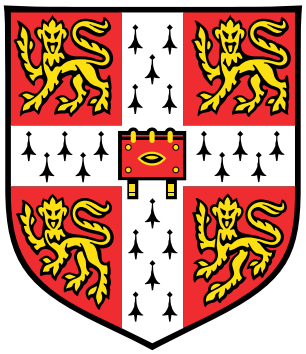
Fact-Checking Fake News



Bart Melman

Supervisors:
Dr Marcus Tomalin,
Prof. Bill Byrne

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

St Edmund's College August 2019

Declaration I

I, Bart Melman of St Edmund's College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. Total number of words in the text is 11712 and the total number of words in the text, captions and headers is 13079.

Signed:

Date: 12-Aug-2019

Declaration II

I adapted the ESIM implementation of the following Github repository: `https://github.com/coetaur0/ESIM` in the following ways: (1) I adapted code to load and process FEVER data and (2) I added options for part-of-speech embeddings and exact match embeddings. I used the sqlitedict package[1] for Python 3.6 to create a local database. The neural networks are implemented with Pytorch. The part-of-speech tags are estimated with spaCy[2]. See section 4.6 for a detailed explanation.

Signed:

Date: 12-Aug-2019

[1] `https://pypi.org/project/sqlitedict/`
[2] `https://spacy.io/`

Acknowledgements

First and foremost, I would like to thank Dr Marcus Tomalin for the weekly meetings at which I could share and discuss the struggles and progress of the project.

Furthermore, I would like to thank Prof. Bill Byrne on making me focus more on the structure of the report and the presentation of the findings.

Finally, I want to thank James Thorne for helping me with practical problems during the implementation. You challenged many of my ideas and introduced me to the state-of-the-art literature.

Abstract

'Fake News' is recognised as a serious problem by modern society, and developing automated fact-checking systems can help to recognise and disprove it. This thesis introduces an end-to-end automated fact-checking system for the FEVER challenge. The document retrieval stage uses term frequency-inverse document frequency (TFIDF) *Title Normalisation*, a new ranking method that solves the pre-processing problems of rule-based title matching methods. The sentence retrieval stage and label prediction stage use adapted versions of ESIM (Chen et al. 2016). A comparative study of different TFIDF methods shows that using unigrams with titles and TFIDF title normalisation performs best. Title Normalisation improves the absolute performance of the top 100 ranked documents with approximately 8%. Furthermore, I introduce a two-stage document retrieval system and create *Exact Match Embeddings* to deal with out-of-vocabulary words. Finally, I find that refuted claims are harder to retrieve than supported claims with text-based document retrieval models. Title-based document retrieval systems do not show this phenomenon and perform better.

Contents

Nomenclature

AR Autoregressive

FEVER Fact Extraction and VERification

GLUE General Language Understanding Evaluation

IR Information Retrieval

MLP MultiLayer Perceptron

NLI Natural Language Inference

NLP Natural Language Processing

NLU Natural Language Understanding

NSMN Neural Semantic Matching Network

POS Part-of-Speech

QA Question Answering

SQuAD Stanford Question Answering Dataset

TFIDF Term Frequency-Inverse Document Frequency

1 Introduction

1.1 Problem Description

'Fake News' has become a familiar phrase and is recognised as a serious socio-political problem in the modern world. For example, Google has announced that it will spend 300 million dollars in three years on the 'Google News Initiative[3]', where it aims to fight fake news and support journalism. Furthermore, Facebook started an initiative to partner with fact-checking organisations to flag fake news after the 2016 US elections[4]. Moreover, a consortium of companies has set up a fact-checking challenge[5] in Canada with a main prize of one million dollars. Even though there are nonprofit organisations like Politifact[6] and Full Fact[7] that perform fact-checking, their assessments are not automated yet. They use humans to read vast number of texts, and as a result, their assessments come with a delay and their processing capabilities are limited. To solve these problems, we need to create automated fact-checking systems that provide instantaneous feedback to claims made in the media. Automated fact-checking systems generally use a corpus of factual documents to verify a claim or piece of text. An automated fact-checking system can give feedback to the user in multiple ways. The system can assign a score of trustworthiness, flag fake news or provide evidence sentences which either support or refute the claim. This report will focus on the FEVER[8] challenge, a shared task which incorporates all three feedback methods.

1.2 FEVER Challenge

The FEVER challenge is a shared task in which research teams create systems for automated fact-checking. The FEVER 1.0 Shared Task (August 2018) was the first challenge, and the follow-up, the FEVER 2.0 Shared Task (August 2019), finishes at the end of August 2019. At the time of writing, the papers of the FEVER 2.0 Shared Task were not available yet. Therefore, I will discuss the systems and papers from the FEVER 1.0 Shared Task. The goal of the FEVER 1.0 Shared Task is to verify claims with a corpus of 5.4 million Wikipedia pages. The systems are ranked based on their FEVER score, and their performance is shown on the leaderboard[9].

Wikipedia Corpus The Wikipedia corpus is a pre-processed version of the June 2017 Wikipedia dump. The corpus consists of the introductory paragraphs of 5.4 million Wikipedia pages and includes both the text and the titles. The sentences from the text are numbered such that the proof of a claim can be linked to a sentence within the text. The titles of Wikipedia pages are the names of people, companies, movies, etc. and are extremely short. Examples title are: *'Kamal Muara Stadium'*, *'Richard Haddock (1673-1751)'* and *'Turn It Up (Pixie Lott album)'*. Figure 1 shows an example claim and two sentences from the Wikipedia corpus that contain the proof.

