UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Automated Fact-checking for Wikipedia

Author:
Mykola TROKHYMOVYCH

Supervisor:
Diego SAEZ-TRUMPER

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Mykola TROKHYMOVYCH, declare that this thesis titled, "Automated Fact-checking for Wikipedia" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Automated Fact-checking for Wikipedia

by Mykola TROKHYMOVYCH

Abstract

The incoming flow of information is continuously increasing along with the disinformation piece that can harm society. Filtering unreliable content helps keep Wikipedia as free as possible of disinformation, making it one of the most significant reliable information sources. Consequently, Wikipedia's knowledge base is widely used for facts verification academic research. The main goal of our work is to transform recent academic achievements into a practical open-source Wikipedia-based fact-checking application that is both accurate and efficient. We review the primary NLI related datasets and study their relevant limitations. As a result, we propose the data filtering method that improves the model's performance and generalization. We show that transfer learning for NLI models are not working well, and complete model training is needed to achieve the best result on a specific dataset. We come up with an unsupervised fine-tuning of the Masked Language model on field-specific texts for model domain adaptation. Finally, we present the new fact-checking system *WikiCheck* API that automatically performs a facts validation process based on the Wikipedia knowledge base. It is comparable to SOTA solutions in terms of accuracy and can be used on low memory CPU instances.

Acknowledgements

First of all, I would like to thank my supervisor Diego Saez Trumper who helped me during the whole research. That work would be impossible without his vision, support, and ardor. Also, I would like to thank my first mentor Alexander Kuzmenko for sharing his production experience and knowledge at the very beginning of my data science path. I would like to thank Jooble company for technical support in my research. I gratefully thank the Ring company for partially supporting my studies at UCU with Teacher Assistance Stipend. Also, I am very grateful to Ukrainian Catholic University and, personally, to Oleksii Molchanovskyi.

Contents

Declaration of Authorship ii

Abstract iii

Acknowledgements iv

1 Introduction 1
- 1.1 Importance of Fact Checking . . . 1
- 1.2 Motivation . . . 2
- 1.3 Contributions . . . 2
  - 1.3.1 Open problems . . . 2
  - 1.3.2 Research goals . . . 3
- 1.4 Thesis structure . . . 3

2 Related work 4
- 2.1 Problem formulation and datasets review . . . 4
- 2.2 Masked language modeling . . . 5
- 2.3 State of the art solution . . . 5
- 2.4 End-to-end fact verification solutions . . . 6

3 Data exploration 9
- 3.1 General purposes NLI datasets: SNLI and MNLI . . . 9
- 3.2 Wikipedia specific datasets: WIKIFACTCHECK-ENGLISH and FEVER 11
- 3.3 NLI datasets review summary . . . 13

4 System architecture 14
- 4.1 Application design . . . 14
- 4.2 Model level one. Wikipedia search API . . . 15
  - 4.2.1 Solution introduction. . . . 15
  - 4.2.2 Candidates selection validation. . . . 15
- 4.3 Model level two. Natural language inference model . . . 16
  - 4.3.1 General model architecture . . . 16

5 Experiments and validation 18
- 5.1 Model level one. Improving the performance of search . . . 18
  - 5.1.1 Improving performance of search . . . 18
  - 5.1.2 Using NER models for performance tuning. . . . 19
  - 5.1.3 NER experiments results . . . 20
  - 5.1.4 Limitations and alternatives . . . 21
- 5.2 Model level two. Building sentence-based NLI model. . . . 21
  - 5.2.1 Experiment setup . . . 21
  - 5.2.2 Masked language models. Efficiency testing . . . 21
  - 5.2.3 Sentence embeddings creation . . . 22
  - 5.2.4 Initial NLI models building. Comparing with SOTA . . . 23

5.2.5 Masked language models unsupervised fine-tuning 24
5.2.6 Performance on other datasets and transfer learning approach . 24
5.3 Building Wikipedia domain-specific NLI model 26
5.3.1 Experiment setup 26
5.3.2 Data preparation 26
5.3.3 Model training and validation 27
5.4 Analyzing models stability. Training on filtered data 29
5.4.1 Training on filtered data. Experiments with FEVER 29
5.4.2 Model performance depending on the length of text. FEVER dataset 30
5.5 WikiCheck: A complete Fact-Checking system based on Wikipedia . . 31
5.5.1 Adapting model for production 31
5.5.2 Difference from SOTA solutions 32
5.5.3 Experiment setup 32
Accuracy 32
Efficiency 33
5.5.4 Experiment results 34
FEVER Accuracy 34
Efficiency 34

6 Conclusions and Future Work **37**
6.1 Conclusions 37
6.2 Future Work 38

Bibliography **39**

List of Figures

3.1 Distribution of length of hypothesis in training dataset of SNLI (left picture) and MNLI (right picture) . . . 10
3.2 SNLI dataset top-15 the most frequent hypothesis and their classes counts . . . 10
3.3 Distribution of length of sentences across datasets (left picture), rate of sentences with punctuation (right picture) . . . 13

4.1 Automated fact checking software architecture. . . . 14
4.2 Model one validation process and example. . . . 16
4.3 Sentence based Siamese classifier with BERT encoder. . . . 17

5.1 BERT-like models encoding efficiency. . . . 22
5.2 Example of input and output for masked sentence. . . . 24
5.3 Confusion matrix for *bart-base* model (cleaned hypothesis training and validation). . . . 28
5.4 Classes distributions before and after filtering. . . . 29
5.5 Hypothesis length (FEVER) vs model's accuracy. . . . 30
5.6 Fact Checking system flow. . . . 32
5.7 Fact Checking system flow for FEVER validation. . . . 34
5.8 Confusion matrix for *WikiCheck* model . . . 35
5.9 Fact Checking system efficiency . . . 36

List of Tables

2.1 Literature review, word-based solutions. 6
2.2 Literature review, sentence-based solutions. 7
2.3 Top FEVER task solutions review 8

3.1 Samples from MNLI and SNLI datasets 9
3.2 Samples from WIKIFACTCHECK-ENGLISH dataset 11
3.3 FEVER data sample. Article linking. 12
3.4 FEVER data sample. SNLI-style relation dataset. 12
3.5 NLI datasets comparison 13

5.1 Comparing Model one configurations performance 20
5.2 Experiments results on SNLI datasets 24
5.3 Training on SNLI dataset and testing on SNLI and MNLI. 25
5.4 Training on MNLI dataset and testing on SNLI and MNLI. 25
5.5 Full training on specific dataset vs. training on SNLI and classifier fine tuning on FEVER and MNLI 25
5.6 Training on original FEVER dataset 27
5.7 Training on FEVER dataset with cleaned hypothesis 28
5.8 Training model on filtered FEVER dataset 30
5.9 Complete fact checking system FEVER accuracy 34
5.10 Fact checking system efficiency, seconds 36

List of Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
NLI	**N**atural **L**anguage **I**nference
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
LSTM	**L**ong **s**hort-**t**erm **m**emory
USE	**U**niversal **S**entence **E**ncoder
MLM	**M**asked **L**anguage **M**odel
MLP	**M**ulti**l**ayer **P**erceptron
SOTA	**S**tate-**o**f-**t**he-**A**rt
TF-IDF	**T**erm **f**requency–**i**nverse **d**ocument **f**requency
POS	**P**art-**o**f-**s**peech
API	**A**pplication **P**rogramming **I**nterface
NSP	**N**ext **S**entence **P**rediction
SRL	**S**emantic **R**ole **L**abelling
GLUE	**G**eneral **L**anguage **U**nderstanding **E**valuation
SNLI	**S**tanford **N**atural **L**anguage **I**nference (dataset)
MNLI	**M**ulti**G**enre **NLI** (dataset)
FEVER	**F**act **E**xtraction and **Ver**ification (dataset)
CV	**C**omputer **V**ision
AR	**A**verage **R**ecall
NER	**N**amed-**e**ntity **r**ecognition
CPU	**C**entral **p**rocessing **u**nit
GPU	**G**raphics **p**rocessing **u**nit
NN	**N**eural **n**etwork
VPS	**V**irtual **p**rivate **s**erver
RAM	**R**andom-**a**ccess **m**emory
R	**R**EFUTES
S	**S**UPPORTS
NEI	**N**OT **E**NOUGH **I**NFO

Dedicated to our bright future

Chapter 1

Introduction

1.1 Importance of Fact Checking

Disinformation can influence elections, stock prices, and even how we treat ourselves for a virus. False facts are spreading faster than the truth and can negatively impact society and business (Vosoughi, Roy, and Aral, 2018).

Social networks allow people to post information wherever and whenever they want. (Bovet and Makse, 2019) showed that in 2016, Trump supporters' activity influenced the dynamics of the top fake news spreaders, which impacted US presidential elections. In 2013, $130 billion in stock value were lost just because of one fake tweet about an "explosion" that injured Barack Obama[1].

Manual fact-checking is time-consuming and can come too late. Automation reduces time to "stick" in the audience's minds (Cazalens et al., 2018). It can prevent propaganda by filtering manipulation and false facts in nearly real-time. That is the reason why researchers are working on creating Automated Fact-Checking systems using Wikipedia as a ground-truth knowledge base (Thorne et al., 2018c; Nie, Chen, and Bansal, 2018; Yoneda et al., 2018; Hanselowski et al., 2018). However, most of that work remains as academic research. That is why usually people proceed with manual fact-checking and usually use Wikipedia as the initial source of open and valid information (Heilman and West, 2015).

Wikipedia is the world's largest repository of human knowledge. It makes this website the most visited knowledge base on the planet according to Similarweb[2]. Wikipedia became one of the resources in the global web dedicated to fighting against fake news, developed numerous practices and policies for information validation and verification by an army of community fact-checkers (McDowell and Vetter, 2020). Our work is developing an end-to-end fact-checking system that relies on the Wikipedia knowledge base, so anyone can verify the fact by providing an initial claim.

We are using best practices and results from the Natural Language Inference (NLI) domain to automate the usual human fact verification process with the help of Wikipedia encyclopedia articles. Natural Language Inference (NLI) model aims to predict an entailment relation label given a claim-hypothesis pair. The goal is to determine whether the truth of the hypothesis follows from the truth of the premise (claim) (Dagan, Glickman, and Magnini, 2006). We consider the NLI model a core block of an automated fact-checking system and set the goal of making it as accurate as possible.

[1] Forbes, Can 'Fake News' Impact The Stock Market? http://bit.ly/fake_news_impact.
[2] Similarweb. Top Dictionaries and Encyclopedias. http://bit.ly/wiki_similarweb.

1.2 Motivation

Automated fact-checking and NLI has a significant social impact, and it is currently developing very fast in academia. However, there is a gap between research achievements and applicability in real life. This project aims to review the state-of-the-art solutions to the NLI problem, reproduce the results, and develop an end-to-end open-source tool to perform automated fact-checking. An essential part of our research is measuring the efficiency of models as a crucial part of real-life applications. We intend to experiment with different model configurations to reveal their strengths and weaknesses and select the best configuration for our specific case.

Finally, we want to contribute to the community with the open-source tool for automated fact-checking based on open Wikipedia knowledge. We believe such an accessible system will help to fight against disinformation and fake news.

1.3 Contributions

1.3.1 Open problems

Previous works in NLI field (Zhang et al., 2020; Liu et al., 2019a; Chen et al., 2016; Kiela, Wang, and Cho, 2018; Talman, Yli-Jyrä, and Tiedemann, 2019) were more concentrated on accuracy of the models. However, the speed of the application is crucial for its practical usage. Also, most SOTA solutions are implemented for GPU usage, require significant computational resources, or do not have published code. It makes it difficult to reproduce the results and use models in practice. The reproducibility, efficiency, and accuracy of NLI models remain an open problem.

Transformer-based models made a big boost for all NLP tasks, including NLI. SOTA results presented by Zhang et al., 2020; Liu et al., 2019a are based on Masked Language Models (MLM). However, these solutions present a word-based approach, when the most recent research for sentence-based NLI models like Chen et al., 2016; Kiela, Wang, and Cho, 2018; Talman, Yli-Jyrä, and Tiedemann, 2019 are not using MLM for modeling. Moreover, sentence-based models are less accurate compared to word-based models on the SNLI dataset[3]. To sum up, one of the open challenges is using transformers for sentence-based NLI models and improving their accuracy. At the same time, using sentence-based models have crucial efficiency benefits for general automated fact-checking application, that will be shown later in Sections 4.3 and 5.5.1

Another open problem is the scarcity of high-quality NLI datasets for model training. Most of the real-world datasets for fact-checking are small or incomplete to train machine learning models. Although there are relatively big (semi)synthetic datasets, they reported having many artifacts left by crowd workers that were creating them (Gururangan et al., 2018). Creating a new dataset is costly as it requires manual annotation.

Moreover, most benchmark datasets are from the general knowledge domain. There is an open question of practical usage of the models trained on that data for a specific field like Wikipedia.

One more open problem is a software architecture for end-to-end fact-checking. The general system and each independent stage need to deal with the trade-off between speed and accuracy. At the same time, this process requires deep analysis of vast amounts of data. Some possible approaches are described as a solution

[3]SNLI published results comparison. https://nlp.stanford.edu/projects/snli/.

for the FEVER task presented by Nie, Chen, and Bansal, 2018; Yoneda et al., 2018; Hanselowski et al., 2018. Although they show an end-to-end approach for fact verification problem solving, the efficiency and usability of such solutions for real-world applications remain an open question.

1.3.2 Research goals

The main goal of the research is to transform academic research on Automated Fact-Checking into a practical open-source application used for fact verification using the Wikipedia knowledge base. We formalized and stated the following research tasks that will help us to achieve our main goal:

1. Analyze previous works in the field of Automated Fact-Checking and Natural Language Inference.
2. Review related datasets that can be used for model training. Define the specific features and limitations of NLI data, and design a methodology for data quality improvement.
3. Develop sentence-based NLI models and compare their accuracy and efficiency with SOTA models.
4. Test the NLI models generalization ability by training and testing models using datasets from different domains.
5. Build domain-specific NLI model. Formulate unsupervised learning and transfer learning solutions for models domain adaptation.
6. Implement an open-source fact-checking API based on the best NLI model. Compare its performance with SOTA solutions. Measure and analyze its efficiency.

