BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

DESIGNING A MULTILINGUAL FACT-CHECKING DATASET FROM EXISTING QUESTION-ANSWERING DATA

TVORBA VÍCEJAZYČNÉ DATOVÉ SADY PRO FACT-CHECKING Z EXISTUJÍCÍCH DAT PRO ODPOVÍDÁNÍ NA OTÁZKY

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. DANIEL KAMENICKÝ**
AUTOR PRÁCE

SUPERVISOR **Ing. MARTIN FAJČÍK**
VEDOUCÍ PRÁCE

BRNO 2023

Master's Thesis Assignment

143409

Institut:	Department of Computer Graphics and Multimedia (UPGM)
Student:	**Kamenický Daniel, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Software Engineering
Title:	**Designing a Multilingual Fact-Checking Dataset from Existing Question-Answering Data**
Category:	Speech and Natural Language Processing
Academic year:	2022/23

Assignment:

1. Describe the problem of multilingual fact-checking and the importance of using sources from different languages.
2. Research available data sources for multilingual fact-checking and describe their problem formulation.
3. Research how existing question-answering (QA) datasets can be converted into fact-checking (FC) datasets.
4. Design an automatic approach for QA to FC dataset conversion
5. Implement your design.
6. Analyze the properties of the newly converted dataset.
7. Evaluate the difficulty of the problem introduced in your dataset by the baseline model.
8. Create an A1 poster presenting your work.

Literature:

- Thorne, James, et al. "FEVER: a Large-scale Dataset for Fact Extraction and VERification." *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018.
- Park, Jungsoo, et al. "FaVIQ: FAct Verification from Information-seeking Questions." *arXiv preprint arXiv:2107.02153* (2021).
- Nørregaard, Jeppe, and Leon Derczynski. "DANFEVER: claim verification dataset for Danish." *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*. 2021.
- SLÁVKA, Michal. Multilingual Open-Domain Question Answering. Brno, 2020. *Master’s thesis*. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Martin Fajčík
- Ruder, S. and Sil, A., 2021, November. Multi-Domain Multilingual Question Answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts* (pp. 17-21).

Requirements for the semestral defence:
assignment items 1-4

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Fajčík Martin, Ing.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	17.5.2023
Approval date:	2.3.2023

Abstract

This thesis adresses the lack of multilingual fact-Checking datasets, which contain annotated evidence grounding the supporting or refuting verdict for a fact. Therefore, this work explores the conversion into the fact-checking dataset from an already existing question-answering dataset. In this work, two approaches for converting question-answer pairs into claims are studied. The first approach is to create a dataset based on a monolingual pre-trained seq-2-seq model T5. The model is trained on an English dataset and the inputs and outputs are translated into the desired languages. The second approach is to use the multilingual mT5 model, which can take input and generate output in the desired language. For multilingual model, training datasets need to be translated. The main problem of this work is the machine translation, which achieved around 30 % success rate in a *low-resource* languages. The experiments showed better results for claims generated from monolingual model using machine translation. On the other hand, the claims generated from multilingual model achieved a success rate of 73 % compared to monolingual model with a success rate of 88 %. Finally, to analyze possible biases label specific claim biases, a logistic-regression based TF-IDF classifier is trained. The classifier, that computes the probability of the claim's veracity just from itself achieves accuracy close to 0.5 for both converted datasets. Thus the converted datasets can be challenging for fact-checking models.

Abstrakt

Tato práce se zabývá nedostatkem vícejazyčných datových sad pro kontrolu faktů, které by obsahovaly důkazy podporující nebo vyvracející fakt. Proto se tato práce zabývá převodem datového souboru pro kontrolu faktů z již existujícího datového souboru otázek a odpovědí. V této práci jsou studovány dva přístupy ke konverzi datové sady. Prvním přístupem je vytvoření datové sady založené na jednojazyčném předem natrénovaném seq-2-seq modelu T5. Model je trénován na anglickém datovém souboru. Vstupy a výstupy jsou překládány do požadovaných jazyků. Druhým přístupem je využití vícejazyčného modelu mT5, který přebírá vstup a generuje výstup v požadovaném jazyce. Pro vícejazyčný model je zapotřebí přeložit trénovací datové sady. Jako hlavní problém této práce se ukázal překlad, který v málo zdrojovém jazyce dosáhl kolem 30 % úspěšnosti. Experimenty ukázaly lepší výsledky v tvrzeních generovaných z jednojazyčného modelu s využitím strojového překladu. Na druhou stranu, tvrzení generované z vícejazyčného modelu dosáhly úspěšnosti 73 % oproti tvrzením z jednojazyčného modelu s dosaženou úspěšností 88 %. Modely byly vyhodnoceny modelem ověřování faktů založeném na TF-IDF. Dosažená přesnost modelu na obou datových sadách se blíží 0,5. Z toho lze usoudit, že výsledné datové sady mohou být náročné pro modely ověřování faktů.

Keywords

Natural Language Processing, Fact-Checking, Information Retrieval, Multilingual, Transformers, mDPR

Klíčová slova

Zpracování Přirozeného jazyka, Ověřování Faktů, Získávání Informací, Transformers, mDPR

Reference

KAMENICKÝ, Daniel. *Designing a Multilingual Fact-Checking Dataset from Existing Question-Answering Data.* Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Martin Fajčík

Rozšířený abstrakt

Vzhledem k tomu, že se nepravdivé informace a falešné zprávy stále šíří po internetu a sociálních sítích, je pro udržení pravdivého digitálního prostředí nezbytná potřeba ověřovacích prostředků. V dnešní digitální době je na internetu dostupné obrovské množství informací a pro jednotlivce může být obtížné rozlišit mezi důvěryhodnými a nedůvěryhodnými zdroji. Ověřování faktů může pomoci identifikovat a opravit chyby, zabránit šíření dezinformací a podpořit důvěru v informacích prezentovaných v textu přirozeným jazykem.

Ověřování faktů se týká zjišťování pravdivosti informací, které se nacházejí v textové podobě, ať už na webových stránkách, sociálních médiích, novinách nebo jiných zdrojích. S rozsahem a různorodostí dostupných informací je stále obtížnější rozlišit mezi pravdivými a nepravdivými tvrzeními.

V oblasti zpracování přirozeného jazyka se využívají techniky strojového učení a umělé inteligence ke zpracování a analýze textu. Existují metody automatického ověřování faktů, které se snaží identifikovat a klasifikovat pravdivá, nepravdivá a sporná tvrzení. Tyto metody zahrnují extrakci a porovnání informací z různých zdrojů, analýzu jazyka a kontextu, a také využívají dostupná data a znalosti pro srovnání a validaci tvrzení.

Přestože se technologie zpracování přirozeného jazyka neustále zlepšuje, ověřování faktů je stále náročný úkol. Existují různé výzvy, kterým je třeba čelit, jako jsou rychlost a objem informací, přítomnost zaujatosti a dezinformace, a také složitost samotného jazyka.

Ověřování faktů je zásadní pro důvěryhodnost a spolehlivost informací, které konzumujeme. Správné rozpoznání pravdivých a nepravdivých tvrzení má široké uplatnění, například v novinářství, vědeckém výzkumu, právu, ale také pro obyčejného člověka při orientaci ve světě informací.

Tato práce se zabývá problematikou ověřování faktů v oblasti zpracování přirozeného jazyka. Přesněji v nedostatcích vícejazyčných datových sad, které jsou potřeba k správnému natrénování vícejazyčného modelu. Na základě nejlepšího vědomí autora a získaných znalostí bylo zjištěno, že v dnešní době existuje pouze jedna datová sada X-Fact pro vícejazyčné ověřování faktů. Tato datová sada obsahuje jeden z velkých nedostatků a to chybějicí důkazy podporující nebo vyvracející fakta, které jsou obsaženy v datové sadě. Toto zjištění bylo hlavním motivem pro vytvoření této práce, jelikož je v prácí zastáván názor, který klade důraz pro zakomponování důkazu do datové sady.

To vedlo k myšlence převést existující datové sady otázek a odpovědí do datových sad ověřující fakta. Pokus o konverzi datové sady byl již představen v publikaci FaVIQ. Zdá se, že konverze má slibné výsledky, jelikož dnešní datové sady otázek a odpovědí obsahují vše potřebné k jejich konverzi na datové sady pro kontrolu faktů. Páry otázek a odpovědí převedené na tvrzení s důkazy získanými z modelu získávání informací se ukázaly jako dostatečné pro konverzi do datového souboru pro kontrolu faktů. Datová sada pro kontrolu faktů pak obsahuje všechny požadované části a to zejména důkazy faktů. Celá práce zkoumá rožšíření konverze pro vícejazyčné datové sady.

V této práci jsou navržené dva přítupy pro vytváření vícejazyčné datové sady. Prvním přístupem je natrénování jednojazyčného seq-2-seq modelu T5 na anglické datové sadě. Tento model pak převádí otázky a opovědi z různých jazyků získané z vícejazyčných datových sad otázek a odpovědí. Otázky a odpovědi jsou přeložené pomocí strojového překladu do anglického jazyka a poté předány modelu pro vygenerování požadovaného tvrzení. Tvrzení je poté přeloženo zpátky do požadovaného jazyka. Druhým přístupem je natrénování vícejazyčného modelu, který dokáže převzít vstup a vygenerovat tvrzení již v požadovaném jazyce. Pro natrénování vícejazyčného modelu je zapotřebí přeložení trénovacích datových sad. Tyto dva přístupy jsou poté v práci porovnávány.

Modely byly poté vyhodnoceny na základě správnosti překonvertovaných tvrzení. Výsledky ukázaly, že model T5 dosáhl úspěšnosti 88 % oproti modelu vícejazyčného modelu mT5, který dosáhl úspěšnosti 73 %. Tyto výsledky musí být podloženy faktem, že využité modely pro strojový překlad dosahovaly poměrně špatných výsledků a to zejména na jazycích s nízkými zdroji (low-resource). Z překorventovaných tvrzeních a ostatních důležitých částí (důkaz, označení pravdivosti) byly vytvořené výsledné datové sady. První datová sada obsahuje tvrzení, které byly překonvertovány modelem T5. Druhá datová sada obsahuje tvrzení, které byly překonvertovány modelem mT5. Datové sady byly poté vyhodnoceny na základě obtížnosti pro model ověřování faktů založeném na TF-IDF. Výsledky ukázaly, že dosažená přesnost modelu na obou datových sadách se blížila hodnotě 0.5. To znamená že úspěšnost modelu ověřování faktů není o nic lepší jak náhodný výběr. Z toho lze usoudit, že výsledné datové sady mohou být náročné pro modely ověřování faktů. Na základě všech informací a výsledků uvedených v této práci je závěrem, že konverze datových sad otázek a odpovědí může být velmi přínosná pro budoucnost vícejazyčných modelů ověřování faktů.

Designing a Multilingual Fact-Checking Dataset from Existing Question-Answering Data

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Ing. Martin Fajčík. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Daniel Kamenický
May 16, 2023

Acknowledgements

I would like to thank my supervisor Ing. Martin Fajčík for his guidance, enthusiasm, patience and willingness to answer all my questions or help with any issues along the way. Thank You.

Contents

Chapter 1

Introduction

As false information and fake news continue propagating throughout the internet and social networks, the need for fact-checking operations becomes necessary in order to maintain a truthful digital environment. In today's digital age, there is a vast amount of information available online, and it can be difficult for individuals to distinguish between credible sources and unreliable sources. Fact-checking can help to identify and correct errors, prevent the spread of misinformation, and promote trust in the information presented in natural language text. Additionally, in certain fields like journalism, fact-checking is a crucial step to ensure the credibility of the news and protect the public from being misinformed.

From a social and psychological perspective, humans have been proven irrational [27] and vulnerable when differentiating between real and fake news. In other words, fake news can gain broad public trust relatively easier than truthful news because individuals tend to trust fake news after constant exposure, if it confirms their pre-existing beliefs, or simply due to the obligation of participating socially and proving a social identity.

Based on these findings, efforts are being made to come up with a solution that can distinguish between real and fake news. Fact verification efforts are divided into two distinct approaches, namely manual and automated fact-checking. Manual fact-checking is time consuming and not a sustainable long-term solution like the manual solutions of PolitiFact.com [1], FactCheck.org [2], FEVER [53], X-Fact [19] and many others [51][62][33]. The amount of disinformation increases over time and the people verifying the facts cannot keep up with the increase in disinformation. On the other hand manual fact-checking is considered to be the most reliable method of verifying the accuracy of information. The other approach is automated fact-checking that has the advantage of being able to perform verification quickly and on a large scale. Automated fact-checking systems can analyze large amounts of text and identify potential errors or inaccuracies much faster than a human fact-checker could. However, there are also some limitations to automated fact-checking. One of the main limitations is that automated fact-checking systems rely on pre-existing knowledge and data, which can be incomplete or out of date. Additionally, automated fact-checking systems may not be able to understand the nuances and context of natural language text, which can lead to false positives or negatives.

Another major disadvantage is that information can spread beyond the boundaries of language in which it was created. Nowadays, there are several fact-checking models that are based on the English language [38][53][33]. To verify a fact from another language,

[1] https://www.politifact.com

[2] https://www.factcheck.org

a translation is needed, but as the results of this work showed, the translation may not always be correct or not everything can be translated. The problem is all the more obvious for *low-resource* languages. These language can have relatively small vocabulary, limited training data or they exhibit complex grammar, sentence structures, and rich morphology. Therefore there is an effort to create a multilingual model for verifying facts. However, quality data is required to train a multilingual model. Based on the author's best knowledge, one multilingual dataset named X-Fact [19] was found, which is intended for training and validating a multilingual model, but the individual samples of the dataset lack evidence where the confirmation or refutation of the fact occurred.

This leads to an idea to convert existing question-answer datasets to fact-checking datasets. The first attempt to convert dataset was introduced in paper [38]. The conversion seems to has a promising results as today's question-answer datasets contain everything we need for converting them into fact-checking datasets. It is shown that question-answer pairs converted into the claim with the evidence retrieved from information retrival model are sufficient for the conversion into the fact-checking dataset. A fact-checking dataset then contains all the required parts. Especially evidence, which is considered as the main deficiency in current state-of-the-art and also the research subject of this work.

The rest of the thesis is organised as follows. Chapter 2 introduces the basics of all the topics related to and essential to this thesis. The datasets that are considered to be related to this thesis and all the information taken from it's knowledge are described in Chapter 3. Subsequently, the information retrieval models are described in Chapter 4. Chapter 5 presents all the proposed systems for the conversion of dataset. The experimental setup with all the metrics used in this thesis is then described in Chapter 6. The experimental evaluation of the converted datasets and other experiments performed within this thesis are discussed in Chapter 7. Finally, the thesis is concluded in Chapter 8.

Chapter 2

Preliminaries

Nowadays, systems are based on existing pre-trained models. Most of the models benefit from prior information that has been developed in the past and try to adapt it for a specific use case. One of the basic qualities required to create something new and beneficial to society is to understand the work that has already been published and be able to apply and possibly modify it to achieve better results.

Several new papers were published, such as the BERT model [15], the T5 model[42] or the GPT model [41], which are based on the knowledge gained from the paper discussing transformer [56]. Nowadays, there are publications of models such as [35] that are still being experimented with. There is still an ongoing effort to find out the limits of the models.

One of the model is the Recurrent Neural Network [50][17]. The architecture of RNN is a variation of a basic neural network. RNNs are good for processing sequential data such as natural language processing and audio recognition. But there is one major issue that the RNNs suffer from short-term-memory problems. One of the mostly known issue is the vanishing gradient problem. This is the main problem during the training as the gradients for the weights at each layer are computed via the Chain Rule, their gradient values will exponentially shrink as it propagates through each time step, eventually "vanishing".

Long Short-Term Memory [57] (LSTM) and Gated Recurrent Unit [10] (GRU) were proposed to mitigate the vanishing gradient problem. LSTMs use a memory cell that can store information over long periods of time and gating mechanisms that allow them to selectively forget or remember information. GRUs are similar to LSTMs, but they have fewer gating mechanisms and are computationally less expensive.

However RNN, LSTM, and GRU are all types of recurrent neural networks that are designed to work with sequential data, such as natural language text. They are all based on the idea of having a „memory“ that allows the network to keep track of previous inputs, enabling the network to model sequential dependencies in the data. Additionally, all three architectures involve passing information between units of the network in a recurrent fashion. This means that the output of a unit can serve as input to another unit in the network, allowing information to flow backwards and forwards in the network.