[3]https://fortune.com/2019/03/20/google-new-tools-fight-fake-news/
[4]https://www.theguardian.com/technology/2016/dec/15/facebook-flag-fake-news-fact-check
[5]`https://leadersprize.truenorthwaterloo.com/en/`
[6]https://www.politifact.com/
[7]https://fullfact.org/
[8]`http://fever.ai/`
[9]http://fever.ai/2018/task.html

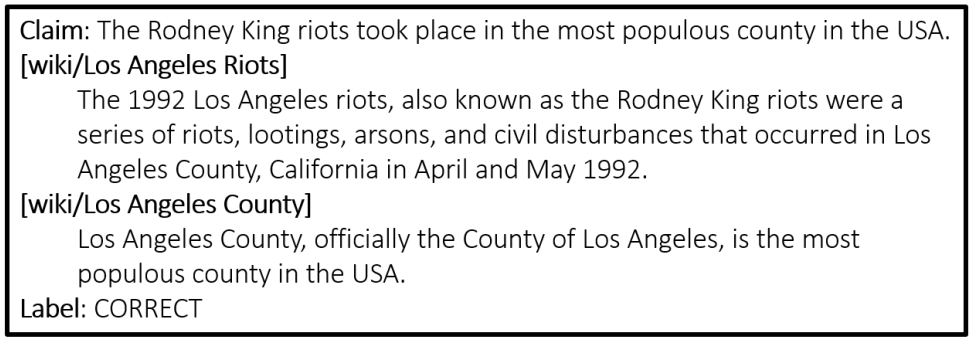


Figure 1: Example claim with two required evidence sentences from different documents.

Task The objective of the challenge is to predict the label of a claim as *CORRECT*, *REFUTED* or *NOT ENOUGH INFO*. If the predicted label is *CORRECT* or *INCORRECT*, one also has to supply a set of five proposed evidence sentences from the Wikipedia corpus to support the prediction.

Claim Dataset The FEVER (Thorne et al. 2018) claim dataset consists of 185,445 claims which are manually verified from the introductory paragraphs of 4.3 million Wikipedia pages. The claim dataset was created in two steps. First, annotators generated claims by changing the sentences of approximately $50,000$ popular pages. Secondly, other annotators were asked to label the claims by providing proof found in the corpus. There are three labels: *SUPPORTED*, *REFUTED* or *NOT ENOUGH INFO*, and for the first two, the annotators also had to include the evidence sentence(s) which led to the classification. The evidence sentences can originate from multiple pages. Figure 2 shows the format of each claim from the claim dataset.

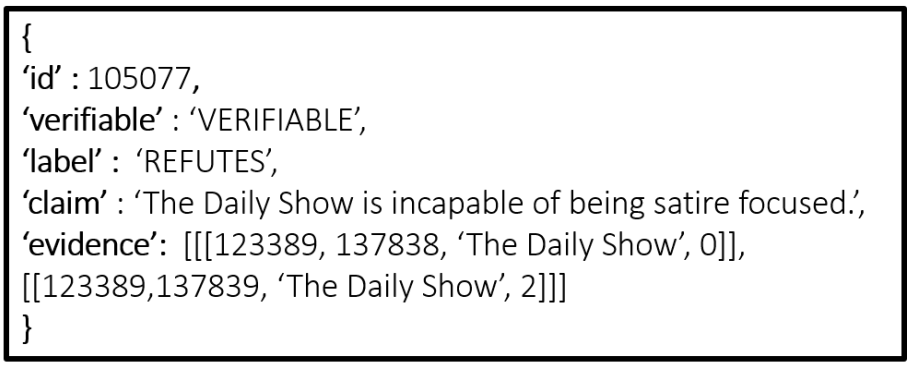


Figure 2: Example claim in the provided dictionary format with two different proofs which are both correct. The format of each proof is: [*Annotator ID, Evidence ID, Wikipedia Title, Sentence Number*]. The *Annotator ID* indicates which annotator found the proof, the *evidence ID* indicates to which evidence it belongs (sometimes there are multiple sentence that form the evidence), the *Wikipedia Title* is the title of the page in which an evidence sentence can be found and the *Sentence Number* indicates which line of the page that contains the evidence.

FEVER Score The FEVER score labels a prediction as correct if the proof of at least one of the annotators is completely covered by the proposed evidence set and if the predicted label is correct. The leaderboard of the FEVER Challenge shows the precision,

accuracy and F1 score, but the ranking is solemnly determined by the FEVER Score. The scoring metrics are described in detail in section 4.5.

1.3 Contributions

Existing fact-checking systems generally consist of three stages: (1) document retrieval, (2) sentence retrieval and (3) label prediction. The systems have an orthogonal design, meaning that the model of a stage from one system can be replaced by a different model from another system whilst keeping the rest of the stages the same. The state-of-the-art systems indicate the overall performance, but they do not show the performance of the individual stages. This has two shortcomings. First, it is unclear which of the stages needs most improvement and secondly, we do not know if a better system can be constructed by combining individual stages from different existing systems.

