question are configured as global tokens. Thereby, the question can attend to all parts of the potentially relevant evidence.

It is worth noting that, while the model does show linear memory complexity in the input sequence length, its complexity is quadratic with respect to the number of global tokens. Lee-Thorp et al. [45] argue that if a large number of global tokens is required to achieve good performance, the model does not effectively have linear complexity anymore.

Big Bird

Big Bird [99] is a model that is conceptually close to Longformer [8]. It also uses local and global attention to limit the scope of tokens attending to each other, reducing runtime complexity. Additionally, Big Bird uses random tokens to which all tokens can attend. The paper contains theoretical proofs showing that their attention mechanism is as powerful and expressive as full self-attention.

The authors of Big Bird themselves acknowledge similarities to Longformer and point out two key differences: The way global-local attention is implemented in Big Bird differs from Longformer in that it uses relative rather than absolute position encodings. Also, they train the global tokens using Contrastive Predictive Coding (CPC) [86] loss.

Big Bird comes in two configurations, Internal Transformer Construction (ITC) and Extended Transformer Construction (ETC). For ITC, the global tokens are fixed for all samples. For example, in their HotpotQA [98] experiments with ITC, they set the first 128 tokens to global. ETC, on the other hand, behaves more like Longformer global tokens: Additional tokens are made global to store more

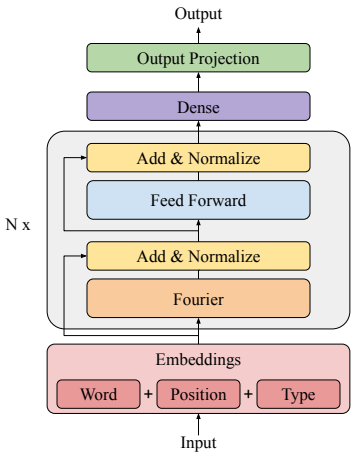


Figure 2.6.: Architecture of an FNet from [45]

context. For HotpotQA [98], they set one token per evidence paragraph, one token per sentence and all question tokens to global. Big Bird ETC outperforms Longformer on HotpotQA [98], TriviaQA [38] and WikiHop [93].

FNet

The abovementioned methods modify the self-attention component to reduce memory complexity. In contrast, the idea of FNet [45] is to do away with it completely. Every self-attention sub-layer in the Transformer architecture is replaced by a 2D Fourier transformation – one along the sequence dimension and one along the hidden dimension. The architecture is depicted in Figure 2.6 and looks very similar to the traditional Transformer in Figure 2.4.

The authors argue that Fourier transformations are suited as replacements for self-attention because they effectively mix tokens. Thereby, they allow the feed-forward layer to access all tokens and learn connections between them. Also, due to the duality of the transformation, the repeated application in the subsequent encoder blocks can be interpreted as alternating transformations between time (sequence) and frequency (hidden) domain. Thus, the feed-forward layers can alternatingly operate on and across tokens.

Fourier transformations are simple linear transformations without learnable parameters, which, due to optimised implementations, are very fast to compute on GPUs. Depending on the algorithm used, they can reduce the computational complexity of the model to $\mathcal{O}(n \log n)$. However, the authors claim that for shorter sequences and when using TPUs, it is faster to use a different algorithm with $\mathcal{O}(n^2)$. The paper reports 92-97% of BERT [20] accuracy on GLUE [89] while training up to 80% faster. They also report much faster training and results comparable to the state of the art on the Long Range Arena benchmark [81].

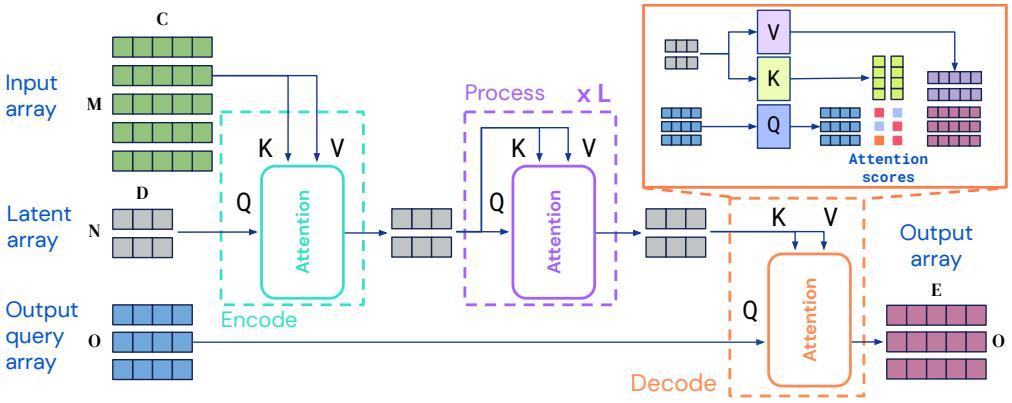


Figure 2.7.: Architecture of Perceiver IO from [36]

Perceiver IO

Perceiver IO [36] is an extension of the previously introduced Perceiver [37]. Deepmind presented both models with a focus that differs from the other efficient Transformers we evaluate here. Rather than aiming to reduce the complexity of the self-attention module, they present an architecture that should be efficient as a whole for multiple tasks. The authors evaluate it on vision, text and multi-modal tasks.

Its core idea, as depicted in Figure 2.7, is to project a potentially large input array into a significantly smaller latent space using cross-attention (`Encode`). The core `Process` phase is virtually equivalent to a standard Transformer self-attention block and is repeated several times. Finally, an output query array is used in the `Decode` phase to extract the output information from the latent space, again using cross-attention. The output query array is the main innovation that Perceiver IO adds to Perceiver. It allows the model to generalise to more tasks and renders it linear in the output size. For entailment, it is just another learnable parameter.

The reduction of computational costs comes from the fact that the repeated self-attention is not applied to the whole input sequence but only to the significantly smaller latent space. Thus, while it still scales quadratically with respect to the latent size, its complexity is linear in the input and output sizes.

The authors report results matching a BERT [20] baseline on GLUE [89] without tokenisation. That is, the input characters are simply encoded as raw UTF-8 bytes.

3. Problem Definition

This chapter formalises the fact-checking task that we tackle. First, we introduce the notation we will use in the rest of the thesis. Then, we present an exact problem definition, runtime challenges that using Transformer models entails, and, finally, the research gap.

3.1. Task Formulation

In the following, we put forward a formal definition of the fact-checking task. Let us call a claim that consists of m tokens $\mathbf{c} = [c_1, \ldots, c_m]$ and a knowledge base $E = \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$. It consists of k evidence documents, which are in turn sequences $\mathbf{e} = [e_1, \ldots, e_n]$ of varying lengths n. For a given $\mathbf{c}$, the task is to find a function $f_{\boldsymbol{\theta}}(\mathbf{c}, E)$ that makes a prediction $\hat{y}$ of the real class $y \in C$ where $C = \{\text{SUP}, \text{REF}, \text{NEI}\}$. That is, the problem is the classification of whether a claim is supported (SUP), refuted (REF) or if not enough information is available to tell (NEI).

We call a sample a tuple of a claim and the corresponding label, i.e., $x_i = (\mathbf{c}_i, y_i)$. The M samples in the training data set $X = \{(\mathbf{c}_1, y_1), \ldots, (\mathbf{c}_M, y_M)\}$ are drawn independently from an unknown distribution p_{data}. The most prevalent objective is to optimise accuracy, which is defined in Equation 3.1 using Iverson bracket notation[1].

$$\text{acc}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{M} \sum_{i}^{M} 1 \left[y_i = \hat{y}_i \right]. \tag{3.1}$$

However, as accuracy is not a good measure for imbalanced data sets, such are often also evaluated on measures like F-scores[2]. While other measures exist, they are mostly specific to certain evaluation data sets and less commonly reported in the literature.

[1] $[P] = \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$

[2] https://en.wikipedia.org/wiki/F-score, accessed 2022-02-07

3.2. Approach

For finding $\hat{y} = f_{\boldsymbol{\theta}}(\mathbf{c}, E)$, the model has two elements at its disposal to base its predictions on: The first are its parameters $\boldsymbol{\theta}$, which we call implicit knowledge. Models only based on them have shown impressive results, but they come with the drawbacks that, on the one hand, they have to be re-trained every time the knowledge base should be updated and, on the other hand, it is expensive to train large models, rendering them costly to scale up [61].

Therefore, we consider a second source of knowledge, namely the explicit evidence base E. Let us assume we had an oracle model f^* that always correctly predicts the class like an infallible human would. It follows, that it makes the prediction correctly given the full evidence base E, i.e., $\forall(\mathbf{c}_i, y_i) \in X, f^*(\mathbf{c}_i, E) = y_i$. Mostly, not all information in E is necessary to verify a given claim $\mathbf{c}$, but only a small subset $E'_{\mathbf{c}} \subset E$ where generally $|E'_{\mathbf{c}}| \ll |E|$. We call $E'_{\mathbf{c}}$ a gold evidence set for claim $\mathbf{c} \iff (f^*(\mathbf{c}, E'_{\mathbf{c}}) = y) \wedge (\nexists \mathbf{e} \in E'_{\mathbf{c}} : f^*(\mathbf{c}, E'_{\mathbf{c}}) = f^*(\mathbf{c}, E'_{\mathbf{c}} \setminus \{\mathbf{e}\}))$. In simple terms, a gold evidence set contains sufficient evidence to verify a claim, but not more. There might exist multiple such sets for any claim $E'^{(1)}_{\mathbf{c}}, \ldots, E'^{(q)}_{\mathbf{c}}$.

We refer to all evidence that is part of any gold evidence set, i.e., $\mathbf{e} \in G_{\mathbf{c}}$ with $G_{\mathbf{c}} = \bigcup_{i=1}^{q} E'^{(i)}_{\mathbf{c}}$, as *gold evidence* and all evidence that is not, i.e., $\mathbf{e} \notin G_{\mathbf{c}}$, as *noise*. Finding an approximation of $E'_{\mathbf{c}}$ is called the *retrieval* step $g_{\boldsymbol{\eta}}$ of the fact-checking pipeline, i.e., $\hat{E}'_{\mathbf{c}} = g_{\boldsymbol{\eta}}(\mathbf{c}, E)$. It is necessary, as current entailment models cannot process all the evidence in the knowledge base. The retriever is optimised to find parameters $\boldsymbol{\eta}^*$ which predict a subset that contains all the necessary evidence while also containing as little noise as possible, i.e.,

$$\begin{aligned} \boldsymbol{\eta}^* = \arg\min_{\boldsymbol{\eta}} \quad & |g_{\boldsymbol{\eta}}(\mathbf{c}, E)| \\ \text{s.t.} \quad & E'_{\mathbf{c}} \subseteq g_{\boldsymbol{\eta}}(\mathbf{c}, E). \end{aligned} \tag{3.2}$$

Practically, retrievers are implemented using a scoring function that assigns a score of relevancy for every piece of evidence $\mathbf{e} \in E$ to a claim $\mathbf{c}$. The ideal version of this function is given in Equation 3.3.

$$\text{score}^*(\mathbf{e}, \mathbf{c}) = 1\,[\mathbf{e} \in E'_{\mathbf{c}}] \tag{3.3}$$

The retriever selects k pieces of evidence with the highest scores. Of course, this approach might lead to a situation where relevant evidence is not correctly retrieved and, hence, the model is left to make a decision based on its implicit knowledge $\boldsymbol{\theta}$.

Finally, the task formulation becomes

$$\begin{aligned} \hat{y} &= f_{\boldsymbol{\theta}}(\mathbf{c}, \hat{E}'_{\mathbf{c}}) \\ &= f_{\boldsymbol{\theta}}(\mathbf{c}, g_{\boldsymbol{\eta}}(\mathbf{c}, E)) \end{aligned} \tag{3.4}$$

where we call $g_{\boldsymbol{\eta}}$ the *retrieval* and $f_{\boldsymbol{\theta}}$ the *entailment* model.

3.3. Runtime Challenges

The models that we and related work use are Transformers, which receive a sequence of text as input, typically constructed as

$$\mathbf{s} = \mathbf{c} \parallel \text{SEP} \parallel \left(\mathop{\Big\|}_{\mathbf{e} \in \hat{E}'_{\mathbf{c}}} \mathbf{e} \right) \tag{3.5}$$

where $\parallel$ denotes the concatenation of two sequences and SEP a separator token.

As these cannot handle arbitrarily long input sequences, they are truncated after a certain sequence length O, so that $\mathbf{s}' = (s_1, \ldots, s_{\min(O, |\mathbf{s}|)})$. Hereinafter we call the final sequence length $N = |\mathbf{s}'|$. Traditional Transformer models using self-attention have a memory complexity of $\mathcal{O}(N^2)$ due to the multiplication QK^T in Equation 2.4, as we explained in Section 2.1.3. Hence, using commonly available current hardware, sequence lengths $N > 512$ tokens become prohibitively expensive, limiting the amount of evidence a model can process [99].

3.4. Research Gap

Under this setting, using more efficient Transformer models with sub-quadratic computational complexity gives rise to the following potential advancements of the state of the art in fact-checking:

1. **Improving model performance**: As efficient Transformers allow for processing longer sequence lengths N, more evidence could be provided to the model, i.e. the size of $\hat{E}'_{\mathbf{c}}$ could be increased. Even though for most data sets, $E'_{\mathbf{c}}$ is relatively small, a prediction $\hat{E}'_{\mathbf{c}}$ of it where $E'_{\mathbf{c}} \subseteq \hat{E}'_{\mathbf{c}}$ might contain substantial noise leading to longer input sequences. Let us consider a noisy retrieval model which retrieves a relevant piece of evidence $\mathbf{e}$ at position p, and an entailment model can handle k pieces of evidence. If $p > k$, it cannot consider the relevant evidence and has to rely on its implicit knowledge $\boldsymbol{\theta}$, which might not contain the relevant

information. Hence, increasing k should improve model performance in such a setting.