Thus, even though the vanishing gradient problem has been mitigated, there is still the problem that RNN type models are very slow to train, so much so that truncated back-propagation [46] is used to update the parameters.

2.1 Transformer

Since RNN-based models use sequential data flow, they cannot be parallelized as well as the other type of neural networks called transformer [56]. Transformers are a type of neural network architecture that have revolutionized natural language processing once again. The transformer architecture utilizes a self-attention mechanism, which allows the model to attend to different parts of the input sequence during training.

This self-attention mechanism is able to learn non-trivial alignments between words in the input sequence, which helps the model to better understand the relationships between words in a sentence. This is in contrast to previous approaches such as recurrent neural networks, which process the input sequence one word at a time and have difficulty capturing long-term dependencies. Transformers replaced RNNs in many areas, such as language translation, text classification, question answering, and became new state-of-the-art architecture for language modeling [25].

2.1.1 Model architecture

The network employs an encoder-decoder architecture much like RNNs shown in figure 2.1. Only some parts of the architecture are described in this paper. A more precise and complete description of all parts can be found in the publication [56]. The difference is that the input sequence can be passed in parallel compared to RNNs which need to take the input sequence word by word because of the hidden state dependencies. The input sentence is passed to the encoder that generates the input embeddings simultaneously for each word. The decoder then generates an output sequence of symbols. At each step, the model is auto-regressive and consumes the previously generated symbols as the next input when generating the next step.

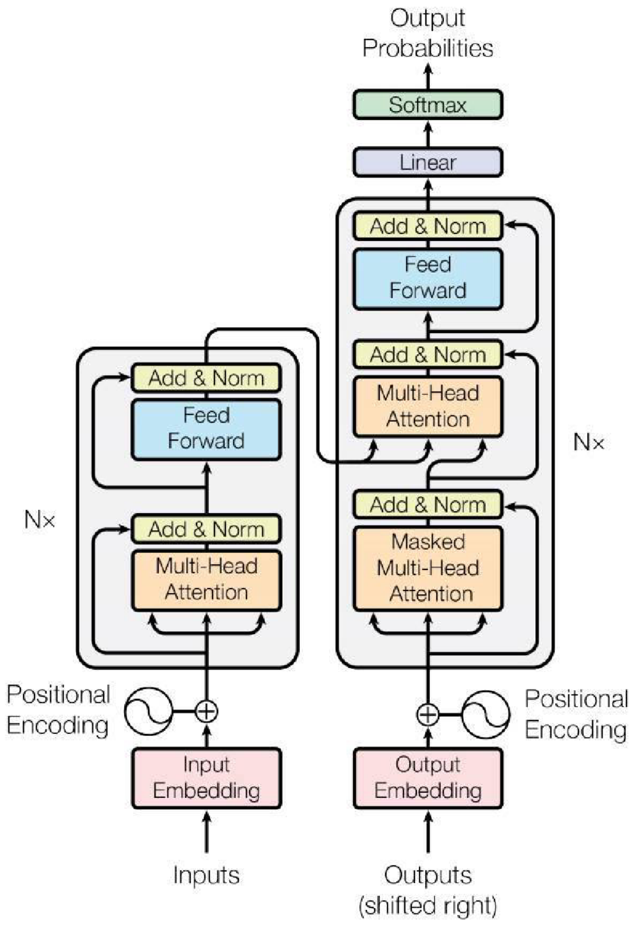

Figure 2.1: Encoder-Decoder structure of the transformer architecture [56].

Scaled Dot-Product Attention

Scaled Dot-Product Attention is a key component of the self-attention mechanism used in transformers for natural language processing. It involves computing the dot product between a query vector and a set of key vectors, and then using the resulting scores to weight the corresponding value vectors.

Query vector, key vector, and value vector are components of the attention mechanism. In the attention mechanism, the input sequence is mapped into these three vectors using learned linear projections. The query vector is used to attend to specific parts of the input sequence by computing the similarity between the query vector and each key vector. The similarity scores are then used to weight the corresponding value vectors, which are summed up to produce the attended output.

To obtain the query, key, and value vectors, three weight matrices, $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, and $\boldsymbol{W}_v$, are multiplied with the input sequence, and they project the input sequence into the query vector, key vector, and value vector, respectively:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_q, \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_k, \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_v. \tag{2.1}$$

where $\boldsymbol{X}$ is the input sequence, each weight matrix is learned during training to optimize the performance of the model, and the size of the weight matrices depends on the desired dimensionality of the query, key, and value vectors.

Each of the vectors extracts different components of the input token. Thus, for each input token x_i, the query vector q_i, the key k_i and the value v_i are extracted, where i represents the index of the token in the input sequence. The computation is processed simultaneously. Matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are then used to compute the attention matrix for each word using formula:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{2.2}$$

The computation of the attention function on the set of queries is performed simultaneously, packed in the matrix $\boldsymbol{W}_q$ and keys and values are also packed into matrices $\boldsymbol{W}_k$ and $\boldsymbol{W}_v$. Scaled Dot-Product Attention allows the model attend to different parts of the input sequence based on the query, capturing complex dependencies between tokens in the sequence.

Multi-Head Attention

Multi-Head Attention is an extension of the Scaled Dot-Product Attention mechanism used in transformers for natural language processing. It allows the model to attend to different parts of the input sequence simultaneously, by computing multiple attention functions in parallel. The input query vector $\boldsymbol{Q} \in \mathbb{R}^{d_q}$, key vector $\boldsymbol{K} \in \mathbb{R}^{d_k}$, and value vectors $\boldsymbol{V} \in \mathbb{R}^{d_v}$ are linearly projected h times with different learned linear projections $\boldsymbol{W}_q^i \in \mathbb{R}^{d_q \times d_h}$, $\boldsymbol{W}_k^i \in \mathbb{R}^{d_k \times d_h}$, and $\boldsymbol{W}_v^i \in \mathbb{R}^{d_v \times d_h}$ respectively, as each of these linear projections are learned independently.

Here, d_q, d_k, and d_v represent the dimensions of the query vector, key vector, and value vectors, respectively. h represents the number of times these vectors are linearly projected, and i represents the index of each projection. $\boldsymbol{W}_q^i$, $\boldsymbol{W}_k^i$, and $\boldsymbol{W}_v^i$ represent the learned linear projections for query, key, and value vectors, respectively. The outputs from each of the attention head is then concatenated and once again projected, resulting in the final values, as depicted in Figure 2.2.

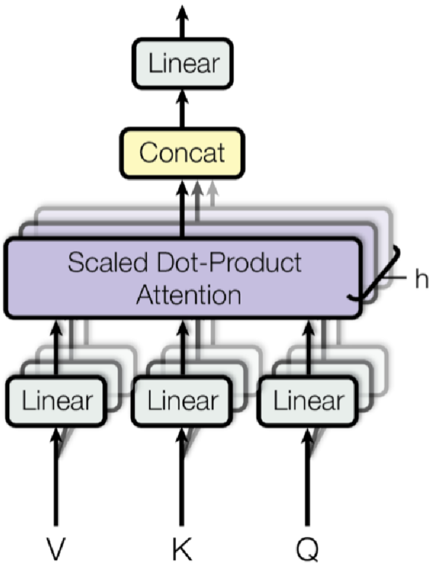

Figure 2.2: Multi-Head Attention layer

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions.

2.2 BERT

BERT [15] stands for Bidirectional Encoder Representations from Transformers and it is a model that is pre-trained on unsupervised tasks using large amount of data. The pre-trained BERT model can be fine-tuned with just one additional output layer, without needing to retrain the entire model, for a wide range of tasks, such as question-answering and Natural Language Inference.

Architecture

BERT's model architecture consists of a multi-layer bidirectional Transformer encoder based on the original implementation of transformer [56]. The architecture is identical with the transformer model section 2.1.

BERT model was published in two different model sizes: $\text{BERT}_{\text{BASE}}$ model consists of 12 layers (i.e., transformer blocks), with the hidden size of 768 and 12 self-attention heads resulting in a total of 110M parameters and $\text{BERT}_{\text{LARGE}}$ model consists of 24 layers (i.e., transformer blocks) with the hidden size of 1024 and 16 self-attention heads resulting in a total of 340M parameters.

Training

The model was pre-trained on two unsupervised tasks namely Masked Language Modeling and Next Sentence Prediction which will be explained below. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

The process of the fine-tuning and the pre-training is shown in figure 2.3, where `[CLS]` is a special classification token that is the first token of every sentence, `[SEP]` is token for differentiation of the sentence pairs, input embeddings are denoted as E, the final hidden vector for special token `[CLS]` is denoted as C and the final hidden vector for the i^{th} input token is denoted as T_i. The final hidden vector refers to the output of the BERT model for a given input sequence. Specifically, BERT consists of a stack of transformer layers, where each layer takes in an input sequence and generates a new sequence of hidden states. The final hidden vector is the hidden state corresponding to the special `[CLS]` token that is added at the beginning of the input sequence.

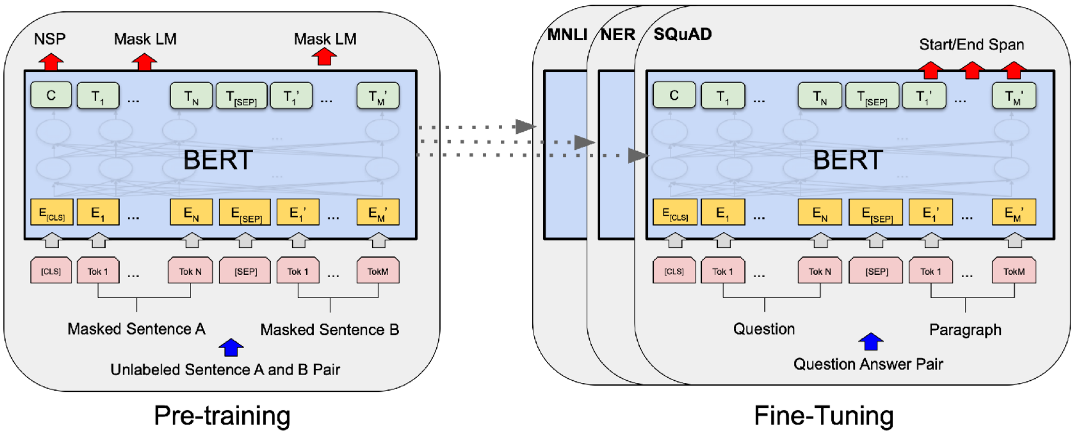

Figure 2.3: The question-answering example of overall pre-training and fine-tuning procedures for BERT [15]. For different tasks (NER [48], MNLI [63] and QA) the same initialization of the pre-trained model parameters is used.

Masked Language Modeling (MLM)

Masked Language Modeling (MLM) is a task in natural language processing that involves masking some words in a sentence and then predicting the masked words based on the context.

The BERT model was pre-trained by masking 15 % of the input tokens and then predicting these masked tokens. In the publication [15], they refer to this procedure as a “masked Language Modeling” (MLM), although it is often referred to as a Cloze task in the literature [52].

However, since the `[MASK]` token does not appear during fine-tuning, this leads to mismatch in pre-training and fine-tuning. To mitigate this, masked tokens are not always masked by the `[MASK]` token, but only in 80 % of the time. In the remaining 10 % of the time, the masked token is replaced by a random token, and in the last 10 % of the time the masked token is not changed.

Next Sentence Prediction (NSP)

For some important tasks like Question-Answering, it is not enough to train only on language modeling, but it also rely on understanding the relationship between two sentences, that is not directly captured by MLM. Therefore, the model is trained on pairs of sentences to predict whether the next sentence is following the first sentence in the original text or not. This is done by choosing sentence `A` and `B` in the way that in the 50 % of the time the sentence `B` is the actual next sentence that follows `A` and 50 % of the time it is a random sentence from the corpus. The sentences are separated with the special token `[CLS]` and the result of the next sentence prediction is then denoted as C shown in figure 2.3. However, this objective was shown to be redundant since RoBERTa paper [30].

2.3 Text-to-text transfer transformer

The Text-to-Text Transfer Transformer [42] (T5) is a language model that is based on the transformer architecture and is trained in a text-to-text framework, meaning that it is trained to generate text outputs from text inputs. T5 is trained on a diverse set of tasks, including machine translation, summarization, question answering, and text classification, among others. This makes it a highly versatile language model that can be fine-tuned for a wide range of downstream tasks with relatively little additional training data.

Architecture

The T5 model is based on the Transformer encoder-decoder architecture [56] as it has been found to perform well in both generative and classification tasks. The encoder and decoder are each designed to be similar in size, specifically, both consist of 12 blocks, each block comprising self-attention, optional encoder-decoder attention, and a feed-forward network [7]. The "key" and "value" matrices of all attention mechanisms have an inner dimensionality of $d_{kv} = 64$ and all attention mechanisms have 12 heads. The feed-forward networks in each block consist of a dense layer with an output dimensionality of $d_{\text{ff}} = 3072$ followed by a ReLU [1] nonlinearity and another dense layer. All other sub-layers and embeddings have a dimensionality of $d_{\text{model}} = 768$. The model contains about 220 million parameters and uses a dropout probability of 0.1 for regularization. A dropout probability of 0.1 is applied everywhere dropout is used in the model.

Dropout probability is a regularization technique used to prevent overfitting in neural networks. During training, a certain proportion of randomly selected neurons in a layer are dropped out, meaning their outputs are set to zero.

Training data

To create a dataset, Common Crawl was used as a source of text. Due to the scraped text data a significant portion of it is not natural language. It consists of gibberish, duplicate text, menus, error messages, and other non-useful content. To clean up Common Crawl's web extracted text, several heuristics were employed. These include retaining lines that end in a terminal punctuation mark, discarding pages with fewer than five sentences, removing any page containing a word on the "List of Dirty, Naughty, Obscene or Otherwise Bad Words"[1], removing any line with the word Javascript, among others. The dataset was then posprocessed with use of `langdetect`[2] to filtered out any pages that were not classified as English with a probability of at least 0.99.

To assemble the base data set, the web extracted text from April 2019 was downloaded, and the aforementioned filtering was applied. This produced a collection of text known as the "Colossal Clean Crawled Corpus" (C4).

[1] https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words

[2] https://pypi.org/project/langdetect/

Pre-training tasks

An objective that does not require labels but teaches the model generalizable knowledge is necessary for leveraging unlabeled data to pre-train the model. Inspired by BERT's "Masked Language Modeling" objective (described in section 2.2) and the "word dropout" regularization technique [9], an objective is designed that randomly samples and then drops out 15 % of tokens in the input sequence. All consecutive spans of dropped-out tokens are replaced by a single sentinel token. A token ID that is unique to the sequence is assigned to each sentinel token. The sentinel IDs are special tokens which are added to the vocabulary and do not correspond to any wordpiece. The target then corresponds to all of the dropped-out spans of tokens, delimited by the same sentinel tokens used in the input sequence plus a final sentinel token to mark the end of the target sequence.

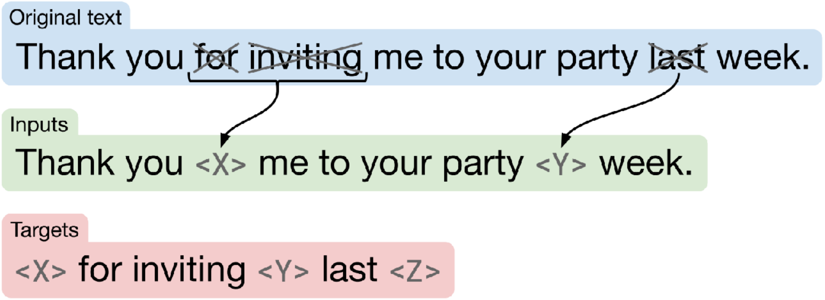
Original text
Thank you ~~for inviting~~ me to your party ~~last~~ week.
Inputs
Thank you <X> me to your party <Y> week.
Targets
<X> for inviting <Y> last <Z>

Figure 2.4: Schematic of the objective for pre-training.

Downstream tasks

Downstream tasks were chosen to measure general language learning abilities. Therefore diverse set of benchmarks were used including machine translation, question answering, abstractive summarization, and text classification. Specifically the performance of the model was measured on the GLUE [61] and SuperGLUE [60] text classification, CNN/Daily Mail abstractive summarization [21], SQuAD question answering [43] and WMT English to German, French and Romanian translation [8].

Input and output format

A „text-to-text“ format is used to cast all the tasks considered for training a single model in order to provide a consistent training objective for both pre-training and fine-tuning. In this format, the model is given some text as context or conditioning and is then required to produce some output text. The maximum likelihood objective is used to train the model, using "teacher forcing" [64] regardless of the task. To indicate which task the model should perform, a task-specific prefix is added to the original input sequence before it is fed to the model. The example of the input sequence is shown in section 5.2.