The objective of this report is to create a better understanding of challenges within fact-checking systems and to improve the performance of existing fact-checking systems. The main focus is to create document retrieval systems that are robust and scalable. The contributions of this report are:

Contribution 1 : *A **comparative study** of different methods of term frequency-inverse document frequency(TFIDF) for document retrieval. I investigate the effect of the tokenisation method, n-gram, TFIDF method and source (text/title).*

Contribution 2 : *A **mathematical formalisation** of the rule-based document retrieval process of current systems and the introduction of a numerical method called 'Title Normalisation', which is robust and does not require a long list of rules to pre-process titles.*

Contribution 3 : *A **label specific scoring method** to quantify the difference in performance between retrieving documents from correct claims and refuted claims.*

Contribution 4 : *I find that **unigram-based** document retrieval methods work well and that there is no need to use memory-intensive bigram-based methods.*

Contribution 5 : *I introduce a **two-stage document retrieval system** which uses a computationally efficient method to recover a selection of 100 documents and uses a second (more complex) model with additional features to improve the ranking of those documents. It allows the number of selected documents to be claim-specific: more claims are selected for longer/more complex claims.*

Contribution 6 *New input embedding to deal with OOV words in fact-checking systems called **'Exact Match Embeddings'**.*

Contribution 7 *An **end-to-end fact-checking system** to evaluate the performance of adding the 'Exact Match Embeddings' and part-of-speech tag info to word input embeddings.*

1.4 Structure of Report

The remainder of the thesis is structured as follows: Chapter 2 discusses related challenges and their best-performing systems. Furthermore, it contains an overview of the leading fact-checking systems of the FEVER 1.0 Shared Task. Chapter 3 provides general statistics of the Wikipedia corpus and the claim database. Chapter 4 describes the design choices of the individual stages and comments on the expected results. Furthermore, it discusses the scoring metrics used and the final system configuration. The results are shown in Chapter 5 and Chapter 6 contains the conclusion. The discussion and suggestions for future research are presented in Chapter 7.

2 Literature Review

2.1 Related Areas

The following three areas relate closely to fact checking: information retrieval, question answering and natural language inference.

Information retrieval (IR) is the process of extracting relevant unstructured text (documents) from a given query. With algebraic models, words are mapped to vectors and the queries are matched to the text using for example cosine similarity (Salton et al. 1975). There are also statistical models that apply topic modelling such as latent Dirichlet allocation (Blei et al. 2003). With respect to fact checking, information retrieval can most closely be described as an 'entity linking problem' (Khalid et al. 2008). Instead of finding texts that are closest to the entire claim, one matches the documents to the claim's entities. An example claim of FEVER is 'Nicholas Brody is a character on Homeland.' With entity linking, only the words 'Nicholas', 'Brody' and 'Homeland' are matched to the documents. The words 'is', 'a', 'character' and 'on' are neglected.

Question Answering (QA) is the task of giving a concise and relevant piece of text as answer to a question. There are multiple datasets such as SQuAD (Rajpurkar et al. 2018), WEBQUESTIONS (Berant et al. 2013) and WikiMovies (Miller et al. 2016), that require a system to select a span of text from a paragraph as proof for a question. The questions can be factual and more subtle. Example questions of the Stanford Question Answering Dataset (SQuAD) are: "When did the 1973 oil crisis begin?" and "Why are there more poor people in the United States and Europe than China?" Example answers are 'Oktober 1972' and 'greater tendency to take on debts'.

Natural Language Inference (NLI) is discovering the directional relationship between two pieces of text. Typically, two pieces are classified as *entailed* if they incorporate the same meaning, *contradicting* if they have a conflicting meaning and *neutral* if they discuss different topics. The label prediction stage of the FEVER challenge is a textual entailment (NLI) problem.

2.2 Other Challenges

This section highlights the most popular challenges and leaderboards in the areas of QA and NLI. These challenges are similar to FEVER, because QA also requires a document retrieval stage and the label prediction stage of FEVER is an NLI task.

2.2.1 GLUE platform

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al. 2018) is a platform which consists of nine sentence or sentence pair natural language understanding (NLU) tasks. Furthermore, it contains a language diagnostic dataset which is designed to evaluate the performance on a wide range of linguistic phenomena. These phenomena are categorised into four categories: *lexical semantics*, *predicate-argument structure*, *logic*

and knowledge and *common sense*. These are again subdivided into tasks such as *interval/numbers*, *propositional structure* and *lexical entailment*. The leaderboard[10] shows the performance of systems on the nine datasets and the performance on the diagnostic dataset. The state-of-the-art systems are adapted versions of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2018) and XLNET (Yang et al. 2019). The highest ranked models, such as XLNET-Large ensemble and MT-DNN ensemble Liu et al. (2019), are often ensemble methods. The GLUE platform is ideal for creating ensemble methods, because it shows for every individual system what it's strengths and weaknesses are. The GLUE platform can therefore be used to find systems that are dissimilar and have weaknesses in different areas.