2. **Reducing computational cost**: On the other hand, using more efficient models might allow for achieving the same model performance while using fewer resources, i.e., having a lower memory footprint or incurring lower computational costs (e.g. memory consumption or forward pass duration) [82]. This might be achieved either solely through a more efficient entailment model or because handling larger sequence lengths allows for using less expensive retrieval approaches, such as sparse methods or skipping the passage retrieval step.

4. Method

Having elaborated the research problem and the potential for improvement, we will now lay out our approach to building an improved fact-checking pipeline. Overall, we follow the typical retrieve-then-entail architecture. In this chapter, we will detail the methodology and the framework under which we perform our experiments. First, we describe the retrieval component in Section 4.1. We experiment with sparse and dense approaches but will find it best to focus on sparse retrieval with BM25 for the following experiments. Then, in Section 4.2, we come to the entailment components, which form the focus of this work. After establishing baselines, we will explore more efficient Transformers, their differences, and how we use them for fact-checking.

4.1. Retrieval

The first step of a traditional fact-checking pipeline, as illustrated in Figure 2.1, is retrieving documents that are relevant for the claim. While most of our experiments rely on it, there are also claim-only methods, which we briefly explore in Section 5.3.2.

This phase of the pipeline corresponds to a typical IR problem. We make use of the classical approach BM25 and more novel deep learning methods in the following subsections.

In terms of technical implementation, we first load all of our evidence base for the respective data sets into an ElasticSearch[1] index, which serves as our database. We then use Haystack[2] to retrieve the relevant documents. The way this is done depends on the retrieval method used, as explained in the following subsections. Finally, the IDs of the top k results for each claim are saved to a JSONL[3] file for further processing. We always retrieve $k = 100$ document IDs, but only significantly fewer can be used by the entailment models, as we will see.

[1] https://www.elastic.co/elasticsearch/, accessed 2022-02-09
[2] https://haystack.deepset.ai/, accessed 2022-02-09
[3] https://jsonlines.org, accessed 2022-02-18

For the data sets FEVER [83] and FEVEROUS [2], the retrieval is supervised, i.e., the gold evidence for each claim is known. Therefore, we can measure how well the relevant evidence was retrieved. For these two data sets, the following two measures are used to compare to other methods in the literature:

- **Fully supported**: This measure is the fraction of claims for which at least one gold evidence set was fully retrieved, i.e., all documents in that set were retrieved. NEI claims are ignored, as, by definition, the knowledge base does not contain any evidence that supports or refutes them.
- **Oracle accuracy**: In contrast to *fully supported*, NEI claims are always considered correctly retrieved for *oracle accuracy*.

For FaVIQ [59], no gold evidence is given. While the authors do provide a set of reference documents, they retrieved them using TF.IDF. Hence, they cannot be used for supervised training and merely act as a baseline to compare to.

We report the results of our retrieval methods in Section 5.3.1

4.1.1. Sparse Retrieval

For sparse retrieval, the only method we experiment with is BM25, as it is commonly used as a baseline method and yields competitive results [7]. When loading the documents into the ElasticSearch document store, a search index is built based on the estimated importance of the search term. In our experiments, we find it helpful to prepend the document's title to the pure text (i.e., the Wikipedia article content). The improvements in retrieval scores are likely due to the fact that, for Wikipedia, the title often describes well what the article is about. As BM25 is based on term frequency, adding the title emphasises its importance and aids the linking to direct matches. We retrieve the evidence using Haystack's `ElasticsearchRetriever`[4] without any adjustments to vanilla BM25 in terms of function and parameters.

4.1.2. Dense Retrieval

In Section 2.2, we introduced multiple approaches that use deep learning for retrieval under the term *dense retrieval*. The ones that we consider are all based on the same principle: The documents in the knowledge base are projected into an embedding space using a Transformer model and those closest to the embedded claim are retrieved using MIPS. Our baseline retrieval method is

[4] https://haystack.deepset.ai/components/retriever#bm25-recommended, accessed 2022-02-23

DPR [39], where we use Haystack's `DensePassageRetriever`[5]. For creating the embeddings, we use DPR models pre-trained for QA hosted on Hugging Face Transformers[6] without fine-tuning them.

4.1.3. Gold Retrieval

To finish this section, we introduce the concept of gold evidence retrieval, which, strictly speaking, is not a retrieval method at all. It is merely a tool to synthetically create retrieval results with which entailment methods can be tested. For doing so, we artificially enrich the retrieval results with the gold evidence, i.e., the evidence that is required for verifying a claim. We experiment with the following configurations:

- `pure`: In this setting, the gold evidence is the only evidence provided to the model. It reduces the fact-checking problem to NLI, in that only the relevant evidence is there, and the model solely needs to perform the entailment without having to deal with noise.
- `prepend`: Here, the gold evidence is inserted at the beginning of the retrieval results of another retrieval method. For example, we experiment with data sets where the real retrieval was done using a sparse retriever and the gold evidence is moved to the beginning. The results are like `pure` gold retrieval results with noise following it. Hence, the entailment model has to learn to ignore the noise.
- `insert`: This setting is identical to `prepend`, except that the gold evidence is inserted at a certain rank p.
- `random_insert`: In this case, the gold evidence is randomly inserted between ranks p_0 and p_1.

In Section 5.3.1, we will come back to these configurations when compiling synthetic retrieval data sets. We will approximate uniformly distributed evidence and construct another set, where the gold evidence is found only relatively far in the back.

4.2. Entailment

Given a claim and the relevant evidence that has been retrieved in the previous step, the entailment model predicts a veracity verdict. In this section, we explain

[5] https://haystack.deepset.ai/components/retriever#dense-passage-retrieval-recommended, accessed, 2022-02-23

[6] `facebook/dpr-ctx_encoder-single-nq-base` for document encoding and `facebook/dpr-question_encoder-single-nq-base` for claim encoding, accessed 2022-02-23

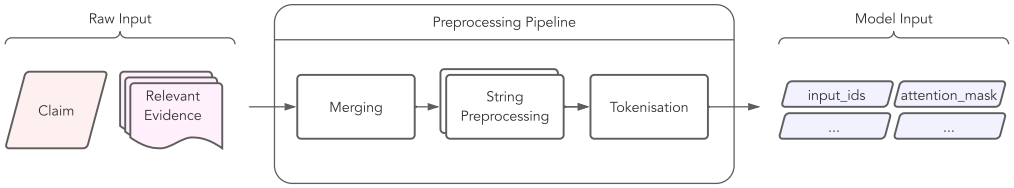


Figure 4.1.: Overview of the preprocessing pipeline that is run for every sample: The raw input is first merged into one string. Then string preprocessing is performed, and, finally, it is tokenised to be input into the model.

how we perform the input preparation (4.2.1) and the training (4.2.2). Then, we elaborate which efficient Transformers we use for this task and how we implement them (4.2.3).

4.2.1. Preparation

Before the input can be fed into the model, it has to be prepared. First, the retrieval results from the document retrieval step have to be accessed. Then, a preprocessing pipeline is run.

Accessing Retrieval Results

For performance reasons, the retrieval step saves only the IDs of the evidence documents it deems relevant in a file. To access the actual text, we implemented both an accessor that fetches it from ElasticSearch and another one that loads the whole data set into memory from the source files and accesses it from there. While the file-based approach is faster for a sufficiently large number of claims, it requires the whole data set to fit into memory. Therefore, we choose a method for an experiment depending on the number of claims and evidence size. For instance, the FEVER evidence base is only around 7GB on disk. Thus, we can fit it into memory. However, the FEVEROUS base is around 46GB. Therefore, we use the ElasticSearch-based accessing solution in this case.

Preprocessing Pipeline

For preprocessing, we implemented a modular pipeline that is depicted in Figure 4.1. The following three steps are run sequentially:

1. **Merging**: We want to feed the entailment model the claim and the evidence in one joint sequence without any intermediary padding. Therefore, first, the claim and the evidence have to be merged into one string that

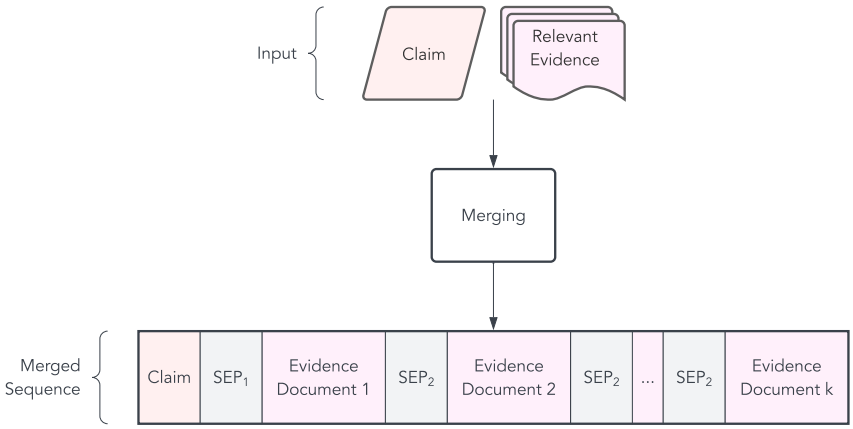


Figure 4.2.: Default merging method in the preprocessing pipeline (`SepTokenMerger`): The claim and all evidence documents are concatenated with separator tokens between them.

can be input into the model. We implement two versions with various configuration options. The first, `ClaimOnlyMerger`, completely disregards the evidence and only feeds the claim into the model. We use it for claim-only experiments, see Section 5.3.2. The more important one, `SepTokenMerger` merges the claim and the evidence with a separator token SEP_1 between them, as depicted in Figure 4.2. It also supports separators between the pieces of evidence (SEP_2), as some pre-trained models we used were trained like this. The output of merging is a single string per sample, i.e., one claim and its evidence.

2. **String preprocessing**: In this step, the merge outputs are preprocessed on a string level. We remove some special characters and perform Unicode normalisation.
3. **Tokenisation**: Finally, the strings are tokenised as usual for NLP models. We use appropriate tokenisers from Hugging Face Transformers[7] depending on the model that will be used later. We list them in Table 4.1. The tokeniser's output is finally the input to the model. It contains at least the input IDs of the tokens and the attention mask. For Longformer [8], it also contains the global attention mask[8].

In a vanilla Transformer with full self-attention, all tokens can attend to all other tokens. With our pre-processing, this means that the evidence documents need not be independent of each other. Not only can the claim attend to all evidence and vice versa, but also all evidence can attend to all other evidence.

Additionally to the core functionality, we built a callback system into the pipeline to collect statistics and intermediary outputs for analysis purposes.

[7] https://huggingface.co/docs/transformers/tokenizer_summary, accessed 2022-02-23
[8] for details, see Subsection 4.2.3

Tokeniser t	Based on	Chars / Token c_t	Reference Tokens r_t
RoBERTa	BPE [77]	4.50	1.00
Longformer			
FNet	SentencePiece [44]	4.08	0.91
Perceiver IO	Byte Encoding	1.00	0.22
Big Bird	SentencePiece [44]	4.55	1.01

Table 4.1.: Different models use different tokenisers. This table lists what they are based on, how many characters one token contains on average, and to how many reference tokens a token generated by the respective tokeniser corresponds.

Reference Tokens

The models we are evaluating use different tokenisers. Due to the different mechanisms, the number of characters tokens represent and, hence, the amount of information they contain differ. As we examine how models perform relative to the amount of evidence information provided to them, tokens are not directly comparable to each other. Therefore, we convert tokens to *reference tokens*, which are defined as the tokens produced by the RoBERTa tokeniser.

Given that a token generated by tokeniser t on average represents c_t characters, we define the number of reference tokens a token generated by t corresponds to as $r_t = c_t / c_{\text{RoBERTa}}$. For example, an FNet token corresponds to only $r_{\text{FNet}} = c_{\text{FNet}} / c_{\text{RoBERTa}} = 4.08/4.50 = 0.91$ reference tokens. Of course, $r_{\text{RoBERTa}} = 1$.

The converted tokens represent the same number of characters and, hence, contain the same amount of information, no matter by which tokeniser they were generated. Table 4.1 details to how many reference tokens the tokens produced by the other tokenisers correspond.

4.2.2. Training

As the inputs are computed, the model can now be trained for the classification problem of entailment, i.e., returning a verdict of whether a claim is *supported*, *refuted*, or, for some data sets, if *not enough information* is available. Unless stated otherwise, all models were trained with the AdamW optimiser [51] and a constant learning rate of $5\text{e}{-6}$. The effective batch size for all experiments was 8. If we could not fit a full batch into GPU memory, we performed gradient accumulation[9] accordingly. All models were trained on the designated training splits of the data sets, and hyper-parameter tuning was done on the validation

[9] https://pytorch-lightning.readthedocs.io/en/latest/advanced/training_tricks.html#accumulate-gradients, accessed 2022-04-15

splits. In the following, we elaborate on the training objective and the resulting loss function.

Objective

The measure most commonly used in the literature to compare multiple fact-checking methods with each other is accuracy. However, since it is not a differentiable function, we train our models using cross-entropy loss. This standard practice is motivated by a maximum likelihood estimation of the ideal parameters, as derived by Goodfellow et al. [25].