Training

All the tasks were formulated as text-to-text tasks. This allowed always to the training use standard maximum likelihood and a cross-entropy loss. For optimization the AdaFactor was used. At test time the greedy decoding was used (i.e. choosing the highest-probability logit at every timestep).

The model was pre-trained on the C4 dataset for 524,288 steps. The maximum sequence length was set to 512 and the batch size was set to 128 sequences. The „inverse square root“ was used as the learning rate schedule: $1/\sqrt{\max(n, k)}$, where n is the current training iteration and k is the number of warm-up steps. The warm-up steps were set to 10^4. This set a constant learning rate of 0.01 for the first 10^4 steps and then exponentially decreased the learning rate until pre-training was over.

The model was then fine-tuned on all the downstream tasks for 262,144 steps. This value was chosen as a trade-off between the high-resource tasks, which benefit from additional fine-tuning, and low-resource task, which overfit quickly. The maximum sequence length and batch size were same as in pre-training. The learning rate was set to a constant value of 0.001. Every 5000 steps the checkpoint was saved and the model reported results to the model checkpoint corresponding to the highest validation preformance.

2.4 Multilingual T5

The mT5 model [65] is multilingual variant of the T5 model. The model was created base on the T5's recipe as closely as possible. Thus the architecture and the training procedure is almost identical as for T5 model. The main differences are in the dataset that was used for pre-trainig and some minor changes in the pre-training procedure.

Architecture

The „T5.1.1“ recipe[3] was used as the basis for mT5's model architecture and training procedure, which closely follows that of T5. The improvements in T5.1.1, such as the use of GeGLU nonlinearities [49], scaling both d_{model} and d_{ff} (instead of just d_{ff} in the larger models), and pre-training on unlabeled data only with no dropout, were also incorporated into mT5. The base variant of the mT5 model was used in this work that has 580M parameters. The increase in parameter counts compared to the corresponding T5 model variants comes from the larger vocabulary used in mT5. Further details on T5 can be found in section 2.3.

Training data

The training data (multilingual C4) used for pre-training the mT5 model follows the same methodology as for C4 dataset that was used for pre-training T5 model. However the C4 dataset was constructed to handle only English language. Thus from C4 any pages that were not classified as English with a probability of at least 0.99 were filtered out by `langdetect`[4]. In contrast, for mC4 `cld3`[5] was used to identify over 100 languages.

The removal of lines that did not end in an English terminal punctuation mark was an important heuristic filtering step in C4. Instead of using English terminal punctuation marks, a "line length filter" is applied that requires pages to contain at least three lines of text with 200 or more characters. The creation of mC4 then followed C4's filtering by

[3] https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md#t511

[4] https://pypi.org/project/langdetect/

[5] https://github.com/google/cld3

deduplicating lines across documents and removing pages containing bad words as the same in the C4 dataset. Finally, the primary language of each page is detected using `cld3` and those with a confidence below 70 % are removed. After applying these filters, the remaining pages are grouped by language. All languages with 10,000 or more pages are included in the corpus.

Training

In pre-training mT5 model, a major factor was how to sample data from each language. The choice ultimately became a zero-sum game: If low-resource languages were sampled too often, the model may overfit, and if high-resource languages were not trained on enough, the model may underfit. Therefore, the approach used in [12][2] was followed, where lower-resource languages are boosted by sampling examples according to the probability $p(L) \propto |L|^{\alpha}$, where $p(L)$ is the probability of sampling text from a given language during pre-training and $|L|$ is the number of examples in the language. The hyperparameter α was set to value 0.3 and was used to control how much the probability of training on low-resource languages was boosted.

mT5 was only pre-trained on mC4 excluding any supervised training. mT5 model variants are pre-trained for 1 million steps on batches of 1024 length and input sequences with length of 1024. The same "inverse square root" learning rate schedule was used as in the pre-training of T5 model. Dropout was not applied during pre-training, as done in the "T5.1.1" recipe. The same self-supervised objective as T5 was used, with 15 % of tokens masked 2.3.

2.5 TF-IDF

TF-IDF stands for *term frequency-inverse document frequency* and is a measure used in the fields of information retrieval (IR) and machine learning that can quantify the importance of terms (words, phrases, lemmas, etc.) in a document amongst a collection of documents (also known as a corpus).

TF-IDF can be broked down into two parts, the term frequency (TF) and the inverse document frequency (IDF).

Term frequency

Term frequency (TF) is a simple technique used in natural language processing (NLP) to represent the importance of each term in a document. The idea of TF is to count the number of occurrences of a term in a document and use this number as a measure of the importance of the term in the document. Term frequency, TF(t,d), is the relative frequency of term t within document d,

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{2.3}$$

where $f_{t,d}$ is the count of the term occurences in the document.

Inverse document frequency

The IDF of a term is a measure of how rare that term is across the corpus. The idea behind IDF is that terms that appear frequently in a single document, but rarely in the rest of the corpus, are more important for understanding the content of that document. On the other hand, terms that appear frequently in many documents are less important, since they do not provide as much information about any individual document. The basic equation for the IDF can be written as follows,

$$\mathrm{IDF}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}, \tag{2.4}$$

where t denotes the term, D denotes the corpus, and N denotes the number of documents that are in the corpus.

Term frequency–inverse document frequency

Once we have computed the term frequency (TF) and inverse document frequency (IDF) for term t in a document, we can use them to calculate the TF-IDF weight for that term in that document. The TF-IDF weight is calculated by multiplying the TF value of the term in the document by the IDF value of the term across the corpus,

$$\text{TF-IDF}(t, d, D) = \mathrm{TF}(t, d) \cdot \mathrm{IDF}(t, D), \tag{2.5}$$

Classifier

Once the TF-IDF scores are computed for each term in a document, they can be used to train a classifier. In general, the classifier is an algorithm that takes input data and predicts the label that the data belongs to. It predicts the labels by learning a function that separates data into different labels. The function is often defined in terms of a set of parameters that are learned from training data. Once the classifier is trained, it can be used to make predictions on new, unseen data.

One common approach is to use a linear classifier, such as logistic regression, which learns the parameters on a set of weights for each term in the TF-IDF vector. The weights are then used to compute a score for each label, and the label with the highest score is then predicted.

2.6 Helsinki OPUS-MT

Translation is one of the important parts of creating a multilingual dataset in this work. Since professional translators are hard to find and their work is time and money consuming, the best alternative is to choose a machine translation model that performs well to do the work for them. Thus, the OPUS-MT [54] translator was chosen, which is based on machine translation using several different models. The starting point is OPUS [55], a growing collection of public parallel datasets that is the primary fuel for open data-driven machine translation.

> It serves aligned bitexts for a large number of languages and language pairs, providing publicly available data sets for machine translation from various domains and sources. Currently, the released data sets cover over 600 languages and additional regional language variants that are compiled into sentence- aligned bitexts for more than 40,000 language pairs. In total there are ca. 20 billion sentences and sentence fragments that correspond to 290 billion tokens in the entire collection. The released data sets amount to about 12 TB of compressed files. [55]

Public data with good language coverage is crucial for good machine translation. OPUS-MT builds on that collection and provides public translation solutions.

OPUS-MT provides the public with state-of-the-art translation solutions and is the main centre for pre-trained translation models. OPUS-MT is based on Marian [22], an efficient implementation of neural machine translation (NMT) in pure `C++` with minimal dependencies. Marian is a production-ready framework and includes optimized routines that enable a scalable approach to the development and exploitation of modern MT systems.

OPUS-MT models have been fully integrated into the transformers library by converting them to PyTorch. Models are available from the Huggingface model hub. The pre-trained models are based on state-of-the-art transformer-based neural machine translation. Models were pre-trained on freely available parallel corpora collected in the large OPUS repository. The architecture is based on a standard transformer setup with 6 self-attentive layers in both encoder and decoder networks with 8 attention heads in each layer.

Chapter 3

Datasets

Natural Language Processing (NLP) is a subfield of artificial intelligence that deals with the interaction between humans and computers using natural language. This subfield is very popular nowadays and is slowly becoming part of everyday life. One of the proofs is the published ChatGPT model [41], which is the most discussed topic nowadays.

However, any model is based on adequate training and and the quality of the data used for training. Statistical models used in natural language processing are getting bigger and bigger, with even billions of trainable parameters. Therefore, there is a great emphasis on creating new datasets. Large models require a large collection of examples in order to learn their parameters to approximate probability distributions over the data.

NLP datasets are used to train models that can then be used for various tasks such as text classification, entity recognition, machine translation, etc. Fact checking is also one of the vast areas. Although disinformation crosses country and language boundaries, most of the work focuses on statements and assertions in English. The actual development of automated fact checking in other languages is much more challenging. There are far fewer fact-checkers in languages other than English, and thus a multilingual dataset itself will be small and less effective in developing a fact-checking system.

The lack of this annotated data leads to the training of multilingual models that can replace the work of fact-checkers to some extent. The existence of a subfield of question answering in NLP that is more widespread was the main motivation for this work.

Datasets are usually divided into sets: `training`, `development` and `testing`. This split into three sets allows us to compare multiple different models against each other. Each of the given set is used for something different.
Training set — used to estimate the model's parameters.
Development set — used to evaluate different checkpoints of the proposed models during training and to select the best performing checkpoint.
Test set — is used to validate the results performed on the development set. These should be questions that do not overlap in any way with any other set.

3.1 X-Fact

X-Fact [19] is a publicly available multilingual dataset for factual verification of naturally existing real world claims. The dataset contains short statements in 25 languages and and is labeled for veracity by expert fact-checkers.

The dataset contains 31,189 short assertions in 25 different languages from 11 different language families.

X-FACT is compiled from several fact-checking sources. All sources were taken from the International Fact-Checking Network [36] (IFCN) list of nonpartisan fact-checkers and the Duke Reporter's Lab. This data was then divided into 7 labels, namely `True`, `Mostly-True`, `Partly-True`, `Mostly-False`, `False`, `Unverifiable` and `Other`. An example from the dataset is shown in table 3.1.

Claim	Muslimische Gebete sind Pflichtprogramm an katholischer Schule. Muslim prayers are compulsory in Catholic schools.
Label	Mostly-False (Grösstenteils Falsch)
Claimant	Freie Welt
Language	German
Source	`de.correctiv.org`
Claim Date	March 16, 2018
Review Date	March 23, 2018

Table 3.1: Example from X-FACT [19].

Since this work focuses on creating a dataset, the dataset was taken as one of the samples that exist today. The X-Fact dataset was mostly created by real fact-checkers and thus it is a very natural dataset, but it is also time and resource consuming. One of the negative is that the dataset is the merging of several different rating scales from multiple languages, which can lead to the point that the annotations can be non-agreeing. The major shortcoming however is that in the dataset there are no explicit annotation of documents relevant to claims. The ability to provide grounding for predicted veracity verdict is often more important than the verdict itself [28].

This shortcoming was considered to be the main problem in training a fact-checking model, and therefore it was the main motivation of this work. This work is based on the judgment that providing evidence for a fact is important because it allows the fact-checking model to make informed judgments about the truthfulness of a claim. Without evidence, a fact-checking model would have no basis on which to evaluate the veracity of a claim. In other words, the model would not be able to distinguish between true and false claims.

3.2 FaVIQ

This work was greatly inspired by the collection method introduced in FaVIQ Dataset [38]. The authors built a large-scale fact-checking dataset consisting of 188,000 assertions. Some of them were derived from an existing corpus of ambiguous information-seeking questions. The ambiguity of the questions allows to automatically construct true and false statements. Therefore, it allows to reflect more questions with a different ways of interpretation, leading to different answers. The use of ambiguity is shown in figure 3.1.

Despite considerable interest in developing general-purpose fact-checking models, it is difficult to build a large-scale dataset for fact-checking with real-world assertions. Existing claims are either authored by people in the crowd, introducing subtle subjective biases that are difficult to control. To mitigate the subjective biases the claims are manually verified by professional fact-checkers, making them expensive and limited in scale. This led to the idea of converting existing question answering datasets into fact-checking dataset.

The dataset was created from two QA datasets namely Natural Questions (NQ) [24] and AmbigQA [32]. NQ is a large dataset consisting of English information retrieval queries obtained from Google search engine. AmbigQA provides disambiguated question-answer pairs for NQ dataset, highlighting the ambiguity inherent in information-seeking questions.

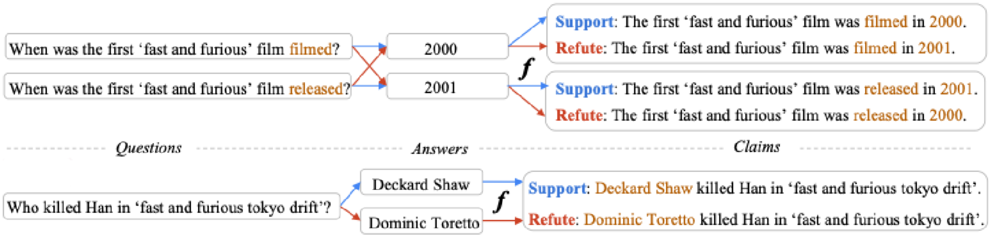
When was the first 'fast and furious' film filmed?
When was the first 'fast and furious' film released?
2000
2001
f
Support: The first 'fast and furious' film was filmed in 2000.
Refute: The first 'fast and furious' film was filmed in 2001.
Support: The first 'fast and furious' film was released in 2001.
Refute: The first 'fast and furious' film was released in 2000.
Questions
Answers
Claims
Who killed Han in 'fast and furious tokyo drift'?
Deckard Shaw
Dominic Toretto
f
Support: Deckard Shaw killed Han in 'fast and furious tokyo drift'.
Refute: Dominic Toretto killed Han in 'fast and furious tokyo drift'.

Figure 3.1: An example of a `refute` and `support` claim on FAVIQ, constructed using ambiguity in the information-seeking question and with the use of the reference answer (Deckard Shaw) and the incorrect prediction from DPR. ***f*** is a T5 model that transforms question-answer pairs to claims. [38].

FaVIQ is composed from two sets. The first set, denoted as the `A set`, consists of claims that were created with ambiguous questions and their disambiguation. The second set is labeled as the `R set` and consists of claims that were created with reference answer for the `support` claim and the incorrect prediction from the Dense Passage Retrieval (DPR) [23] model for `refute` claim. The claims are then generated by T5 model. The statistics of the `R set` and `A set` are shown in table 3.2.

		Total	Support	Refute
Train	A	17,008	8,504	8,504
	R	140,977	70,131	70,846
Dev	A	4,260	2,130	2,130
	R	15,566	7,739	7,827
Test	A	4,688	2,344	2,344
	R	5,877	2,922	2,955

Table 3.2: FaVIQ dataset statistics [38]. *A* includes claims derived from ambiguous questions, while *R* includes claims from regular question-answer pairs.

The question-answer pairs were converted into claims using a trained neural model that maps (`question`, `answer`) pairs to claims. A claim is marked as `support` if the pair from which the claim is generated contains the correct answer to the questions, otherwise the claim is marked as `refuted`. The model was firstly `pre-trained` on very similar dataset with thousands of examples and then `fine-tuned` using a dataset of 250 valid and invalid question-answer pairs that were manually converted. The T5-3B model [42] was then trained using 150 statements for training and 100 statements for validation.

In this work, the same methodology is followed in converting QA datasets to Fact-Checking, however, disambiguated questions are not used. This work is based on the system called CORA [5]. The system returns top-K documents for the question and then generate the answer based on these documents. The data from [38] was used for training the T5 and mT5 model to generate claims.

3.3 TyDi QA

The typologically diverse question answering dataset (TyDi QA) [11] is a dataset that consists of 11 typologically diverse languages with 204 thousands question-answer pairs. Of these, 167 thousands are one-way annotated, to be used for training, and 37 thousands are 3-way annotated, comprising the dev and test sets shown in table 3.3.

Language	**Train**	**Dev**
	(1-way)	(3-way)
English	9,211	1031
Arabic	23,092	1380
Bengali	10,768	328
Finnish	15,285	2082
Indonesian	14,952	1805
Japanese	16,288	1709
Kiswahili	17,613	2288
Korean	10,981	1698
Russian	12,803	1625
Telugu	24,558	2479
Thai	11,365	2245
Total	**166,916**	**18,670**

Table 3.3: Number of samples from each language represented in the dataset [11].