In summary, this work contributes with an end-to-end open-source solution for Automated Fact-Checking that builds on previous research but focusing on making it usable in real-life scenarios, and more specifically, using Wikipedia as a knowledge base for fact-checking.

1.4 Thesis structure

In this work, we start revising relevant research on the field of Automated Fact-Checking. Next, we proceed with the exploratory data analysis, where we review NLI problem-related datasets. In the next chapter, we present our solution architecture describing their components and implementation details. Then, we describe our experiments for all parts of the automated fact-checking system and general application testing. All experiments are aimed to answer our research questions. Finally, we summarize our results and present the main conclusions of this work.

Chapter 2

Related work

In this chapter, we organize related work in four major categories: *(i)* fact-checking problem formulation and review; *(ii)* language modeling; *(iii)* NLI state of the art solutions; *(iv)* End-to-end fact verification solutions. For each paper, we review the main contribution and realization details. Such analysis allows us to have a general overview of that topic and the most recent results.

2.1 Problem formulation and datasets review

The problem of fact-checking was initially used in journalism as an essential part of news reporting. One of the first datasets published on this domain consisted of 221 labeled claims - related to politics - checked by Politifact[1] and Channel4 with related sources of evidence (Vlachos and Riedel, 2014). After that, similar data collection but much more extensive, containing 12.8K labeled claims from Politifact, was released by Wang, 2017. However, this can be considered a small collection of data to train large deep-learning models.

In 2015 SNLI dataset was presented and became the primary benchmark dataset used for the NLI problem (Bowman et al., 2015). Even though it is not specialized in specific topics like politics, it is large enough (570K pairs of sentences) to train large models. SNLI consists of pairs of sentences with relation labels: entailment, contradiction, or neutral. It is created by crowd workers, showing them a sentence and asking them to generate three new sentences (hypotheses) for each entailment class (Bowman et al., 2015). In 2018 the MultiNLI dataset was released, which is almost the same as SLNI(Bowman et al., 2015), but has improving topics coverage and complexity of claims (Williams, Nangia, and Bowman, 2017).

As the primary benchmark dataset, SNLI has one crucial drawback: models that did not even look at the evidence perform well on the NLI task. This behavior is explicitly reviewed by (Gururangan et al., 2018), where authors reveal linguistic annotation artifacts in SNLI. They show specific words in texts which are highly correlated with certain inference classes.

There are also alternatives for SNLI and MNLI. Sathe et al., 2020 release the *WIKIFACTCHECK-ENGLISH* dataset (124K triplets of sentences), which consists of real-world claims from Wikipedia. Similarly, the FEVER dataset presented by (Thorne et al., 2018a) has a more complex structure based on the Wikipedia dump. A more detailed description of these four datasets is presented in Chapter 3.

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

2.2 Masked language modeling

The most crucial part of a modern NLI solution is language models. The recent state-of-the-art solutions are built on top of them. Therefore, to create a valuable NLI model, we also need to review the literature about language modeling. The most recent language modeling results are based on transformers architecture.

One of the most valued recent contributions to NLP is the BERT architecture, which stands for Bidirectional Encoder Representations from Transformers (Devlin et al., 2018). BERT model made a revolution in the NLP field. It significantly moved state-of-the-art scores for several NLP tasks by presenting new architecture for language modeling. It showed point absolute improvement 7.7% on GLUE score (Wang et al., 2018). The authors present a solution allowing it to be bidirectional and utilize the masked language model (MLM) and next sentence prediction (NSP) as a pretraining objective. The MLM training process is built on masking some of the tokens and predicting them based only on their context (Devlin et al., 2018). Training using NSP loss is working by choosing the two sentences as training sample, 50% of the time, one is following another one (labeled as IsNext), then another half represented by random sentences from the corpus (labeled as NotNext). Training models with NSP loss is beneficial for NLI problem according to Devlin et al., 2018. Then RoBERTa model was presented, which improved previous results of BERT. Liu et al., 2019b introduce a replication study of BERT, their research is built on removing the NSP loss, using a much bigger dataset for training that consists of 160GB of text, and increasing the number of pretraining steps from 100K to 500K.

Another relevant work for our research is the Sentence-BERT. Authors present a way to train sentence embeddings instead of word embeddings using the pretrained transformer model and Siamese network (Reimers and Gurevych, 2019). This approach allows the dump of precalculated sentence embeddings, reusing them for different tasks, improving the model's efficiency, and making transformer models like BERT and RoBERTa possible to use in high-load production tasks.

2.3 State of the art solution

We divided SOTA solutions in two groups: *(i)* sentence-based and *(ii)* word-based. The difference is that in the case of a word-based solution, sentences are represented as a set of word vectors, while in the sentence-based, a single vector is used as a sentence representation and then used for building the model that will solve the NLI task. Sentence-based solutions are usually faster, more applicable in real life as vectors can be cached. However, word-based solutions are more precise. For each paper, we defined the contribution along with the approach and SNLI score. We structured this analysis in Table 2.1 and Table 2.2.

Gong, Luo, and Zhang, 2017; Chen et al., 2017 represent models that are not based on BERT, as were created earlier. However, they present very different approaches along with good results. When Liu et al., 2019a; Zhang et al., 2020; Pilault, Elhattami, and Pal, 2020 present results of models based on BERT architecture with various modifications and learning strategies.

Most of the best sentence-based models are build using LSTM architecture. Also, we see that scores for those types of models are lower than word-based. However, such models can be used in production high-load tasks as they are lighter, faster. Better efficiency of sentence-based models is caused by their ability to cache intermediate results like sentence embeddings and lighter architecture.

TABLE 2.1: Literature review, word-based solutions.

Name, source	Description, contribution	Explanation of approach	SNLI Score
Neural Natural Language Inference Models Enhanced with External Knowledge (Chen et al., 2017)	Use external knowledge from words meaning. State-of-the-art performance with a relatively small number of parameters of $\sim$ 4M.	Use information about synonymy, antonym, hypernym and hyponymy existance in attention layer.	88.6
Natural Language Inference over Interaction Space (Gong, Luo, and Zhang, 2017)	Combines both NLP and computer vision (CV) approaches. Based on a high-level understanding of the sentence pair relation.	Create a tensor representation of pairs of texts using their word embeddings, manipulate it to extract semantic features, and do the classification.	88.9
Multi-Task Deep Neural Networks for Natural Language Understanding (Liu et al., 2019a)	Training BERT model on multiple natural language understanding (NLU) tasks simultaneously, benefiting from a regularization.	Use BERT model along with Lexicon encoder, adding extra information about position and word's segment. Fine-tune the model simultaneously for four different NLP tasks	91.6
Semantics-aware BERT for Language Understanding (Zhang et al., 2020)	Integrates contextualized features into language model. Extend the language representation model with semantics. State-of-the-art result for SNLI.	Use semantic role labeling (SRL) model with BERT to parse the predicate-argument structure. Fine-tune the model separately for different tasks.	91.9
Conditionally Adaptive Multi-Task Learning: Improving Transfer Learning in NLP Using Fewer Parameters & Less Data (Pilault, Elhattami, and Pal, 2020)	The latest state-of-the-art result, using idea and results of (Liu et al., 2019a). The current approach to learning different sets of parameters while fine-tuning different NLP tasks. Demonstrate faster fine-tuning as most parameters are frozen and dataset balanced across different tasks that reduce data to around 60%.	Upgrade standard transformer architecture with five additions: conditional attention, conditional alignment, conditional layer normalization, conditional adapters, and multi-task uncertainty sampling to have a specific approach to each NLP task-saving unified architecture.	92.1

2.4 End-to-end fact verification solutions

In this section, we review the top solutions presented as the solution for FEVER Shared Task presented by Thorne et al., 2018c along with end-to-end production systems. The FEVER challenge was to implement an automated fact verification

TABLE 2.2: Literature review, sentence-based solutions.

Name, source	Description, contribution	Explanation of approach	SNLI Score
Sentence Embeddings in NLI with Iterative Refinement Encoders (Talman, Yli-Jyrä, and Tiedemann, 2019)	Hierarchical BiLSTM model with Max Pooling for building sentence embeddings and further tuning for NLI task. Present error analysis.	Use advanced architecture based on iterative refinement strategy. Build sentence embeddings and then use the MLP model to use those vectors in NLI task	86.6
Dynamic Meta-Embeddings for Improved Sentence Representations (Kiela, Wang, and Cho, 2018)	Project utilize dynamic meta-embeddings for sentence embeddings composition and building further model on top of them.	Use composition of different embeddings like Word2Vec (Mikolov et al., 2013) or fasttext (Bojanowski et al., 2017). Learns the weights for the composition of defined vectors and uses BiLSTM to compose the sentence embeddings used for the NLI task.	86.7
Enhanced LSTM for Natural Language Inference (Chen et al., 2016)	Present carefully designing sequential inference that outperforms complicated network architectures and state new state-of-the-art result for sentence based models	Use the BiLSTM block to represent a word and its context and inference composition before the final prediction. Use Local Inference Modeling for determining the overall inference between these pair of texts.	88.6

system. It differs from standard NLI formulation, as here we need not just classify relation between two sentences but also pick the evidence (hypothesis) sentence from a knowledge base. That makes this task more complicated and more close to the real-world scenario at the same time. We will analyze the top solutions of that task in this section. As a baseline system presented by Thorne et al., 2018b, most of the solutions are multistage models that perform document retrieval, sentence selection, and sentence classification. Baseline exploits TF-IDF-based retrieval to find the relevant evidence and an NLI model to classify the relationship between the returned evidence and the claim. The top scorers present their approach to solve the problem and improve results. Main ideas of those works presented in comparison Table 2.3. As we can see, all solutions are very similar but have specific features.

Also, we previously mentioned that there are production solutions that offer fact validation services like Logically[2] or Claimbuster (Hassan et al., 2017). However, they have crucial drawbacks. For example, the knowledge bases used in these systems are usually proprietary or unknown. Also, the actual accuracy of systems is not tested against benchmark datasets. The Logically service is offered as an "automated fact-checking," but the service mixes manual fact-checkers and some automated process.

[2]Logically AI https://www.logically.ai

TABLE 2.3: Top FEVER task solutions review

Name, source	**Explanation of approach**	**FEVER Score**
FEVER: a large-scale dataset for Fact Extraction and Verification (Thorne et al., 2018b)	Uses tf-idf based retrieval to find the relevant evidence and an MLP based NLI model to classify the relationship between the returned evidence and the claim.	0.28
UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification (Hanselowski et al., 2018)	Uses entity linking and WikiMedia search API for article search. Uses Glove and FastText embeddings for the NLI model. Make a detailed analysis of each stage of the model.	0.61
UCL Machine Reading Group: Four Factor Framework For Fact-Finding (HexaF) (Yoneda et al., 2018)	Uses four stage architecture adding aggregation logic on top. Uses logistic regression for document and sentences retrieval stage.	0.62
Combining Fact Extraction and Verification with Neural Semantic Matching Networks (Nie, Chen, and Bansal, 2018)	Implemented neural models to perform deep semantic matching from raw textual input for document and sentence retrieval stages. Added WordNet features in order to improve NLI model	0.64

Chapter 3

Data exploration

The most recent research in the NLP field is strongly bound to data. The SOTA results are achieved not only because of innovative models but also because of significant amounts of data, accurate filtering techniques, and understanding of data nature. The next step in the research is review and analysis of data to get valuable insights from them.

As for our research, we considered using multiple datasets for training and validation. This chapter will review and discuss the primary datasets used to solve the NLI problem and their characteristics. We divided all datasets into two subgroups: SNLI and MNLI, and another one are FEVER and WIKIFACTCHECK-ENGLISH. The first group is standard datasets used as a benchmark for NLI tasks when datasets from another group represent domain-specific data, which will be used for domain adaptation. Also, the FEVER dataset has a more complex structure that allows us to use it for more specific task formulation.

3.1 General purposes NLI datasets: SNLI and MNLI

SNLI and MNLI are the most used benchmark datasets. They allow us to compare results with the most recent SOTA results, as most recent papers validate them. SNLI and MNLI datasets consist of claim, related hypothesis, and label, which is either neutral, contradiction, or entailment (Table 3.1).

TABLE 3.1: Samples from MNLI and SNLI datasets

Dataset	**Claim**	**Hypothesis**	**Label**
MNLI	The Old One always comforted Ca'daan, except today.	Ca'daan knew the Old One very well.	neutral
MNLI	At the other end of Pennsylvania Avenue, people began to line up for a White House tour.	People formed a line at the end of Pennsylvania Avenue.	entailment
SNLI	A man inspects the uniform of a figure in some East Asian country.	The man is sleeping	contradiction
SNLI	An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor.	neutral

It is important to mention that all classes are well balanced and have almost the same amount of samples. We analyzed distributions of the length of three classes'

claims and hypotheses and found that the claims' length is equally distributed. At the same time length of the hypothesis are different within different classes. Figure 3.1 shows that the entailment class hypothesis is usually shorter than others. It can influence the model that could learn the length of a sentence instead of its meaning. The MNLI dataset situation is much better as distributions of the length of texts are more balanced than in SNLI, but the neutral class sentences are usually longer. Moreover, we see that the MNLI hypotheses are larger than SNLI, making it closer to the real world. We should also mention that MNLI sentences come from different semantic domains, so this dataset is more generalized than SNLI, which is made of only image captions (Williams, Nangia, and Bowman, 2017).

In our exploration, we found out the reason why certain words in the hypothesis are highly correlated with specific classes, as was discussed by Gururangan et al., 2018. We defined top-15 the most frequent hypothesis used by annotators and analyzed the classes to which they correspond. We found out that frequent hypotheses are usually used in either entailment or contradiction class, represented in Figure 3.2. It is also not natural behavior as the model will learn only the sense of hypothesis instead of the desired relation between claim and hypothesis. In Chapter 5 we analyze how filtering out such patterns from training will influence the models' validation results.