Given a labelled sample $\mathbf{x} = (\mathbf{c}, y)$, where $\mathbf{c}$ is the claim and y the veracity label, we encode y in a one-hot fashion in $\bar{\mathbf{y}} \in \{0, 1\}^{|C|}$, i.e., $\bar{y}_i = [i = y]$. The output of the last layer of our model are unnormalised logits $\mathbf{z} \in \mathbb{R}^{|C|}$. Using the softmax function, they are scaled to be interpretable as probabilities and collected in a prediction vector $\hat{\mathbf{y}} \in \mathbb{R}^{|C|}$ with

$$\hat{y}_i = \operatorname{softmax}(z_i) = \frac{\exp(z_i)}{\sum_i \exp(z_i)} \tag{4.1}$$

where each entry $\hat{y}_i$ is the model's predicted probability that class i is the real class y. Finally, the loss for one sample is computed as

$$\mathcal{L}(\bar{\mathbf{y}}, \hat{\mathbf{y}}) = -\sum_{i=0}^{|C|} \bar{y}_i \log \hat{y}_i \tag{4.2}$$

and, given that $\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{c}, \hat{E}'_{\mathbf{c}})$, the training optimisation objective becomes

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=0}^{M} w_i \mathcal{L}\left(\bar{\mathbf{y}}^{(i)}, f_{\boldsymbol{\theta}}\left(\mathbf{c}^{(i)}, \hat{E}'_{\mathbf{c}^{(i)}}\right)\right). \tag{4.3}$$

To account for class imbalance in the training data set, we set loss weights inversely proportional to the class frequencies using the vector $\mathbf{w} \in [0, 1]^M$, whose components we set to

$$w_i = \frac{1}{\sum_{j=0}^{M} 1[y_i = y_j]}. \tag{4.4}$$

Practically, we used the PyTorch implementation `CrossEntropyLoss`[10].

[10] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, accessed 2022-02-24

4.2.3. Efficient Transformers

Traditional Transformer models build on full self-attention on the complete input sequence, which, as we have shown in Subsection 2.1.3, results in quadratic memory complexity with respect to the input length. This severely limits the amount of evidence the model can process. The longest sequence length that these models are usually trained on is 512 tokens[11]. As Figure 5.7b shows, using longer sequence lengths with them results in infeasibly high memory usage on our hardware. Hence, we experiment with models that mitigate this limitation. It is intuitive to expect that using more evidence can improve prediction accuracy. This hypothesis relies on two assumptions. First, models can, in fact, make use of the evidence they receive as input. Second, the current sequence length limitation is prohibitive in that it prevents models from seeing all the necessary evidence, i.e., the gold evidence.

In Figure 4.3a, we depict an evaluation of RoBERTa [49] predictions on FEVEROUS [2]. It shows that samples for which the model has seen at least a part of the gold evidence are much more likely to be correctly classified. Hence, it is reasonable to assume that the models actually use the evidence they see, supporting assumption one.

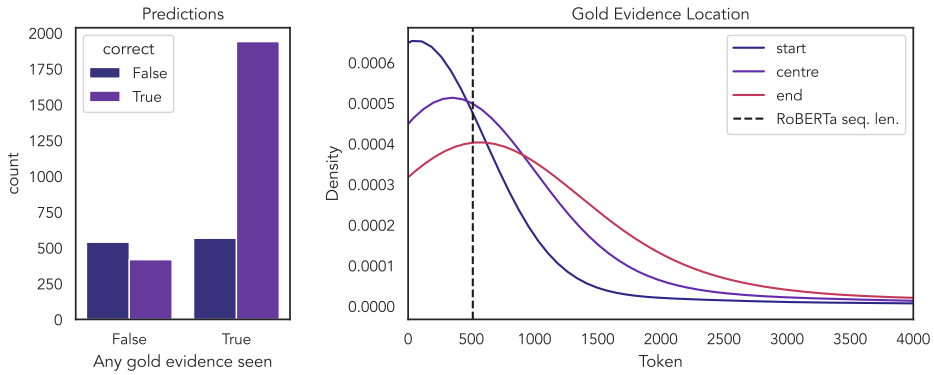
The second assumption is addressed in Figure 4.3b containing an analysis of where the gold evidence is located in the sequence that is put into the model. The Kernel Density Estimate (KDE) plot shows the start, centre and end of the gold evidence in the sequence tokenised with a RoBERTa tokeniser. The data is for the FEVEROUS data set, where the evidence has been retrieved using BM25. We find that the RoBERTa model will not see large parts of the gold evidence, as it is cut off at token index 512. Therefore, at least in this case, longer input sequences might prove beneficial.

The rest of the thesis elaborates on this idea and analyses in which settings and to which extent the use of more efficient Transformers that can deal with longer sequence lengths can outperform traditional models. This subsection explains the efficient Transformer models we experiment with, starting with Longformer. Detailed results will be reported in Subsection 5.3.3. An overview of the hyper-parameters can be found in Table A.2.

Longformer

Longformer [8] is an obvious choice for our experiments, as its concept of global attention tokens seems well suited for the entailment task. While it appears essential for the claim to be able to attend to all evidence tokens, it might be

[11] see configurations in Section 5.3.2



(a) Samples for which the model sees the gold evidence are much more likely to be correctly classified.

(b) For FEVEROUS sparse retrieval, models with a sequence length of 512 are unable to see significant parts of the gold evidence.

Figure 4.3.: Motivation for using longer sequences as inputs to the entailment model

less crucial for all pieces of evidence to attend to each other. To set the global attention for claim tokens, we tokenise the merged string and use the statistics collected in the pipeline to deduce which tokens the initial claim corresponds to. In an ablation experiment (Section 5.3.5), we found it best to activate the global attention only for the claim and no other special tokens.

We perform experiments with the Hugging Face implementation `allenai/longformer-base-4096`[12] that was trained on MLM for sequences of up to 4096 tokens. In terms of hidden dimension size, the number of attention heads and hidden layers, the model is equivalent to `roberta-base`.

Big Bird

As Big Bird [99] is relatively similar to Longformer but different in some important ways (see Section 2.2.3), we examine which influence these differences have for our task. We use the Hugging Face implementation with the checkpoint `google/bigbird-roberta-base`[13], which has a hidden size of 768, 12 attention heads and 12 layers. There is also a large version (`google/bigbird-roberta-large`[14] with a hidden size of 1024, 16 attention heads and 24 hidden layers), but we only experiment with the base version as is it is approximately the same size as the Longformer checkpoint and `roberta-base`. The model has been pre-trained with MLM on English Wikipedia, `bookcorpus` [102] and `cc_news`[15] on sequences of up to a length of 4096 tokens.

[12] https://huggingface.co/allenai/longformer-base-4096, accessed 2022-03-04
[13] https://huggingface.co/google/bigbird-roberta-base, accessed 2022-03-16
[14] https://huggingface.co/google/bigbird-roberta-large, accessed 2022-03-16
[15] https://huggingface.co/datasets/cc_news, accessed 2022-03-16

Unfortunately, the Hugging Face implementation[16] only supports ITC and not ETC mode. Therefore, we cannot activate global attention for the claim tokens and, hence, cannot make a perfect comparison with Longformer. The attention comes in two implementations, `original_full` and `block_sparse`. The contributors recommend using `original_full` for sequences shorter than 1024 tokens, as there is no benefit in using `block_sparse`. For our experiments, we still used `block_sparse` to ensure a fair comparison.

FNet

Our motivation for experimenting with FNet [45] comes from the major speed benefits it brings despite its relatively simple architecture. We fine-tuned the models on our data sets and test the following checkpoints from Hugging Face:

- `fnet-base`[17]: a model pre-trained on the C4 data set [66] with MLM and NSP
- `fnet-base-finetuned-mnli`[18]: a fine-tuned version of `fnet-base` on the GLUE MNLI data set [89]

Both of these models were trained on a sequence length of only 512 tokens. As we want to experiment with longer sequences, we have to extend the position embeddings. We experiment with randomly initialising the extended position embeddings (`random`) and with repeating the existing ones, as it was done by Longformer [8] (`repeat`). We find the latter to work better in most cases. All FNets are trained with a learning rate of 10^{-5} and an effective batch size of 8. In initial tests, we find almost no difference in performance between the two checkpoints, which is why we only use `fnet-base` for our final experiments.

Perceiver IO

Perceiver IO [36] is a potentially well-suited model for our experiments as it puts forward a novel approach that differs from the other methods we consider in that it provides an explicit mechanism for extracting information. Usually, not all documents in the evidence base are necessary for verifying a claim. Mostly, not even all the retrieved evidence is. The way our experiments are set up, we feed complete documents, of which only a few passages are pertinent. It is conceivable that the Perceiver IO encoder can be taught to extract the relevant evidence and project it into latent space. The reasoning of whether it

[16] https://huggingface.co/transformers/v4.9.2/model_doc/bigbird.html, accessed 2022-03-28
[17] https://huggingface.co/google/fnet-base, accessed 2022-03-08
[18] https://huggingface.co/gchhablani/fnet-base-finetuned-mnli, accessed 2022-03-08

supports or refutes the claim then happens in the `Process` module, which can be significantly smaller.

All our Perceiver IO models were trained with a learning rate of 10^{-5} and an effective batch size of 16. We use the implementation and weights from the Hugging Face model `deepmind/language-perceiver`[19] that has been pre-trained with MLM on English Wikipedia and C4 [66]. It has a maximum sequence length of 2048 tokens, where each token is the raw UTF-8 encoding of an input character. In the encoding phase, the input sequence is projected into a latent space of size 256×1280.

As 2048 Perceiver IO tokens only contain the information of roughly 455 reference tokens, we need to use larger versions for evaluating longer sequence lengths. Like with FNet, we test extending them using repetition and random initialisation. We also experiment with linear and nearest-neighbour interpolation since the other techniques do not train well within our training budget.

[19] https://huggingface.co/deepmind/language-perceiver, accessed 2022-03-10

5. Evaluation

After having described the methodology, this chapter now elaborates on the evaluation of our methods. We start with some practical aspects (5.1) and the data sets on which we evaluate (5.2). We go on to present the results (5.3), where we will find BM25 to work best for retrieval and that using efficient Transformers yields gains for entailment both in predictive and computational performance. The best-performing model in our experiments is Longformer. We then discuss our results (5.4) and, finally, explore possibilities for future work (5.5).

5.1. Practical Aspects

This section details the software and hardware setup we used for our experiments to aid reproducibility.

5.1.1. Software

All our code is implemented in Python[1]. For retrieval, we use ElasticSearch[2] as the document storage and Haystack[3] as the framework to access it. Our deep learning models are built with PyTorch[4], PyTorch Lightning[5] and TorchMetrics[6]. All Transformer model implementations are from Hugging Face Transformers[7]. Deep learning models were trained using PyTorch's Distributed Data Parallel (DDP)[8] with CUDA[9]. We tracked and visualised our experiments with Weights &

[1] version 3.9.5, https://www.python.org, accessed 2022-02-02
[2] version 7.10, https://www.elastic.co/elasticsearch/, accessed 2022-02-09
[3] version 1.1.0, https://haystack.deepset.ai, accessed 2022-02-09
[4] version 1.10, https://pytorch.org, accessed 2022-02-09
[5] version 1.5.5, https://www.pytorchlightning.ai, accessed 2022-02-09
[6] version 0.6.1, https://torchmetrics.readthedocs.io, accessed 2022-02-09
[7] version 4.15, https://huggingface.co/docs/transformers, accessed 2022-02-09
[8] https://pytorch.org/docs/stable/generated/torch.nn.parallel.DistributedDataParallel.html, accessed 2022-02-09
[9] version 11.3, https://docs.nvidia.com/cuda/, accessed 2022-03-07

Biases [9][10]. Finally, we used the following packages for various utility functions: numpy[11], pandas[12], seaborn[13], matplotlib[14] and loguru[15].

5.1.2. Hardware

We ran our experiments on an Ubuntu[16] server equipped with an Intel Core i7-7820X CPU clocked at 3.60GHz, 64GB of RAM at 3GHz and two NVIDIA GeForce RTX 3090 GPUs with 24GB of VRAM each, interconnected with an NVLink.

5.2. Data

Of the various data set options presented in Section 2.2, we decided to evaluate on FEVER [83], FEVEROUS [2] and FaVIQ [59]. This combination strikes a good trade-off between widely used and sufficiently challenging options.

5.2.1. FEVER

The FEVER (*Fact Extraction and VERification*) data set [83] has been intensively studied, and many fact-checking methods have already been evaluated on it since its introduction in 2018 [7]. It forms part of KILT [62], a benchmark for various knowledge intensive tasks. The FEVER claims were written by human annotators based on a 2017 dump of around 50,000 popular Wikipedia pages as the knowledge base. In contrast to its successor FEVEROUS [2], this first version of the series only contains the introductory sections of the Wikipedia articles, not the full documents. With a median length of only $|\tilde{\mathbf{e}}| = 350$ characters and a mean length of $|\bar{\mathbf{e}}| = 548$ characters, which corresponds to roughly 122 reference tokens, this makes the individual pieces of evidence $\mathbf{e} \in E$ comparatively short. The training data set consists of around 145,000 claims, of which 80,000 are supported, 30,000 are refuted, and for 35,000, there is not enough information. The validation and test data sets consist of about 20,000 claims each, with balanced classes.