2.2.2 SQuAD

SQuAD consists of questions generated by crowdworkers on a set of Wikipedia pages, where the answer is a segment of the text. An example question is 'What are the agents the immune system detects known as?' and the answer is 'pathogens'. The systems consist of two stages: document retrieval and reading comprehension (recovering a span of relevant text from the text). Similar to FEVER, the system must abstain from providing evidence if there is there is no valid answer in the text. There is a wide interest in this challenge and many systems have been created by companies such as Facebook, Microsoft and Google. All of the top-ranked ensemble methods incorporate BERT or XLNET as one of their systems. There are no papers written from the top ensemble methods. Example questions and answers are given in section 2.1

2.2.3 SNLI & MNLI

The Stanford Natural Language Inference[11] (SNLI) corpus (Bowman et al. 2015) consists of 570,000 English sentence pairs and the objective is to predict how the sentences relate to each other. The three labels are: *entailment*, *contradiction* or *neutral*. An example sentence pair of a contradiction is 'A man inspects the uniform of a figure in some East Asian country' and 'The man is sleeping'. Most of the FEVER systems were developed originally for this challenge, since it incorporates the same labels for the prediction stage. An improved version of the SNLI dataset is the MultiGenre NLI (MultiNLI) dataset (Williams et al. 2017), which has 433 thousand sentences, but is a more diverse corpus. It is part of the GLUE platform and consists of an evaluation leaderboard for a matched dataset and a mismatched dataset.

2.3 NLI Models

The final stage of the FEVER system uses an NLI model to predict the label of a claim. FEVER systems use adapted versions of leading NLI models (see section 2.4 and as a result, the architectures of the most popular and best performing NLI models are described in this section.

[10] https://gluebenchmark.com/leaderboard
[11] https://nlp.stanford.edu/projects/snli/

2.3.1 BERT

The BERT architecture is the baseline of many of the top systems on the GLUE leaderboard, such as Roberta (Yinhan Liu 2019), MultiTask DNN (Liu et al. 2019) and SpanBERT (Joshi et al. 2019). The BERT model consists of a bidirectional transformer in contrast to a left-to-right transformer (OPENAI GPT, Radford et al. (2018)) or bidirectional LSTMs (ELMo embeddings, Peters et al. (2018)). The advantage is that the bidirectional transformer is able to capture bidirectional representations that are jointly conditioned on the left and right context in all layers. The left-to-right transformer only captures left-to-right dependencies and does not take the context to the right of a word into consideration and the bidirectional LSTMs capture the forward and backward dependencies, but not the joint context.

In the pre-training procedure, parts of the input sentence are randomly masked and the objective is to predict the masked words. Denoising auto-encoders (Vincent et al. 2008) also predict the sentence itself, but BERT only predicts the masked words. The input embeddings are the sum of token embeddings, segmentation embeddings and position embeddings. The model is pre-trained on a large unlabelled dataset and is fine-tuned on the applied task. Apart from the output layers, the remaining architecture remains the same. Consequently, it can easily be trained for various tasks.

2.3.2 XLNET

XLNET is part of the top three ensemble methods of the SQuAD leaderboard and has the second position at the GLUE leaderboard (11 August 2019). It is a generalised autoregressive pre-training method which maximises the likelihood over all permutations of the factorisation order. As a result, it can capture bi-directional context. XLNET solves the following two limitations of BERT: (1) BERT suffers from a pretrain-finetune discrepancy, because it is pre-trained on an input symbol [MASK] which never occures in the finetuning stage. XLNet does not suffer from this issue, because the AR language modelling does not require the input to be corrupted. (2) BERT assumes that the masks are independently reconstructed, whereas this independence assumptions does not hold with language models. XLNET's AR formulation factorises the maximum likelihood using the product rule and does not assume independence. As a result, it takes the dependencies between masked words into account.

2.3.3 ESIM

Most of the fact-checking systems of FEVER use adapted versions of the enhanced sequential inference method (ESIM) (Chen et al. 2016). In short, it uses bi-directional LSTM layers with soft alignment between the encoded premise and hypothesis. A detailed explanation is given in section 4.2.1. The added value of the paper is that it is a relatively simple implementation which only uses two bidirectional LSTM layers which still obtains good results. The initial implementation uses 300 dimensional Glove (Pennington et al. 2014) word embeddings. Peters et al. (2018) improved the performance by adding embeddings from language models (ELMo).

2.4 FEVER Systems

This section contains the system descriptions of the top-ranked FEVER systems of the FEVER 1.0 shared task. The systems are compared based on their implementations for

the different stages. Current fact checking systems consist of the following three stages:

Stage 1: Document Retrieval The first step is to reduce the number of documents (5.4 million) by only selecting the ones which are relevant to the claim (typically about 5 documents). Current systems, such as Thorne et al. (2018), use implementations of TFIDF (see section 4.1.2), with unigrams and/or bigrams. Other systems, such as Yoneda et al. (2018) and Malon (2019), use title matching.

Stage 2: Sentence Retrieval The sentences from the relevant documents are used to find the potential evidence of the claim. The objective is to provide a maximum of five evidence sentences which either support or refute the claim. Most systems used adaptations of ESIM.

Stage 3: Label Prediction In the final stage, the label of the claim is predicted from the set of evidence sentences. Most systems use adapted versions of existing NLI models, such as ESIM or transformer models (Vaswani et al. 2017).