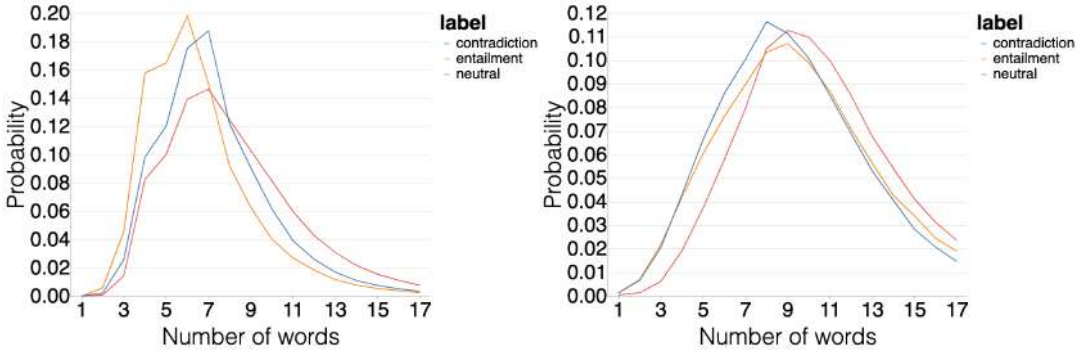

FIGURE 3.1: Distribution of length of hypothesis in training dataset of SNLI (left picture) and MNLI (right picture)

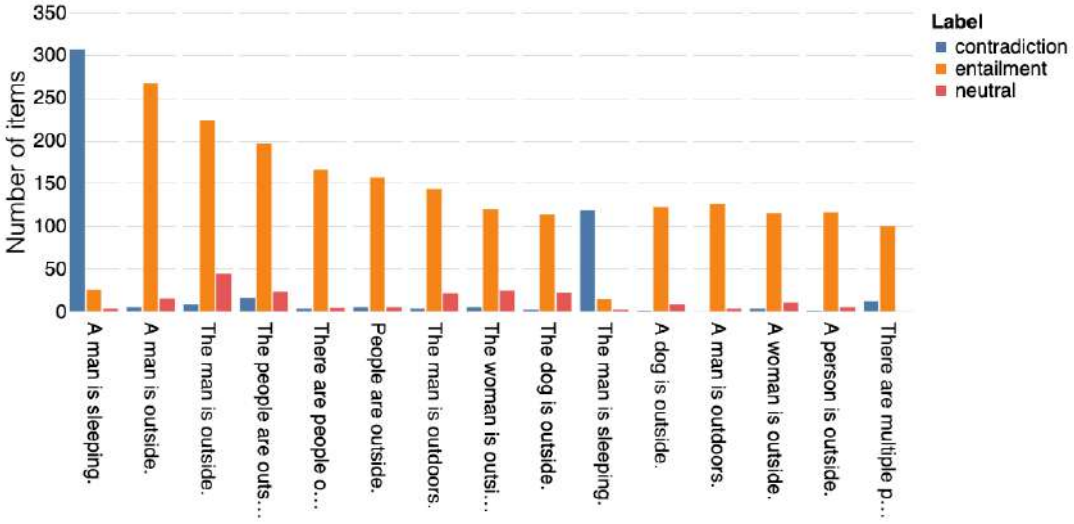

FIGURE 3.2: SNLI dataset top-15 the most frequent hypothesis and their classes counts

3.2 Wikipedia specific datasets: WIKIFACTCHECK-ENGLISH and FEVER

In this section, we review Wikipedia domain-specific datasets. Wikipedia text is created *in real-life scenarios* differing from the artificially created ones mentioned above. Later, in Chapter 5 we compare the performance of models trained on SNLI and MNLI applied on the Wikipedia-related ones.

WIKIFACTCHECK-ENGLISH was presented as a large-scale dataset of factual claims, context, and evidence documents extracted from the English Wikipedia, along with manually written claims refuted by the evidence documents (Sathe et al., 2020). The most relevant part for our research are triplets of context, the refuted claim and valid claim as they will allow us to use it in classical formulation when we have to classify relation. However, after initial investigation, we found out that most of the sentences are either unrelated or too easy to classify, having only one token changed. Also, we revealed that such cases are not minor and could not be automatically filtered out. Such examples are presented in Table 3.2. We link each example with an ID that corresponds to the line number in the training file, which can be found in the official Github repo[1]. As a result of the initial analysis of the WIKIFACTCHECK-ENGLISH dataset, we decided not to proceed with it for our further research.

TABLE 3.2: Samples from WIKIFACTCHECK-ENGLISH dataset

ID	Context	True claim	Refuted claim
0	A lunar sortie (or lunar sortie mission) is a human spaceflight mission to the Moon.	In contrast with lunar outpost missions, lunar sorties will be of relatively brief duration.	In contrast with lunar outpost missions, lunar sorties will be of relatively long duration.
5	Richard M. Dick Bond (April 23, 1921 – March 25, 2015) was an American politician from Washington State.	He served the 6th district from 1975 to 1987.	He served the 7th district from 1975 to 1987.
13	Liometopum venerarium is an extinct species of Miocene ants in the genus Liometopum.	Described by Heer in 1864, fossils of the species were found in Switzerland.	Described by Heer in 1864, fossils of the species were brought to Switzerland.

FEVER dataset consists of 185,445 claims generated by altering sentences extracted from Wikipedia and subsequently verified without knowing the sentence they were derived from. The claims are classified as *SUPPORTED*, *REFUTED*, or *NOT-ENOUGH-INFO* by annotators (Thorne et al., 2018a). This dataset differs from other previously discussed, as it represents another problem formulation because it not only implies classifying the relation between two pieces of text but also linking the given claim with corresponding evidence in a knowledge base. That knowledge base is a Wikipedia dump. This is a more generalized problem setting that is very close to the real-life scenario, simulating what humans could do to fact-check a given claim. This dataset is the main validation for our research as it is of good quality, represents written speech, and Wikipedia domain-specific.

[1]WIKIFACTCHECK-ENGLISH Github repo https://github.com/wikifactcheck-english/wikifactcheck-english.

TABLE 3.3: FEVER data sample. Article linking.

Claim	**Evidence Articles**
Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.	Fox_Broadcasting_Company, Nikolaj_Coster-Waldau
Hermit crabs are arachnids.	Arachnid, Hermit_crab, Decapoda
There is a capital called Mogadishu.	Mogadishu

TABLE 3.4: FEVER data sample. SNLI-style relation dataset.

Claim	**Hypothesis**	**Label**
Roman Atwood is a content creator.	He is best known for his vlogs, where he posts updates about his life daily.	SUPPORTS
Adrienne Bailon is an accountant.	Adrienne Eliza Houghton (née Bailon; born October 24, 1983) is an American singer-songwriter, recording artist, actress, dancer, and television personality. née name at birth singer-songwriter	REFUTES
Selena recorded music.	Selena began recording professionally in 1982. Selena Selena (film)	SUPPORTS

The original dataset consists of a claim, label, and evidence link. In case the label is *NOT ENOUGH INFO*, there is no corresponding link. The evidence link is the name of the Wikipedia article with the number of the sentence in that article. Also, we have the actual Wikipedia dump that allows us to find out the evidence sentence. In our case, we used the FEVER dataset to build two types of datasets: The first one consists of a claim and article link as in Table 3.3. The second one contains a claim, corresponding evidence sentence from Wikipedia dump and label. That is the SNLI style of the dataset with only one difference that we have two classes: *REFUTES* and *SUPPORTS*—the sample of such data presented in Table 3.4.

Also, we review FEVER dataset characteristics and got some insights that differentiate it from SNLI and MNLI. First of all, the FEVER dataset has a different distribution of sentence length. It can be found in Figure 3.3 (left part) that claims are usually short when the evidence sentences are longer than in MNLI and SNLI.

FEVER hypotheses are sentences from the summary section of related articles taken from the Wikipedia dump. Usually, these texts include most of the main information about a related topic that explains their big irregular length. Also, Wikipedia dump sentences include tags to related named entities at the end of a sentence. For example, the sentence *"Selena began recording professionally in 1982.\tSelena\tSelena (film)"* includes tags *Selena* and *Selena (film)*. These tags directly influence the length of the sentence and potentially add extra noise to it. We consider filtering these artifacts in Section 5.3.2.

Also, an interesting pattern found that the dataset has a different rate of punctuation signs in sentences. In the Figure 3.3 (right part) we showed the ratio of sentences that include punctuation (not including "." and ","). This also differentiates FEVER, as almost 65% of its hypothesis include punctuation. Interesting that SNLI is almost free of punctuation at all.

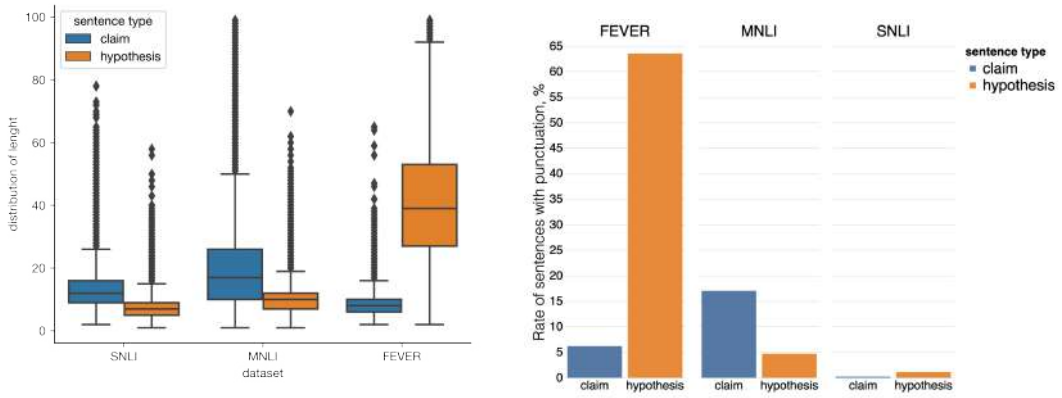

FIGURE 3.3: Distribution of length of sentences across datasets (left picture), rate of sentences with punctuation (right picture)

3.3 NLI datasets review summary

In order to summarise our analysis, we made a comparison table 3.5 that contains a brief summary of datasets reviewed in this section.

TABLE 3.5: NLI datasets comparison

Dataset name	Domain	Number of samples	Classes	Description
SNLI	Image captions. (general domain)	570K sentence pairs	entailment, contradiction, and neutral	One of the first good quality NLI datasets widely used for benchmarking solutions.
MNLI	Wide range of styles, degrees of formality, and topics. (general domain)	433K sentence pairs	entailment, contradiction, and neutral	Data are formatted in the same way and comparable in size with SNLI. It includes a more diverse range of text domains (Williams, Nangia, and Bowman, 2017).
WIKI FACT CHECK ENGLISH	Wikipedia texts. (specific domain)	124K triplets + 34K manually refuted claims	True, Refuted	The dataset presented as a set of actual claims and evidence, that differ from synthetic SNLI and FEVER (Sathe et al., 2020).
FEVER	Wikipedia texts. (specific domain)	145,449 of train and 19,998 of test	not enough info, support, refute	It is generated by altering sentences from Wikipedia. It stimulates claim verification against textual sources (Thorne et al., 2018a).

Chapter 4

System architecture

One of our goals is to implement an open API that will automatically perform the fact validation process. In this chapter, we propose our fact-checking system architecture. We decompose the application into two major parts of the candidates selection model (Model level one) and NLI classification model (Model level two). We present a possible approach to building and validating each of the models.

4.1 Application design

The main idea of our approach is to reproduce the human way to do the fact-checking process. In such formulation, the initial input is usually a claim, which is the piece of text, that should be checked.

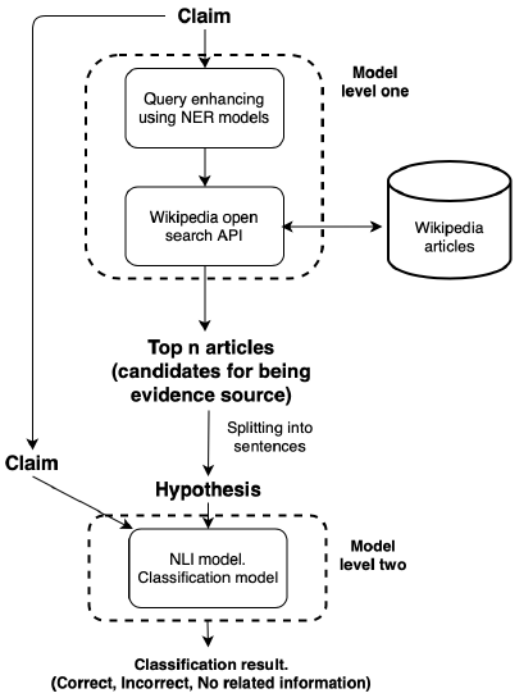

FIGURE 4.1: Automated fact checking software architecture.

Firstly, we try to find related facts, evidence of correctness, or wrongness of such a claim using trusted sources. Having found hypothetical evidence related to our claim, we compare these two pieces of text to decide if they demonstrate the same fact or not. At this stage, we can conclude that the found hypothesis either supports

the initial claim, refutes it, or does not relate to it. Having that vision, we propose a possible architecture of automated fact-checking software (Figure 4.1).

The critical point of our architecture is splitting it into two logical parts. Firstly, having a claim, we use it to extract related articles that possibly include desired evidence. We exploit open search API, which copies human behavior of looking for related information using Wikipedia. Note that we are aware of Wikipedia's credibility issues (Saez-Trumper, 2019). However, given its size and openness, Wikipedia is still one of the most used sources for (pre)research and fact-checking (Tomaszewski and MacDonald, 2016; Back et al., 2016; Heilman and West, 2015). Here we experiment with query enhancing techniques to improve the search quality. After the first stage, we have a set of articles that possibly include desired evidence of correctness or wrongness of the initial claim. We split each article into sentences and pass each of them to the second stage model. In that phase, having claim and hypothesis sentences, we exploit the NLI classification model to define their relationship and, as a result, say if our initial claim is correct, incorrect, or we have no related information in our knowledge base regarding it.

4.2 Model level one. Wikipedia search API

In this section, we discuss the usage of a combination of open-source models. It is essential to mention that we did not intend to contribute to the domains of full-text search, named entities recognition, or query enhancing. We reviewed and used previous research works that helped us achieve our goal to implement an automated fact-checking system.