[10] version 0.12.9, https://wandb.ai/, accessed 2022-02-10
[11] version 1.21.4, https://numpy.org, accessed 2022-02-10
[12] version 1.3.5, https://pandas.pydata.org, accessed 2022-02-10
[13] version 0.11.2, https://seaborn.pydata.org, accessed 2022-02-10
[14] version 3.5.1, https://matplotlib.org, accessed 2022-02-10
[15] version 0.5.3, https://loguru.readthedocs.io, accessed 2022-02-10
[16] version 20.04, https://ubuntu.com, accessed 2022-04-15

FEVER poses the fact-checking task explicitly as the combination of retrieval and entailment. It provides for every claim $\mathbf{c}$ multiple evidence sets $E_{\mathbf{c}}^{'(1)}, \ldots E_{\mathbf{c}}^{'(j)}$, each of which contains sufficient relevant evidence to verify the claim. Thereby, the retrieval step can be trained in a supervised manner. In fact, FEVER retrieval is annotated on a passage level, so the annotators even marked the passages within $\mathbf{e} \in E_{\mathbf{c}}^{'(i)}$ that contain the relevant information. However, we ignore this for the purposes of our evaluation, as we are focusing on entailment rather than on retrieval. For the three-way classification problem of entailment, FEVER of course also contains for every claim a label $y \in C$ where $C = \{\text{SUP}, \text{REF}, \text{NEI}\}$. Naturally, NEI claims have no evidence sets, i.e. $j = 0$. For non-NEI samples, the mean number of evidence sets is $\bar{j} = 1.85$.

Typically, the data set is evaluated on LA and the so-called FEVER score, which is a measure that takes into account both retrieval and entailment results. Since we are focusing on the entailment component of the pipeline, we do not compute it in our experiments. While sometimes precision, recall and F1-score are computed, the balanced nature of the evaluation splits allows the intuitive choice of accuracy.

It is worth noting that the FEVER data set has been thoroughly studied, and several issues have been raised. Schuster et al. [76] find that claim-only methods, i.e., methods that take as input only the claim and no evidence at all, achieve very competitive performance. They trace this result not only to the implicit world knowledge these methods possess (which are not an issue with the data set per se) but also find idiosyncracies in the claims. For example, negation phrases are highly correlated with the REF label. Further, Bekoulis et al. [7] lament that FEVER should not be used as the only data set to evaluate fact-checking systems since its claims are artificially generated and based only on Wikipedia, positioning it relatively far from real-world claims. Finally, Bekoulis et al. [7] note that only around 16% of the claims need multiple sources of evidence, making FEVER essentially a single-hop data set. While these issues pose real challenges, they are acceptable for the types of experiments we are running.

5.2.2. FEVEROUS

FEVEROUS (*Fact extraction and VERification Over Unstructured and Structured information*) [2] is the third version of the FEVER [83] series, and it is designed to be more challenging than its predecessors. As its name implies, it extends FEVER by including not only unstructured but also structured information. By that, the authors mean that, besides the Wikipedia articles' texts, lists and tables are also in the evidence base. Like its predecessors, the data set also contains labels for the retrieval step, i.e., gold evidence in terms of text passages, list items or

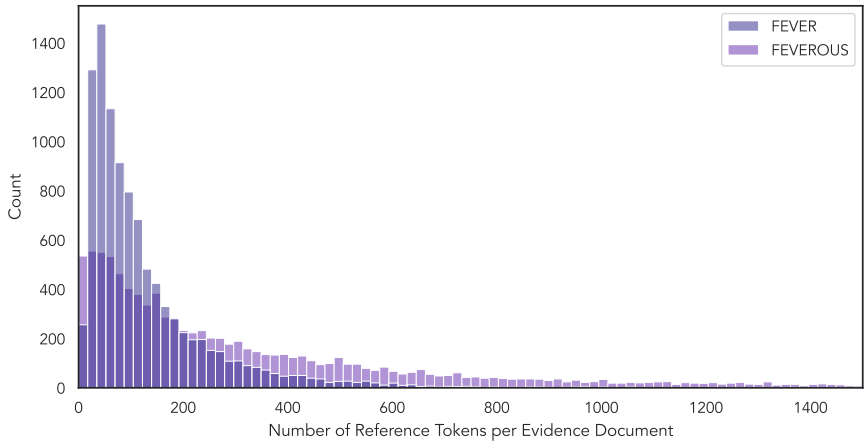


Figure 5.1.: Comparison of the number of reference tokens per evidence document for FEVER [83] and FEVEROUS [2]. FEVEROUS documents are significantly longer than FEVER documents.

table cells that are relevant for verifying a claim. FEVEROUS consists of around 87,000 claims labelled with the same classes as FEVER (*Supported*, *Refuted* and *Not Enough Information*). The authors claim to have taken special care to avoid biases that could be exploited. Another important difference to FEVER is that the FEVEROUS evidence base does not only contain the introductory sections of the Wikipedia articles but the complete pages. Also, it consists of all articles from the English Wikipedia, not only a small subset like FEVER [83].

These challenges, which render finding the relevant pieces of evidence significantly harder, are an important reason for why we chose to evaluate on this data set. However, as the focus of this work does not lie on structured information, we filter out all claims with evidence sets that contain tables or lists. Consequently, we evaluate on a subset of FEVEROUS with a total of 3,463 out of 7,891 validation and 29,169 out of 71,292 training claims. The following statistics of the validation split of our subset show that FEVEROUS is significantly larger than FEVER: While the mean number of evidence sets is $\bar{j} = 1.78$, the mean length of an evidence document is $|\bar{\mathrm{e}}| = 2033$ which corresponds to 449 reference tokens. An illustration of the number of tokens per evidence documents can be found in Figure 5.1.

5.2.3. FaVIQ

The FaVIQ data set [59], which was presented in 2021, is aimed to address some of the shortcomings of FEVER. Most importantly, it aims to contain realistic claims. The authors achieve this by constructing the data set from questions

asked by real users who do not know the answers. Two types of sets exist: An `A` set, which is constructed from ambiguous questions and an `R` set, which is made up of regular questions. With a total of 188,000 claims, it contains approximately as many samples as FEVER.

Like FEVEROUS, it is based on the newer, significantly larger Wikipedia dump from KILT [62], containing 5.9 million articles of full length – not only the introductory section such as in FEVER. The authors claim that the task is, hence, much more challenging. FaVIQ does not provide annotated gold evidence for the retrieval component of the pipeline. It also reduces entailment to a two-way classification problem, where $C = \{\text{SUP}, \text{REF}\}$, dropping the class NEI.

In contrast to FEVER, FaVIQ claims are automatically generated from questions, leading to homogeneous sentence structures and sometimes grammatical errors or typos in the phrasing. Also, due to its novelty, little research exists about the data set. We evaluate on it nonetheless, as the larger knowledge base is well suited for our experiments.

5.3. Results

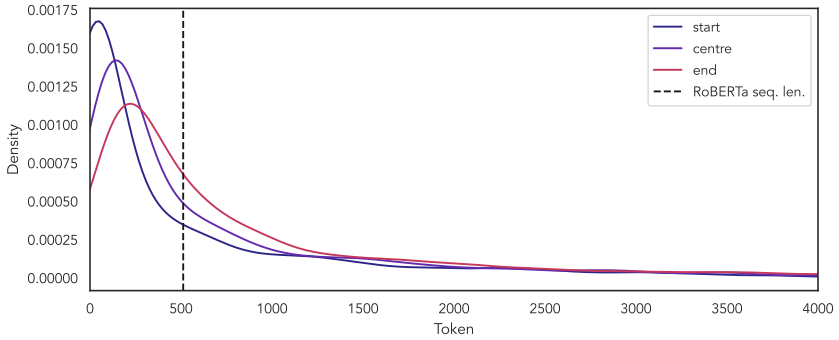
In this section, we report the results we achieved. Starting with the retrieval results (5.3.1), we move on to the entailment baselines (5.3.2). Then, we report the results of our complete fact-checking pipeline (5.3.3). Finally, we outline computational costs (5.3.4) and detailed studies about noise resistance and Longformer global attention (5.3.5).

5.3.1. Retrieval Results

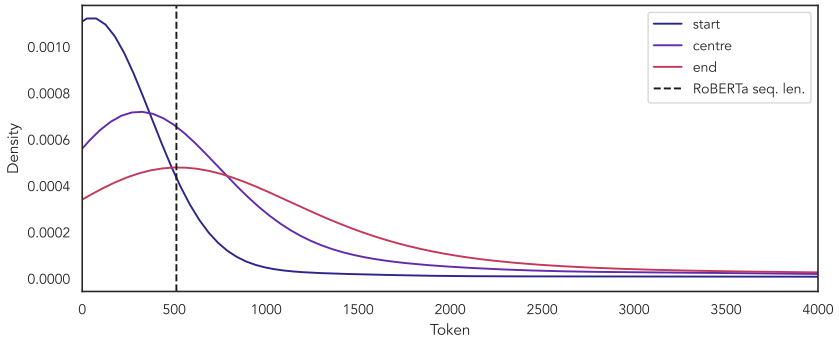
While retrieval is not the main focus of this work, its outputs are the inputs for the entailment and, thereby, its foundation. Thus, we report our results for FEVER [83] and FEVEROUS [2] on the validation splits here. Since FaVIQ [59] provides no retrieval labels, we cannot compute its evaluation measures.

Table 5.1 shows the results we achieved using ElasticSearch BM25 retrieval (see Subsection 4.1.1) and DPR [39] (see Subsection 4.1.2). Of course, the higher the number of documents considered (k), the better the results. In all settings, BM25 outperforms the dense approach. We discuss potential reasons for this in Subsection 5.4.1.

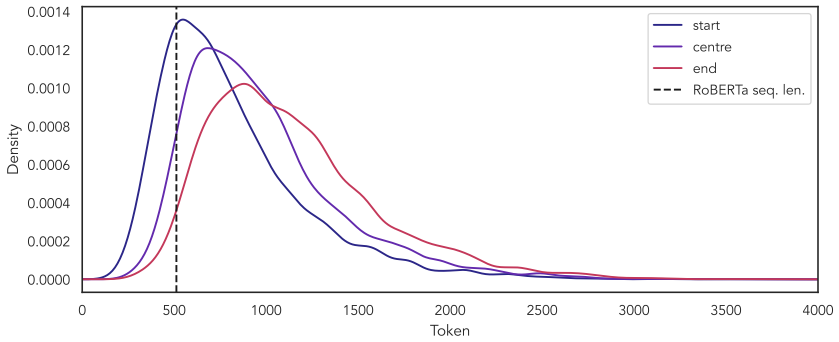
In order to assess whether it makes sense to use models that can handle longer sequence lengths, it is important to know if the additional information would even be helpful. This depends in part on the location of the gold evidence in the tokenised sequence. We depict these location for BM25 retrieval in Figure 5.2a



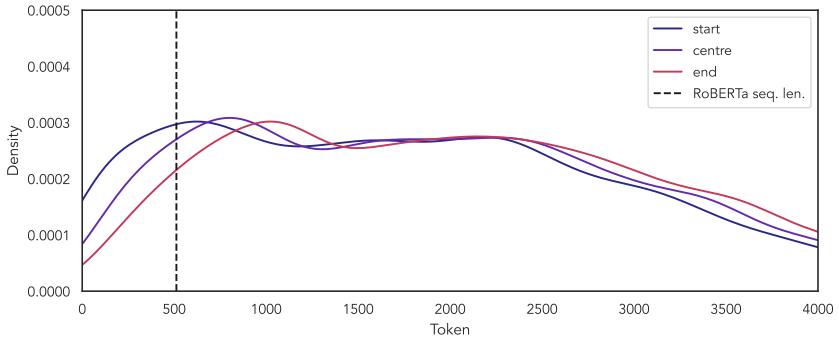
(a) FEVER BM25 retrieval: Almost all evidence is within a range that RoBERTa can still see.



(b) FEVEROUS BM25 retrieval: A lot of evidence is retrieved after token 512.



(c) FEVER *Gold Far Back*: Most evidence starts after token 512.



(d) FEVER *Uniform Gold*: Gold evidence is uniformly distributed.

Figure 5.2.: These plots show where the gold evidence is located in the model input sequence for different retrieval settings.

	FEVER [83]		FEVEROUS [2]	
k	**ES**	**DPR**	**ES**	**DPR**
		fully supported		
1	37.26	36.35	73.50	38.18
2	47.73	45.08	81.26	46.35
5	60.44	53.59	86.87	54.25
10	68.95	58.70	89.37	58.88
20	75.26	62.95	91.12	63.27
50	82.82	68.03	93.62	68.57
		oracle accuracy		
1	58.18	57.57	77.33	47.13
2	65.16	63.39	83.97	54.11
5	73.63	69.06	88.77	60.87
10	79.30	72.47	90.90	64.83
20	83.51	75.30	92.41	68.58
50	88.55	78.69	94.54	73.12

Table 5.1.: Retrieval Results as *fully supported* and *oracle accuracy* within top k results: ES denotes sparse retrieval using ElasticSearch and BM25, DPR is dense retrieval with DPR [39]. We find that, in all settings, the simple sparse approach outperforms the dense method.

and 5.2b. They are almost identical for DPR. We find that, while, for FEVER, most gold is located before token 512, for FEVEROUS, there is a non-negligible amount of evidence after that boundary.

To directly evaluate the usefulness of models that can handle longer sequence lengths, we constructed synthetic retrieval results where the gold evidence was inserted at certain ranks. The results are based on BM25 retrieval on FEVER. They were then modified with `random_insert` gold retrieval, as explained in Subsection 4.1.3. We constructed the following splits:

1. **Gold Far Back**: By setting $p_0 = 5$ and $p_1 = 7$, we created retrieval results where all the gold evidence lies just after the last token that RoBERTa models with sequence lengths of 512 can see. Hence, in this situation, these models will not see any gold evidence and, therefore, find themselves in a claim-only setting. Since corresponding experiments have shown quite bad results, we expect models with such short sequence lengths to be unable to compete with longer ones. The distribution of gold evidence is depicted in Figure 5.2c.
2. **Uniform Gold**: In this setting, the gold evidence is approximately uniformly distributed over a sequence of length 4096. We achieve this by setting $p_0 = 0$ and $p_1 = 30$. For models to perform well on these results, they have to be able to process the whole sequence and not just focus on

some positions. The gold distribution is illustrated in Figure 5.2d.