Table 1 shows the leaderboard of the FEVER 1.0 shared task. The FEVER baseline is included to indicate the initial benchmark of these systems.

Table 1: 2018 FEVER 1.0 Shared Task Final Leaderboard

The performance in terms of F1, accuracy and FEVER score of the best fact checking systems of the FEVER 1.0 shared task challenge. The accuracy denotes the label prediction accuracy. The FEVER baseline system is ranked 20^{th}.

			Evidence		FEVER
Rank	Publication	Model	F1	Accuracy	Score
1	Nie et al. (2019)	NSMN*	0.5322	0.6798	0.6398
2	Yoneda et al. (2018)	ESIM	0.3521	0.6744	0.6234
3	Hanselowski et al. (2018)	ESIM + Extension	0.3733	0.6522	0.6132
4	Malon (2019)	Transformer Network	0.6471	0.6074	0.5704
...	...	...	...	...	...
20	Thorne et al. (2018)	MLP**	0.1866	0.4892	0.2771

* Neural Semantic Matching Network

** Multi-layer perceptron

FEVER Baseline System The baseline system by Thorne et al. (2018) is a simple baseline which combines existing systems from question answering with a feature-based NLI model.

- **Document Retrieval** The first stage uses the document retrieval stage from Facebook's DrQA (Chen et al. 2017) system for question answering with Wikipedia pages. It ranks documents based on cosine similarity between binned unigram and bigram TFIDF vectors.
- **Sentence Retrieval** uses TFIDF vector similarity based on bigrams (similar to document retrieval stage).

- **Label Prediction** uses a multi-layer perceptron (MLP) (Riedel et al. 2017) with a single hidden layer and has the term frequencies and TFIDF cosine similarity between the claim and evidence as input features.

Neural Semantic Neural Networks (NSMN) The leading system by Nie et al. (2019) uses NSMN, an adapted version of ESIM, which adds shortcut connections between the input and matching layer and simplifies the output stage by only using max-pooling and a single affine layer. Furthermore, they add WordNet features (Miller 1995) and number embeddings to improve the score.

- **Document Retrieval** A document is selected if there is an exact match between the title and a span of text of the input claim. In case no titles are returned, the same rule is applied to the individual words of the claim. An NSMN is applied to disambiguate documents ($\approx 10\%$ documents) which have additional information between brackets in the title. An example is: "Savages (band)". The algorithm is case-insensitive except for the first word. Furthermore, 'a', 'an' and 'the' are removed at the start of the claim.
- **Sentence Retrieval** uses a NSMN with a single output to predict the probability that a sentence contains the proof of a claim. The five sentences with the highest probability are selected as evidence sentences.
- **Label Prediction** uses a NSMN with three output labels to predict the label of a claim from the evidence set.

Four Factor Framework for Fact Finding (HEXAF) The second ranked FEVER system by Yoneda et al. (2018) uses a four staged system which splits the label prediction module into two stages.

- **Document Retrieval** uses a logistic regression with features of the claim, title and text to predict the probability that the document contains evidence for the claim. The features include (1) the position and capitalisation within the claim, (2) the presence of stop words, (3) token match counts between the first sentence of the article and the claim, (4) whether the name is truncated and (5) whether there are words in between brackets in the claim. The model is trained on a balanced dataset of positive and negative examples. The text is pre-processed by making the claim and text lowercase and the titles are truncated to the first parenthesis.
- **Sentence Retrieval** uses a logistic regression with following features: (1) the position of the sentence within the article, (2) the length, (3) whether the article name is present, (4) token matching between the sentence and the claim and (5) the score from the document retrieval stage.
- **Label Prediction** *Stage I:* The ESIM model is used to predict the label probabilities of the individual evidence sentences. *Stage II:* These probabilities are aggregated to a single vector and used by an MLP to predict the final label probabilities. The ESIM model is pretrained on SNLI and fine-tuned on the FEVER dataset. The title is added to the sentence to deal with pronouns in the evidence sentence.

Multi-Sentence Textual Entailment for Claim Verification The third ranked system by Hanselowski et al. (2018) redefines the document retrieval stage as an entity linking problem and uses a considerably different approach to the previously mentioned approaches.

- **Document Retrieval** uses a constituency parser from AllenNLP (Gardner et al. 2018) to detect entity mentions. Every noun phrase is considered as a potential entity mention. Sometimes, a movie or song can be an adjactive or another part-of-speech. To account for this, the words between the main verb and the entire claim itself are also considered as a potential claim. The MediaWiki API[12] is used to find matching articles which have the most overlap with the query. For each entity mention, the seven highest ranked pages are stored.
- **Sentence Retrieval** consists of a model which consists of two ESIM models which independently predict the positive and negative score for a claim-evidence pair. These two scores are combined with a modified hinge loss to rank the documents. The word representations are a combination of the GLOVE embedding and the FastText (Joulin et al. 2016) embeddings, because they are both pre-trained on Wikipedia.
- **Label Prediction** uses an ESIM model for every evidence-claim pair in the evidence set. The encodings of the five systems are then used to compute five attended vectors using the cosine similarity of the claim and the individual sentences. The final stage consists of a maxpooling layer and a three layer MLP.