4.2.1 Solution introduction.

As it was discussed before, we decided to use Wikimedia API[1] as the first level model used for candidates selection. It allows us to look for up-to-date articles related to our claim using the official search engine. In 2014 Wikipedia started to use Elasticsearch as a base of new search infrastructure to all of the wikis[2]. So, the first level model performs a full-text search through the whole Wikipedia index. Although we cannot influence the Wikipedia search engine, which is a significant limitation of our approach, but we can improve the query itself, which showed an excellent boost for our task.

4.2.2 Candidates selection validation.

On this stage we used aforementioned FEVER dataset to validate our solution.At this stage, we are not training an NLI model yet, and we do not risk overfitting; we use the whole joined FEVER dataset for validation. We use a claim column that corresponds to the model's input and an evidence column containing information about the ground truth Wikipedia page link we desire to get as an output.

That is important to mention that samples with the class *NOT ENOUGH INFO* do not include Wikipedia page link, so they were filtered out for this experiment leaving 123142 rows from joined FEVER dataset. One more important thing is that the FEVER dataset was collected regarding Wikipedia pages dumped in June 2017.

[1] WikiMedia API https://www.mediawiki.org/wiki/API:Search.

[2] Wikimedia moving to Elasticsearch https://diff.wikimedia.org/2014/01/06/wikimedia-moving-to-elasticsearch/.

At the same time, Wikimedia API returns results from an up-to-date index. It means that some links could be changed. We ran a script that went through FEVER evidence links and validated their existence to detect such cases. After that process, we filtered out 5312 (4.3%) more rows with changed or deleted links that we cannot validate. Finally, we obtained 117830 rows of fine-grained test datasets used to evaluate Model-level one.

The validation process itself is represented in Figure 4.2 with corresponding example. The general idea is given a claim we pick up a set of candidate articles using Wikimedia API. Then we compare the obtained set with ground truth items provided in the FEVER dataset. We use the Average Recall (AR) metric for results validation and comparison. The recall for one specific search is calculating as following:

$$\text{Recall} = \frac{true\ positives}{(true\ positives) + (false\ negatives)}$$

The AR is just the average of recall metrics for each search. This metric does not consider the negative samples retrieved, but we are interested in all relevant candidates finding, so the recall works perfectly for our purposes.

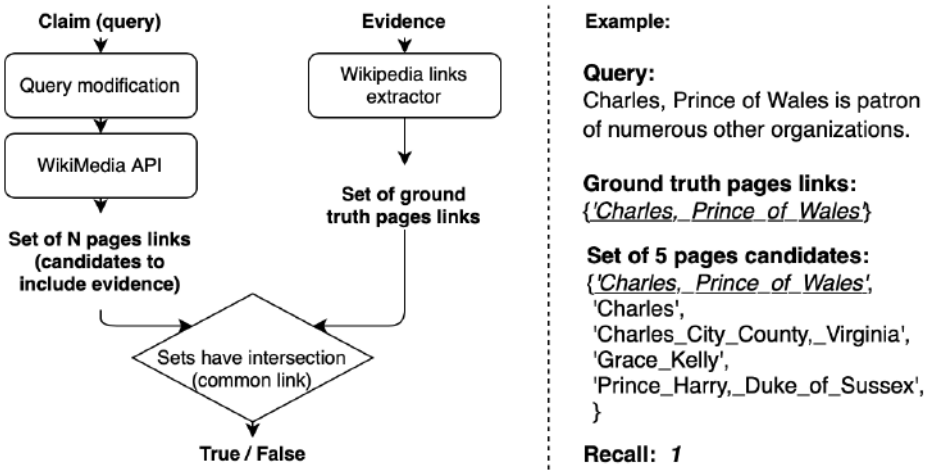

FIGURE 4.2: Model one validation process and example.

4.3 Model level two. Natural language inference model

4.3.1 General model architecture

Model level two is the Natural language inference model that is making three classes classification. It aimed to define exact sentences in predefined candidates' articles that are evidence of correctness or wrongness of a given claim. It has three possible outputs: *SUPPORTS*, *REFUTES*, and *NEI*. We are using the sentence-based model to have crucial benefits for real-world usage and general solution performance. In this subsection, we present a general idea of our solution and justify our choices.

The general idea of the presented NLI model is a Siamese network using a BERT-like model as a trainable encoder for sentences. The model architecture is presented in Figure 4.3. The idea goes from Conneau et al., 2017a, with the difference that we are not using multiplication of sentence vectors in concatenation layer, but using

only original vectors and their absolute difference. The approach is not new, that was used by Reimers and Gurevych, 2019 for training sentence embeddings, but it was not presented as an efficient solution for the NLI problem. In our work, we enhance that approach and use it to build end-to-end fact-checking API for Wikipedia.

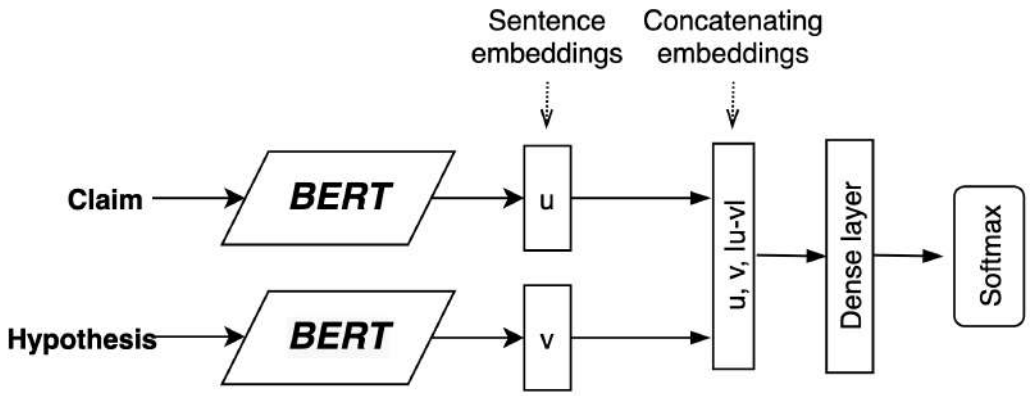

FIGURE 4.3: Sentence based Siamese classifier with BERT encoder.

In our problem formulation, we need to compare one claim with multiple hypothesis sentences. The sentence-based model approach enables to calculate claim embedding only once and reuse it for every hypothesis. Also, embeddings for the hypothesis can be batch processed and precalculated in advance. The presented architecture enables to cache intermediate results of embeddings and reuse them for online prediction. In the following sections, we present the results of our experiments with different BERT-like encoders and training strategies.

Chapter 5

Experiments and validation

This Chapter describes the experiments done for different configurations of models at all levels along with general architecture performance. Here we present results of experiments using Named entity recognition (NER) models for search improvement, different approaches to building sentence-based models, training and validating models using different datasets, and comparing results with SOTA solutions. As for our architecture, we use the model represented in Figure 4.3. We will also experiment on how different masked language models influence the accuracy and efficiency of the solution. Finally, we will test our approach on the FEVER shared task and compare results with top solutions.

5.1 Model level one. Improving the performance of search

Model level one goal is selecting candidates for further analysis. It is a crucial building block of the final solution as, without well-picked candidates, we will not get good enough results. In this section, we review our experiments aimed to improve the basic approach to candidate selection. The detailed validation process, including data and metrics used, was reviewed in Section 4.2.2. So in the following sections, we concentrate on experiments and results discussion.

5.1.1 Improving performance of search

The first important hyperparameter that we can control for model level one is N, the number of candidates to extract. The bigger is N, the higher probability of finding the desired candidate in the obtained set. At the same time, the number of candidates selected on the first stage influences the general model efficiency, as the more candidates we select on the first stage, the more time we need to process them on the second stage. So our further investigation aimed to find an approach that maximizes the first-stage model, Average Recall (AR), with the tricky parameter N that aims to minimize computations on the second stage.

The initial experiment of passing the raw query to the search API and extracting the top ten candidates showed an AR of about 0.628. Increasing the number of candidates to thirty and fifty resulted in the 2-3% boost of AR. So, we concluded that increasing the number of candidates is not a solution in our case.

After the initial experiment, we performed an error analysis of the initial model and found insights that helped us to increase the model's AR. FEVER is the Wikipedia-based dataset, and, logically, it usually contains claims about some named entities that have been mentioned in some article within Wikipedia. We found the pattern that claims usually contains the named entity and desired evidence (hypothesis) is inside the corresponding Wikipedia article during error analysis. However, provided API could not find the correct link to that entity in top N

suggestions. So, we decided to proceed with query modification in order to help the Wikipedia API.

5.1.2 Using NER models for performance tuning.

The initial modification we have done is performing named entities recognition with the given claim. We used several strategies of using obtained information. Firstly, we performed an extra API call with only all joined named entities in the query if found. For example, if we claim *"Ryan Gosling has been to a country in Africa"*, we do an extra query of *"Ryan Gosling Africa"* using Wikipedia API. One more strategy is making separate API calls for each entity found. We have a merged set of candidates obtained from all queries for a specific claim as a final result. Another improvement was to add related words to named entities for some experiments (query augmentation). We detected them using patterns found in the dataset when related words stand just after named entity in brackets, such as *"The Chaperone (film)"*. In summary, we used two major strategies:

1. Extra query with only all joined named entities. It was named NER_merged.

2. Extra queries for each entity found in the claim. We named it NER_separate.

We did not train any NER models but used pretrained ones. For our experiments we used models *"en_core_web_sm"* and *"en_core_web_trf"* from spaCy tool[1] framework and *"ner-fast"* model from Flair[2]. Given that we are designing a solution for real-life scenarios - our approach must provide answers as fast as possible. It means that models may not have SOTA performance, but they have an optimal ratio between AR and efficiency. When spaCy models they were reported to have 0.84 and 0.898 F-score (Honnibal et al., 2020) comparing to 0.892 F-score of Flair model on Ontonotes dataset (Akbik et al., 2019). We have done a set of experiments measuring the efficiency of each model and the AR performance of the whole Model one solution using a specific NER model. Also, we have done experiments for different modifications discussed before. We have got the result presented in Table 5.1.

One more important metric we wanted to evaluate is efficiency in time. Although we cannot influence Wikipedia API time of response, we are modifying a query. It takes time and computational resources, so we need to measure it. In this experiment, we measure only the time used to modify a query without an actual API call, which helps to compare approaches fairly.

The same computational resources are used for all experiments. We are using a CPU-only 2,4 GHz Quad-Core Intel Core i5 instance with 8Gb RAM. The critical metric for efficiency is time - seconds per 1000 queries. The result we got is presented in Table 5.1. Also, the number of candidates picked by Model level one can be different from parameter N, as we are doing a different number of additional queries that return a different number of results for different configurations. So we also measure the N_returned metric. That is the average number of candidates returned after Model level one. It influences the accuracy and efficiency of Model level two. Based on the three described metrics, we will select the best configuration for our specific case.

[1] spaCy https://spacy.io.
[2] Flair https://github.com/flairNLP/flair.

TABLE 5.1: Comparing Model one configurations performance

Configuration	**AR (higher is better)**	**sec per 1000 queries (lower is better)**	**N returned, (lower is better)**
No NER model N=10	0.628	0	9.11
No NER model N=30	0.645	0	25.02
No NER model N=50	0.649	0	39.16
SpaCy en_core_web_sm NER_merged N=10	0.810	5.01	15.33
SpaCy en_core_web_sm NER_merged N=30	0.833	5.01	44.02
SpaCy en_core_web_sm NER_merged N=50	0.840	5.01	70.67
SpaCy en_core_web_sm NER_separate N=10	0.834	5.01	10.12
SpaCy en_core_web_sm NER_separate + related N=10	0.839	5.01	10.11
SpaCy en_core_web_trf NER_merged N=10	0.827	82.5	15.09
SpaCy en_core_web_trf NER_separate N=3	0.874	82.5	6.93
SpaCy en_core_web_trf NER_separate N=5	0.892	82.5	11.68
SpaCy en_core_web_trf NER_separate N=10	0.911	82.5	23.47
SpaCy en_core_web_trf NER_separate + related N=10	0.913	82.5	23.44
Flair ner-fast NER_merged N=10	0.861	86.2	15.54
Flair ner-fast NER_separate N=3	0.879	86.2	**6.27**
Flair ner-fast NER_separate N=5	0.895	86.2	10.58
Flair ner-fast NER_separate N=10	0.914	86.2	21.30
Flair ner-fast NER_separate + related N=10	**0.915**	86.2	21.28

5.1.3 NER experiments results

As we can see in Table 5.1, usage of NER models dramatically increased the performance of candidate selection. An increasing number of N did not show a significant impact. Also, a big N for the first stage can be harmful to the second stage, where we should compare each sentence of picked articles with the given claim. We found out that the NER_separate strategy gives better results than NER_merged. Also, non-NN based model en_core_web_sm is much faster than other models, but it provides a lower accuracy. However, it can be used when efficiency is such crucial that we can sacrifice accuracy.

We should mention here that these results may be improved by fine-tuning corresponding models for our specific Wikipedia data but left this for future research.

As for our needs, we decided to use the "Flair ner-fast NER_separate N=3" configuration for further experiments and API development. It provides a relatively high accuracy of 0.879 AR with only 6.27 candidates returned. The main reason for such a decision is relatively high AR with the lowest number of candidates returned. According to our experiments with a complete fact-checking system presented in Section 5.5.4, the most time-consuming part is a sentence-embeddings calculation used in the NLI model. The number of returned articles from Model one directly

influences the time for embeddings calculation. So, we sacrifice 4% of AR to increase the efficiency of sentence embeddings by almost three times, compared with the configuration with the highest AR.

5.1.4 Limitations and alternatives

As we concluded in Section 5.1.3, query modification with extracted named entities improves the quality of the information retrieval stage. However, this part of the system still has room for improvement, which was not covered in our research.

In our experiments, we are using all possible types of entities extracted by the NER model. However, some of them can be useless and even bring unnecessary noise to the article retrieval. Filtering inappropriate entity types like numbers or money can increase the system's efficiency and reduce the number of returned candidates, which is beneficial for the speed of the second stage.