5.3.2. Entailment Baselines

In order to adequately judge the performance of our methods, we compare them to various baselines: random guessing, claim-only models and standard methods with access to evidence.

Random Guessing

The lowest bar, which every model should beat, is random guessing. Since entailment is a classification problem, one would expect a probability for randomly guessing correctly of $p = 1/\#classes$. For FEVER [83] and FEVEROUS [2], both of which have three classes, this results in an accuracy of $0.\dot{3}$. For FaVIQ, with its two classes, it is 0.5.

Claim-only

Claim-only methods, sometimes called *retrieval-free* methods [97, 11], receive as input only the claim and no explicit evidence whatsoever. To predict a claim's veracity, they have to either rely on the world knowledge encoded in their parameters (implicit knowledge) or exploit biases in the data set. The latter is, of course, an undesirable side effect of data sets that is present in FEVER [7], but actively prevented in FEVEROUS [2].

Roberts et al. [70] showed that claim-only approaches could achieve surprisingly good results. However, they found that only to be the case with the largest configuration of their model, the 11-billion-parameter Transformer T5 [66], which can pack a lot of world knowledge into its parameters. Smaller models, such as RoBERTa [49] (355M parameters) or BART [47] (110M parameters), typically underperform without any explicit evidence. This has been experimentally shown by the claim-only experiments with small Gopher models [65] and the FaVIQ baseline [59].

For all of our baseline experiments, we use RoBERTa [49]. While we expect it to underperform in a claim-only setting, it yields good results when provided with explicit evidence, as exemplified by the GLUE MNLI task [89][17]. It was also used by FaBULOUS [12], the top-scoring team of the FEVEROUS [2] shared task[18]. Concretely, we experiment with the following two configurations:

[17] https://gluebenchmark.com/leaderboard, accessed 2022-03-01
[18] https://fever.ai/2021/task.html, accessed 2022-03-01

Method	Input	FEVER	FaVIQ	FEVEROUS
Random*	-	33.3	50.0	33.3
RoBERTa**	claim	66.0	56.0	57.1
Gopher†	claim	51.1	-	-
RoBERTa**	claim + gold	**95.7**	-	**85.3**
Gopher†	claim + gold	77.5	-	-
Paper Baseline‡	claim + retrieved	51.4	**66.9**	-

Table 5.2.: Results of the entailment baselines: The columns show LA for the different data sets. All results refer to the validation split. For FaVIQ, the results are on the `A` set. RoBERTa indicates our experimental results with RoBERTa large [49] (355M parameters, sequence length of 512 tokens), based on the checkpoint `roberta-large-mnli` and fine-tuned. The Gopher results (280B parameters) are from Rae et al. [65] and were obtained using few-shot prompting.
*Theoretical results
**Our experimental results. No *claim + gold* results for FaVIQ, since there is no annotated FaVIQ gold evidence.
†Results from Rae et al. [65], who only evaluated on FEVER.
‡Baselines of the data set papers. For FEVER: document and passage retrieval using TF.IDF, entailment using an MLP on TF.IDF features and DA [58]. For FaVIQ: retrieval using DPR [39] and entailment using BART [47]. No paper baseline for FEVEROUS due to our own split.

- `roberta-base`[19]: 12 layers, 12 attention heads, 125M parameters
- `roberta-large-mnli`[20]: 24 layers, 16 attention heads, 355M parameters

Both configurations were trained with position embeddings for up to 512 tokens, limiting the maximum sequence length to this value. For the hidden dimension, both use a size of 768. Implementation-wise, we use `ClaimOnlyMerger` in the merge step of the pre-processing pipeline. After the encoder part of the model, a linear layer projects the hidden representations down to logits in the dimension of the number of classes for the data set, i.e., two or three. Predictions are then made through a softmax of the logits.

With Evidence

In this setting, we allow the model to use evidence, as most fact-checking methods do. To do so, a retriever, as described in Section 4.1, retrieves the evidence relevant to the claim. Then, the claim and the evidence are pre-processed and concatenated using `SepTokenMerger`. Finally, the tokenisation pads or truncates the sequence to a pre-defined sequence length.

[19] https://huggingface.co/roberta-base, accessed 2022-03-01
[20] https://huggingface.co/roberta-large-mnli, accessed 2022-03-01

Model	Seq. Len.	LA	Training	Peak GPU Usage
Unit	*tokens*	*%*	*hours*	*GB*
RoBERTa-large	512	78.80	5.5	17.5
Longformer	512	79.05	**5.1**	**11.3**
Longformer	1024	86.01	8.6	15.6
Longformer	2048	**89.97**	16.2	16.0

Table 5.3.: Results on *Gold Far Back*: Longformer with longer inputs clearly outperforms RoBERTa. LA reported on the validation split.

We also use RoBERTa [49] due to the aforementioned reasons. However, RoBERTa has the important limitation of quadratic complexity in the input sequence length, which it shares with all other traditional Transformers. Therefore, all these baseline experiments are run on sequence lengths of only up to 512 tokens.

Results

Table 5.2 summarises the baseline results. We find that claim-only methods perform significantly worse than models with explicit evidence at their disposal. Also, it becomes apparent that FEVER is already solved relatively well if gold evidence is provided. While Gopher forms a strong baseline given that it is based only on few-shot prompting, it is still clearly outperformed by the specifically fine-tuned RoBERTa.

5.3.3. Fact-Checking Results

In this section, we report the results of the complete fact-checking pipeline with efficient Transformers as entailment components. While these results depend on the retrieval component, the focus is on entailment. To compare different entailment methods with each other, we evaluate them on the same retrieval outputs. We start with the synthetic retrieval results *Gold Far Back* and *Uniform Gold* (see 5.3.1). Then, we move to experiments based on real retrieval results for FEVER, FaVIQ and FEVEROUS.

Gold Far Back

In an initial experiment to verify the usefulness of using longer sequence lengths, we ran a RoBERTa baseline and a Longformer on the *Gold Far Back* input split. Due to hardware constraints, and the quadratic development of memory usage of RoBERTa models with respect to the sequence length, we were unable to

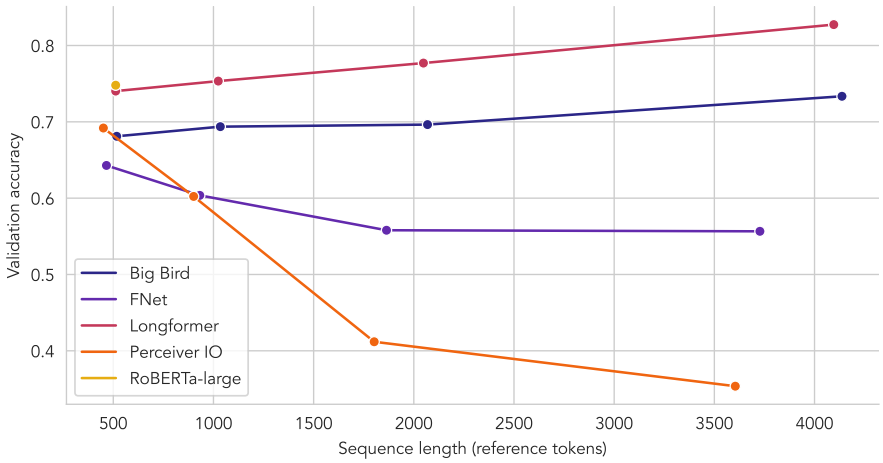


Figure 5.3.: Accuracy vs. Sequence Length on *Uniform Gold*: Longformer and Big Bird are the only models in our experiments that are able to make use of the additional gold evidence they see. Longformer outperforms RoBERTa with it. The performance of FNet and Perceiver IO decreased with additional evidence.

train RoBERTa on longer sequences. As the results listed in Table 5.3 show, Longformer clearly outperforms RoBERTa when using the longer sequence lengths. At equal sequence lengths of 512 tokens, Longformer performs about as well as RoBERTa while training slightly faster and using considerably less GPU memory.

Uniform Gold

In this experiment, we evaluate how well models are able to find the gold evidence that is approximately uniformly distributed across the input sequence. The prediction results are listed in Table 5.4 and visualised with respect to the sequence length in Figure 5.3. Like in the previous setting, we find that Longformer beats RoBERTa when allowed to use longer sequence lengths. Big Bird, while showing increasing performance given more evidence, consistently performs worse than Longformer. FNet clearly underperforms all models and decreases in performance as the sequence length is increased. While Perceiver IO starts with results that are closer to RoBERTa and Longformer, the performance plummets with longer sequences and ends up close to random guessing in the longest setting. All training details can be found in the appendix in Table A.1.

FEVER with BM25 Retrieval

Given our findings from the synthetic data sets and our computational budget, we decided to focus our following experiments on Longformer. The first data

Model	Sequence Length		Label Accuracy (LA)
Unit	*tokens*	*ref. tokens*	*%*
RoBERTa-large	512	512	75.59
Longformer	512	512	74.03
Longformer	1024	1024	75.34
Longformer	2048	2048	77.70
Longformer	4096	4096	**82.74**
FNet	512	466	64.29
FNet	1024	932	60.68
FNet	2048	1864	55.80
FNet	4096	3727	55.66
Perceiver IO	2048	451	70.45
Perceiver IO	4096	901	60.86
Perceiver IO	8192	1802	41.18
Perceiver IO	16384	3604	35.36
Big Bird	512	517	68.11
Big Bird	1024	1034	69.37
Big Bird	2048	2068	69.64
Big Bird	4096	4137	74.08

Table 5.4.: Results on *Uniform Gold*: *Tokens* are the real sequence length of the model input while *reference tokens* denote the equivalent number of reference tokens. Longformer clearly outperforms all other methods.

Model	Seq. Len.	FEVER	FaVIQ	FEVEROUS
Unit	*tokens*		*LA in %*	
RoBERTa-base	256	73.61	62.63	65.35
RoBERTa-base	512	74.24	63.12	68.06
Longformer	256	73.06	63.99	65.32
Longformer	512	74.79	**64.95**	68.47
Longformer	1024	75.23	60.31	**69.94**
Longformer	2048	76.19	58.15	68.32
Longformer	4096	**76.43**	57.91	65.58

Table 5.5.: Results on FEVER, FaVIQ and FEVEROUS validation splits with BM25 retrieval: Providing Longformer with longer input sequences increases performance up to a certain point.

set on which we experiment with real retrieval inputs is FEVER. Table 5.5 lists the results of Longformer with BM25 retrieval on it. We find that longer sequence lengths consistently improve performance. However, between 2048 and 4096 tokens, the difference is relatively small, indicating limited gains from the additional evidence. Despite the simplicity of our retrieval method and the fact that we use no passage retrieval or re-ranking at all, we achieve a performance of up to 97-99% of the state of the art (MLA [43], e-FEVER [78] and KGAT [50], see Table 2.2).

FaVIQ with BM25 Retrieval

The results for FaVIQ with BM25 retrieval are listed in Table 5.5. The LA values refer to the validation split on the A set. Longformer with a sequence length of 512 reaches approximately the performance of the baselines in the data set paper, which is a validation accuracy of 65.1 with TF.IDF and BART [47].

However, we find that longer sequence lengths do not improve but rather hurt performance. Unfortunately, the retrieval step for FaVIQ is not annotated, i.e., the data set does not provide the documents or passages that contain the gold evidence. Therefore, we cannot trivially analyse whether the poor performance is due to retrieval or entailment. Hence, we focus on other data sets for further analysis and experiments.

FEVEROUS with BM25 Retrieval

Lastly, we evaluate longer sequence lengths on FEVEROUS with BM25 retrieval. The results are depicted in Table 5.5. Since we are evaluating exclusively on text-only claims, there is no related work to compare to directly. However, we can

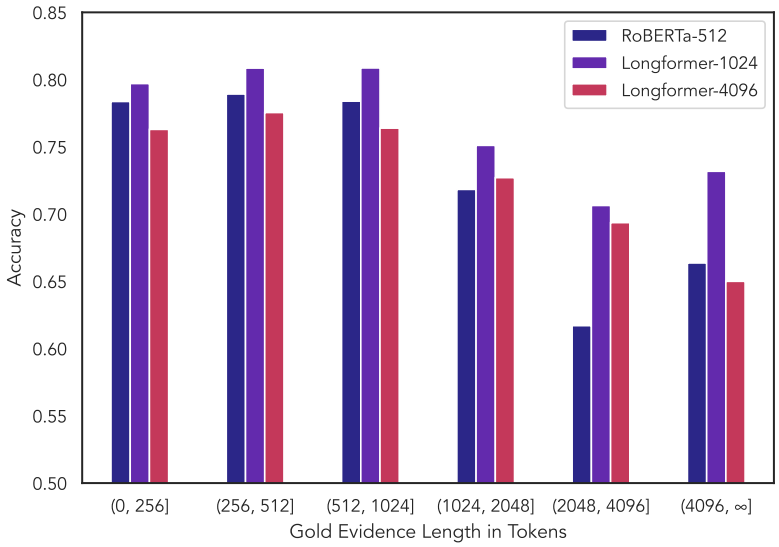


Figure 5.4.: Label Accuracy of FEVEROUS claims grouped by the length of the corresponding gold evidence: RoBERTa and Longformer show comparable performance for claims for which the gold evidence is short. However, the performance gap between Longformer and RoBERTa gets larger with longer sequences. We attribute the increase in the last group to the small number of samples in it. *Not Enough Info* claims were excluded from this plot, as no gold evidence exists for them. The results are based on BM25 retrieval (see Table 5.5).

conclude that there is a certain point up until using longer sequences improves performance. After that, the performance goes down again. An analysis of the accuracy depending on the length of the gold evidence is depicted in Figure 5.4. Using Longformer with longer sequence lengths is especially beneficial for claims whose gold evidence is long. An example confusion matrix is illustrated in Figure 5.5.