Due to its typological diversity, the QA abilities of the model are tested in many distinctive cultural settings. To provide a realistic information-seeking task and avoid priming effects, the questions were written by people who wanted to know the answer but did not yet know the answer. This was achieved by presenting the person with a short prompts consisting of the first 100 characters of Wikipedia articles. After reading it, the person needed to ask a question to which they wanted to know the answer, but this information was not represented in the short prompts. Annotators were asked to provide a question about anything interesting that came to mind, no matter how unrelated to the topic. This allows annotators even more freedom to ask about topics that really interest them, including topics not covered in the Wikipedia articles shown in table 3.4. The data are collected directly in each language without the use of translation.

Article:	Apple is a fruit...
Question:	What disease did Steve Jobs die of?

Table 3.4: Example showing the part of an Wikipedia prompts and a possible question based on the Wikipedia prompts [11].

The dataset contains 10 languages: *English*, *Arabic*, *Bengali*, *Finnish*, *Indonesian*, *Japanese*, *Kiswahili*, *Korean*, *Russian*, *Telugu*, *Thai.*

Each question was then matched with a Wikipedia article by searching for the question text in Google search, restricted to the Wikipedia domain for each language, and then the annotators selected the top-ranked result. The top-ranked result had to be selected based on whether the Wikipedia article contained the answer to the question or whether the question could not be answered using any article (or that no single passage is a satisfactory answer). If such a passage is found annotators were then asked to select the minimum response. Most often, the answer is a few words, but in some cases can span most of a sentence.

3.4 XOR-TyDi QA

The Cross-lingual Open Retrieval Question Answering (XOR-TyDi QA) dataset [4] is a collection of multilingual question-answer pairs that was created for development and evaluation of cross-lingual open-domain QA models. The dataset TyDi QA contains only answers of the same language. Unlike TidyQA, the dataset XOR-TyDi QA was built on questions from the pairs that are lacking same-language answers. Thus the dataset contain questions paired with target documents and answers in English only. These pairs are unanswerable in original dataset. XOR-TyDi QA includes 40k information-seeking questions from across 7 diverse non-English languages: *Arabic*, *Bengali*, *Finnish*, *Japanese*, *Korean*, *Russian* and *Telugu* shown in Table 3.5.

Language	Ar	Bn	Fi	Ja	Ko	Ru	Te
Size	20,379	5,704	12,110	9,564	5,847	11,218	8,196

Table 3.5: Dataset size of the XOR-TYDI QA corpus.

The question-answer pairs in the XOR-TyDi QA dataset were collected from Wikipedia articles as it was derived from TyDI QA dataset. The data was human curated and annotated to ensure a high level of quality and relevance. The professional translation service Gengo4 was used to translate all collected questions into English. As named entities were crucial for quality control, translators were instructed to translate them carefully by searching for common English translations from the English Wikipedia or other external sources. Manual quality assessments by native speakers were performed on 50 sample translations, and more than 95 % of the translations were found to be correct.

Language	Train	Dev	Test
Ar	18,402	708	1,269
Bn	5,010	427	267
Fi	9,768	615	1,727
Ja	7,815	433	1,316
Ko	4,325	371	1,151
Ru	9,290	568	1,360
Te	6,759	351	1,086

Table 3.6: Dataset size of the XOR-TYDI QA corpus splited into sets.

To evaluate the performance of models on the XOR-TyDi QA dataset, the data was divided into a training set, a validation set, and a test set that is shown in Table 3.6. Models were trained on the training set and their performance was evaluated on the validation and test sets.

3.5 Other datasets

In addition to X-Fact, there are many other datasets [53][59][33][51][62] that are designed to support training of modern models. However, most of the datasets, are created in English only and there is quite a small spectrum that deals with multilingual datasets.

One of the many datasets, for example, is the FEVER: Fact Extraction and VERification dataset [53]. It consists of 185,445 statements created by modifying sentences extracted from Wikipedia and then verified without knowledge of the sentence from which they were derived. Claims are classified by annotators as `Supported`, `Refuted` or `NotEnoughInfo`. For the first two classes (Supported and Refuted), the annotators also labeled the sentences constituting the necessary evidence for their judgment, which is one of the key shortcomings of the X-Fact dataset. However, the dataset is constructed in English only.

Another dataset is, for example, the Politifact dataset. This is a high-quality dataset that collects data from the PolitiFact website [59]. The dataset contains 21,152 passages that are checked by professional fact-checkers. The dataset is expanded every year with additional data. All passages are divided into 6 categories, namely: `true`, `mostly true`, `half true`, `mostly false`, `false`, and `pants on fire`. It also lists the sources where the statement appeared, which can be crucial for gaining different insights on fact-checking.

There are other datasets [33][51][62] that deal with fact-checking, but to the best of the author's knowledge no other multilingual datasets have been published. Nowadays, it is important to emphasize multilingual datasets as information/disinformation is disseminated beyond the boundaries of the language in which the disinformation originated[1].

[1] https://eufactcheck.eu/

Chapter 4

Information retrieval

Ranking retrieval is a process of ordering a set of documents in response to a user query in such a way that the most relevant documents appear at the top of the list. Formally, given a query q and a collection of documents $D = \{d_1, d_2, ..., d_n\}$, the ranking retrieval function $f(q, D)$ assigns a score to each document d_i in D, such that the documents are sorted in descending order of their scores, and the top-ranked documents are considered the most relevant to the query.

Information retrieval models are statistical models used to represent the process of retrieving relevant information from a large collection of documents or other data sources. The goal of information retrieval models is to assist users in finding the most relevant documents or information in response to a specific query.

There are various information retrieval models, but some of the most common include the Boolean model [44], the vector space model [47], the probabilistic model, the latent semantic indexing (LSI) model [26], and neural network models [16]. The Boolean model uses Boolean logic to match documents to queries. The vector space model represents documents and queries as vectors in a high-dimensional space, where each dimension corresponds to a term in the vocabulary. It computes the similarity between them using measures such as cosine similarity.

> Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis. [20]

The probabilistic model, on the other hand, uses probabilistic techniques to rank documents based on the likelihood that they are relevant to a query, taking into account factors such as term frequency and document length. The LSI model uses singular value decomposition to identify hidden relationships between terms in the corpus, while neural network models such as convolutional neural networks (CNN) [37] and transformers [56] show promising results when applied to information retrieval tasks. One of the transformer-based document retrieval models is Multilingual Dense Passage Retrieval (mDPR) [5]. It is a dense retrieval method that encodes the query and documents into dense representations using pre-trained multilingual language models such as mBERT [40], and then compares the query and documents in a vector space to retrieve the most relevant passages.

Information retrieval models can also be divided into sparse and dense models based on their approach to representing and matching textual data. Sparse information retrieval models, such as the classical Boolean model, represent textual data as binary vectors that indicate whether or not each term occurs in a document. In contrast, dense information retrieval models use continuous, dense representations of textual data that capture more subtle information about the relationships between terms, that is the key advantages.

Information retrieval models are essential in natural language processing and enable effective retrieval over large text corpus.

4.1 Dense passage retrieval

Dense Passage Retrieval (DPR) [23] is a method of dense retrieval model used for information retrieval tasks that involves retrieving relevant passages of text from a large corpus. DPR encodes both the query and documents into dense representations using pre-trained transformer models such as BERT [15], and then matches them in a vector space to retrieve the most relevant passages.

Dense passage retrieval uses a dense encoder $E_P(\cdot)$ which maps any input passage to a d-dimensional real-valued vectors and builds an index for all the M passages that will be used for retrieval:

$$E_P(\mathbf{x}, \mathbf{m}) : (\mathbb{R}^{n \times d}, \mathbb{R}^{n \times 1}) \rightarrow \mathbb{R}^d \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^{n \times d}$ is a matrix representing the input document tokens, $\mathbf{m} \in \mathbb{R}^{n \times 1}$ is a vector representing the attention mask for the input tokens, and $E_P(\mathbf{x}, \mathbf{m}) \in \mathbb{R}^d$ is the output vector representing the encoded document.

A different encoder $E_Q(\cdot)$ is applied at the run-time, which maps the input query to a d-dimensional vector and obtains the k passages that are closest to the query vector:

$$E_Q(\mathbf{x}, \mathbf{m}) : (\mathbb{R}^{n \times d}, \mathbb{R}^{n \times 1}) \rightarrow \mathbb{R}^d \tag{4.2}$$

where $\mathbf{x} \in \mathbb{R}^{n \times d}$ is a matrix representing the input query tokens, $\mathbf{m} \in \mathbb{R}^{n \times 1}$ is a vector representing the attention mask for the input tokens, and $E_Q(\mathbf{x}, \mathbf{m}) \in \mathbb{R}^d$ is the output vector representing the encoded query.

These encoders are two different BERT networks with output on the token [CLS], so $d = 768$. The input tokens are first tokenized and passed through the BERT model, which consists of multiple transformer layers. Each transformer layer applies self-attention and feed-forward neural networks to the input tokens to compute a contextualized representation of each token. The output of the final transformer layer is then used as the encoded representation of the input document or query.

In particular, the E_P takes as input a matrix $\mathbf{x}$ of document tokens and a vector $\mathbf{m}$ of attention masks, where each element of $\mathbf{m}$ is either 0 or 1 to indicate whether a corresponding token should be masked out or not. The E_P passes $\mathbf{x}$ and $\mathbf{m}$ through the BERT model to obtain a matrix of contextualized token representations. The attention mask is used to ensure that the masked tokens do not affect the output of the BERT model. The matrix of contextualized token representations is then aggregated into a single vector representation of the document using an attention mechanism.

Similarly, the E_Q takes as input a matrix $\mathbf{x}$ of query tokens and a vector $\mathbf{m}$ of attention masks. The E_Q passes $\mathbf{x}$ and $\mathbf{m}$ through the same BERT model to obtain a matrix of contextualized token representations. The E_Q also uses an attention mechanism to aggregate the matrix of contextualized token representations into a single vector representation of the query.

The similarity is defined as a dot product between the query vector and the document vector:

$$sim(q, p) = E_Q(q)^T E_P(p). \tag{4.3}$$

Training

Encoder training is done to maximize the cosine similarity between the query and relevant passages and minimize the cosine similarity between the query and irrelevant passages. The goal is to create a vector space that contains relevant pairs of question and passage vectors with smaller distance (i.e., higher similarity) than the irrelevant ones. Let $D = \{\langle q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^- \rangle\}_{i=1}^m$ be the training data that contains m instances. Each instance contains one question q_i and one positive passage p_i^+ along with n negative passages $p_{i,j}^+$ shown in table 4.1. The loss function is then optimized as the negative log-likelihood of the positive passage:

$$L(q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-) = -\log \frac{e^{sim(q_i, p_i^+)}}{e^{sim(q_i, p_i^+)} + \sum_{j=1}^n e^{sim(q_i, p_{i,j}^-)}}. \tag{4.4}$$

Question:	What is the capital of France?
Positive passage:	The capital of France is Paris, one of the most famous cities in the world, known for its art, fashion, and cuisine.
Negative passage 1:	France is a beautiful country located in Western Europe. It is home to a diverse range of landscapes, from the snowy peaks of the Alps to the sandy beaches of the Mediterranean coast.
Negative passage 2:	The Eiffel Tower is a famous landmark in France that attracts millions of visitors every year. It was built in the late 19th century as part of a World's Fair held in Paris.

Table 4.1: Example of a query, positive passages and negative passages.

For retrieval problems, it is often the case that positive examples are available explicitly, while negative examples must be selected from an extremely large set. Relevant passages to the question may be listed in the QA dataset or may be found by using the answer. All other passages could be considered irrelevant in this case. Therefore, negative passages were selected using three different procedures:

1. the first procedure, the so-called *random procedure*, was to select a random passage from the corpus,

2. the second procedure, the so-called *BM25 procedure*, was to select passages that were returned as best by BM25 model [44] and contain most of the question tokens but do not contain the answer,

3. the last procedure, the so-called *in-batch negatives*, consisted of selecting a positive passage and pairing it as negative with another question that appeared in the training set.

Using *in-batch negatives* has proven to be an effective approach because it's an easy and memory-efficient way to reuse negative examples already in a batch rather than creating new ones. The experiments in DPR paper [23] led to the conclusion that combining the second and third approaches is the best approach. That is, they took the negative passages from the *in-batch negatives* and added 7 negative passages from the *BM25 procedure*.

4.2 Multilingual dense passage retrieval

Multilingual dense passage retrieval [5] (mDPR) is a retrieval model used for information retrieval tasks. It is a variant of the dense passage retrieval [23] (DPR) model and is designed to be multilingual and cross-lingual, meaning that it can handle queries and documents in multiple languages. mDPR encodes both the query and the documents into a dense representation in the same way as DPR. However, since mDPR is designed as a multilingual model, it uses multilingual BERT (mBERT) [40].

mDPR is used to rank passages based on their relevance to a given query. It does so by representing each passage as a dense vector, using a combination of language models and transformer networks.

To rank passages for a given query, mDPR uses the following basic equation:

$$\text{Score}(\mathbf{P},\mathbf{Q}) = \text{cosine\_similarity}(\mathbf{P},\mathbf{Q}), \tag{4.5}$$

where P is the dense vector representation of the passage, Q is the dense vector representation of the query and `cosine_similarity` is a measure of the similarity between two vectors, calculated as the dot product of the vectors divided by the product of their magnitudes:

$$\text{cosine\_similarity}(\mathbf{P},\mathbf{Q}) = \frac{\mathbf{Q}\cdot\mathbf{P}}{\|\mathbf{Q}\|\|\mathbf{P}\|} = \frac{\sum_{i=1}^{n} Q_i P_i}{\sqrt{\sum_{i=1}^{n} Q_i^2}\sqrt{\sum_{i=1}^{n} P_i^2}}. \tag{4.6}$$

mDPR can handle queries by taking into account the full context of the passages it is ranking, in addition to evaluating their relevance to a given query. This makes it an attractive option for tasks such as question answering, where the meaning of the query may not be clear from the individual terms alone.

mDPR has been shown to be an effective method for ranking passages based on their relevance to a given query. It has been compared to other retrieval models, including Okapi BM25, and has demonstrated strong performance in a number of different benchmarks [44]. As such, it is often used as a baseline method for evaluating the performance of other retrieval models and has the potential to be a useful tool in a variety of information retrieval tasks.

Comparsion of mDPR with DPR

Let $D = \{\langle q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-\rangle\}_{i=1}^m$ be m training instances. Each instance consists of a question q_i, a passage that answers the question p_i^+ (a positive passage), and n passages that do not answer the question $p_{i,j}^{-}$ (negative passages). The mDPR model is then trained in the same way as the DPR model with equation 4.4, as described in the section on DPR.

DPR is originally initialized with the English BERT [15]. For multilingual search and to take advantage of cross-language transfer, the mDPR model was initialized with mBERT [40], while all other aspects of training remained the same [67].

Several works [3][31][45][58] have shown that monolingual BERT models can be more efficient than mBERT in various natural language processing tasks, but the gains are not consistent. In a publication [67], the results of the comparison of the monolingual model with the multilingual model were published. The results showed that it is possible that there are "better" monolingual BERT models for the target language than mBERT. However, the advantage of the mBERT model is that it can be easily pre-finetuned by using the MS MARCO dataset [34] and leveraging datasets from other languages. Obviously, for monolingual (non-English) BERT, it is no longer possible to pre-fine-tune the model on MS MARCO. A recommendation is included in the publication:

> Using a monolingual BERT backbone can yield a model that is more effective than using mBERT, but the monolingual model is not consistently better. Thus, it seems "safer" to just use mBERT as the backbone. [67]

A comparison of the results of all experiments can be found in the publication [67]. Part of the results table is presented in Table 4.2.

Language	Ar	En	Fi	Id	Ko
multilingual DPR	0.900	0.841	0.856	0.860	0.785
monolingual DPR	0.894	0.805	0.893	0.888	0.820

Table 4.2: Results from the publication [67] with a performance metric Recall@100.

To achive this results monolingual DPR was trained for all languages separately. The languages represented in the experiment were: *Arabic*, *English*, *Finnish*, *Indonesian* and *Korean*. Multilingual DPR was trained for all languages. The models were trained in 40 epochs with the corresponding training data (single or multiple languages) with 128 batch size without using transaltions. The monolingual DPR models were trained on data used from Mr. TYDI dataset [66]. The multilingual DPR was trained on MS MARCO dataset. The Recall@100 metric was then used for the evaluation of the models. Recall@100 is a metric used to evaluate the performance of information retrieval models. It measures the percentage of relevant documents that are retrieved among the top 100 results for a given query.