Another problem is that pretrained NER models cannot extract valuable keywords like diseases, animals, or food names. As a result, we observe $\sim$ 6% of samples where we could not extract any named entities. For that case, a part-of-speech (POS) tagger can be used for noun extraction, which is an alternative for the NER model.

In summary, more experiments are needed to confirm all mentioned hypotheses. We leave that for future research.

5.2 Model level two. Building sentence-based NLI model.

5.2.1 Experiment setup

We considered using three datasets for our further experiments: MNLI, SNLI, and FEVER. SNLI and MNLI are benchmark datasets used for initial training and comparing our approach with SOTA results. FEVER dataset is used as a domain-specific one and used for final fine-tuning (training) and validation. Each dataset used has its predefined train and test parts that were also used for our experiment evaluation. All training procedures were done using the RTX2070 GPU instance.

5.2.2 Masked language models. Efficiency testing

It is costly to train a masked language model from scratch. Only big companies and wealthy organizations can afford it. However, usually, they open the trained models for the community that allows reusing them. A great example of a platform for sharing pretrained models is the Hugging Face platform[3], which is an open-source solution, which we used to get pretrained models. As for our further experiments, we used three different language models.

The first one is *bert-base-uncased*. This model is uncased, so it does not differentiate between "NLP" and "nlp" words. The model was pretrained on BookCorpus dataset (Zhu et al., 2015) and English Wikipedia (excluding lists, tables, and headers) (Devlin et al., 2018). That is a basic lightweight masked language model.

The second one is the *bart-base* model presented by Facebook. Bart uses a bidirectional encoder (like BERT) and a left-to-right decoder (like GPT). BART is particularly effective when fine-tuned for text generation and works well for comprehension tasks (Lewis et al., 2019). This model was reported to work well for summarizing tasks when training on semantic similarity (Yoon et al., 2020). We hypothesize

[3]Hugging Face. Models `https://huggingface.co/models`.

that this is related to models' ability to detect text meaning, which can also be used on NLI problems.

Finally, we also use another model in our experiments: ALBERT, one of the top-performing models based on the GLUE score. It is also reported to be more memory efficient, so we experimented with it.

Previously, we mentioned that we are going to use "base" models. Although we sacrifice the quality of embeddings, we want our solution to be as fast as possible, and that decision provides about three times a boost in time according to our investigation. We tested the efficiency of each model by measuring the average time for one sentence encoding. We used all unique claims from the SNLI test dataset that is 3,323 sentences, for testing data. Also, we measured time in two modes. The first is processing using a batch of 32 items, and the second is one-by-one processing. We used CPU instance in order to reproduce the inference environment. The results of this experiment are presented in Figure 5.1. We find out that "base" models are much faster than "large" ones. Also, as was expected, batch processing is more efficient and should be used if possible. As for further experiments, we will concentrate on classical BERT and BART models as they have different natures that can be beneficial.

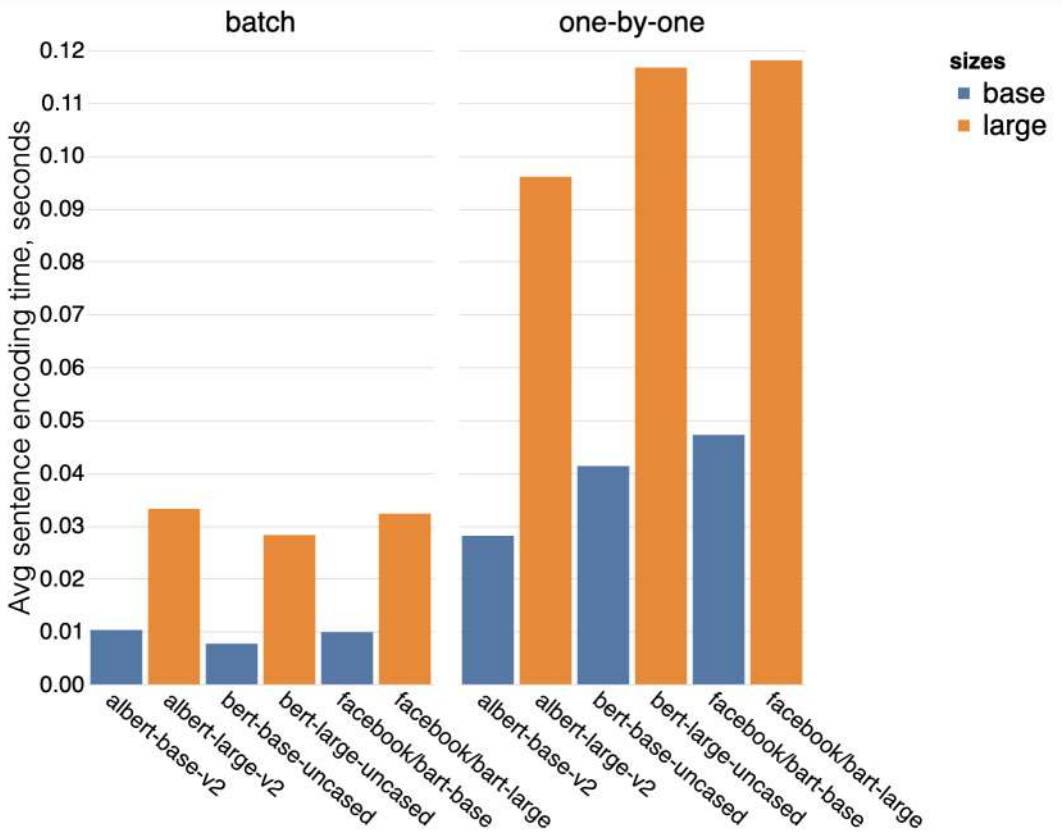

FIGURE 5.1: BERT-like models encoding efficiency.

5.2.3 Sentence embeddings creation

There are several methods for sentence embeddings creation out of masked language models. The main approaches are described as pooling methods by Reimers and Gurevych, 2019. The first method is using embedding got from *[CLS]* token

from the BERT model. That is a special tag initially designed to represent the entire sentence, used for sentence classification tasks. Another method is taking an average of all word-level tokens, which is a relatively straightforward approach previously used, for example, in Word2Vec models sentence embeddings creation. Another approach is computing a max-over-time of the output vectors, which performed well in the case of the BiLSTM encoder of InferSent (Conneau et al., 2017b). It is essential to mention that to get relevant results on the NLI downstream task, it is necessary to fine-tune the whole model, including MLM for each approach (Reimers and Gurevych, 2019). As a basic approach for our models, we will use the average of word embeddings as it is reported as best performing by (Reimers and Gurevych, 2019), comparable to other architectures and the most intuitive.

5.2.4 Initial NLI models building. Comparing with SOTA

The initial experiment was to reproduce the results of the SOTA models and also measure their efficiency. In order to have a general overview, we decided to pick both word and sentence-based models. For the first iteration we picked SemBERT (Zhang et al., 2020) and HBMP (Talman, Yli-Jyrä, and Tiedemann, 2019) models. Those models represent single model architecture without an ensemble, have top-performing results, and have official repositories with code[4][5].

The experiment of reproducing results is computationally expensive, so we used Microsoft Azure NC6 Promo virtual machine with six cores, 56 GiB RAM, and one K80 GPU. For consistency, we used the same configuration for all further experiments. Here, we measure the accuracy and computation speed on inference using the SNLI test set. By computation speed on inference, we mean time needed to classify relations between pairs of sentences. We measure the speed of the model on both GPU and CPU instances. Using CPU instance simulates the actual scenario for our API, which is running on a small instance on the Wikimedia Cloud[6].

The code for SOTA results from official repositories is implemented to work on GPU only, so we could not fairly compare performance on that code on CPU. We compared the performance of all models on GPU instance, which allows us to compare them.

We evaluated models on 9,824 samples from the SNLI testset. The results are presented in the comparison Table 5.2. We also included the results for our custom BERT-based models in the comparison table. By custom architecture, we mean the siamese model presented in subsection 4.3 with different MLM encoders.

As we can see, the word-based model SemBert has the best accuracy results. However, it is 25x times slower on inference than sentence-based model HBMP. It does not allow caching, saving intermediate results. Although it is very accurate, it is difficult to use in real-world scenarios. Sentence-based models have worse accuracy than word-based, but they are significantly faster. In this experiment, we also tested the Universal sentence encoder, which can be substituted by the BERT encoder. Although it is the most efficient model among the testing set, USE has the lowest accuracy in the NLI task, and it was decided not to use it for further experiments.

[4]Github repository for SemBERT model https://github.com/cooelf/SemBERT.
[5]Github repository for HBMP model https://github.com/Helsinki-NLP/HBMP.
[6]https://wikitech.wikimedia.org/wiki/Portal:Cloud_VPS

TABLE 5.2: Experiments results on SNLI datasets

Model	**Accuracy on SNLI dataset**	**Efficiency on inference CPU**	**Efficiency on inference GPU**
SemBERT	**91.9%**	-	0.51 $\frac{s}{sample}$
HBMP	86.6%	-	0.02 $\frac{s}{sample}$
Siamese + *bert-base-uncased*	85.2%	0.1 $\frac{s}{sample}$	0.006 $\frac{s}{sample}$
Siamese + *bart-base*	86.9%	0.12 $\frac{s}{sample}$	0.006 $\frac{s}{sample}$
Siamese + *albert-base*	84.98%	0.08 $\frac{s}{sample}$	0.006 $\frac{s}{sample}$
Universal sentence encoder	78.7%	0.036 $\frac{s}{sample}$	**0.004** $\frac{s}{sample}$

5.2.5 Masked language models unsupervised fine-tuning

Supervised fine-tuning has a significant improvement in accuracy. However, usually, data collection for such training is expensive as manual annotators are required. We decided to use an unsupervised fine-tuning of language model that can potentially help us improve the domain-specific model and not require annotated data. As part of this experiment, we fine-tuned *bert-base-uncased* and *bart-base* on the WikiText dataset (Merity et al., 2016). That is a collection of over 100 million tokens extracted from the set of verified Good and Featured articles on Wikipedia that aims to provide domain adaptation for our models. It is important to mention that the following dataset is much bigger than the most popular usual NLI datasets, such as SNLI or MNLI. The unsupervised fine-tuning was done on the RTX2070 GPU instance. To fine-tune, we followed the experiment setup by (Devlin et al., 2018) but trained only for masked LM problem formulation and only for one epoch. Reproducing the original setup, we selected 15% of tokens at random. Then 80% of tokens selected were changed to [MASK] special tag, 10% were switched to another token, and 10% remained original. Example of input and output such strategy is presented in Figure 5.2. As a result of such an experiment, we got two fine-tuned models that we will use in further experiments. We expect these models will perform better on the Wikipedia-specific FEVER dataset.

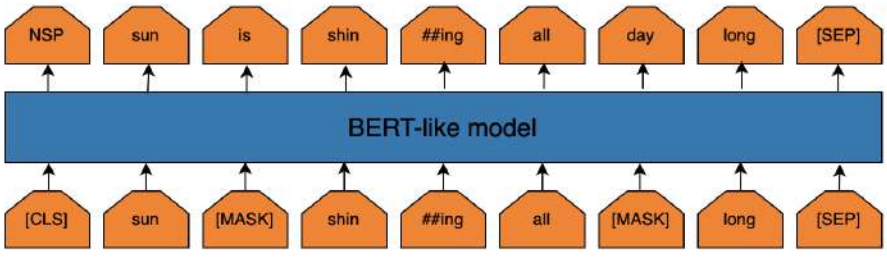

FIGURE 5.2: Example of input and output for masked sentence.

5.2.6 Performance on other datasets and transfer learning approach

This section will review the performance of trained models for different datasets, including domain-specific dataset FEVER. That experiments are done to test models'

TABLE 5.3: Training on SNLI dataset and testing on SNLI and MNLI.

Model	**Accuracy on SNLI dataset**	**Accuracy on MNLI dataset**
Siamese + *bert-base-uncased*	85.2%	59.16%
Siamese + *bart-base*	86.9%	63.19%
Siamese + *albert-base*	84.98%	58.58%

TABLE 5.4: Training on MNLI dataset and testing on SNLI and MNLI.

Model	**Accuracy on SNLI dataset**	**Accuracy on MNLI dataset**
Siamese + *bert-base-uncased*	65.33%	76.1%
Siamese + *bart-base*	66.93%	77.85%
Siamese + *albert-base*	66.33%	80.65%

TABLE 5.5: Full training on specific dataset vs. training on SNLI and classifier fine tuning on FEVER and MNLI

Model	**MNLI classifier fine tuned vs. full train**	**FEVER classifier fine tuned vs. full train**
bert-base-uncased	64.8% / 76.1%	70.1% / 79.81%
bart-base	67.6% / 77.85%	74.4% / 85.24%
bert-base-uncased + fine tuned	65.4% / 76.29%	69.7% / 82.45%
bart-base + fine tuned	68.1% / 77.35%	73.0% / 85.62%

ability to generalize knowledge and work on different texts. The initial experiment was straightforward; we were training models on one SNLI or MNLI and validated on test sets of both datasets.

In such problem formulation, we could not directly use FEVER as it has only two classes with provided evidence sentences if transformed to SNLI style (Table 3.4), because the *NOT ENOUGH INFO* class is not supported by any sentence. Therefore, for the current experiments, we decided to drop all examples of *NOT ENOUGH INFO* class and use only the other two classes for that experiment. In Section 5.3.2 we propose and implement a solution to overcome this problem.

The results we got are presented in Table 5.3 and Table 5.4. As we can see, models trained on one dataset have poor performance on another data, which is a general trend for all models. However, we want to single out BART models, which have better generalization power as we see a significantly lower drop in accuracy than other models.