5.3.4. Computational Cost

Now that we have seen how the different models perform in terms of prediction accuracy, we will analyse the computational cost that comes with them. A first overview of the model sizes is given in Figure 5.6, which depicts the number of parameters of the models. In this respect, RoBERTa-large is clearly the largest model we consider, followed by Perceiver IO. FNet has the fewest parameters, and Big Bird, Longformer and RoBERTa-base are about in the middle. From a global perspective, all our models are smaller by several orders of magnitude than many of today's huge models. T5 [66] has around 100 times as many parameters as Longformer, GPT-3 [14] almost 1000 times as many.

To empirically measure the runtime performance of our models, we ran short

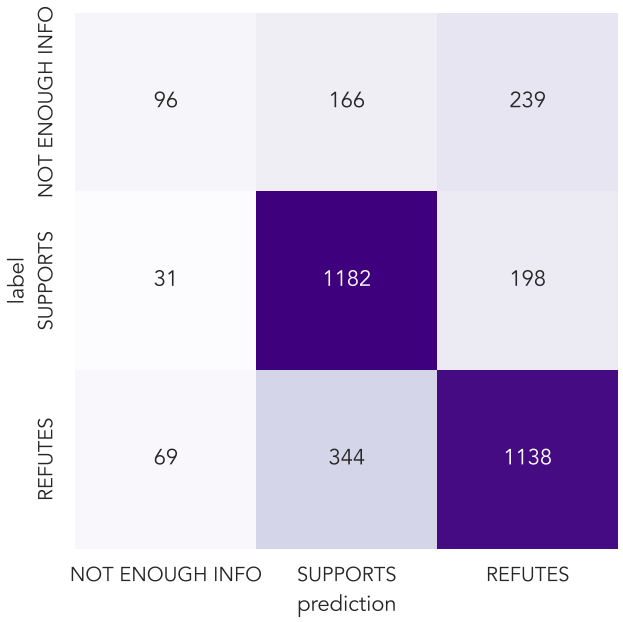


Figure 5.5.: Confusion matrix on FEVEROUS BM25-retrieval with Longformer-1024: The class *Not Enough Info* is almost never correctly predicted by this model.

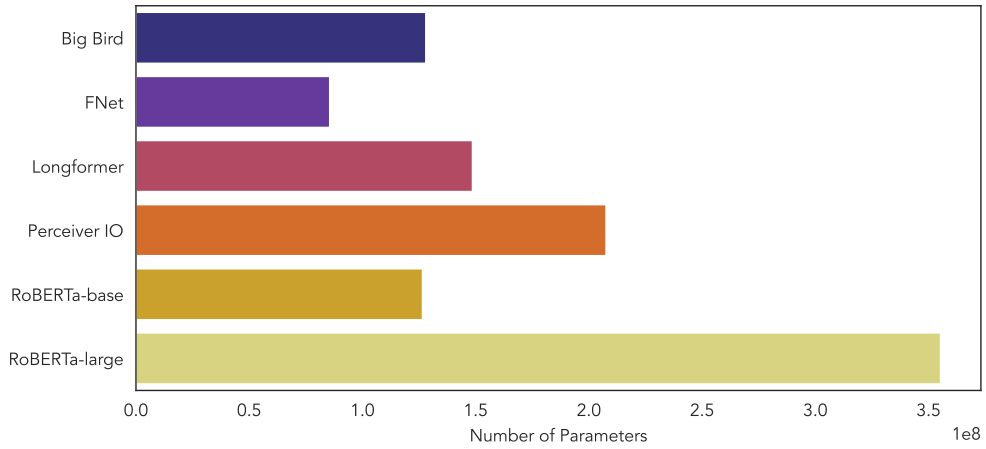
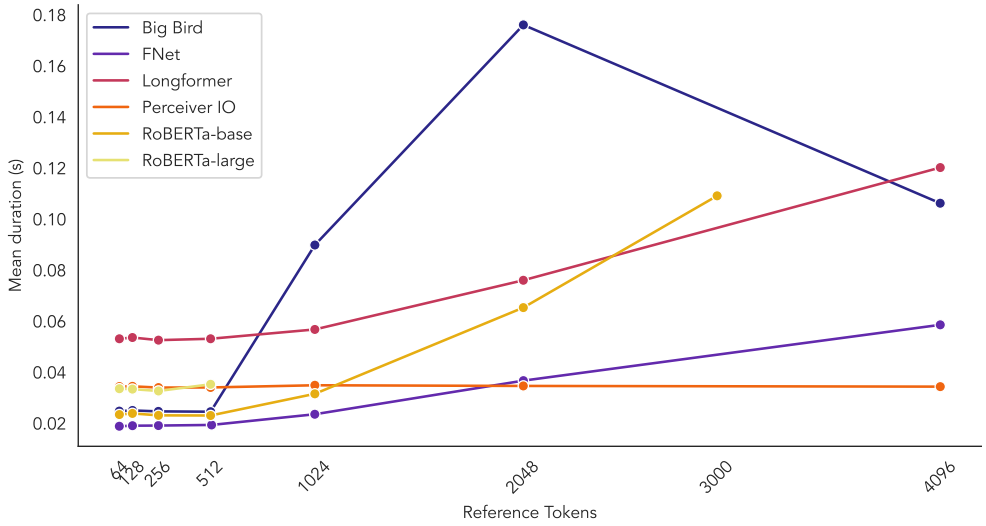
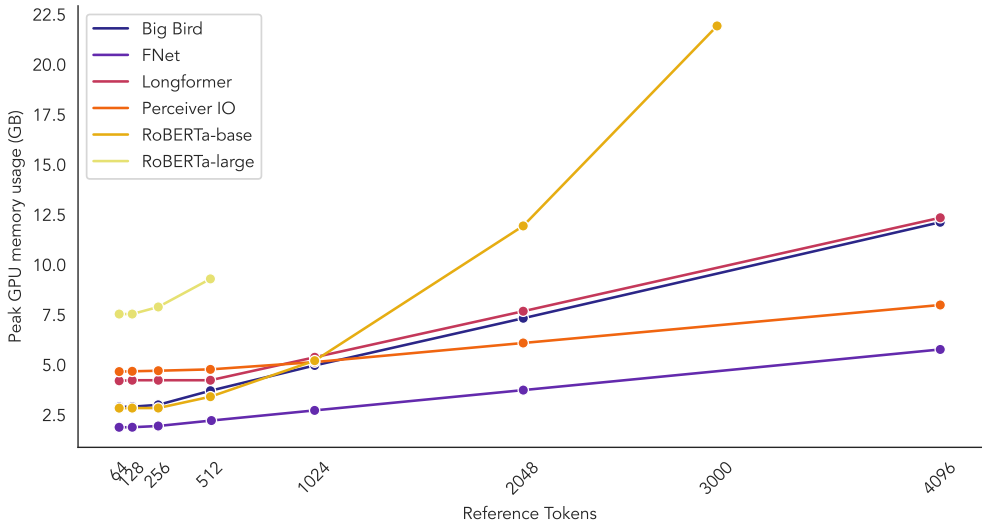


Figure 5.6.: Model sizes in terms of parameters



(a) Duration of the forward pass



(b) Peak GPU memory usage per device

Figure 5.7.: Computational cost with respect to the number of input reference tokens: RoBERTa scales quadratically. FNet, Longformer and Big Bird generally scale linearly. The Big Bird forward pass duration is adversely affected by the attention mechanism used, which is not ideal for shorter sequence lengths. For Perceiver IO, memory usage increases linearly while forward pass duration stays almost constant.

trainings on FEVER with a batch size of one to also fit long sequences. The findings are presented in Figure 5.7. It is worth noting that these results depend on the model implementations we used. Figure 5.7a shows the mean duration of a forward pass depending on the model and the sequence length. Figure 5.7b depicts GPU memory usage.

The results confirm the theoretical finding of the quadratic memory complexity of RoBERTa. The longest sequence that we could fit into GPU memory with RoBERTa-base was 3000 tokens long.

Longformer and Big Bird show linear growth of memory usage with the sequence length. However, we observe that Longformer also proves to be significantly slower to train than RoBERTa for short sequences, rendering it less beneficial in this setting. For Big Bird, we observed that the implementation we used (`block_sparse`) comes with a significant runtime overhead that is only amortised with longer sequences. This drawback can likely be mitigated by using the `original_attention` implementation for shorter sequences.

For FNet, our findings support the authors' claims: The model displays an increase in time and memory requirements that appears linear in the sequence length. Also, there do not seem to be large constants hidden in the Big-O notation, as FNet proves to be the by far most lightweight model we evaluate.

Perceiver IO shows almost constant runtime and linearly increasing GPU memory usage. That is due to its architecture, in which the core process module is independent of the input size and is only affected by the latent size. The only component that changes is the encoder cross-attention.

5.3.5. Detailed Studies

In this last subsection, we report the results of two detailed studies. The model that performed best in our experiments was Longformer. One of its key innovations is the global attention mask. In the first study, we look at how it affects predictive performance.

Secondly, we study our models' resistance to noise. This is motivated by the results showing that model performance does not always increase monotonically with longer input sequences. In our experiments, longer input sequences contained not only more evidence but also more noise. The second study examines this effect.

Claim	SEP	BOS	EOS	CLS	LA
✓	X	X	X	X	**74.79**
✓	X	✓	✓	✓	74.67
✓	✓	✓	✓	✓	74.23
X	✓	✓	✓	✓	73.37

Table 5.6.: Ablation study about the performance impact of using global attention for specific tokens: The checkmarks and crosses indicate whether the global attention was set to `true` or `false` for the claim tokens or the special tokens SEP (separator), BOS (beginning of sentence), EOS (end of sentence) and CLS (classification). The column LA indicates the validation accuracy of a Longformer on FEVER with BM25 retrieval. The first row is the configuration we used in the end. Therefore, its performance matches the Longformer-512 result on FEVER in Table 5.5.

Input	LA
claim	66.04
claim || BM25 results	73.69
claim || gold	**95.77**
claim || gold || BM25 results	86.47

Table 5.7.: Comparison of validation LA on FEVER with RoBERTa-256 depending on the input. || denotes concatenation. The claim-only setting performs worst. The best results are obtained when the claim is fed in only with the gold evidence. If after the gold evidence the other retrieval results are added, performance drops by almost ten percentage points.

Longformer Global Attention

In order to determine how important the global attention mask is for Longformer to perform well, we conduct an ablation study where we experiment with enabling and disabling it for the claim tokens and several special tokens. The results are shown in Table 5.6. We find that setting the global attention to `true` for the claim makes a difference, albeit relatively small. For all special tokens, it seems to be better to set the global attention to `false`. However, there are only minuscule differences between the experiments with the special tokens, which might be due to random weight initialisations.

Noise Resistance

In this short analysis, we study the influence of additional evidence that is not relevant for verifying a claim. In our formulation, we call that evidence *noise*, which is the set $E \setminus G_{\mathbf{c}}$. For this evaluation, we train a RoBERTa model with a sequence length of 256 tokens on FEVER entailment with varying inputs. All models are fine-tuned separately. We compare the following input settings:

- **claim**: only the claim
- **claim || BM25 results**: the claim and BM25 retrieval results
- **claim || gold**: only the claim and the gold evidence
- **claim || gold || BM25 results**: the claim, the gold evidence, and noise from the original BM25 retrieval results (with gold removed)

The results are shown in Table 5.7. We find that concatenating noise after the gold evidence leads to a performance drop of almost ten percentage points. This finding is an indication that the model is not perfectly able to select the relevant evidence from the input sequence, even if it is always located directly after the claim.

5.4. Discussion

After having presented the results, we now turn to discussing them and drawing our conclusions. We evaluate retrieval (5.4.1), the usefulness of evidence to entailment models (5.4.2) and assess the models we used (5.4.3). Then, we review the impact of using more efficient Transformer models on the computational cost (5.4.4) and the effects of feeding complete documents on interpretability (5.4.5). To conclude this section, we discuss the overall usefulness of longer sequences for improving predictive performance in fact-checking (5.4.6).

5.4.1. Retrieval

For retrieval, we found that sparse methods consistently outperformed dense ones. On the one hand side, we attribute this to the fact that Wikipedia articles are well named, in that their titles accurately describe what the article is about. Secondly, FEVER and FEVEROUS contain many claims concerning named entities that are often directly in the title of the relevant article. Lastly, DPR [39] was developed for passages and not whole documents. It is conceivable that it is challenging for the embedding model to project all relevant information in a document into one embedding. Possibly, that is why the exact keyword matching of BM25 works better. This explanation is supported by the observation that the gap between sparse and dense retrieval is much larger for FEVEROUS, of which the documents are much longer than those of FEVER. While we believe that fine-tuning DPR could have improved the results, we decided not to pursue this path due to its significant computational cost.

While the results we obtain with BM25 beat the baseline put forward by Thorne et al. [83], they are far from state-of-the-art. For FEVER, Bekoulis et al. [7] report the best approach at the time of their writing to be by Chakrabarty

et al. [15]. It is a solution tailored to the data set that uses a combination of the Google Custom Search API, Named Entity Recognition (NER) and dependency parsing. Reportedly, it achieves a fully supported accuracy of 94.4 for $k = 3$. However, for the purposes of our experiments, we only intended to compare entailment models for a given retrieval method. Therefore, the simple baseline we established was sufficient.