Transformer Networks at FEVER The system by Malon (2019) is the first FEVER system to apply Transformer networks (Vaswani et al. 2017) for label prediction stage. Furthermore, it uses a unique score aggregation method which is rule based. Normally, an MLP is used for score aggregation.

- **Document Retrieval** uses the FEVER baseline system to find matching documents using TFIDF. It adds documents whose titles match named entities and titles that feature entirely in the claim. The authors recognised that there are many questions with regard to movies. Therefore, they also include the document 'X (film)' if it exists and 'X' is retrieved.
- **Sentence Retrieval** is based on the baseline system and selects the sentences with the highest TFIDF to the claim.
- **Label Prediction**
  - **Transformer Model** The label prediction is performed using the transformer model from Radford et al. (2018) which is pre-trained for language modelling. The ESIM system has a single token for OOV words which makes it impossible to distinguish OOV named entities. With the transformer model, this is solved by allocating 10,000 more indices for OOV words. These indices are randomly initialised embeddings and a hash is taking from all the words to determine the corresponding index. The classes are weighted equally to account for the unbalanced dataset.

[12] https://www.mediawiki.org/wiki/API:Main_page

– **Rule based Score Aggregation** Instead of feeding all the sentences in the transformer, the class probabilities are computed for every evidence sentence-claim pair. As most claims only require a single sentence as evidence, they use the following rules to determine the label of a claim: (1) if any evidence supports the claim, the claim is classified as supported, (2) if any evidence refutes the claim and no evidence supports the claim, the claim is classified as refuted and (3) if no evidence sentence proofs or refutes the claim, the claim is classified as not having enough information.

3 Data

The FEVER challenge consists of a claim dataset and a Wikipedia corpus where the evidence of the claims can be found. Section 1.2 described how the datasets are formed. In this section, I present general statistics of both data sets, and I comment on how these should be interpreted.

Wikipedia dataset The Wikipedia corpus consists of the introductory paragraphs from 5.4 million Wikipedia pages and includes both the text and the titles. Table 2 shows the general statistics of the Wikipedia Corpus. The titles of Wikipedia pages are the names of people, companies, movies, etc. Therefore the titles are short (2.8 words on average). The titles can also contain numbers to indicate when an event happened.

Table 2: Wikipedia Corpus: General Statistics

Wikipedia pages	5.4 million
Sentences per page (mean)	12
Words per sentence (mean)	23
Words per title (mean)	2.8
Unique words title	1.6 million
Unique words text	4.1 million

Claim dataset The FEVER (Thorne et al. 2018) claim dataset consists of 185,445 claims. Table 3 shows the composition of the different subsets. The training set is not balanced, and there are roughly three times as many supported observations relative to the number of refuted observations.

Table 3: Claims : Observations per Label

Amount of claims per label for the training, development (dev), test and reserved set.

	Supported	**Refuted**	**NEI***	**Total**
Training	80,035	29,775	35,639	145,449
Dev	3,333	3,333	3,333	9,999
Test	3,333	3,333	3,333	9,999
Reserved	6,666	6,666	6,666	19,998
Total	93,367	43,107	48,971	185,445

(*) *NEI: not enough info*

Table 4 shows that 87.5 per cent of the claims can be verified by a single evidence sentence. In case there are multiple sentences required to prove a claim, these mostly originate from multiple pages. It is much harder to prove a claim which requires multiple evidence sentences because they also originate from different documents. Another observation is that claims with the REFUTED label on average require fewer sentences as evidence. The reason is that a REFUTED claim only requires the annotator to find a single evidence sentence to disprove part of the claim, whereas a SUPPORTED claim requires all parts of the claim to be validated.

Table 4: Claims: Nr Evidence Sentences

The percentage of claims for which at least one annotator finds evidence which consists of one, two or three sentences. The checkmark indicates the percentage of times that the evidence sentences are found on different Wikipedia pages.

Nr sentences	Different Wiki-Pages	Train		DEV	
		CORRECT	REFUTED	CORRECT	REFUTED
1		0.875	0.901	0.899	0.921
2		0.170	0.142	0.145	0.116
2	✓	0.152	0.132	0.128	0.102
3		0.025	0.019	0.017	0.015
3	✓	0.024	0.018	0.017	0.014

Table 5 shows how the maximum potential accuracy increases with the number number of provided evidence sentences. It is not possible to obtain a FEVER score of a 100% on the train or dev set. There are a few claims which require up to 41 evidence sentences (See Appendix A.1 for example) and according to the competition we are only allowed to supply a maximum of five evidence sentences. This is not a problem since only in one or two per cent of the claims more than two evidence sentences are required.

Table 5: Claims: Maximum Potential Accuracy

This table shows the maximum potential accuracy versus the number of predicted evidence sentences per claim.

Nr sentences	Train		DEV	
	CORRECT	REFUTED	CORRECT	REFUTED
1	0.875	0.901	0.899	0.921
2	0.983	0.987	0.987	0.989
3	0.994	0.996	0.995	0.996
4	0.997	0.998	0.998	0.998
5	0.999	0.998	0.999	0.998

4 Methodology

This section describes the models and scoring metrics used for the three stages of the fact checking system. In addition to that, it motivates the design decisions that are made in the process and discusses what results are expected.