One more experiment was to train the whole model on one dataset and then fine-tune only the last dense layer (Figure 4.3), which is responsible for classification specifically for the target dataset. Using such a strategy is computationally efficient as the heavy MLM remains frozen but pre-trained on NLI tasks. Also, in such problem formulation, we can compare SNLI, MNLI, and FEVER, as we can set a different number of classes in the final layer without losing problem generalization. We expected to get good results for such a training strategy. In our experiment, we trained models on SNLI, then fine-tuned the last layer on MNLI or FEVER train set and tested on the corresponding test set. We also trained individual models for each

dataset with all layers unfrozen and compared performance with the transfer learning approach. The results of this investigation are presented in Table 5.5.

We can see that the transfer learning approach of fine-tuning classifier slightly improves models' performance. However, they are still much lower than the results of fully trained models on specific datasets. It reveals that language models take a significant role in classification models and should be fine-tuned for domain-specific texts.

An important finding from results presented in Table 5.5 is that unsupervised fine-tuning of language model can be even more advantageous than a supervised pre-fine tune on another dataset. We see that in the case of full training of model on FEVER dataset, the performance of unsupervised fine-tuned models is higher comparing to raw ones, especially for a *bert-base-uncased* model.

5.3 Building Wikipedia domain-specific NLI model

As we showed in previous experiments, NLI models fully trained on a specific dataset perform much better than fine-tuned models. So we decided to train the FEVER-specific NLI model that will be the primary building block of a fact-checking system for Wikipedia. In this section, we review our training approach and validation results.

5.3.1 Experiment setup

All the experiments conducted in this section are done on the RTX2070 GPU instance. We are using original FEVER data for all experiments adapting it for our particular needs. Also, we are using a predefined train/dev split for all experiments. As for training parameters, we are using a batch size of ten and default parameters provided in Sentence-Transformers framework[7].

5.3.2 Data preparation

For previous experiments, we used a model trained only for predicting two classes *REFUTES (R)* and *SUPPORTS (S)*, as there is no hypothesis presented for all samples of *NOT ENOUGH INFO (NEI)* class. That is an important difference between the FEVER dataset comparing to SNLI and MNLI (see section 5.2.6). Therefore, we need to generate samples for *NEI* class. We use a negative sampling strategy inspired by the approach followed by *Hanselowski et al., 2018*.

For all training samples of R or S classes, we generate samples of the NEI class. We take the original claim as used in the R or S class example and change a hypothesis for such samples. As a new hypothesis for the NEI class sample, we use a randomly chosen sentence from the same article, which corresponds to the original hypothesis. We limit the pool of sentences with those that were not previously used in another sample. This approach is a heuristic, aiming to pick such pairs of claims and hypotheses, which are related by topic but not support or refute each other.

As for validation, we are using a predefined testing set. We are using another strategy for filling *NEI* class samples for the testing set. In this case, we are taking the original claim for *NEI* samples, using model level one (see Figure 4.1) to pick article candidates for such sample and then randomly select one sentence from such articles. These samples might be more accessible for models to predict than those

[7] Sentence-Transformers framework `https://www.sbert.net`.

built for the training set. However, that approach seems to be better for validation as we are not biasing to specific claims and have the original distribution of classes.

In addition, we estimate the quality of filling NEI class samples strategy. For that experiment, we randomly pick 200 generated samples and manually annotate them. As a result, we found only one example where the hypothesis refutes the given claim. It means that the proposed filling strategy has an estimated error rate of 0.5%, which is acceptable. However, as all *NEI* samples are artificially constructed, we consider validating the model with and without such samples just using *S* and *R* labeled samples.

Another important thing to highlight is that the original sentences from the Wikipedia dump include some tags at the end of the sentence, separated by tabulation symbol. We consider building models with and without them. Such filtering decreased the average number of symbols in the hypothesis from 212 to 136 characters. We will experiment with how such filtering influences the accuracy results of our models.

5.3.3 Model training and validation

As mentioned in the previous section, we use two types of oversampled train datasets: reduced and original hypothesis sentences. So we consider a training set of models for each case. As for validation scores, we also compare results on the reduced and original test sets and include performance on only *S* and *R* classes as they are not synthetic. Results of our experiments are presented in Table 5.6 and Table 5.7.

TABLE 5.6: Training on original FEVER dataset

Model	**FEVER original**	**FEVER clean**	**FEVER original only R and S classes**	**FEVER clean only R and S classes**
Siamese + *albert-base*	71.92%	70.93%	66.22%	63.38%
Siamese + *bert-base-uncased*	72.16%	71.14%	66.93%	63.83%
Siamese + *bart-base*	74.25%	73.28%	68.25%	65.70%
Siamese + *bert-base-uncased* + unsupervised fine tuned	71.91%	70.95%	65.99%	63.14%
Siamese + *bart-base* + unsupervised fine tuned	**74.60%**	73.60%	68.57%	66.24%

We can conclude that cleaning FEVER tags leads to a drop in accuracy for all models from obtained results. However, these tags are specific for the FEVER dump and are not usable in real scenarios when we would have plain text. Also, we concluded that in case we train on cleaned texts, we are getting 1% boost on clean texts validation comparing to model trained using original texts. If we want to use the model in a real scenario, we should consider training the final model using samples with the cleaned hypotheses.

As we see, unsupervised fine-tuning of models using domain-specific data made a boost for *bart-base* model. It is the best-performing model for all cases. Also, we

TABLE 5.7: Training on FEVER dataset with cleaned hypothesis

Model	**FEVER original**	**FEVER clean**	**FEVER original only R and S classes**	**FEVER clean only R and S classes**
Siamese + *albert-base*	72.40%	71.85%	67.11%	65.50%
Siamese + *bert-base-uncased*	71.97%	71.67%	67.17%	66.13%
Siamese + *bart-base*	74.20%	74.72%	68.11%	68.76%
Siamese + *bert-base-uncased* + unsupervised fine tuned	72.02%	71.76%	67.01%	66.19%
Siamese + *bart-base* + unsupervised fine tuned	74.18%	**74.82%**	68.34%	69.33%

should mention that the best performing model is the fine-tuned *bart-base* model trained on clean text, which achieved 74.82% accuracy.

We noticed a tendency that accuracy only considering the *R* and *S* classes is usually much lower than for all classes. In order to understand the reasons behind that, we looked into the confusion matrix (Figure 5.3). We found out that the model has difficulties predicting the *R* class as there are many false negatives for that sample. Approximately more than 13% of accuracy we are losing just on *REFUTES* (*R*) class. It is important to mention that models trained on the cleaned dataset generally perform much better when we consider only the *R* and *S* classes, which is another benefit of such pre-processing technique. We assume that the filtered tags do not include a valuable signal of the relation between claim and hypothesis but add additional noise, reducing performance. Data preparation and model stability should be reviewed in more detail. We go deeper on this problem in the following section.

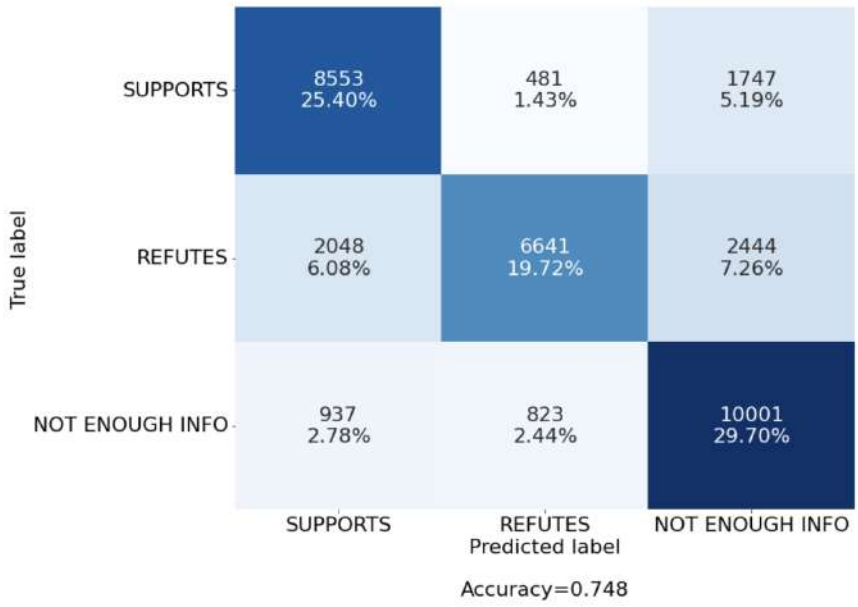

FIGURE 5.3: Confusion matrix for *bart-base* model (cleaned hypothesis training and validation).

5.4 Analyzing models stability. Training on filtered data

It is important to understand how models perform in different conditions. Stability of model performance (*a.k.a* generalization) is a crucial characteristic that determine usability in a production environment. This section will experiment with filtering the training dataset and how it is related to model performance. Also, we will explore how the length of texts relates to accuracy.

5.4.1 Training on filtered data. Experiments with FEVER

As it was shown in Figure 3.2, some hypotheses are duplicated multiple times and correspond only to one class, which can lead to model overfitting. As it was discussed by (Gururangan et al., 2018), such annotation artifacts have a significant impact on model accuracy. FEVER dataset has the same issue. So we considered filtering and balancing dataset and experiment with how it influences model performance. As a base dataset for filtering, we took the cleaned training dataset used in section 5.3.3, also the same training procedure was done in order to get comparable results.

We used three main steps during data filtering. Firstly, we filtered out absolute duplicates by fields 'claim' and 'hypothesis'. That reduced number of samples by 8.8% concerning the original size. Interestingly, some samples were absolute duplicates by 'claim' and 'hypothesis' but contradicted labels. For example, pair of claim *"Meryl Streep is an award losing actress."* with corresponding hypothesis *"Streep has also received 30 Golden Globe nominations, winning eight more nominations, and more competitive wins than any other actor."* has both *SUPPORTS* and *REFUTES* labeled samples.

After that, we proceed to filter samples with the duplicated hypothesis. For that, we selected the set of all samples with the same hypothesis sentence. Then we found the difference N_{diff} in number of samples of SUPPORTS and *REFUTES* classes. Next, we picked the random number N_{drop} from 0 to N_{diff}, which corresponds to the number of samples to drop. Then we randomly picked N_{drop} samples to drop from major class in order to equalize the distribution of contradicting classes among one hypothesis. Such operation was done only for those hypotheses that correspond to at least ten samples. After applying such filtering methodology, the dataset reduced by an additional 6.9% compared to the original FEVER size.

Finally, we randomly undersampled the number of *NOT ENOUGH INFO* cases, equalizing it to the size of the *SUPPORTS* class which is the second largest one. This

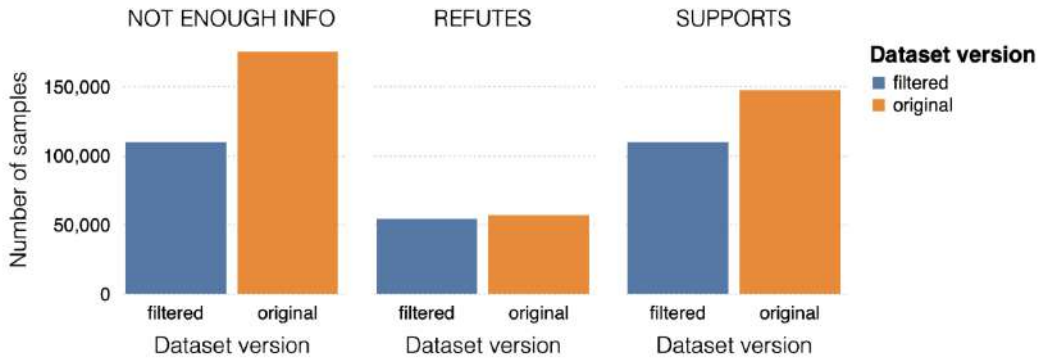

FIGURE 5.4: Classes distributions before and after filtering.

action additionally decreased the filtered training dataset by 12.2% with respect to the original size. However, it is important to add here that all NEI samples are artificially created by the heuristic described in section 5.3.2, so they are less informative for model comparing manually created samples coming in the original FEVER dataset. The distribution of labels of a filtered dataset compared to the initial one is shown in Figure 5.4. As a result, we got the filtered dataset with the total number of samples reduced by 27.9% concerning the original size.

After that we proceed with training model on filtered data. We compared 3 main models results in Table 5.8

TABLE 5.8: Training model on filtered FEVER dataset

Model	**FEVER clean**	**FEVER clean filtered**	**FEVER clean only R and S classes**	**FEVER clean filtered only R and S classes**
Siamese + *albert-base*	72.73%	72.40%	71.04%	68.46%
Siamese + *bert-base-uncased*	73.39%	73.04%	71.85%	70.49%
Siamese + *bart-base*	74.84%	75.53%	70.68%	71.47%
Siamese + *bert-base-uncased* + unsupervised fine tuned	73.48%	73.38%	71.11%	70.44%
Siamese + *bart-base* + unsupervised fine tuned	75.42%	**75.91%**	71.32%	71.91%

5.4.2 Model performance depending on the length of text. FEVER dataset

During the FEVER data exploration (Chapter 3, Figure 3.3), we found out an important variation on hypothesis length, measured as the number of characters. To understand the impact of hypothesis length on the NLI task performance, we took results of *bart-base* model on the FEVER test set and plotted accuracy for different hypothesis lengths. Here, we consider only those length values corresponding to at least 1% of samples from the test set. Also, for each value, we plotted a 95% confidence interval. The results of that experiment are presented in Figure 5.5.

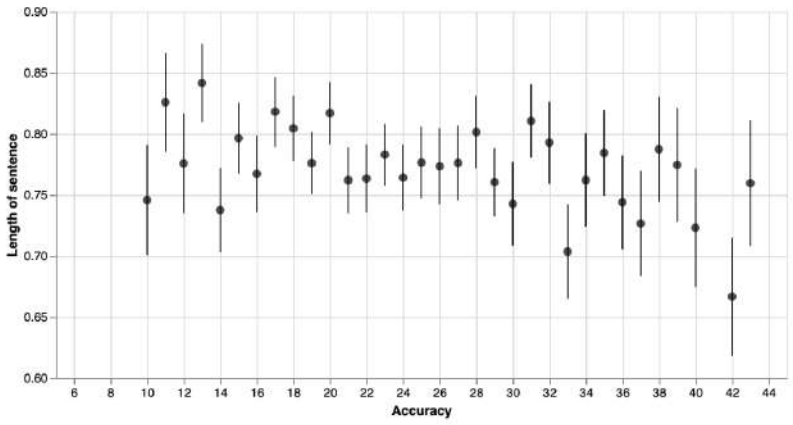

FIGURE 5.5: Hypothesis length (FEVER) vs model's accuracy.