5.4.2. Usefulness of Explicit Evidence

An underlying assumption of this project was that models are, in fact, able to use the evidence presented to them to verify claims. First, we presented a positive indicator in Figure 4.3a. Then, in Table 5.2 we compare claim-only methods with such that receive gold evidence. In that, we directly compare relying on implicit knowledge to seeing explicit knowledge. We find the latter to clearly outperform the former, showing how explicit evidence is useful. Finally, we can say that the experiments *Gold Far Back* and *Uniform Gold* also show that it is helpful for the models we examine and that it benefits the entailment results.

5.4.3. Models

The models we evaluated used a variety of different techniques to reduce the quadratic complexity of Transformers. As varied their concepts, so too did their usefulness to our task. In the following, we assess the success we had with the models and argue potential reasons.

RoBERTa The RoBERTa model proved to be a strong baseline that was not easy to beat in predictive performance. However, due to its quadratic complexity, it was prohibitively costly to upscale to longer sequence lengths. Hence, it could not see more evidence than what fitted into its 512 tokens.

Longformer Longformer turned out to be the best-performing model for our task. It proved to be quick to fine-tune, was able to use the additional gold evidence it saw when increasing its sequence length, and scaled linearly in the input sequence length in terms of memory usage and forward pass time. In terms of experimental conditions, it likely benefited from the fact that we could fine-tune it from weights that were trained for 4096 tokens.

As we have seen in the ablation study (Subsection 5.3.5), the global tokens do not seem to be vital for it to perform well. The local and dilated window attention seems to be enough to discover the relevant parts of the input sequence.

Big Bird For Big Bird, too, we used a checkpoint that has been trained on a sequence length of 4096 tokens. While we observe that the prediction performance increases with longer inputs, the model is outperformed by Longformer by a consistent margin of 6-8% at all sequence lengths.

As our implementation did not support ETC mode, we could not enable global attention for the claim tokens, which might have harmed performance. However, as we have shown in Subsection 5.3.5, the effects of the global attention do not seem pronounced enough to explain the differences we find between Longformer and Big Bird. It appears as though the random tokens that Big Bird uses are not helpful for the model either. We cannot completely explain the performance gap and leave the reasoning for future work.

In terms of memory consumption, Big Bird behaves very similarly to Longformer. For forward pass duration, the `block_sparse` implementation was very slow for sequences of lengths 1024 and 2048, but it improved for 4096. We consider this a non-issue, given that our focus lies on longer sequences and that for shorter ones, the original attention implementation could be used.

FNet While our empirical analysis supports the authors' claims about a small computational footprint, in terms of predictive performance, FNet was appreciably outperformed in all settings by other models. Adding more evidence seemed only to worsen the results. It is worth noting that we fine-tuned the model from a checkpoint with only 512 tokens and, for longer sequences, extended the positional embeddings by repeating them. While this might explain why it could not handle longer sequences well, it is no reason why FNet struggled at its initial sequence length. However, given that FNet also seems to underperform RoBERTa on GLUE MNLI [89] by a substantial margin of around 20%[21], our results are not entirely unexpected.

Perceiver IO Our Perceiver IO checkpoint has been trained on 2048 tokens, which, due to the model's byte tokeniser, only corresponds to 451 reference tokens. In this setting, Perceiver IO is the model that gets closest to RoBERTa-large and Longformer in terms of predictive performance. However, we find that its performance rapidly decreases when trained with longer sequences.

We attribute this to multiple factors. First, our experiments showed that Perceiver IO is difficult to fine-tune to sequences longer than it was trained on. Extending position embeddings using random initialisation, repetition, and linear interpolation did not work well. The only method that trained reasonably within our computational budget was nearest neighbour interpolation. We

[21] https://paperswithcode.com/sota/text-classification-on-glue-mnli, accessed 2022-03-17

conjecture that the projection into latent space relies on the position embeddings, and the model needs to be re-trained more than others for this change of sequence length. We believe the encoder module to be sensitive to changes in the input and that it takes many iterations to recover from the change.

Secondly, for being able to retain the weights of the processing module, we had to keep the latent array size the same. That, however, introduced a bottleneck that we believe to have hurt performance. It is conceivable that this happens either because the encoder has difficulties extracting the relevant information from the input or because it could not be compressed in a way that the process models could still understand. In any case, these issues are not inherent problems of the model itself. With a higher computational budget, the model could be pre-trained from scratch with MLM and a larger input and latent array. Alternatively, it could be trained with a traditional tokeniser to accommodate more information in shorter sequence lengths. Still, the issue of it being hard to fine-tune to different input lengths remains.

In terms of computational cost, Perceiver IO showed desirable properties. Its forward pass duration and GPU memory usage only increased relatively slowly with the sequence length. However, it is worth noting that the latent size likely also needs to be scaled up to accommodate longer input sequences. We believe that the degree to which this is necessary is decisive for whether using Perceiver IO can be useful as a more efficient Transformer for fact-checking.

5.4.4. Reducing Computational Cost

One of our research questions was whether it is feasible to use more efficient Transformer models to reduce the computational cost while achieving the same or similar accuracy as current models, i.e., in our case, RoBERTa.

In a setting where only the entailment component of a pipeline is replaced by a more efficient Transformer and the rest is left the same, we found this rather not to be the case. FNet and Big Bird performed too poorly to match RoBERTa's predictive performance. Longformer outperformed it while using less memory, but, in turn, it was significantly slower. This might be advantageous if the lower memory consumption allows for using larger batch sizes or in an inference setting where memory is expensive. However, given our results, it seems unlikely that there are significant gains to be made. The most promising option might be Perceiver IO, which, while achieving worse results than RoBERTa-large by about 5% LA, used around one third less memory. In a setting where some predictive performance should be sacrificed for a smaller computational footprint, it might prove beneficial.

However, Longformer's capability to handle longer documents allowed us to skip the passage retrieval step while still obtaining performance very close to state-of-the-art methods for FEVER. This significantly reduces the computational costs of the entire pipeline. The amount of the savings depends on the complexity of the passage retrieval step in related work. KGAT [50] uses a dedicated LSTM or BERT model for passage retrieval, while MLA [43] has a custom-made deep method for it. With Longformer, we can remove these components completely and thereby simplify and speed up the pipeline.

5.4.5. Interpretability

In a real-world scenario, users often want to understand why a model classified a claim as supported or refuted. This fits the definition of *interpretability* by Biran and Cotton [10]: *"systems are interpretable if their operations can be understood by a human, either through introspection or through a produced explanation"*. For fact-checking, an intuitive way to give an explanation is to exhibit the concrete passages of evidence the model considered to come to its verdict.

In our setting, the entailment model was always fed complete evidence documents, which could be rather long. In the end, the model predicted a class without providing an explanation of which passages formed the basis for the verdict. This is an inherent drawback of feeding whole documents. With passage retrieval, it is transparent for the user which parts of the evidence the entailment model used to reach its verdict. If complete documents are fed, this is not directly the case anymore. With our approach, the pipeline only exhibits its sources on a document level.

5.4.6. Using Longer Sequences

Most importantly, we set out to research whether using longer input sequences that potentially contain more gold evidence is helpful for entailment. We found this to be the case with relatively short documents, such as with FEVER, and longer documents, such as with FEVEROUS. When comparing the results of the BM25 retrieval with the synthetic splits we constructed, it becomes apparent that the advantages of longer input sequences are more pronounced if the evidence is more spread out. This might be caused by either longer documents or corresponding retrieval characteristics. In the case *Gold Far Back*, where gold evidence only appears relatively late in the sequence (Figure 5.2c), we see significant improvements of Longformer-4096 over RoBERTa-512. This is also the case when the gold evidence is spread out uniformly across the sequence (Figure 5.2d). However, in the BM25 retrieval results that we examined

(Figure 5.2), the gold evidence is mainly located at the beginning with a rapid drop after around ranks 3 to 5.

Looking at the entailment results on the BM25 retrieval outputs listed in Table 5.5, we observe that performance increases with longer sequences up until a certain point, after which it starts decreasing again. We conjecture that this behaviour is due to the tradeoff between additional information and noise. While adding more gold evidence aids the model in its predictions, the longer sequences also add noise that can confuse it. Our study about noise resistance showed a drop in performance if noise is appended after the gold evidence. Hence, we believe that performance starts decreasing once there is too little gold evidence in the additional input to outweigh the added noise. The fact that the noise hurts performance indicates that the models we examined are not perfectly able to select the relevant evidence in the input sequence. If they were, we would observe monotonically increasing performance with longer sequences.

Taking this together, using longer sequence lengths can benefit prediction accuracy, especially when the data set consists of long documents or the retrieval method produces relatively spread-out results.

5.5. Future Work

Given our findings, we propose four directions future work could explore: considering multi-hop retrieval (5.5.1), improving models' resistance to noise (5.5.2), enhancing performance on the class *Not Enough Information* (5.5.3) and studying interpretability (5.5.4).

5.5.1. Multi-Hop Retrieval

Using more efficient Transformers allows entailment models to see more evidence. Nonetheless, there are cases where they will fail. One of them is if the gold evidence is not correctly retrieved in the first place and the model does not have the necessary information in its implicit knowledge. Retrieval failure is especially likely if a claim requires multi-hop reasoning. Our approach will often not work in such a setting. Future work might explore incorporating efficient Transformers with dedicated multi-hop retrieval methods [97] or iterative retrieval approaches that allow the entailment component to influence the retrieval. Thereby, the retrieval step could make a more informed choice of relevant evidence that is not only similar to the claim but also helpful for reasoning about its veracity.

5.5.2. Noise Resistance

In our detailed study about noise, we found that adding evidence that is not relevant to a claim hurts prediction performance on RoBERTa. Future work might elaborate on this issue by improving Transformers' capacity to find the relevant evidence in the input sequence and thereby increase their resistance to noise. However, even if long-sequence models make mistakes, they can be improved theoretically. Models with too short sequence lengths to even see the gold evidence will never correctly classify the sample unless they make a lucky guess.

5.5.3. Claims with *Not Enough Information*

As the example in Figure 5.5 illustrates, the class *Not Enough Information* (NEI) is almost never correctly predicted by our models. The problem of *knowing what you don't know* has been recognised as challenging in related work for QA [69] and fact-checking [65, 12]. For QA, multiple data sets [69, 30] and methods [80, 32, 92] have already been developed to address it. While some aim to tackle the problem using additional training data, others propose separate models that predict whether the model's response truly answers the question. Future work could try to transfer these approaches to the fact-checking task to improve model performance on predicting NEI.

5.5.4. Interpretability

Feeding complete documents and removing the explicit passage retrieval step makes interpretability more challenging. The pipeline can not directly reveal the specific parts of the evidence it considered for its verdict anymore. In future research, one could try to develop methods by which entailment models can disclose which parts of the input documents were relevant for the decision. This could be achieved by highlighting the relevant parts in the documents using attention activations[22] [94] or more advanced relevancy analysis, such as propagation based on the Deep Taylor Decomposition [16]. Alternatively, entirely separate models could generate textual explanations for the verdict [3].

[22] see https://github.com/jessevig/bertviz, accessed 2022-03-18

6. Conclusion

This thesis project tackled the fact-checking task where claims are verified against an explicit knowledge base. Our goal was to investigate whether using models that can handle longer input sequences for entailment would improve predictive and computational performance. We experimented with the more efficient Transformer models Longformer [8], Big Bird [99], Perceiver IO [36] and FNet [45]. These methods use different techniques to achieve sub-quadratic runtime and memory complexity in the input length. For our analysis, we built a complete fact-checking pipeline consisting of document retrieval and entailment. As our entailment models could handle long input sequences, it was not necessary to use explicit passage retrieval. We evaluated our models on the data sets FEVER [83], FEVEROUS [2] and FaVIQ [59].

For retrieval, we experimented with sparse (BM25 [71]) and dense (DPR [39]) approaches and found the former to outperform the latter in our setting. Hence, we used it as the basis for all of our entailment experiments.

The focus of this work lay on analysing the effects of the more efficient Transformer models in the entailment component. As a baseline, we compared them to a RoBERTa [49] model that could handle only a sequence length of up to 512 tokens due to its quadratic memory complexity. Our efficient Transformers could handle up to 4096 tokens. After establishing that entailment models could make use of explicit evidence, we found that providing more evidence was helpful and allowed for outperforming our baseline on FEVER, FEVEROUS and FaVIQ. However, this was only the case with Longformer. Big Bird, while increasing in performance with longer sequences, was consistently outperformed. FNet's results were significantly worse than all other models. Perceiver IO achieved comparable performance to RoBERTa at a sequence length of 512 tokens while using one third less memory. Nevertheless, we found it challenging to extend it to longer input sequences within a limited hardware budget.

Removing the explicit passage retrieval step and, therefore, letting the selection of relevant evidence to the entailment model requires future work to regain interpretability. However, it also simplifies the fact-checking pipeline and significantly reduces its computational cost. We showed that, for FEVER, using sparse document retrieval with Longformer achieves 97-99% of the performance of state-of-the-art methods that use costly dense document retrieval, explicit passage retrieval and a custom-tailored fact-checking approach.

In conclusion, our work showed that using efficient Transformers for fact-checking can lead to gains in predictive performance and savings in computational cost. Furthermore, it allowed the entailment models to consider longer evidence as input. Future work incorporating them with modern retrieval methods might achieve new levels of state-of-the-art performance.

Bibliography

[1] J. Ainslie, S. Ontañón, C. Alberti, V. Cvicek, Z. K. Fisher, P. Pham, A. Ravula, S. K. Sanghai, Q. Wang, and L. Yang. Etc: Encoding long and structured inputs in transformers. In *EMNLP*, 2020.