Chapter 5

Proposed system for dataset conversion

Following text is based on the findings presented in paper [38] about creation of a multilingual dataset for fact verification from an existing question-answering dataset. The main goal was to see if it is possible to automatically create a challenging dataset without help of human translators and annotators.

This work also goes trough this topic with additional comparison where two different approaches for dataset creation are attempted. The XOR-TyDi QA dataset was used for the conversion. The dataset consist of 7 languages: *Arabic*, *Bengali*, *Finnish*, *Japanese*, *Korean*, *Russian* and *Telugu*. This work assumes that the question-answer pairs are in the same language. It is proceed in the way when someone asks a question, he receives the answer in the same language in which the question was asked. However, this may not always be true for the evidence that contains the necessary information for the model to answer the question. The evidence may be in a different language than the language in which the question is represented. Even so, the model should be able to take the information from the other language and then return the answer in the desired language shown in Figure 5.1.

Question	What is one of the most popular musical instruments that is used in many genres of music?
Original Passage	В мире существует множество видов музыкальных инструментов, каждый из которых имеет свои особенности. Одним из наиболее популярных инструментов является гитара, которая используется во многих жанрах музыки.
Translated Passage (for readers)	There are many types of musical instruments in the world, each of which has its own characteristics. One of the most popular instruments is the guitar, which is used in many genres of music.
Answer	Guitar

Figure 5.1: Example of question in English an the information in the Russian language with the generated answer in English.

5.1 Answer generation

5.1.1 Data

The data which were used to return the documents are from the February 2019 Wikipedia dumps of 13 diverse languages contain all XOR-TyDi QA languages [5]. These 13 languages have a large number of Wikipedia articles and a variety of both Latin and non-Latin scripta continua. The data were extracted from Wikipedia using wikiextractor [6] and then each article was splitted into 100-token segments. In [5] they also filtered out the short articles with fewer than k (i.e., k = 20 in the paper) tokens resulting in 43.6M passages in total.

5.1.2 mDPR

One of the main part of the proposed system is article document retrieval. The implementation of the mDPR model taken from [5] covered all the steps described in this subsection. The mDPR produced dense embeddings of a question and all multilingual passages, thereby retrieving passages across all languages.

In order to answer a question, we need to find the record in which the answer to the question is located. This is the document retrieval task to returns a certain number of the most relevant documents in which the answer might be located. These documents are returned in different languages. The documents are then evaluated based on an `exact match score` metric (explained in Section 6.4) with the correct answer, since the answers are represented in the dataset from which the question is taken. Thus, the label "has_answer" was added to each sample to indicate whether the target answer is present in the document.

5.1.3 Generation of positive and negative question-answer-passage triplets

The generation model (mGEN) taken from [5] was trained to output an answer in the target language based on the retrieved multilingual passages. The resulting dataset must contain not only positive answers but also negative answers. They were created in the FaVIQ paper [38] using two approaches, particularly with the help of ambiguous questions and documents retrieved by document retrieval. In this work, only the approach with help of documents retrieved mDPR was used.

Let $Q^* = \{q_1, q_2, ..., q_n\}$ be all questions and $A^* = \{a_1^+, a_2^+, ..., a_n^+\}$ be all answers to the questions from the XOR-TyDi QA dataset [4].

Positive triplets

To generate a positive answer a^+, the questions q_i were submitted to mDPR, which retrieved the top 100 documents. Let P be all passages returned from mDPR for the question q_i. Since mGEN was trained with a set of 15 documents, these 100 documents denoted as P retrieved by mDPR were then post-processed in the way that only the top 15 documents that have the "has_answer" label set to *True* are taken into mGEN. If there were less than 15 documents with a label set to True, the sample was not considered in further triplet formation. Based on these documents mGEN then generates an answer a^+ for each question $q_i \in Q^*$ with the resulting scores shown in table 5.1.

The generated answer a^+ was then postprocessed with a goal to eliminate answer that was not present in any of the passages P retrieved by mDPR. Therefore, let $\mathrm{EM}(a^+, P_i)$ be the exact match score metric 6.4 that calculates the score with each passage $P_i \in P$ returned from mDPR. The EM was ten evaluated in the way $\mathrm{EM}(a^{+}, P_i) = 1$. If the result score was equal to 1 the answer a^{+} was presented in the passage P_i. The answer a^{+} is paired with the passage P_i and question q_i. This resulted into formation of the final positive triplet. Rest of the passages in P were not used in the final output. Samples that had no P_i passage that contained the answer a^{+} information were not converted to the final positive triplet. The results are shown in table 5.1.

Dataset	Postprocess	Number of Samples	F1 score	EM Score
Dev	Original	3473	46.08 %	34.35 %
	EM procedure	1145	61.09 %	49.78 %
Train	Original	61340	54.36 %	42.96 %
	EM procedure	29751	73.69 %	62.91 %

Table 5.1: Results using F1 and EM metrics (explained in Section 6.4) from the postprocess procedure for positive answers. The column named “Postprocess” indicates the specific procedure step. In the EM procedure the data was filtered in the way to eliminate answers that did not contain an passage with the evidence of the answer.

Negative triplets

To generate a negative answer a^{-}, the same procedure was followed as for a positive answer a^{+}, with the only difference that the top 15 passages P that have the “has_answer” label set to *False* are taken to generate the answer. The label set to false means they do not contain any answer match. Based on these documents mGEN generated an answer a^{-} for each question $q_i \in Q^{*}$ with the resulting scores shown in table 5.2.

The generated answer a^{-} was then postprocessed with a goal to eliminate answer in the way $\mathrm{F1}(a^{-}, A^{*}) = 1$. This score indicates that generated answer a^{-} was likely correct, as it was included in the original dataset answers A^{*} and since the purpose of this answer was to be incorrect, it was not considered in further triplet formation. Then for each answer a^{-} the EM was ten evaluated for each passage $P_i \in P$ in the way $\mathrm{EM}(a^{-}, P_i) = 1$. If the EM score was equal to 1 the answer a^{-} was presented in the passage P_i. The answer a^{-} was paired with the passage P_i and question q_i. This resulted into formation of the final negative triplet. Rest of the passages P were not used in the final output. Samples that had no P_i passage that contained the answer a^{-} information were not converted to the final negative triplet. The results are shown in table 5.2.

Dataset	Postprocess	Number of Samples	F1 score	EM Score
Dev	Original	3473	17.72 %	11.66 %
	F1 procedure	3068	6.86 %	0.00 %
	EM procedure	1080	9.01 %	0.00 %
Train	Original	61340	25.50 %	19.33 %
	F1 procedure	49452	7.59 %	0.00 %
	EM procedure	18037	9.13 %	0.00 %

Table 5.2: Results using F1 and EM metrics (explained in Section 6.4) from the postprocess procedure for negative answers. The column named "Postprocess" indicates the specific procedure step. In the F1 procedure the data was filtered in the way to eliminate answers that was likely correct as they were included in the original dataset. In the EM procedure the data was filtered in the way to eliminate answers that did not contain an passage with the evidence of the answer.

5.2 Conversion system

The mGEN model only generated a set of correct and incorrect answers to the question, but the final result to be achieved in this work is to create a multilingual fact-checking dataset. So the questions and answers need to be converted into the claims. Based on the question the mGEN generates the answer. As a result we have a question-answer pairs. From these pairs, the T5 model 2.3 was trained to generate a claim. The question contains information about what the answer specifically refers to and the answer itself is the information we want to transfer to the claim. From this point the execution flow was split into two approaches. The first approach was to train a multilingual T5 model [65] (mT5) with the use of the dataset that has the question-asnwer pairs in all considered languages. The second apporach is to train the monolingual T5 model [42] with dataset that contains only the data in English language.

Training

The training of these two models was done using data that were collected from the original paper FaVIQ [38]. The data consist of three datasets. The first dataset was used for pre-training and has 60k examples of the question-answer pairs and the target outputs that stand for the target claims. It includes also 10k examples for validation [38]. The dataset was collected from one of the authors of the Faviq paper [38]. The sample from the pre-training dataset is shown in table 5.3.

Dataset:	SQuAD
ID:	570bf0896b8089140040fada
Question:	Why did the committee debate adding a shift function?
Answer:	would allow more than 64 codes to be represented by a six-bit code
Claim:	Committees debated on adding a shift function because it would allow more than 64 codes to be represented by a six-bit code.

Table 5.3: Sample from dataset used for pre-training.

The other two datasets are manually annotated samples in English for the purpose of fine-tuning a monolingual T5 model. The datasets were collected from one of the authors of the Faviq paper [38]. They were created in the same way as `R Set` and `A Set` described in section 3.2. They consist of the same data structure that was needed for pre-training. Each dataset has the structure of one example as: *question*, *positive answer*, *negative answer*, *positive claim* and *negative claim*. Each dataset has ≈ 150 samples for training and ≈ 100 samples for validation. The sample from the fine-tuning dataset is shown in table 5.4. These datasets were used for fine-tuning of the models. There was no difference between positive and negative question-answer pairs that were used for fine-tuning, as mentioned by the author of these datasets. Thus the samples from both datasets were separated into two examples (positive and negative). Therefore, for fine-tuning task the dataset had 600 samples for training (each dataset consist of 150 samples of positive and negative example, i.e. $2^2 \cdot 150$) and 400 samples for validation (each dataset consist of 100 samples of positive and negative example, i.e. $2^2 \cdot 100$). The datasets were created in english language. Therefore, theses datasets need to be translated into all desired languages for training of a multilingual mT5 model.

Question:	Who was the comedian who said chase me?
Positive Answer:	duncan norvelle
Positive Claim:	duncan norvelle was the comedian who said chase me
Negative Answer:	chevy chase
Negative Claim:	chevy chase was the comedian who said chase me

Table 5.4: Sample from dataset used for fine-tuning.

English

The pre-trained model from the huggingface library, namely “t5-base”, was used for the monolingual T5 English model. The „t5-base“ model was pre-trained on the large C4 corpus [56]. The base version of T5 has 220M parameters that can be tuned.

The model was first pre-trained on the first dataset on 10 epochs. One epoch means that the model went through all the samples from the dataset exactly once and tried to predict the correct outputs. Each epoch was run with a batch size of 12, which means that 12 samples from the datasets were taken and computed at a time. The maximum sequence length was set to 256. A loss function was then calculated for each batch and gradients were calculated to update the model weights. Gradient accumulation was also used, meaning that the weights were only updated in some iteration of the training process. The gradient accumulation was set to 3 in this case. It means that every third iteration of training loop updated the model weights. At the end of each pre-training epoch, the checkpoint was evaluated based on the Rouge score, exact match score and F1 metrics 6.4. After 10 epochs, the performance of the model began to fluctuate. Thus, the model was not pre-trained for any further epochs and moved to the fine-tuning phase.

The model was fine-tuned using two other datasets on 900 epochs. The maximum sequence length and the batch size were the same as in pre-training. The gradient accumulation was also set to 3. During fine-tuning, a checkpoint was evaluated every 20 epochs. Fine-tuning was stopped in the 900 epoch, as it became clear that the model was already performing well in manual tests. Manual tests consisted of manually created questions and answers. Tests were used to check that the model generates correct claims.

After training the model, the resulting claims were generated using translation of the questions and answers into English. The model generated claims in English, which were then translated back into the desired language. Translations were made using the machine translation model described below. The model input structure was created by adding the prefixes "question:" before the target question and "answer:" before the target answer: "`question: Who was the comedian who said chase me? answer: duncan norvelle`". The recommended generation method[1] from the HuggingFace library was used. The max length of the sequence was set to 256. Generation was based on the beam search [18] which reduces the risk of missing hidden high probability word sequences by keeping the most likely "num_beams" of hypotheses at each time step and eventually choosing the hypothesis that has the overall highest probability. The "num_beam" parameter was set to 5. The parameter "early_stopping" was set to True with the "no_repeat_ngram_size" parameter set to 3. The "n-grams" penalty ensured that no "n-gram" (i.e. 3-gram in this case) appears twice by manually setting the probability of next words that could create an already seen "n-gram" to 0. The resulting scores for the T5 model are shown in table 5.5.

Multilingual

For the multilingual T5 model, which was trained for all languages, a pre-trained model "google/t5-base" from the huggingface library was used. The base version of mT5 has 580M parameters that can be tuned. To train the model, the datasets collected from one of the authors of FaVIQ paper were translated into all desired languages and then combined into target datasets. For training, the same approach as for the T5 model was used, except that only 3 epochs were performed for pre-training and then 900 epochs were used for fine-tuning. The batch size was the same as for T5 training, with the use of gradient accumulation. The difference between the number of pre-training epochs was due to the size of the training data. Since the original dataset was translated into all target languages, the size of the training dataset was almost 500,000 samples. The generation method with the same parameters was used as for monolingual T5 English model.

After the training process, the input was passed to the model to generate the claim. The structure of the input to the model was the same as shown in the T5 model except that a prefix for the target language was added: „`language: en question: Who was the comedian who said chase me? answer: duncan norvelle`". The resulting scores for the mT5 model are shown in table 5.5.

As can be seen the T5 model achieved better results than the mT5 model in automatic metrics. This was caused with the bad performance of the translation models as the metrics were calculated on the validation dataset that was used for fine-tuning the mT5 and T5 models. These datasets were firstly translated into all desired languages for the training of the mT5 model.

Therefore two errors are for the bad performance of the mT5 model. The error#1 is because of translations in some languages did not even reach 50 % success rate (described in Section 7.1), therefore the target claims were translated into claims that were not correct prediction in the target language for the model. The error#2 is that after the translation of the sample into the target language, the question and answer did not represent the same meaning as the translated claim. Thus the mT5 generated relevant claim for the

[1] https://huggingface.co/blog/how-to-generate

Model	Metric	Score
T5	Rouge-1	85 %
	Rouge-2	78 %
	Rouge-L	84 %
	EM	19 %
mT5	Rouge-1	66 %
	Rouge-2	51 %
	Rouge-L	61 %
	EM	6 %

Table 5.5: Evaluation of the trained models based on Rouge and Exact Match score metrics (explained in Section 6.4).

input question and answer but the target claim was different because of the translation. Unfortunately, these are still only hypotheses and experiments have not been performed to explain this problem.

5.3 Translation

The translation was performed using the Helsinki OPUS-MT [54] described in Section 2.1. For each language, a different model was used. Each model was trained for the target language. Since the Helsinki OPUS-MT lacked a translator model for Korean, a second translator model was used in this work, namely „facebook/m2m100_1.2B“. The model was trained for Many-to-Many multilingual translation and covers Korean as well.

All models were taken from the huggingface library, where the models are already pre-trained for the translation task. Translation is one of the main limitations of this work, as after manual analyses it was found that not all models translated the input to the target language correctly, especially for non-Latin languages. An example of a query translated from English to Telugu and then translated back to English is shown in table 5.6. On the other hand, the models work well in Latin languages, as can be seen in section 7.1.

Original query:	The chicken is afraid of the road.
Translated query:	The cold night's scared.

Table 5.6: Example of a translated query from English to Telugu and back to English.

However, the sentences in the original dataset are not that complex and each translator model performed well. The back-translation is evaluated in Section 7.1.

5.4 Final dataset

All the parts that are needed for the final pipeline are described and therefore the final pipeline structure can be interpreted. Both approaches are almost identical except the translation part. For the monolingual approach, the pipeline is shown in figure 5.2.

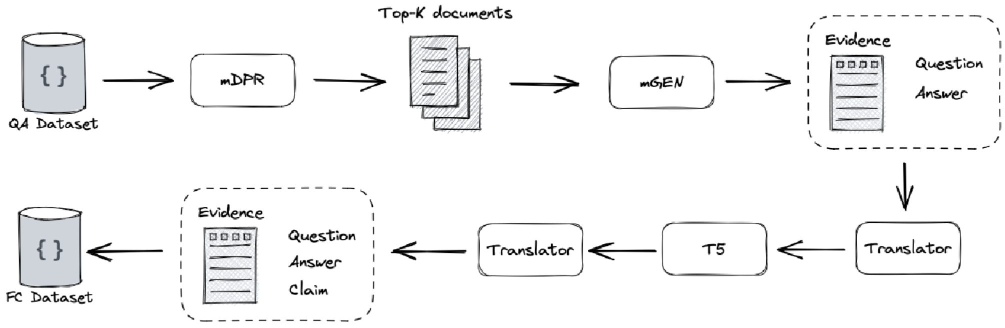

Figure 5.2: Pipeline for the approach with the English T5 model.

For the multilingual model the pipeline is shown in figure 5.3.