4.1 Stage 1: Document Retrieval

Current leading FEVER systems, such as Nie et al. (2019), Hanselowski et al. (2018) and Malon (2019), use a rule based document retrieval process which selects a document if its entire title features in the claim. Even though the rule based document retrieval process works well for the Wikipedia Database with short titles that mostly contain proper nouns, it has some properties which make it unsuitable for other datasets.

Firstly, the rule based document selection will probably fail with databases that have documents with longer titles, because the probability that a long title has an exact overlap with the claim is small. It is likely that for many claims, no documents will be selected.

A second disadvantage of using a rule-based method is that it requires careful pre-processing of the titles. Rule-based systems have a large list of rules on how to perform tokenisation and deal with punctuation, stop words and words between brackets. The words and tokens which are removed are mainly frequently occurring ones like 'a' and '('. It is not clear that the pre-processing rules will still work well with other datasets, which makes this method not scalable. Some rules are extremely specific to the Wikipedia corpus. For example, one of the rules is to remove the word 'movie' from all the titles.

The third disadvantage is that an 'exact match' is a strict measure which fails if the title contains additional information which is irrelevant. The development set mainly contains claims about movies. For example, the movie 'savages' has two related documents: 'savages' and 'savages (movie)'. A hypothetical claim 'Savages is a movie' does not select the second document.

A numerical based document retrieval process could deal better with these kind of cases. Words in the title that frequently occur in that dataset can be assigned a lower weight to reduce its impact on the ranking. In the context of constructing fact checking systems which are robust to different datasets, I revisit the process of using TFIDF (see section 4.1.2) and came up with a new method called *Title Normalisation* which combines insights from the rule based systems and TFIDF to create a numerical score.

The document retrieval process is a computationally expensive one, since it requires the model to compare the claim to the entire Wikipedia corpus (5.4 million Wikipedia pages). Current systems therefore use simple rules or numerical methods to rank documents. I introduce a *two-stage document retrieval process* which uses a simple method to select approximately 100 documents and uses a more complicated model with more features to reduces this selection to 5 documents.

The structure of this section on document retrieval is as follows. First, I describe the different methods in which the text can be tokenised and preprocessed. Secondly, I describe the title normalisation algorithm. Thirdly, I describe the different features used in the two stage document retrieval process and the different neural networks that can be applied.

4.1.1 Tokenisation

For the document retrieval process, a bag-of-words model is used with unigrams and bigrams. I investigate four different ways of tokenising the titles and text. With *tokenisation*, words keep there original casing. With *lemmatisation*, the stem of the words are stored in the database (e.g. working → work) and words keep their original casing. Direct stemming does not take the part-of-speech into consideration, whereas lemmatisation does. Lemmatisation therefore generates a more informative vocabulary than with stemming. *Lemmatisation + part-of-speech tag* is an adapted version of *lemmatisation* where the part-of-speech tag is attached to the word (e.g. working → workVERB). The advantage is that it can distinguish words which can be both a verb and a noun (e.g. work can be a verb and a noun). With the previous methods, the words keep their original casing. With *lowercasing*, all capital letters are converted to lower case letters. In the preprocessing stage, all symbols (e.g. '$' or '(') are treated as individual words. This means that '£100' is split into two words: '£' and '100'.

4.1.2 TFIDF

Term frequency-document inverse frequency (TFIDF) is a technique used to reflect the importance of a key-word to a document in a corpus. TFIDF is defined as

$$\text{TFIDF}(t, d) = \text{TF}(t, d)\text{IDF}(t), \tag{1}$$

where t is the term, d is the document, TF is the term frequency and IDF is the inverse document frequency. I explore the following variations of the term frequency:

$$\text{TF(t,d)} = I[c_{t,d} \geq 1] \text{ (constant)} \tag{2a}$$

$$\text{TF(t,d)} = c_{t,d} \text{ (raw count)} \tag{2b}$$

$$\text{TF(t,d)} = \frac{c_{t,d}}{\sum_{t' \in d} c_{t',d}} \text{ (term frequency)}, \tag{2c}$$

where $c_{t,d}$ is the number of times term t occurs in document d. To be clear, the constant term TF(t,d) is one if the term occurs at least once and 0 if it does not occur at all. The constant and raw count term frequency have a bias towards selecting longer documents. Alternatives are to use the term frequency or double normalisation[13]. I do not use double normalisation, because it requires us to tune a parameter K and the experiments take too long to tune this parameter. The inverse document frequency is defined as

$$IDF(t) = \log \frac{N}{n_t}, \tag{3}$$

where N is the total number of documents and n_t the number of documents in which term t features at least once. There are multiple methods of combining the scores. The simple implementation is to compute the sum of the TFIDF values of the claim with respect to the document text or title:

$$SCORE = \sum_{t \in claim} \text{TFIDF}(t, d), \tag{4}$$

[13] `https://nlp.stanford.edu/IR-book/html/htmledition/maximum-tf-normalization-1.html`

where t is a term in the claim and d is the document. Another approach is to use the cosine similarity which is defined as

$$cosine\ similarity = \frac{\sum_{t \in claim} \text{TFIDF}(t, d)\text{TFIDF}(t, claim)}{\sqrt{\sum_{t \in claim} \text{TFIDF}(t, claim)^2}\sqrt{\sum_{t \in d} \text{TFIDF}(t, d)^2}}, \tag{5}$$

where the normalisation term forces the cosine similarity to be between zero and one.