As we can see from Figure 5.5, there is no obvious dependency between the length of the hypothesis and accuracy, but still, there is a tendency for a more extended hypothesis to have a worse accuracy. However, more experiments are needed to confirm this relationship. We leave that for future research.

5.5 WikiCheck: A complete Fact-Checking system based on Wikipedia

In previous sections, we were analyzing each block separately. However, it is important to understand the performance of our end-to-end system. By end-to-end, we mean a system that can receive a sentence (claim), query Wikipedia looking for evidence to contrast with (hypothesis), and then apply the NLI model on that pair (*claim, hypothesis*), returning whether the claim is *SUPPORT*, *REFUTED* or if there is *NOT ENOUGH INFO* on Wikipedia about it.

In this section, we provide details of *WikiCheck*, our proposed system architecture. Also, we provide key characteristics of time efficiency and accuracy, compare our model performance on the FEVER dataset with other SOTA solutions.

5.5.1 Adapting model for production

After performing numerous experiments with models, we designed heuristics that can be used in real scenarios to make it work faster and more accurately. In this subsection, we describe the data processing pipeline using the flow presented in Section 4.1 and present the implementation solutions we implemented (or discarded).

There are several steps that the model passes before the final prediction. The general flow is presented in Figure 5.6. Each step should be as fast and accurate as possible to deliver good results for the user. This trade-off between time and accuracy is one of the main challenges we are dealing with, and here describe and justify our choices.

The model starts with passing claims to the *NER* model, which detects named entities and forms a set of queries to pass to the Wikipedia search API. At the step of article search, we have had the I/O bound as the instance is not loaded, but we are waiting for the search service answer. We consider using paralleling or asynchronous processing on this step, which can improve general application timing. Also, we are using machine that has a fast connection with the search API to reduce response time. The same problem applies to the next step when we retrieve the Wikipedia articles. Those two steps are highly dependant on network conditions, and we did not optimize them in this research. However, we made all the experiments on the same machine to make results comparable.

After Wikipedia articles texts are collected, we continue with the NLI model. From the very beginning, we decided to use a sentence-based model as it has obvious benefits. First of all, using such an approach, we can calculate embeddings for claim and hypothesis separately. Consequently, we can calculate embedding for the claim only once as it is the same for all pairs. Also, considering results from Figure 5.1, we are using batch processing for hypothesis calculation which decreases the amount of time needed for this step by three times comparing to one-by-one processing.

Also, it is important to mention that it is possible to pre-compute and cache vectors for all Wikipedia articles and extract them instead of texts. It moves sentence

embeddings calculation to offline, so we do not spend time for it on inference. The main drawback is that such a solution requires many storage resources.

Having embeddings for the claim and hypotheses, we apply a classification model, a simple MLP model, and get results for each claim-hypothesis pair. We consider all pairs as output as it gives more interpretability and generalization.

FIGURE 5.6: Fact Checking system flow.

5.5.2 Difference from SOTA solutions

Most of the SOTA solutions were created during the FEVER competition, where the main criterion for model comparison is an accuracy. Our research orientation is shifted towards usability. It means that the model should be not only accurate but also fast and interpretable.

All top solutions Nie, Chen, and Bansal, 2018; Yoneda et al., 2018 and Hanselowski et al., 2018 are following the baseline presented by Thorne et al., 2018b and build three-staged models. These stages are article selection, sentence selection, NLI classification. We present a two-staged solution with ML-based aggregation on top. We consider using only document retrieval and NLI models for fact verification. We do not apply sentence selection logic in order to avoid missing important information.

Our documents retrieval process is very similar to the one presented by Hanselowski et al., 2018. However, we are using pretrained NER models for query extending. According to our experiments (see Section 5.1.2), such an approach showed a significant boost (over 30%) on recall comparing to basic querying.

Also, as for the NLI model, we are using a sentence-based approach. It significantly improves the speed of the NLI model on inference, sacrificing a little the accuracy of the results.

Moreover, our simple NLI model achieves almost SOTA result for sentence-based models, being more efficient (Section 5.2).

5.5.3 Experiment setup

There are two main characteristics of fact-checking applications that we want to measure: accuracy and efficiency. In this subsection, we are reviewing the experiment setup for measuring each of them.

Accuracy

As for general fact-checking system accuracy validation, we used the original FEVER 1.0 dataset. All experiments were done using the RTX2070 GPU instance. In order to measure application accuracy, we decided to use the official FEVER validation tool [8], which allows us to compare our solution with FEVER competitors.

It gives several metrics used for validation:

[8] Github. Fever-scorer `https://github.com/sheffieldnlp/fever-scorer`.

1. FEVER score: That is a more strict accuracy score. In order to consider the sample correctly classified, it requires the correct label along with the full match of true evidence with predicted. It was the main metric for the FEVER competition, but we will not use it as the primary one. The main reason is that this score is highly biased to NEI class, as in case the correct match we do not need the match of evidence.

2. Accuracy: Standard accuracy score that requires only label match to consider sample correctly classified.

3. Evidence $F_1@k$ score: The score that evaluates the correctness of picked evidence and does not take into account NEI class samples. It calculates as following:

$$F_1@k = 2 \cdot \frac{(Precision@k) \cdot (Recall@k)}{(Precision@k) + (Recall@k)}$$

$$\text{Precision@}k = \frac{true\ positives\ @k}{(true\ positives\ @k) + (false\ positives\ @k)}$$

$$\text{Recall@}k = \frac{true\ positives\ @k}{(true\ positives\ @k) + (false\ negatives\ @k)}$$

The pipeline for FEVER validation differs from the presented fact-checking system flow and has some crucial limitations. The changes are shown in Figure 5.7 and marked with red. As for the validation process, we are not querying Wikipedia texts from MediaWiki API. We are taking the actual texts from the Wikipedia dump dated 2017, provided along with the FEVER dataset. We can find the text by the article name found in the previous step. It allows matching the sentences' ids for validation correctly. However, this approach also has an important limitation. The Search API is looking for articles in the up-to-date version of the Wikipedia database since there are changes from the 2017 year dump. The same document retrieval approach was used by Hanselowski et al., 2018, and the same problem was reported. However, in our case, the time lag is bigger, and as a result, we have 11.51% of articles found by MediaWiki API that do not have a matched text in the dump provided.

As for the aggregation block, we are using two stacked CatBoost models[9], trained on a sample of probability outputs from the NLI model. We are using stacked probability outputs from the NLI model and similarity scores of the top 10 closest hypotheses for label classification. We are using cosine similarity between sentence embeddings from the NLI model to define the relationship between hypothesis and claim. Finally, for each sample, we have a vector of 40 features used to classify the claim. We use padding by zeros in case of a lack of sentences after the article selection phase.

We used CatBoost learning-to-rank model for evidence picking, where each hypothesis was represented by its class probabilities from the NLI model and cosine similarity to the claim. The training loss used is *YetiRankPairwise* presented by Gulin, Kuralenok, and Pavlov, 2011. We do not include aggregation logic for efficiency testing explained in the next section, as it was not included in our final API.

Efficiency

We define efficiency as the speed of the entire system, which means that the faster is the most efficient. For the experiment, we measured the total time and time taken

[9] CatBoost framework `https://catboost.ai`.

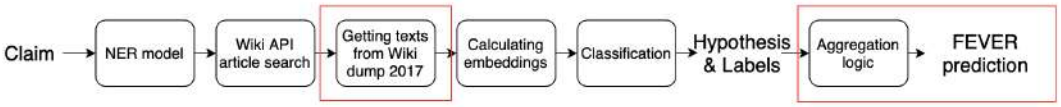

FIGURE 5.7: Fact Checking system flow for FEVER validation.

for each step from the Figure 5.6. We used a random set of one thousand unique claims from the FEVER testing part as a test set. We tested the system using three configurations with different sentence encoders models. For each model, we used a different random set of claims to avoid API caching influence. The system was running on CPU-only 2,0 GHz Intel processor instance with 8Gb RAM provided hosted on the Wikimedia VPS Cloud[10].

5.5.4 Experiment results

This section presents the results of measuring the efficiency and accuracy of the general fact-checking model.

FEVER Accuracy

We compared our fact verification system (*WikiCheck*) with the best performing solutions of FEVER competition. In comparison, we used our final model with a BART-based NLI classifier.

The final results are presented in Table 5.9. As a result, we got a 0.43 Fever score and 0.57 of general accuracy for our *WikiCheck* model.

Also, we analyzed the errors of our models and found out that most of our mistakes are made for *NEI* class. The confusion matrix for both of our models can be found in Figure 5.8. That also explains the relatively low FEVER score. Correctly classified *NEI* class sample does not require evidence match, which is a limitation of the FEVER score.

TABLE 5.9: Complete fact checking system FEVER accuracy

Team/Name	FEVER rank	Evidence F1	FEVER score	Accuracy
UNC-NLP	1	0.5322	0.6398	0.6798
UCL MRG	2	0.3521	0.6234	0.6744
Athene	3	0.3733	0.6132	0.6522
The Ohio St. Uni	7	0.5854	0.4322	0.4989
GESIS Cologne	8	0.1981	0.4058	0.5395
WikiCheck	-	0.3587	0.4307	0.5753

Efficiency

The results for the system efficiency experiment can be found in Figure 5.9 and also can be learned with more details using Table 5.10.

We split the whole application into the logical part and tested each separately. *NER_model* part correspond to using the NER model for named entities extraction,

[10]This is a free service offered by the Wikimedia Foundation to host Wikipedia-related tools: https://wikitech.wikimedia.org/wiki/Portal:Cloud_VPS

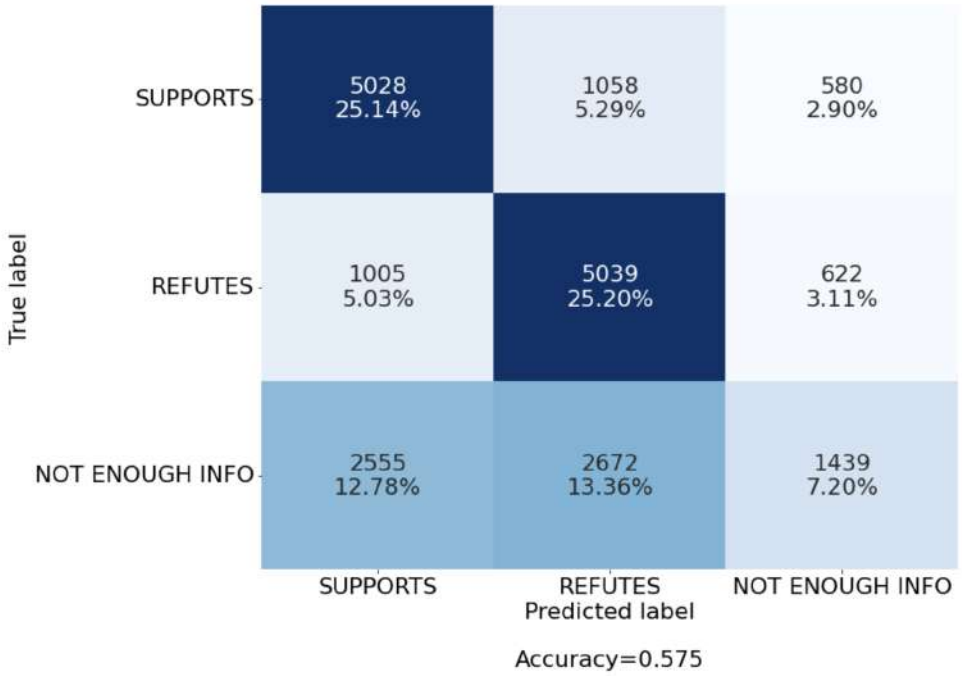

FIGURE 5.8: Confusion matrix for *WikiCheck* model

used for article selection improvement, as reviewed in Section 5.1.2. Wikimedia API usage is represented by two parts: *wiki_search*, responsible for article search, and *wiki_text* corresponding to loading the texts for selected articles. Then we have two stages that represent embeddings calculation. We calculate embeddings for claim and hypothesis separately. The last step is *classification* that in charge of using NLI classifier given the sentence embeddings. Also, we include the *total_time*, which shows the general system efficiency.

As we can see, the most time-consuming parts are text extraction (*wiki_text*) using Wikimedia API and hypothesis embeddings calculation. All the other parts have no significant impact on the model's speed comparing with the two parts mentioned before (Figure 5.9). Loading the text takes about 40% of total application time, and calculation embeddings for the hypothesis take 50%. It was expected that all system parts except embedding calculation have approximately similar timing for different configurations. The approximate time needed for fact-checking process about six seconds. As for the final model used for the system, we will use *bart-base* model, even though it is the slowest one. That is because the difference between models is not significant in terms of general system time efficiency (see Figure 5.9). Future work to improve the system efficiency, should focus on improving text extraction and embedding calculation that uses around 90% of application time.