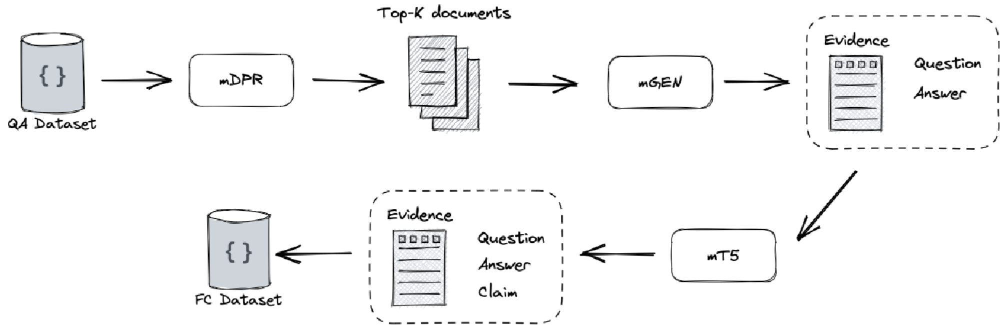

Figure 5.3: Pipeline for the approach with the multilingual T5 model.

As the final result both approaches convert dataset with the same samples and structure. The structure of one sample from the dataset is shown in table 5.4. The datasets consists of the validation dataset and the training dataset. The test dataset for final testing was not converted in this work, since in this work there were no evaluations of final fact-checking model that required the test dataset and therefore it was not needed for the experiments. Experiments were performed on the fact-checking model based on TF-IDF and it did not need a test data set to evaluate 7.4. However, to convert also the test dataset is straightforward since all the required parts for the creation were described.

ID:	324
Original ID:	-7474347795333626880
Language:	ru
Claim:	Эш Кетчум была главным героем в мультике „Покемо“
Label:	support
Evidence:	Эш Кетчум () или — главный герой аниме-сериала «Покемон». В японской версии сериала его зовут Сатоси (в честь создателя «Покемона» Сатоси Тадзири). Помимо аниме, Эш появляется в различного рода продукции, связанной с «Покемоном»: в манге, в полнометражных фильмах, в играх и прочих медиатоварах. Эш основан на персонаже по имени Рэд, главном герое первых игр серии „Pokemon Red“ и „Blue“ и манги Pokemon Adventures. Его заветная мечта — стать Мастером Покемонов, самым лучшим тренером покемонов всех времён. В 28 эпизоде девятнадцатого сезона (Покемон: XYZ), «Ответы в заснеженном лесу!» показана истинная история о том, почему Эш Кетчум так сильно любил покемонов: когда
Question:	Как звали главного героя в мультике „Покемон“?
Answer:	Эш Кетчум

ID:	324
Original ID:	-7474347795333626880
Language:	ru
Claim:	Ash Ketchum was the main character in the cartoon „Pokémon“
Label:	support
Evidence:	Ash Ketchum () or — the main character of the anime series «Pokémon». In the Japanese version of the series, he is called Satoshi (in honor of the creator of «Pokémon» Satoshi Tajiri). In addition to anime, Ash appears in a variety of productions related to «Pokémon»: in manga, in full-length films, in games and other media products. Ash is based on a character named Red, the main character of the first games of the series „Pokemon Red“ and „Blue“ and the manga Pokemon Adventures. His cherished dream is to become a Pokemon Master, the best Pokemon trainer of all time. In episode 28 of the nineteenth season (Pokémon: XYZ), «Ответы в зазеженном лесу!» The true story of why Ash Ketchum loved Pokemon so much is shown: when
Question:	What was the name of the main character in the cartoon „Pokémon“?
Answer:	Ash Ketchum

Figure 5.4: Example of one sample from the dataset. The second sample is the same, translated into English for the reader.

The final datasets consists of 3 different sample types. The first is positive claim, that is matched with the evidence that supports the claim. Second type of sample is a negative claim, that is matched with the evidence that likely refutes the negative claim but containing the positive answer for the claim. To ensure the quality of the datasets and that the task were not that simple for the fact-checking model, the third type of samples were created in such a way that negative claim that is matched with the evidence containing the negative information based on which the claim was generated. The total number of the samples and its split is shown in the table 5.7.

Dataset	Type	Size	Support	Refute	Refute$^+$	Refute$^-$
T5	train	47788	29751	18037	11784	6253
	dev	2225	1145	1080	497	583
mT5	train	47788	29751	18037	11784	6253
	dev	2225	1145	1080	497	583

Table 5.7: Overview of data file sizes. Refute$^+$ stands for negative claims matched with the evidence that contains correct answer. Refute$^-$ stands for negative claims matched with the evidence that supports incorrect answer.

Chapter 6

Experimental setup

In this work Python was selected as the main programming language since it is popular in the machine learning community and contains many machine learning libraries that are used in this work. The vast majority of the code is based on the transformer library with a large collection of pre-trained models running on Pytorch [39] backend.

6.1 Top-K accuracy

Top-k accuracy is a common evaluation metric used in retrieval tasks. It is used to measure the performance of a model in retrieving the most relevant documents or passages for a given query.

Top-K accuracy is calculated as the percentage of queries for which the model was able to retrieve at least one of the top k most relevant documents or passages. The value of k is typically set to a small number to ensure that the model is able to retrieve the most relevant documents or passages (i.e., k=100 in this work).

The basic equation for calculating top-k accuracy is as follows:

$$\text{Top-k Accuracy} = \frac{\substack{Number\ of\ queries\ with\ at\ least\ one \\ relevant\ document\ in\ top\text{-}K\ ranked\ documents}}{Total\ number\ of\ queries} \tag{6.1}$$

Top-k accuracy is a useful evaluation metric because it focuses on the ability of the model to retrieve the most relevant documents or passages for a given query. It is particularly useful for tasks where it is important to retrieve a small number of highly relevant documents or passages, such as in question answering task. Thus it is a valuable tool for measuring the performance of models in retrieving any relevant documents or passages for a given query.

6.2 F-score

F-score [14] describes model performance using a scale from zero to one. F-score itself is derived from two summary measures: precision and recall. Precision describes the proportion of entities which a model returns that are correct. Recall describes the proportion of all entities that potentially should be found, that a given model actually returns.

Precision and recall

Precision measures the proportion of true positive results among the total number of positive results predicted by a model shown in equation 6.2. In other words, it is the ratio of true positives to the sum of true positives and false positives. High precision means that a model is producing few false positives relative to the number of true positives, indicating a low rate of false positives. However, precision alone may not give a complete picture of a model's performance, and it is often used in conjunction with the recall to provide a more comprehensive evaluation.

$$P = \frac{|true\ positives|}{|true\ positives| + |false\ positives|} \tag{6.2}$$

Recall is a measure of a system's ability to identify all relevant instances of a target concept within a dataset. It is defined as the ratio of true positive results to the sum of true positive and false negative results shown in equation 6.3. In other words, recall measures the proportion of relevant items in the dataset that were correctly identified by the model. A high recall value indicates that the system is effective at finding all relevant instances of the target concept, while a low recall value indicates that many relevant instances were missed.

$$R = \frac{|true\ positives|}{|true\ positives| + |false\ negatives|} \tag{6.3}$$

F1 score

These metrics can be balanced out together. It is noted that these extreme situations, which were previously exploited, contrast with each other: when everything is returned, only a baseline precision is achieved, and returning just one thing typically results in a very low recall measure. Thus, it is a common practice to combine precision and recall with a weighted harmonic mean, known as an F-score:

$$F_\beta = (1+\beta^2)\frac{PR}{\beta^2 P + R} \tag{6.4}$$

The balance between precision and recall is determined by the coefficient β in this equation, with high values favoring recall. A harmonic weighted mean of precision and recall is obtained in this manner. Typically F-score is used with $\beta = 1$, c.f. its sometimes being called “F1 score”.

6.3 Rouge

The quality of a summary is determined automatically by comparing it to other (ideal) summaries created by humans, using measures included in ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [29]. The number of overlapping units such as n-gram, word sequences, and word pairs are counted between the computer-generated summary to be evaluated and the ideal summaries created by humans. In this work the measures `Rouge-N` and `Rouge-L` were used.

Rouge-N: N-gram co-occurrence statistics

ROUGE-N is an n-gram recall between a candidate summary and a set of reference summaries. Rouge-N is computed as follows:

$$ROUGE\text{-}N = \frac{\sum\limits_{S\in\{Summary\}} \sum\limits_{gram_n\in S} Count_{match}(gram_n)}{\sum\limits_{S\in\{Summary\}} \sum\limits_{gram_n\in S} Count(gram_n)} \tag{6.5}$$

where n stands for the length of the n-gram, $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries ($Summary$).

When multiple references are used, pairwise summary-level Rouge-N is computed between a candidate summary s and every reference, r_i, in the reference set. The maximum of pairwise summary-level ROUGE-N scores is then taken as the final multiple reference Rouge-N score. This can be written as follows:

$$ROUGE\text{-}N_{multi} = \text{argmax}_i ROUGE\text{-}N(r_i, s) \tag{6.6}$$

This procedure is also applied to computation of Rouge-L.

Rouge-L: longest common subsequence

A sequence $Z = [z_1, z_2, ..., z_n]$ is a subsequence of another sequence $X = [x_1, x_2, ..., x_m]$, if there exists a strict increasing sequence $[i_1, i_2, ..., i_k]$ of indices of X such that for all $j = 1, 2, ..., k$, we have $x_{ij} = z_j$ [13]. When considering two sequences X and Y, the common subsequence with maximum length is a longest common subsequence (LCS) that can be obtained.

To apply LCS in summarization evaulation, the summary sentence is viewed as a sequence of words. The intuition is that the longer the LCS of two summary sentences is, the more similar the two summaries are. In it was proposed to use LCS-based F-measure to estimate the similarity between two summaries X of length m and Y of length n. X stands for a reference summary sentence and Y stands for candidate summary sentence, as follows:

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{6.7}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{6.8}$$

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{6.9}$$

where $LCS(X,Y)$ is the length of a longest common subsequence of X and Y, and $\beta = P_{lcs}/R_{lcs}$ when $?F_{lcs}/?R_{lcs} =?F_{lcs}/?P_{lcs}$. β is set to a very big number. Therefore, only R_{lcs} is considered.

6.4 Exact match score

The exact match score is a metric used to evaluate the accuracy of natural language processing (NLP) models in predicting exact matches between two pieces of text.

In NLP tasks such as question-answering or text classification, the goal is often to predict a specific answer or label that exactly matches the true answer or label. The exact match score measures the percentage of predictions that are exactly correct, with no errors or variations from the true answer.

It is calculated as the number of exact matches between the predicted text and the true text, divided by the total number of examples:

$$\mathrm{EM} = \frac{\text{number of exact matches}}{\text{total number of examples}}. \tag{6.10}$$

The exact match score is a useful metric for evaluating the performance of NLP models in scenarios where precision is more important than recall. It is more strict evaluation metric than other commonly used metrics, such as the F1 score, which may give partial credit for answers that are close but not exact.

6.5 Implementation

The implementation is divided into four main parts: mDPR, mGEN, T5 and translator. Each of these parts can be independent and thus none of the parts is explicitly dependent on the other. One of the main drawbacks is that each part requires different versions of the libraries, especially the pytroch and transformer libraries. These versions are described in the documentation for the code, which can be found on the github repository[1]. However, if the different parts of the code are run in the same order as described in chapter XX, the results will be the same as in this work.

6.5.1 mDPR & mGEN

Since the implementations of multilingual dense passage retrieval and multilingual generator were already implemented in article [5], the code was taken from there. Some minor changes were applied in the code to eliminate some of the issues with the different versions of libraries. However, all the versions of libraries that were mention in the documentation of the code were used.

6.5.2 T5

Two implementations of the T5 model are presented in this work. Implementation for the T5 model was done with the own training loop and “t5-base” pre-trained model was used. However, there is nothing special about the implementation, as it was based on the HuggingFace[2] tutorial. For the mT5 model the 'google/mt5-base' model was used from HuggingFace library. This implementation is based on library simpletransformers[3], that consist of all the required functions for the training process.

For the validation of the models two metrics was used namely rouge score (specificaly rouge-1, rouge-2 and rouge-L) and exact match score. The calculation of the rouges scores

[1] https://github.com/xkamen21/designing-a-multilingual-fact-checking-dataset-from-existing-question-answering-data.git

[2] https://huggingface.co

[3] https://pypi.org/project/simpletransformers/0.26.0/

was done by the library evaluate library from HuggingFace. The implementation of the exact match score was taken from The Google AI Language Team[4].

[4] https://github.com/google-research/language/blob/6019bb3ab669fff3a0bc65feb438caa58c262233/language/orqa/utils/eval_utils.py#L97

Chapter 7

Experiments

The aim of this work was to compare two different approaches of creating fact-checking dataset from existing question-answering dataset. Several experiments were conducted to determine which of the approaches is better. As some stepts of dataset conversion are not easily evaluated automatically (such as translation evaluation and correctness of generated claims), this work resorted to Human Evaluation. To make the results as accurate as possible, most of the experiments consisted of human evaluation of the generated statements. The major part of the experiments consist of comparing the resulting datasets that were converted by the trained mT5 and T5 models. The Section 7.1 deals with the evaluation of translation into other languages, as it turned out during the work 7.1 that this part was the most critical for correct results.

For the claim validation 70 samples from each resulting dataset were randomly selected for experiments. These 70 samples were then given to 5 annotators who proceeded with the evaluation based on the attached instructions. The instructions can be seen in the Appendix A.

Human evaluation of the translations was performed only for the Russian language, as no sufficiently qualified persons were found for the other languages.

7.1 Translation

The translation of datasets using machine learning proved to be one of the main challenges of the entire work. The work deals with diverse languages from several different language families. The languages contain also *low-resource* languages (Bengali, Telugu). These language can have relatively small vocabulary, limited training data or they exhibit complex grammar, sentence structures, and rich morphology. This can lead to difficulties in training the translation model. Latin languages were translated well, but non-Latin languages were often translated into completely different statements, that are different from the original.

Human Evaluation

One of the experiments was to perform a human evaluation of a machine translation into the Russian language. Evaluation was performed on 100 samples from the original XOR-TyDi QA dataset [4]. One sample consisted of two parts. The first part contained Russian questions, which were then paired with English translations. The second part was the evaluation of the reverse translation from English to Russian. For this evaluation, English claim was taken and translated into Russian.

The translation was evaluated by an annotator who is a native speaker. The task of the annotator was to evaluate the translations into three different classes: `correct`, `incorrect` and `excellent`. The class `correct` stands for a translation that is correct with minor errors, the class `excellent` stands for a correct translation without any errors, and the class `incorrect` stands for a translation with important errors that affected the understanding of the sentence. The results can be seen in table 7.1.

Type	Samples	Good	Bad	Excellent
ru-en	100	29 %	26 %	45 %
en-ru	100	18 %	15 %	67 %

Table 7.1: Russian evaluation of translations.

As can be seen in the table, the results are very positive. The model can translate from Russian to English with more than 70 % success rate and from English to Russian with an success rate greater than 80 %.

All Languages

It is difficult to find annotators for each language that are native speakers. Therefore, the validation of the translation for other languages was done by translating an English sentence into the resulting language and then converting it back into English (back-translation). The task of this experiment was to check the translation of individual languages.

Data from the original FaVIQ dataset [38] were used for evaluation. Then 4 statements in the English language were taken from the dataset. These statements were then translated into all 7 languages and then translated back into the English language as you can see in Table 7.2.

Lang	Sequence
en	Additional information can be obtained.
ar	Additional information can be obtained.
bn	You can receive more information.
fi	Further information can be obtained.
ja	You can get additional information.
ko	You can get additional information.
ru	For further information, please click here.
te	More information is available.
en	A sniper bullet hit Matt on the right side of the neck.
ar	Sniper bullet hit Matt on the right side of the neck.
bn	A sniper went right on the right side of the car.
fi	The sniper's bullet hit Matt on the right side of his neck.
ja	The sniper's been shot on the right side of the mat.
ko	The sniper bullet hit the mat on the right side of his neck.
ru	A sniper bullet hit Matt on the right side of his neck.
te	A snooze butt on the right side of the heat.
en	He called his slave army the Black Guard.
ar	His slave army called his Black Guard.
bn	He told his slave soldier in the Black Guard.
fi	He called his slave army the Black Guard.
ja	He called the Slaves Blackwords.
ko	He called the slave army the Black Guard.
ru	He called his army slaves a black guard.
te	He was called the Black Guard of his servant's army.
en	Plants eventually formed chemical defenses against insects.
ar	Plants eventually formed chemical defences against insects.
bn	Finally, the rains were prevented by Comprick.
fi	The plants eventually formed chemical defenses against insects.
ja	Finally, plants established a chemical protection against insects.
ko	The plant eventually formed chemical defenses against insects.
ru	Plants eventually created chemical protection against insects.
te	In time, the fruitage produced by the family produced spiritual protection.