[2] R. Aly, Z. Guo, M. Schlichtkrull, J. Thorne, A. Vlachos, C. Christodoulopoulos, O. Cocarascu, and A. Mittal. Feverous: Fact extraction and verification over unstructured and structured information. *arXiv preprint arXiv:2106.05707*, 2021.

[3] P. Atanasova, J. G. Simonsen, C. Lioma, and I. Augenstein. Generating fact checking explanations. In *ACL*, 2020.

[4] I. Augenstein, C. Lioma, D. Wang, L. C. Lima, C. Hansen, C. Hansen, and J. G. Simonsen. Multifc: A real-world multi-domain dataset for evidence-based fact checking of claims. *arXiv preprint arXiv:1909.03242*, 2019.

[5] J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[6] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[7] G. Bekoulis, C. Papagiannopoulou, and N. Deligiannis. A review on fact extraction and verification. *arXiv preprint arXiv:2010.03001*, 2020.

[8] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[9] L. Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

[10] O. Biran and C. V. Cotton. Explanation and justification in machine learning: A survey. 2017.

[11] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. W. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens. *ArXiv*, abs/2112.04426, 2021.

[12] M. Bouziane, H. Perrin, A. Sadeq, T.-T. Nguyen, A. Cluzeau, and J. Mardas. Fabulous: Fact-checking based on understanding of language over unstructured and structured information. In *FEVER*, 2021.

[13] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

[14] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[15] T. Chakrabarty, T. Alhindi, and S. Muresan. Robust document retrieval and individual evidence modeling for fact extraction and verification. 2018.

[16] H. Chefer, S. Gur, and L. Wolf. Transformer interpretability beyond attention visualization. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 782–791, 2021.

[17] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading wikipedia to answer open-domain questions. In *ACL*, 2017.

[18] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, D. Belanger, L. Colwell, et al. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*, 2020.

[19] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ArXiv*, abs/1901.02860, 2019.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[21] T. Diggelmann, J. L. Boyd-Graber, J. Bulian, M. Ciaramita, and M. Leippold. Climate-fever: A dataset for verification of real-world climate claims. *ArXiv*, abs/2012.00614, 2020.

[22] Eurobarometer. Fake news and disinformation online, 2018. URL https://europa.eu/eurobarometer/surveys/detail/2183.

[23] W. E. Forum. The global risks report 2018. 2018. URL https://www3.weforum.org/docs/WEF_GRR18_Report.pdf.

[24] F. Fusco, D. Pascual, and P. W. J. Staar. pnlp-mixer: an efficient all-mlp architecture for language. *ArXiv*, abs/2202.04350, 2022.

[25] I. J. Goodfellow, Y. Bengio, and A. C. Courville. Deep learning. *Nature*, 521:436–444, 2015.

[26] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909, 2020.

[27] A. Hanselowski, C. Stab, C. Schulz, Z. Li, and I. Gurevych. A richly annotated corpus for different tasks in automated fact-checking. *ArXiv*, abs/1911.01214, 2019.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[29] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. *ArXiv*, abs/2006.03654, 2021.

[30] Q. Heinrich, G. Viaud, and W. Belblidia. Fquad2.0: French question answering and knowing that you know nothing. *ArXiv*, abs/2109.13209, 2021.

[31] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[32] M. Hu, F. Wei, Y. Peng, Z. Huang, N. Yang, and M. Zhou. Read + verify: Machine reading comprehension with unanswerable questions. *ArXiv*, abs/1808.05759, 2019.

[33] D. S. Hutchins, I. Schlag, Y. Wu, E. Dyer, and B. Neyshabur. Block-recurrent transformers. 2022.

[34] M. S. Islam, T. Sarkar, S. H. Khan, A.-H. M. Kamal, S. M. M. Hasan, A. Kabir, D. Yeasmin, M. A. Islam, K. I. A. Chowdhury, K. S. Anwar, A. A. Chughtai, and H. Seale. Covid-19–related infodemic and its impact on public health: A global social media analysis. *The American Journal of Tropical Medicine and Hygiene*, 103:1621–1629, 2020.

[35] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, 2021.

[36] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[37] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*, 2021.

[38] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017.

[39] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Y. Wu, S. Edunov, D. Chen, and W. tau Yih. Dense passage retrieval for open-domain question answering. *ArXiv*, abs/2004.04906, 2020.

[40] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[42] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451, 2020.

[43] C. Kruengkrai, J. Yamagishi, and X. Wang. A multi-level attention model for evidence-based fact checking. In *FINDINGS*, 2021.

[44] T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*, 2018.

[45] J. P. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontañón. Fnet: Mixing tokens with fourier transforms. *ArXiv*, abs/2105.03824, 2021.

[46] H. J. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *KR*, 2011.

[47] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[48] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

[49] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[50] Z. Liu, C. Xiong, M. Sun, and Z. Liu. Fine-grained fact verification with kernel graph attention network. In *ACL*, 2020.

[51] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[52] M. Mahoney. Large text compression benchmark, 2011.

[53] C. D. Manning, P. Raghavan, and H. Schütze. Scoring, term weighting and the vector space model. *Introduction to information retrieval*, 100:2–4, 2008.

[54] S. Mussmann and S. Ermon. Learning and inference via maximum inner product search. In *ICML*, 2016.

[55] U. of Baltimore. The economic cost of bad actors on the internet, 2019. URL https://s3.amazonaws.com/media.mediapost.com/uploads/EconomicCostOfFakeNews.pdf.

[56] Y. Onoe, M. J. Zhang, E. Choi, and G. Durrett. Creak: A dataset for commonsense reasoning over entity knowledge. *ArXiv*, abs/2109.01653, 2021.

[57] W. Ostrowski, A. Arora, P. Atanasova, and I. Augenstein. Multi-hop fact checking of political claims. *arXiv preprint arXiv:2009.06401*, 2020.

[58] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *EMNLP*, 2016.

[59] J. Park, S. Min, J. Kang, L. Zettlemoyer, and H. Hajishirzi. Faviq: Fact verification from information-seeking questions. *ArXiv*, abs/2107.02153, 2021.

[60] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[61] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases? *ArXiv*, abs/1909.01066, 2019.

[62] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, et al. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020.

[63] M. N. Rabe and C. Staats. Self-attention does not need $\mathcal{O}(n^2)$ memory. *arXiv preprint arXiv:2112.05682*, 2021.

[64] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. Compressive transformers for long-range sequence modelling. *ArXiv*, abs/1911.05507, 2020.

[65] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L. A. Hendricks, M. Rauh, P.-S. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. F. J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. M. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J.-B. Lespiau, M. Tsimpoukelli, N. K. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. de Masson d'Autume, Y. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, C. Jones, J. Bradbury, M. Johnson, B. A. Hechtman, L. Weidinger, I. Gabriel, W. S. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. W. Ayoub, J. Stanway, L. L. Bennett, D. Hassabis, K. Kavukcuoglu, and G. Irving. Scaling language models: Methods, analysis&insights from training gopher. 2021.

[66] C. Raffel, N. M. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683, 2020.

[67] A. Rajaraman and J. D. Ullman. *Data Mining*, page 1–17. Cambridge University Press, 2011. doi: 10.1017/CBO9781139058452.002.

[68] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[69] P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. In *ACL*, 2018.

[70] A. Roberts, C. Raffel, and N. M. Shazeer. How much knowledge can you pack into the parameters of a language model? *ArXiv*, abs/2002.08910, 2020.

[71] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389, 2009.

[72] A. Saakyan, T. Chakrabarty, and S. Muresan. Covid-fact: Fact extraction and verification of real-world claims on covid-19 pandemic. In *ACL/IJCNLP*, 2021.

[73] G. Salton and M. McGill. Introduction to modern information retrieval. 1983.

[74] D. Sarkar, R. Bali, and T. Sharma. Practical machine learning with python. In *Apress*, 2018.

[75] M. Schuster and K. Nakajima. Japanese and korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012.

[76] T. Schuster, D. J. Shah, Y. J. S. Yeo, D. Filizzola, E. Santus, and R. Barzilay. Towards debiasing fact verification models. *ArXiv*, abs/1908.05267, 2019.

[77] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *ArXiv*, abs/1508.07909, 2016.

[78] D. Stammbach and E. Ash. e-fever: Explanations and summaries for automated fact checking. In *Proceedings of the 2020 Truth and Trust Online Conference (TTO 2020)*, page 32. Hacks Hackers, 2020.

[79] D. Stammbach and G. Neumann. Team domlin: Exploiting evidence enhancement for the fever shared task. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 105–109, 2019.

[80] C. Tan, F. Wei, Q. Zhou, N. Yang, W. Lv, and M. Zhou. I know there is no answer: Modeling answer validation for machine reading comprehension. In *NLPCC*, 2018.

[81] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[82] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

[83] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018.

[84] J. Thorne, A. Vlachos, O. Cocarascu, C. Christodoulopoulos, and A. Mittal. The fever2. 0 shared task. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 1–6, 2019.

[85] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. *ArXiv*, abs/2105.01601, 2021.

[86] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[87] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[88] D. Wadden, K. Lo, L. L. Wang, S. Lin, M. van Zuylen, A. Cohan, and H. Hajishirzi. Fact or fiction: Verifying scientific claims. *ArXiv*, abs/2004.14974, 2020.

[89] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ArXiv*, abs/1804.07461, 2018.

[90] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, 2019.

[91] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *ArXiv*, abs/2006.04768, 2020.

[92] X. Wang, L. Shou, M. Gong, N. Duan, and D. Jiang. No answer is better than wrong answer: A reflection model for document level machine reading comprehension. In *FINDINGS*, 2020.

[93] J. Welbl, P. Stenetorp, and S. Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.

[94] S. Wiegreffe and Y. Pinter. Attention is not not explanation. In *EMNLP*, 2019.

[95] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, 2018.

[96] C. Wu, F. Wu, T. Qi, and Y. Huang. Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*, 2021.

[97] W. Xiong, X. L. Li, S. Iyer, J. Du, P. Lewis, W. Y. Wang, Y. Mehdad, W. tau Yih, S. Riedel, D. Kiela, and B. Oğuz. Answering complex open-domain questions with multi-hop dense retrieval. *ArXiv*, abs/2009.12756, 2021.

[98] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[99] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

[100] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *ICML*, 2019.

[101] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. Ernie: Enhanced language representation with informative entities. In *ACL*, 2019.

[102] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Appendices

A. Training Details

In this appendix, we disclose additional details about our training. Table A.1 contains more information about the training on the *Uniform Gold* data split.

Model	Sequence Length	Epochs	Batch Size	Training	Peak GPU Usage
Unit	*tokens*	*#*	*#*	*hours*	*GB*
RoBERTa-large [49]	512	3	2	7.1	16.6
Longformer [8]	512	3	2	7.7	11.0
Longformer [8]	1024	3	2	10.7	13.9
Longformer [8]	2048	3	2	17.3	19.7
Longformer [8]	4096	3	1	32.3	19.4
FNet [45]	512	3	8	2.7	10.8
FNet [45]	1024	3	8	5.2	14.4
FNet [45]	2048	3	8	5.6	21.4
FNet [45]	4096	3	4	8.8	21.9
Perceiver IO [36]	2048	6	16	5.0	13.1
Perceiver IO [36]	4096	6	16	5.3	14.4
Perceiver IO [36]	8192	6	16	5.9	17.0
Perceiver IO [36]	16384	3	4	5.3	9.3
Big Bird [99]	512	4	2	8.5	9.1
Big Bird [99]	1024	4	2	14.2	12.0
Big Bird [99]	2048	4	2	24.7	14.9
Big Bird [99]	4096	5	1	52.4	15.4

Table A.1.: Training Details of *Uniform Gold*

In Table A.2, we list the training hyper-parameters (effective batch size and learning rate), the overall number of parameters and the sequence lengths of the checkpoints for all models we experimented with.

Model	**RoBERTa**		**Longformer**	**Big Bird**	**FNet**	**Perceiver IO**
Variant	*base*	*large*	*base*	*base*	*base*	-
Effective Batch Size	8		8	8	8	16
Learning Rate	5e-6		5e-6	5e-6	1e-5	1e-5
Checkpoint (ref. tokens)	512		4096	4137	466	455
Number of Parameters	125M	355M	149M	128M	86M	211M

Table A.2.: Model Hyper-Parameters and Architecture Information

B. Acronyms

BPE Byte Pair Encoding

CNN Convolutional Neural Network

CPC Contrastive Predictive Coding

DPR Dense Passage Retrieval

ETC Extended Transformer Construction

IR Information Retrieval

ITC Internal Transformer Construction

KDE Kernel Density Estimate

LA Label Accuracy

LSTM Long Short-Term Memory

MIPS Maximum Inner Product Search

ML Machine Learning

MLM Masked Language Modelling

MLP Multi-Layer Perceptron

NER Named Entity Recognition

NLI Natural Language Inference

NLP Natural Language Processing

NSP Next Sentence Prediction

QA Question Answering

RNN Recurrent Neural Network

RTE Recognising Textual Entailment

C. Glossary

Claim, Hypothesis A statement that is to be verified

Evidence, Premise A statement that is presumed to be correct and against which *claims* are verified

Retrieval The task of finding *evidence* that is relevant to a given *claim*

Entailment, Verdict Prediction, Claim Verification, RTE The task of classifying whether a piece of *evidence* supports a *claim* or not [7]

Fact-Checking, Fact Verification *Retrieval* and *entailment* combined

Multi-Hop Reasoning *Fact-checking*, where only a set of connected evidence pieces leads to the final verdict [57]

Fake News Detection *Fact-checking* applied to news articles or social media posts [7]

Efficient Transformer Summary term for a Transformer-based model [87] with sub-quadratic computational complexity in the input sequence length [82]

Terms in a line separated by commas denote synonyms.