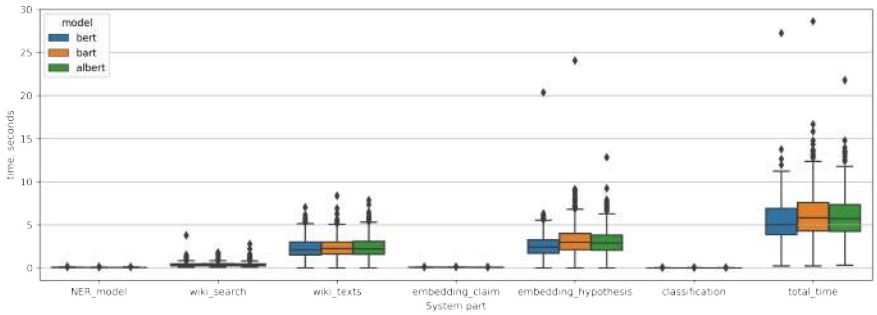

FIGURE 5.9: Fact Checking system efficiency

TABLE 5.10: Fact checking system efficiency, seconds

system parts	model	albert	bart	bert
NER_model	mean	0.059532	0.061278	0.062362
	std	0.016586	0.017219	0.019037
wiki_search	mean	0.388099	0.399130	0.390143
	std	0.204878	0.201289	0.212448
wiki_texts	mean	2.401416	2.372876	2.296560
	std	1.137427	1.063903	1.097486
embedding_claim	mean	0.068034	0.082378	0.074696
	std	0.011248	0.015822	0.014462
embedding_hypothesis	mean	3.048855	3.181426	2.568341
	std	1.355213	1.589718	1.281055
classification	mean	0.009174	0.009596	0.009061
	std	0.005151	0.005550	0.005580
total_time	mean	5.975197	6.106762	5.407904
	std	2.355858	2.473450	2.245171

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The main goal of the thesis is to transform academic research on Natural Language Inference and Automated Fact-Checking into a usable tool that can be used for fact verification using the Wikipedia knowledge base.

We analyzed related research and defined open problems that should be solved in order to achieve our goal. Previous works have not paid attention to the efficiency of their solution but concentrate on accuracy instead when the speed of the models is a crucial characteristic of practical application. Moreover, most SOTA solutions are implemented for GPU usage with batch processing and require code refactoring to use CPU instances on inference. Also, most organizations have limited resources that make it impossible to use SOTA solutions for their needs. The last but not the minor problem is the lack of NLI datasets to train models. The presented datasets have their specific limitations when creating a new dataset is expensive as it requires manual annotation.

During our research, we performed advanced data analysis, tried different modeling methods, and discovered the following insights:

1. FEVER dataset has its limitations and annotation artifacts that can influence the NLI model's performance. We proposed the heuristic filtering technique that led to the model's accuracy increase.

2. We showed that usage of NER models for search increases the quality of results, in case there are named entities in queries, as in the FEVER dataset.

3. We discovered that NLI models lack generalization. Full model training is required to get the best results for a specific dataset. The only classification layer fine-tuning transfer learning does not work well.

4. We proposed using unsupervised fine-tuning of Masked Language Models with domain-specific texts that further increased accuracy on the downstream NLI task.

5. We showed that batch processing is faster than one-by-one. Also, we provided reasoning why sentence-based NLI models are faster than word-based for real-world applications. The proposed possible architecture of sentence-based NLI model that shows comparable to SOTA results, being more efficient.

Finally, we present a new fact-checking system *WikiCheck*[1], that can receive a sentence (claim), query Wikipedia looking for evidence, and then apply the NLI model on that pair (*claim, hypothesis*), returning the relation of each corresponding

[1]Fact checking API WikiCheck https://nli.wmcloud.org.

pair, which is one of *SUPPORT*, *REFUTED* or *NOT ENOUGH INFO*. The presented system has comparable to SOTA results, being in the top-10 solutions of FEVER competition. It can be used on CPU, low memory devices, which makes it more applied. We make all the code for WikiCheck API available on the github[2].

However, the system has its limitations. It needs to be faster, as six seconds on average for one claim is far from desired real-time processing. We should research more on the aggregation stage. We have problems with *NEI* labeled samples, which is a limitation for real-life scenarios. More research should be done for this part of the application. Moreover, WikiCheck API depends on the MediaWiki Search API that we do not have control over.

6.2 Future Work

Many experiments can potentially improve the NLI model. As for future work, we consider experimenting with different methods for sentence-embeddings creation explained in Section 5.2.3. We plan to experiment with full training the NLI models on more than one dataset, where the last one would be the domain-specific dataset. That experiment is similar to the transfer learning experiment but is more computationally expensive. That is why it was not done in this work. Also, we want to experiment with different classifier models applied as the last layer of the NLI model. Our research uses the simple MLP layer, but using more advanced architectures can improve accuracy results. Also, the relation between the length of the hypothesis and the NLI model needs additional research.

The aggregation phase also needs additional research. The possible steps are additional features of claim-hypothesis relation generation, as using only the NLI model's probabilities is not enough to achieve SOTA results.

We consider optimizing the text extraction process for future efficiency improvements, especially recalculating and caching the sentence embeddings for texts. By caching the sentence embeddings, we understand storing them along with the text or in the mirrored database. When text changes, it also goes to the process of recalculating the corresponding embedding. Such a hypothetical approach can be used in high-load systems when the model's efficiency is crucial and does not allow the calculation of text embeddings on inference. Even though it is inefficient in storage, caching will make both the most time-consuming parts useless for the model and make the fact verification process almost instantly. Also, improvements in the network conditions (beyond the scope of this study) might significantly impact efficiency.

There are possible techniques that can be applied for MLM to tune the efficiency of embeddings calculation. Additional research is needed for model size reduction, model distillation, float parameters quantization that can possibly improve performance.

[2]WikiCheck Github repository `https://github.com/trokhymovych/WikiCheck`.

Bibliography

Akbik, Alan et al. (2019). "FLAIR: An easy-to-use framework for state-of-the-art NLP". In: *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 54–59.

Back, David A et al. (2016). "Learning management system and e-learning tools: an experience of medical students' usage and expectations". In: *International journal of medical education* 7, p. 267.

Bojanowski, Piotr et al. (2017). "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. DOI: 10.1162/tacl_a_00051. URL: https://www.aclweb.org/anthology/Q17-1010.

Bovet, Alexandre and Hernán Makse (Jan. 2019). "Influence of fake news in Twitter during the 2016 US presidential election". In: *Nature Communications* 10. DOI: 10.1038/s41467-018-07761-2.

Bowman, Samuel R. et al. (2015). "A large annotated corpus for learning natural language inference". In: *CoRR* abs/1508.05326. arXiv: 1508.05326. URL: http://arxiv.org/abs/1508.05326.

Cazalens, Sylvie et al. (2018). "A Content Management Perspective on Fact-Checking". In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 565–574. ISBN: 9781450356404. DOI: 10.1145/3184558.3188727. URL: https://doi.org/10.1145/3184558.3188727.

Chen, Qian et al. (2016). "Enhancing and Combining Sequential and Tree LSTM for Natural Language Inference". In: *CoRR* abs/1609.06038. arXiv: 1609.06038. URL: http://arxiv.org/abs/1609.06038.

Chen, Qian et al. (2017). "Natural Language Inference with External Knowledge". In: *CoRR* abs/1711.04289. arXiv: 1711.04289. URL: http://arxiv.org/abs/1711.04289.

Conneau, Alexis et al. (Sept. 2017a). "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data". In: pp. 670–680. DOI: 10.18653/v1/D17-1070.

Conneau, Alexis et al. (Sept. 2017b). "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 670–680. DOI: 10.18653/v1/D17-1070. URL: https://www.aclweb.org/anthology/D17-1070.

Dagan, Ido, Oren Glickman, and Bernardo Magnini (2006). "The PASCAL Recognising Textual Entailment Challenge". In: *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*. Ed. by Joaquin Quiñonero-Candela et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 177–190. ISBN: 978-3-540-33428-6.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Gong, Yichen, Heng Luo, and Jian Zhang (2017). "Natural Language Inference over Interaction Space". In: *CoRR* abs/1709.04348. arXiv: 1709.04348. URL: http://arxiv.org/abs/1709.04348.

Gulin, Andrey, Igor Kuralenok, and Dimitry Pavlov (2011). "Winning The Transfer Learning Track of Yahoo!'s Learning To Rank Challenge with YetiRank". In: *Proceedings of the Learning to Rank Challenge*. Ed. by Olivier Chapelle, Yi Chang, and Tie-Yan Liu. Vol. 14. Proceedings of Machine Learning Research. Haifa, Israel: PMLR, pp. 63–76. URL: http://proceedings.mlr.press/v14/gulin11a.html.

Gururangan, Suchin et al. (2018). "Annotation Artifacts in Natural Language Inference Data". In: *CoRR* abs/1803.02324. arXiv: 1803.02324. URL: http://arxiv.org/abs/1803.02324.

Hanselowski, Andreas et al. (Nov. 2018). "UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification". In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Brussels, Belgium: Association for Computational Linguistics, pp. 103–108. DOI: 10.18653/v1/W18-5516. URL: https://www.aclweb.org/anthology/W18-5516.

Hassan, Naeemul et al. (Aug. 2017). "ClaimBuster: The First-Ever End-to-End Fact-Checking System". In: *Proc. VLDB Endow.* 10.12, 1945–1948. ISSN: 2150-8097. DOI: 10.14778/3137765.3137815. URL: https://doi.org/10.14778/3137765.3137815.

Heilman, James M and Andrew G West (2015). "Wikipedia and medicine: quantifying readership, editors, and the significance of natural language". In: *Journal of medical Internet research* 17.3, e62.

Honnibal, Matthew et al. (2020). *spaCy: Industrial-strength Natural Language Processing in Python*. DOI: 10.5281/zenodo.1212303. URL: https://doi.org/10.5281/zenodo.1212303.

Kiela, Douwe, Changhan Wang, and Kyunghyun Cho (2018). "Dynamic Meta-Embeddings for Improved Sentence Representations". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 1466–1477. DOI: 10.18653/v1/D18-1176. URL: https://www.aclweb.org/anthology/D18-1176.

Lewis, Mike et al. (2019). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *CoRR* abs/1910.13461. arXiv: 1910.13461. URL: http://arxiv.org/abs/1910.13461.

Liu, Xiaodong et al. (July 2019a). "Multi-Task Deep Neural Networks for Natural Language Understanding". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4487–4496. DOI: 10.18653/v1/P19-1441. URL: https://www.aclweb.org/anthology/P19-1441.

Liu, Yinhan et al. (2019b). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692. arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

McDowell, Zachary and Matthew Vetter (July 2020). "It Takes a Village to Combat a Fake News Army: Wikipedia's Community and Policies for Information Literacy". In: *Social Media + Society* 6, p. 205630512093730. DOI: 10.1177/2056305120937309.

Merity, Stephen et al. (2016). "Pointer Sentinel Mixture Models". In: *CoRR* abs/1609.07843. arXiv: 1609.07843. URL: http://arxiv.org/abs/1609.07843.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781.

Nie, Yixin, Haonan Chen, and Mohit Bansal (2018). *Combining Fact Extraction and Verification with Neural Semantic Matching Networks*. arXiv: 1811.07039 [cs.CL].

Pilault, Jonathan, Amine Elhattami, and Christopher Pal (2020). *Conditionally Adaptive Multi-Task Learning: Improving Transfer Learning in NLP Using Fewer Parameters Less Data*. arXiv: 2009.09139 [cs.LG].

Reimers, Nils and Iryna Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *CoRR* abs/1908.10084. arXiv: 1908.10084. URL: http://arxiv.org/abs/1908.10084.

Saez-Trumper, Diego (2019). "Online disinformation and the role of wikipedia". In: *arXiv preprint arXiv:1910.12596*.

Sathe, Aalok et al. (May 2020). "Automated Fact-Checking of Claims from Wikipedia". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, pp. 6874–6882. ISBN: 979-10-95546-34-4. URL: https://www.aclweb.org/anthology/2020.lrec-1.849.

Talman, Aarne, Anssi Yli-Jyrä, and Jörg Tiedemann (2019). "Sentence embeddings in NLI with iterative refinement encoders". In: *Natural Language Engineering* 25.4, 467–482. ISSN: 1469-8110. DOI: 10.1017/s1351324919000202. URL: http://dx.doi.org/10.1017/S1351324919000202.

Thorne, James et al. (2018b). *FEVER: a large-scale dataset for Fact Extraction and VERification*. arXiv: 1803.05355 [cs.CL].

— (2018a). "FEVER: a large-scale dataset for Fact Extraction and VERification". In: *CoRR* abs/1803.05355. arXiv: 1803.05355. URL: http://arxiv.org/abs/1803.05355.

Thorne, James et al. (Nov. 2018c). "The Fact Extraction and VERification (FEVER) Shared Task". In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Brussels, Belgium: Association for Computational Linguistics, pp. 1–9. DOI: 10.18653/v1/W18-5501. URL: https://www.aclweb.org/anthology/W18-5501.

Tomaszewski, Robert and Karen I MacDonald (2016). "A study of citations to Wikipedia in scholarly publications". In: *Science & Technology Libraries* 35.3, pp. 246–261.

Vlachos, Andreas and S. Riedel (2014). "Fact Checking: Task definition and dataset construction". In: *LTCSS@ACL*.

Vosoughi, Soroush, Deb Roy, and Sinan Aral (2018). "The spread of true and false news online". In: *Science* 359.6380, pp. 1146–1151. ISSN: 0036-8075. DOI: 10.1126/science.aap9559. eprint: https://science.sciencemag.org/content/359/6380/1146.full.pdf. URL: https://science.sciencemag.org/content/359/6380/1146.

Wang, Alex et al. (2018). "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *CoRR* abs/1804.07461. arXiv: 1804.07461. URL: http://arxiv.org/abs/1804.07461.

Wang, William Yang (2017). ""Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection". In: *CoRR* abs/1705.00648. arXiv: 1705.00648. URL: http://arxiv.org/abs/1705.00648.

Williams, Adina, Nikita Nangia, and Samuel R. Bowman (2017). "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *CoRR* abs/1704.05426. arXiv: 1704.05426. URL: http://arxiv.org/abs/1704.05426.

Yoneda, Takuma et al. (Nov. 2018). "UCL Machine Reading Group: Four Factor Framework For Fact Finding (HexaF)". In: *Proceedings of the First Workshop on Fact*

Extraction and VERification (FEVER). Brussels, Belgium: Association for Computational Linguistics, pp. 97–102. DOI: 10.18653/v1/W18-5515. URL: https://www.aclweb.org/anthology/W18-5515.

Yoon, Wonjin et al. (2020). *Learning by Semantic Similarity Makes Abstractive Summarization Better*. arXiv: 2002.07767 [cs.CL].

Zhang, Zhuosheng et al. (2020). *Semantics-aware BERT for Language Understanding*. arXiv: 1909.02209 [cs.CL].

Zhu, Yukun et al. (2015). "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *arXiv preprint arXiv:1506.06724*.