Table 7.2: Example of back translation.

The results have shown that the machine translation has a problem with languages like Bengali, Japanese or Telugu. Since the translation is part of the process of creating a dataset using the T5 model, the samples from these languages are of lower quality. It would be necessary to use another better model for translation. The other languages that are shown in table 7.2 were translated without major problems. It can be concluded that this methodology works well with Latine languages and could work with better models for other languages.

Additional manual analysis was performed on twenty samples based on back-translation. The results are shown in Table 7.3.

Language	ar	bn	fi	ja	ko	ru	te
Score	65 %	30 %	75 %	35 %	70 %	80 %	25 %

Table 7.3: Results from manual back-translation evaluation over 20 samples.

As can be seen, the results showed that languages like Bengali, Japanese or Telugu were mistranslated more than 65 % of the time. On the other hand, using "facebook/m2m100_1.2B" showed very good results in Korean. Thus, it can be considered that there are better models that could be used for machine translation in future works.

The evaluation of the next parts was based on usage of samples from the languages which translation was done correctly. This approach was selected because the goal of the work was not to evaluate the translation, but the resulting datasets.

7.2 Human evaluation of claims

One of the main parts of the experiments was the human evaluation of the generated claims. The evaluation was performed by five annotators. They were provided 3-way annotation for 70 identical samples from both datasets (one generated by mT5 and the other generated by the T5 model). The annotator's task involved assessing the accuracy of the individual claims in terms of their generation. The factual side of the statement was not taken into account, and thus the annotators should have been purely concentrated on the claim conversion from the question-answer pairs. A/B testing was applied to evaluate the performance and effectiveness of the multilingual T5 model and monolingual T5 English model. The primary objective of A/B testing was to measure the performance of the models. By comparing the performance of the two models, statistical analysis was conducted to determine if there was a significant difference in performance between the models. The exact instructions which the annotators followed can be seen in Appendix A.

The first part of the experiment was to assess whether the claim is generated correctly or incorrectly. In this evaluation, the annotators were not supposed to compare the claims with each other. They only evaluated the correctness of the generation. For a specific case where the statement was generated correctly but the information was not expressed explicitly, or the statement was interesting in some way to the annotator, he could mark the statement as interesting. This third class of evaluation is taken as correct but not entirely specific. The results of each annotator's evaluation are shown in Table 7.4.

Model	Annotator	Correct	Incorrect	Interesting
T5	1	50	13	7
	2	61	9	0
	3	64	4	2
	4	52	11	7
	5	56	2	12
mT5	1	30	34	6
	2	43	25	2
	3	55	7	8
	4	47	16	7
	5	47	10	13

Table 7.4: Results from annotators from 70 samples of the dataset.

It can be seen from the results in Table 7.4 that both models generated statements with very good accuracy. For a more reliable evaluation an objective score was calculated for each individual model. This score was derived from the number of correct results for each sample (the sum of interesting and correct statements), which was then divided by the number of annotators who participated in the evaluation. After all samples were calculated, the values were summed to obtain the final model score. For a perfect evaluation, the model had to score 70 points, which would mean that all annotators would rate all samples as correct or interesting.

The next objective for the annotators was to select claim they would prefer as more accurately generated. Both evaluation ways results are shown in Table 7.5.

Model	Examples	Total Score	Score [%]	Preference [%]
T5	70	62	89	56
mT5		51.4	73	27

Table 7.5: Results of custom metrics over results from annotators. The "Preferences" column contains only preferences from annotators for the model. The remaining 17 % were rated as cannot be judged.

As can be seen, the results show that the T5 model resulted in better score. For T5 the annotators determined 89 % of the generated statements to be correct, compared to the mT5 model, which generated a resulting accuracy of 73 %. Thus, the preference of the generated statements prevailed for the T5 model, when out of the total 70 samples. The annotators were in favor of the T5 model.

The above metrics considered each individual response from the annotators. For a different point of view, table 7.6 shows correctly identified statements using the methodology of `at least one agreement` and `unanimous agreement`.

Type	mT5		T5	
	Count	In precent	Count	In precent
at least one agreement	66	94 %	69	99 %
unanimous agreement	33	47 %	49	70 %

Table 7.6: The table shows the number of samples for which the annotators agreed on the correct one „`unanimous agreement`“ and the number of samples for which at least one annotator determined the sample to be correct „`at least one agreement`“.

Model preference was evaluated based on the majority of individual preferences. In the table 7.5 we can see that 39 samples were preferred by the annotators for the T5 model and only 19 samples were preferred for the mT5 model. The remaining 12 samples could not be judged as they had an inconclusive preference. For a more accurate evaluation, individual preferences were converted into numbers. For the T5 model, the preference was converted to value of -1, for mT5 the preference was converted to value of 1, and the samples that could not be assessed were set to 0. These values were then added up. The result is shown in Figure 7.1.

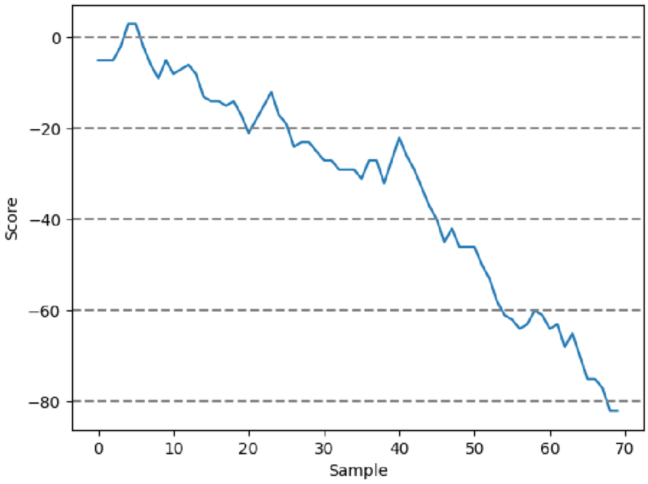

Figure 7.1: Result of custom metric for annotators preference.

As can be seen, most of the preferences were still leaning towards the T5 model, but the difference is no longer so significant. The maximum score that each model could get was 350 for mT5 and -350 for T5. The resulting score ended up at -82 in favor of the T5 model.

Unfortunately, a disclaimer must be given here, as the data that was passed to the annotators contained information about which claim was generated by which model and therefore the annotators could be biased by previous results. Thus, this section should be re-examined and done without identifying models in future works.

However, the results of both models are very positive. If we take into account that the use of the T5 model requires additional computing power for translation and the generation process itself is then that much longer, then the use of the multilingual T5 model is definitely an interesting option.

7.3 Discussion of the final dataset quality

The translation of the text was worse than expected. Therefore, an optimistic and pessimistic estimate of the size of the usable dataset was made. The estimate is based on three baseline values that were evaluated during the experiments. The first value is the number of generated claims, which is described in Section 5.1.3. The second value is the success rate of text translation in individual languages, which was obtained using back translation in Section 7.1. These values are shown in Table 7.7.

Language		Ar	Bn	Fi	Ja	Ko	Ru	Te
# Samples	Train	17900	4108	8655	1891	2224	7909	5101
	Dev	610	311	445	60	161	429	209
Translation		65 %	30 %	75 %	35 %	70 %	80 %	25 %

Table 7.7: Values for the estimation of the final datasets size.

The last value refers to the success rate of the claim generation itself using individual models. The claim generation value was used from two evaluation heuristics. Based on this estimate, an estimate of the size of the useful data in the dataset was then evaluated.

The first heuristic was calculated using an optimistic evaluation, where the success rate value of the claim generation was obtained by the `at least one agreement` method 7.2. In this case, the monolingual T5 model had a success rate of 99 % and the multilingual mT5 model had a success rate of 94 %. For each language, the number of successfully translated samples was calculated in the way that number of samples from each language were miltyplied with the success rate of the translation for the target language. For Arabic, the value of correctly translated samples was calculated: $0.65 * 17900 = 11635$ for the training dataset and $0.65 * 610 = 396$ for the validation dataset. The estimation is that 11635 samples in the training data set and 396 samples in the validation data set were correctly translated. This methodology was performed for all languages.

As a result of multiplying the correctly translated samples with the success of generating an individual model, the value of the number of usable samples in each data set was reached. Thus, for the values calculated for Arabic and the T5 model, the resulting number of usable samples would be calculated: $11635 * 0.99 = 11518$ for the training dataset. This was done for a single model in particular over all languages. The resulting count shows how many usable causes from each language a single data set contains.

The second heuristic was calculated using an pessimistic evaluation, where the success rate value of the claim generation was obtained by the `unanimous agreement` method 7.2. The same methodology was used as for the first estimation. The values of the success rate were 70 % for the T5 model and 47 % for the mT5 model. The final results of the datasets are shown in table 7.8.

Model	Variant	Dataset	Ar	Bn	Fi	Ja	Ko	Ru	Te	Total
T5	Opt	Train	11518	1220	6426	655	1541	6264	1262	**28887**
		Dev	393	92	330	21	112	340	52	**1340**
	Pess	Train	8144	863	4544	463	1090	4429	893	**20426**
		Dev	278	65	234	15	79	240	37	**948**
mT5	Opt	Train	11169	1183	6232	635	1495	6074	1224	**28013**
		Dev	381	90	320	20	108	329	50	**1298**
	Pess	Train	5468	579	3051	311	732	2974	599	**13714**
		Dev	186	44	157	10	53	161	25	**636**

Table 7.8: The estimation number of final samples in each data set. The “Variant” column represents optimistic and pessimistic heuristics.

It can be seen from the results that the converted datasets still have a large number of samples. However, some languages have a smaller number of samples, almost unusable. This is due to poor machine translation where a large number of samples were mistranslated. The heuristics used this fact into account. Heuristics themselves have their limitations. This is because random samples were not selected to evaluate the correct generation of claims. Instead, samples that made sense when translated were used. One sample means the translated question, answer, and claim itself from the target dataset. This may have filtered out samples from low-resource languages and thus cannot guarantee that the models generate claims with the same accuracy for these languages as well.

Nevertheless, using a better translation model for low-resource languages could make the evaluations more accurate. The results could then be more balanced for all languages.

Therefore, the translation turned out to be the main critical point of this work. The conversions of claims themselves turned out to be almost perfect. And so the work shows beneficial information that could be used to convert an existing question-answering dataset into a fact-checking dataset.

7.4 Fact-checking using TF-IDF

In this section, the machine evaluation of resulting dataset was performed with help of logistic regression classifier to determine if a claim is supported or refuted.

Logistic regression is a commonly used classification algorithm that is well suited for binary classification tasks. It works by estimating the probability that an input belongs to a particular class based on the values of its features. The model uses a logistic function to map the input features to the output probabilities.

In the evaluation phase, the logistic regression model was trained on the training data and then used to make predictions on the validation data. The confusion matrix 7.2 was used to evaluate the performance of the models in terms of true positives, true negatives, false positives, and false negatives. The result for the mT5 model is shown in table 7.9.

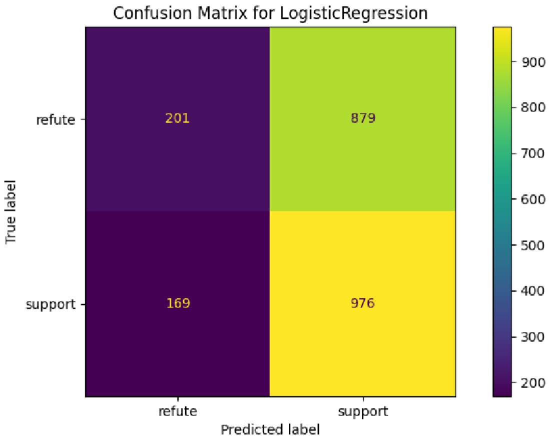

Figure 7.2: mt5 logistic regression

	Precision	Recall	F1-score	# Samples	Accuracy
refute	0.54	0.19	0.28	1080	0.53
support	0.53	0.85	0.65	1145	

Table 7.9: Classification report for dataset from mT5 model.

The classification report shows that the model is better at predicting the „support“ class compared to the „refute“ class. The model has a high precision for both classes, but the recall for „refute“ class is low. This means that the model is not able to correctly identify all „refute“ claims, and is more biased towards predicting „support“ claims.

After evaluating the T5 model on the same dataset, it was found that the confusion matrix 7.3 and classification report 7.10 were almost identical to the T5 model.

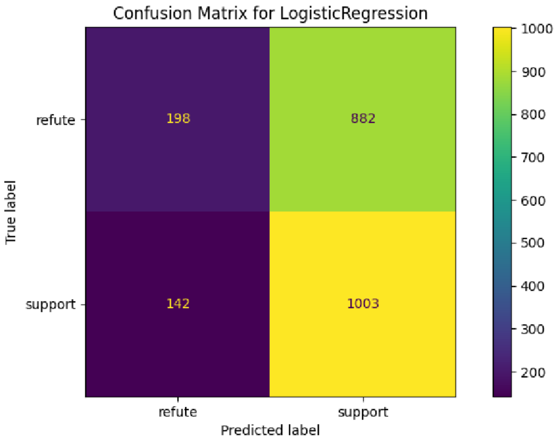

Figure 7.3: T5 logistic regression

	Precision	Recall	F1-score	# Samples	Accuracy
refute	0.58	0.18	0.28	1080	0.54
support	0.53	0.88	0.66	1145	

Table 7.10: Classification report for dataset from T5 model.

These results suggest that the performance of the T5 model is similar to the mT5 model in classifying claims as either supported or refuted. This finding showed that there is almost no difference for machine classification between the two datasets. The classification model was able to correctly predict supported claims on both datasets, but failed to correctly identify refuted claims.

The resulting classification report for all languages from each dataset can be seen in Table 7.11.

Dataset	Lang		Precision	Recall	F1-score	# Samples	Accuracy
T5	Ar	refute	0.55	0.13	0.21	291	0.53
		support	0.53	0.90	0.67	319	
	Bn	refute	0.59	0.46	0.52	170	0.53
		support	0.48	0.60	0.54	141	
	Fi	refute	0.56	0.11	0.18	199	0.56
		support	0.56	0.93	0.70	246	
	Ja	refute	0.50	0.04	0.07	25	0.58
		support	0.59	0.97	0.73	35	
	Ko	refute	0.49	0.26	0.34	68	0.57
		support	0.60	0.80	0.68	93	
	Ru	refute	0.61	0.16	0.25	226	0.50
		support	0.49	0.89	0.63	203	
	Te	refute	0.73	0.11	0.19	101	0.55
		support	0.54	0.96	0.69	108	
mT5	Ar	refute	0.52	0.16	0.24	291	0.53
		support	0.53	0.87	0.66	319	
	Bn	refute	0.58	0.45	0.50	170	0.52
		support	0.47	0.60	0.53	141	
	Fi	refute	0.48	0.08	0.14	199	0.55
		support	0.56	0.93	0.70	246	
	Ja	refute	0.00	0.00	0.00	25	0.55
		support	0.57	0.94	0.71	35	
	Ko	refute	0.45	0.29	0.36	68	0.55
		support	0.59	0.74	0.66	93	
	Ru	refute	0.59	0.19	0.29	226	0.50
		support	0.49	0.85	0.62	203	
	Te	refute	0.44	0.07	0.12	101	0.51
		support	0.51	0.92	0.66	108	

Table 7.11: Classification report for all languages from both datasets.

Chapter 8

Conclusion

The objective of this work was to propose, develop and compare different approaches to converting a multilingual fact-checking dataset from existing question-answering (QA) dataset. The dataset was converted from an existing XOR-TyDi QA dataset. From this dataset, question-answer-passage triplets were constructed in 7 diverse languages from which claims were generated. The proposed systems are based on generative transformer model T5. Two main approaches were tested. The first was to adapt the English monolingual model to a multilingual task using machine translation to translate both input and output. The second approach was to use a natively multilingual model that would take input in any language and generate statements directly in the target language.

In order to create a system for automatic data set conversion, it was necessary to become familiar with the current problems of multilingual fact-checking and the importance of using sources from different languages. Therefore, the work describes the relevant data sources that were needed to convert the datasets. The work also examines an already existing approach for converting datasets. The work is based on this approach.

The answers were generated using the adopted CORA system. An evidence (passage) was added, resulting in a question-answer-passage triplet. These triplets were translated into the English and passed to the T5 model. The model converted question-answer pairs presented in triplet into the resulting claims in English. Each claim was then translated back to the target language. The second approach was based on mT5 model that takes the triplets in the original languages. The mT5 converted the question-answer pair presented in the triplet into the claim. The converted claims were already in the target language without help of translation. Datasets were then created based on the generated claims.

To analyze possible biases label specific claim biases, the logistic-regression based TF-IDF classifier was trained. The classifier achieved accuracy close to 0.5 for both converted datasets. The qualitative value of the dataset was evaluated by several annotators with positive results compare to the FaVIQ where they achieved success rate of 95 %. The claims generated from multilingual model achieved a success rate of 73 % compared to monolingual model with a success rate of 88 %. However, compared to the FaVIQ, only the smaller model variants of the T5 models were used.

The result of this work is that the conversion of a multilingual dataset from an existing QA dataset is possible, but with the use of more complex models both for translation and for generating the resulting claims, it would be possible to achieve much better results. The *low-resource* languages were translated with the success rate around 35 %.

Bibliography

[1] Agarap, A. F. Deep Learning using Rectified Linear Units (ReLU). *CoRR*. 2018, abs/1803.08375. Available at: http://arxiv.org/abs/1803.08375.

[2] Arivazhagan, N., Bapna, A., Firat, O., Lepikhin, D., Johnson, M. et al. Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges. *CoRR*. 2019, abs/1907.05019. Available at: http://arxiv.org/abs/1907.05019.

[3] Armengol-Estapé, J., Carrino, C. P., Penagos, C. R., Bonet, O. D. G., Armentano-Oller, C. et al. Are Multilingual Models the Best Choice for Moderately Under-resourced Languages? A Comprehensive Assessment for Catalan. *CoRR*. 2021, abs/2107.07903. Available at: https://arxiv.org/abs/2107.07903.

[4] Asai, A., Kasai, J., Clark, J. H., Lee, K., Choi, E. et al. XOR QA: Cross-lingual Open-Retrieval Question Answering. In: *NAACL-HLT*. 2021.

[5] Asai, A., Yu, X., Kasai, J. and Hajishirzi, H. One Question Answering Model for Many Languages with Cross-lingual Dense Passage Retrieval. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. and Vaughan, J. W., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021, vol. 34, p. 7547–7560. Available at: https://proceedings.neurips.cc/paper/2021/file/3df07fdae1ab273a967aaa1d355b8bb6-Paper.pdf.

[6] Attardi, G. *WikiExtractor* [https://github.com/attardi/wikiextractor]. GitHub, 2015.

[7] Bebis, G. and Georgiopoulos, M. Feed-forward neural networks. *IEEE Potentials*. 1994, vol. 13, no. 4, p. 27–31. DOI: 10.1109/45.329294.

[8] Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P. et al. Findings of the 2014 workshop on statistical machine translation. In: *Proceedings of the ninth workshop on statistical machine translation*. 2014, p. 12–58.

[9] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R. et al. Generating Sentences from a Continuous Space. *CoRR*. 2015, abs/1511.06349. Available at: http://arxiv.org/abs/1511.06349.

[10] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv, 2014. DOI: 10.48550/ARXIV.1412.3555. Available at: https://arxiv.org/abs/1412.3555.

[11] Clark, J. H., Choi, E., Collins, M., Garrette, D., Kwiatkowski, T. et al. TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages. 2020.

[12] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G. et al. Unsupervised Cross-lingual Representation Learning at Scale. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, p. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. Available at: https://aclanthology.org/2020.acl-main.747.

[13] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. *Introduction to Algorithms, Third Edition*. 3rdth ed. The MIT Press, 2009. ISBN 0262033844.

[14] Derczynski, L. Complementarity, F-score, and NLP Evaluation. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, p. 261–266. Available at: https://aclanthology.org/L16-1040.

[15] Devlin, J., Chang, M., Lee, K. and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*. 2018, abs/1810.04805. Available at: http://arxiv.org/abs/1810.04805.

[16] Dimitrov, P. *Neural network models*. 2023.

[17] Elman, J. L. Finding structure in time. *Cognitive science*. Wiley Online Library. 1990, vol. 14, no. 2, p. 179–211.

[18] Freitag, M. and Al-Onaizan, Y. Beam Search Strategies for Neural Machine Translation. *CoRR*. 2017, abs/1702.01806. Available at: http://arxiv.org/abs/1702.01806.

[19] Gupta, A. and Srikumar, V. X-FACT: A New Benchmark Dataset for Multilingual Fact Checking. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2021.

[20] Han, J., Kamber, M. and Pei, J. 2 - Getting to Know Your Data. In: Han, J., Kamber, M. and Pei, J., ed. *Data Mining (Third Edition)*. Third Editionth ed. Boston: Morgan Kaufmann, 2012, p. 39–82. The Morgan Kaufmann Series in Data Management Systems. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00002-2. ISBN 978-0-12-381479-1. Available at: https://www.sciencedirect.com/science/article/pii/B9780123814791000022.

[21] Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W. et al. Teaching Machines to Read and Comprehend. *CoRR*. 2015, abs/1506.03340. Available at: http://arxiv.org/abs/1506.03340.

[22] Junczys Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K. et al. Marian: Fast Neural Machine Translation in C++. In: *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia: Association for Computational Linguistics, July 2018, p. 116–121. DOI: 10.18653/v1/P18-4020. Available at: https://aclanthology.org/P18-4020.

[23] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L. et al. Dense Passage Retrieval for Open-Domain Question Answering. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, November 2020, p. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550. Available at: https://www.aclweb.org/anthology/2020.emnlp-main.550.

[24] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A. et al. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*. MIT Press. 2019. Available at: https://aclanthology.org/Q19-1026.

[25] Lakew, S. M., Cettolo, M. and Federico, M. A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation. *CoRR*. 2018, abs/1806.06957. Available at: http://arxiv.org/abs/1806.06957.

[26] Landauer, T. K. and Dumais, S. T. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*. 1997, vol. 104, no. 2, p. 211–240. DOI: 10.1037/0033-295x.104.2.211.

[27] Lewandowsky, S., Ecker, U. K. H., Seifert, C. M., Schwarz, N. and Cook, J. Misinformation and Its Correction: Continued Influence and Successful Debiasing. *Psychological Science in the Public Interest*. 2012, vol. 13, no. 3, p. 106–131. DOI: 10.1177/1529100612451018. PMID: 26173286. Available at: https://doi.org/10.1177/1529100612451018.

[28] Lewandowsky, S., Ecker, U. K. H., Seifert, C. M., Schwarz, N. and Cook, J. Misinformation and Its Correction: Continued Influence and Successful Debiasing. *Psychological Science in the Public Interest*. Sage Publications, Inc. 2012, vol. 13, no. 3, p. 106–131. ISSN 15291006. Available at: http://www.jstor.org/stable/23484653.

[29] Lin, C.-Y. ROUGE: A Package for Automatic Evaluation of Summaries. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, p. 74–81. Available at: https://aclanthology.org/W04-1013.

[30] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M. et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*. 2019, abs/1907.11692. Available at: http://arxiv.org/abs/1907.11692.

[31] Martin, L., Müller, B., Suárez, P. J. O., Dupont, Y., Romary, L. et al. CamemBERT: a Tasty French Language Model. *CoRR*. 2019, abs/1911.03894. Available at: http://arxiv.org/abs/1911.03894.

[32] Min, S., Michael, J., Hajishirzi, H. and Zettlemoyer, L. AmbigQA: Answering Ambiguous Open-domain Questions. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020. Available at: https://aclanthology.org/2020.emnlp-main.466.

[33] Mohr, I., Wührl, A. and Klinger, R. Covert: A corpus of fact-checked biomedical covid-19 tweets. *ArXiv preprint arXiv:2204.12164*. 2022.

[34] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S. et al. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *CoRR*. 2016, abs/1611.09268. Available at: http://arxiv.org/abs/1611.09268.

[35] OpenAI. *GPT-4 Technical Report*. 2023.

[36] Orsek, B. *International Fact Checking Network*. Available at: https://ifcncodeofprinciples.poynter.org.

[37] O'Shea, K. and Nash, R. An Introduction to Convolutional Neural Networks. *CoRR*. 2015, abs/1511.08458. Available at: http://arxiv.org/abs/1511.08458.

[38] Park, J., Min, S., Kang, J., Zettlemoyer, L. and Hajishirzi, H. FaVIQ: FAct Verification from Information-seeking Questions. *CoRR*. 2021, abs/2107.02153. Available at: https://arxiv.org/abs/2107.02153.

[39] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR*. 2019, abs/1912.01703. Available at: http://arxiv.org/abs/1912.01703.

[40] Pires, T., Schlinger, E. and Garrette, D. How multilingual is Multilingual BERT? *CoRR*. 2019, abs/1906.01502. Available at: http://arxiv.org/abs/1906.01502.

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

[42] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S. et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. arXiv, 2019. DOI: 10.48550/ARXIV.1910.10683. Available at: https://arxiv.org/abs/1910.10683.

[43] Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. *CoRR*. 2016, abs/1606.05250. Available at: http://arxiv.org/abs/1606.05250.

[44] Robertson, S. and Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*. 2009, vol. 3, no. 4, p. 333–389. DOI: 10.1561/1500000019. ISSN 1554-0669. Available at: http://dx.doi.org/10.1561/1500000019.

[45] Rönnqvist, S., Kanerva, J., Salakoski, T. and Ginter, F. Is Multilingual BERT Fluent in Language Generation? *CoRR*. 2019, abs/1910.03806. Available at: http://arxiv.org/abs/1910.03806.

[46] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. Learning representations by back-propagating errors. *Nature*. Nature Publishing Group UK London. 1986, vol. 323, no. 6088, p. 533–536.

[47] Salton, G., Wong, A. and Yang, C. S. A Vector Space Model for Automatic Indexing. Association for Computing Machinery. 1975.

[48] Sang, E. F. T. K. and Meulder, F. D. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *CoRR*. 2003, cs.CL/0306050. Available at: http://arxiv.org/abs/cs/0306050.

[49] Shazeer, N. GLU Variants Improve Transformer. *CoRR*. 2020, abs/2002.05202. Available at: https://arxiv.org/abs/2002.05202.

[50] Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *CoRR*. 2018, abs/1808.03314. Available at: http://arxiv.org/abs/1808.03314.

[51] Shu, K., Mahudeswaran, D., Wang, S., Lee, D. and Liu, H. FakeNewsNet: A Data Repository with News Content, Social Context and Dynamic Information for Studying Fake News on Social Media. *ArXiv preprint arXiv:1809.01286*. 2018.

[52] Taylor, W. L. “Cloze Procedure”: A New Tool for Measuring Readability. *Journalism & Mass Communication Quarterly*. 1953, vol. 30, p. 415 – 433.

[53] Thorne, J., Vlachos, A., Christodoulopoulos, C. and Mittal, A. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In: *NAACL-HLT*. 2018.

[54] Tiedemann, J. and Thottingal, S. OPUS-MT — Building open translation services for the World. In: *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*. Lisbon, Portugal: [b.n.], 2020.

[55] Tiedemann, J., Aulamo, M., Bakshandaeva, D., Boggia, M., Grönroos, S.-A. et al. *Democratizing Machine Translation with OPUS-MT*. arXiv, 2022. DOI: 10.48550/ARXIV.2212.01936. Available at: https://arxiv.org/abs/2212.01936.

[56] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. *Attention Is All You Need*. arXiv, 2017. DOI: 10.48550/ARXIV.1706.03762. Available at: https://arxiv.org/abs/1706.03762.

[57] Vennerød, C. B., Kjærran, A. and Bugge, E. S. *Long Short-term Memory RNN*. arXiv, 2021. DOI: 10.48550/ARXIV.2105.06756. Available at: https://arxiv.org/abs/2105.06756.

[58] Virtanen, A., Kanerva, J., Ilo, R., Luoma, J., Luotolahti, J. et al. Multilingual is not enough: BERT for Finnish. *CoRR*. 2019, abs/1912.07076. Available at: http://arxiv.org/abs/1912.07076.

[59] Vo, N. and Lee, K. Where Are the Facts? Searching for Fact-checked Information to Alleviate the Spread of Fake News. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*. 2020.

[60] Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J. et al. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *CoRR*. 2019, abs/1905.00537. Available at: http://arxiv.org/abs/1905.00537.

[61] WANG, A., SINGH, A., MICHAEL, J., HILL, F., LEVY, O. et al. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *CoRR*. 2018, abs/1804.07461. Available at: http://arxiv.org/abs/1804.07461.

[62] WANG, W. Y. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2017.

[63] WILLIAMS, A., NANGIA, N. and BOWMAN, S. R. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *CoRR*. 2017, abs/1704.05426. Available at: http://arxiv.org/abs/1704.05426.

[64] WILLIAMS, R. J. and ZIPSER, D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*. 1989, vol. 1, no. 2, p. 270–280. DOI: 10.1162/neco.1989.1.2.270.

[65] XUE, L., CONSTANT, N., ROBERTS, A., KALE, M., AL-RFOU, R. et al. MT5: A massively multilingual pre-trained text-to-text transformer. *CoRR*. 2020, abs/2010.11934. Available at: https://arxiv.org/abs/2010.11934.

[66] ZHANG, X., MA, X., SHI, P. and LIN, J. Mr. TyDi: A Multi-lingual Benchmark for Dense Retrieval. *CoRR*. 2021, abs/2108.08787. Available at: https://arxiv.org/abs/2108.08787.

[67] ZHANG, X., OGUEJI, K., MA, X. and LIN, J. *Towards Best Practices for Training Multilingual Dense Retrieval Models*. 2022.

Appendix A

Introduction into the task

The aim of my thesis is to evaluate two different approaches to building a multilingual dataset for fact-checking models. What does a fact-checking model mean? In simple terms, you can imagine that a fact is passed to the model, for example: ”*President Barack Obama was born in 2012.*“ and the model tries to evaluate whether that fact is `true` or `false`. To train such a model, it is important to have a dataset that contains as many examples as possible. Each example must contain a claim, an evidence, that supports or refutes the claim and the correct label (in my case: `refute/support`). Nowadays, there are already many datasets, but they are all mostly only in English and thus there is a lack of multilingual datasets.

A similar task is a model that answers questions. A question is passed to the model and the model generates an answer. To train such a model, one needs a dataset that contains the question-answer pair, ideally including a record of where the answer to the question occurred.

It can be seen that the approach to training the two models is quite similar. Therefore, I decided to create the fact-checking dataset from another existing multilingual dataset for the question-answering model that contains question-answer pairs. From these pairs, the seq2seq model (a model that has a sequence of words as input and the output is again a sequence of words) was trained to generate a fact, since in the question we have information about what the answer specifically refers to, and the answer itself is the information we want to transfer to the fact.

Question: When was Barack Obama born?	**Answer**: 1961

Table 1: Example of the Question-Answer pair.

Since generating a fact is not a trivial task for the model (especially for multilingual data), two different approaches were therefore taken.

- The first approach is to train the model on the English dataset. The translated input (question and answer) is then passed to the model. The model returns the output (fact) in English, which is then translated back into the original language.
- The second approach is to train a multilingual model that receives the input in the desired language and therefore returns the output in the desired language.

Task for annotators

Automatic evaluation of the results of each approach is difficult to perform and therefore human evaluation is also important. In the attached file you have received an excel that contains the generated claim (translated into the english) from both models. These results need to be manually analyzed to determine whether the resulting fact is understandable or not.

Therefore, I would like to ask you to perform a manual analysis. One line of the file corresponds to one example from the dataset. Each line contains information about the original question and its answer. This is followed by the claim that was generated by model A and then the claim that was generated by model B. Your task is to assign to each claim one of your answers at your discretion: whether the claim is correct, whether the claim is wrong, or whether the claim is generated in an interesting way. An interesting claim contains all the necessary information, but is not quite as specific as you would expect.

Question: Where was Barack Obama born? **Answer**: USA
Claim: Barack Obama was born in Honolulu, Hawaii

Table 2: Example of the generated calim from the Question-Answer pair.

ToDo list for annotators

In summary, you need to add a label to each claim and then select a preference for the claim that you think is better generated:

1. fill in the column to the right of the claim with one of the answers:

correct	incorrect	interesting

Table 3: Labels for the claim validation.

2. fill in the last column with your preference for the generated fact based on the model as follows:

A	B	CBJ

Table 4: Labels for the claim preference.

* (CBJ stands for: cannot be judged)