It is not clear whether the cosine similarity is better than taking the TFIDF summation (Equation 4). First of all, the left term of the denominator does not impact the ranking procedure. Current document retrieval systems take the top K documents of every claim and the left term in the denominator is independent of the words in the document. Secondly, the right term in the denominator might even decrease the performance. The function of the term is not to normalise for the number of words in the document, because the term frequency (Equation 2c) already accounts for that. The actual effect of the term is that it decreases the overall score if the words from the document that do not occur in the claim are infrequent. This should not be an issue. One actually expects that to be the case with refuted statements. For example, if the claim is: 'Jonathan lives in the London', then one wants to assign the same score if a document contains the sentence 'Jonathan lives in Cambridge' or 'Jonathan lives in Grantchester' even though Grantchester will probably feature much less than Cambridge.

Due to computing time constraints, I was unable to run experiments for the cosine similarity. All the results of TFIDF show the performance difference of the summed TFIDF with a constant term (Equation 2a), raw count term (Equation 2b) and term frequency (Equation 2c).

4.1.3 Unigrams and Bigrams

The document retrieval system of the baseline system use TFIDF with a combination of unigrams and bigrams. They hash bigrams with the methodology of Weinberger et al. (2009) into 2^{24} bins. I am unable to run this on my computer and it is a very inefficient way to store information/features from a document. It is also not scalable solution for databases which are larger.

The underlying thought of TFIDF is to assign a higher weight to words that feature in a few number of documents. These words are unique and therefore more meaningful. This certainly applies for unigrams where the size of the vocabulary is relatively small compared to the number of documents/sentences. The chance that a meaningless bigram such as 'is' or 'becomes' only occurs in a few documents is extremely small. The law of large number makes sure that TFIDF can therefore be applied.

The number of observations per bigram is on average much less compared to number of observations per unigram. The law of large number does not apply to this setting and the estimated TFIDF is noisy. There are many verb tenses and it is possible that a bigram such as 'he stopped' only occurs once even though the unigrams 'he' and 'stopped' occur a lot. I would define a meaningful bigram as one where the individual unigrams are infrequent, but if they occur, they often occur together. That means that there is a connection between the two which is meaningful.

In order to constrain the number of bigrams that need to be stored, I only store the bigrams of which the part-of-speech tags of both words are meaningful. For example, a bigram such as 'George Clooney' is meaningful, because it connects two proper nouns to

form a person. I select the following part-of-speech tags as meaningful: *interjunction, noun, number, proper noun, symbol, unknown* and *adjective*. Apart from adjectives, the other part-of-speech tags should not have many forms of the word.

4.1.4 Title Normalisation

The rule based document retrieval systems only select a document if there is an exact overlap between the title and part of the claim. This rule can be applied, because the titles are generally short and contain proper nouns. In contrast to verbs or adjectives, proper nouns remain the same in every context. Intuitively speaking, it also makes sense. It is logical to select a Wikipedia page of Barack Obama if his name features in the claim. Unfortunately, the rule is very strict and the performance can degrade if stop words are part of the title or the title contains brackets. An alternative way to formulate the problem is to accept a document if most of the important words in the title feature in the claim. If we define 'importance' as how unique a word is, we can use TFIDF to assign a value to every word and normalise the score by the maximum value that can be obtained if all the words of the title would feature in the claim. The title normalised score of a document with respect to a claim is defined as:

$$\text{TN-SCORE} = \frac{\sum_{t \in claim} \text{TFIDF}(t, title)}{\sum_{t' \in title} \text{TFIDF}(t', title)}, \tag{6}$$

where $\text{TFIDF}(t, title)$ is the TFIDF value of term t with respect to the title. The terms in the summations of the numerator and denominator can also be squared like with cosine similarity. Due to time constraints, I have only investigated the performance of applying title normalisation.

4.1.5 Two Stage Document Retrieval

The objective of the two stage document retrieval system is to identify a computationally simple model in order to select a subset approximately 100 documents and to use a second system with more features to improve the ranking of this subset. For the first stage, I use title normalisation with the term frequency of Equation 2c to select a maximum of 100 documents (out of 5.4 million) per claim. The second stage extracts a constant number of features from the claim, title and document and uses a neural network to predict the probability that the document contains the evidence. It reduces the number of documents from 100 to 5 documents.

Features The process of generating features of fixed size is inspired by the research of Yoneda et al. (2018). The advantage of fixed size features is that a neural network can be trained with a constant input length. If the number of features are variable, then the neural network has to have a more complicated architecture. The TFIDF value of a claim can be used to predict the probability that a document contains the evidence. However, I think that information about the part-of-speech tag of the words could provide additional information. A TFIDF score of a proper noun might have more value than the same TFIDF score of a verb. I create a feature vector which stores the summed TFIDF values of all words with the same part-of-speech tag. Furthermore, I also calculate the maximum TFIDF value that could be obtained if all words in the claim would feature one time in

the document. Figure 3 shows an example of how I create a feature vector of length 19 for a document with respect to the claim. 'Paul' and 'Cambridge' are both proper nouns, so their TFIDF values are summed and the summed value is one of the features.

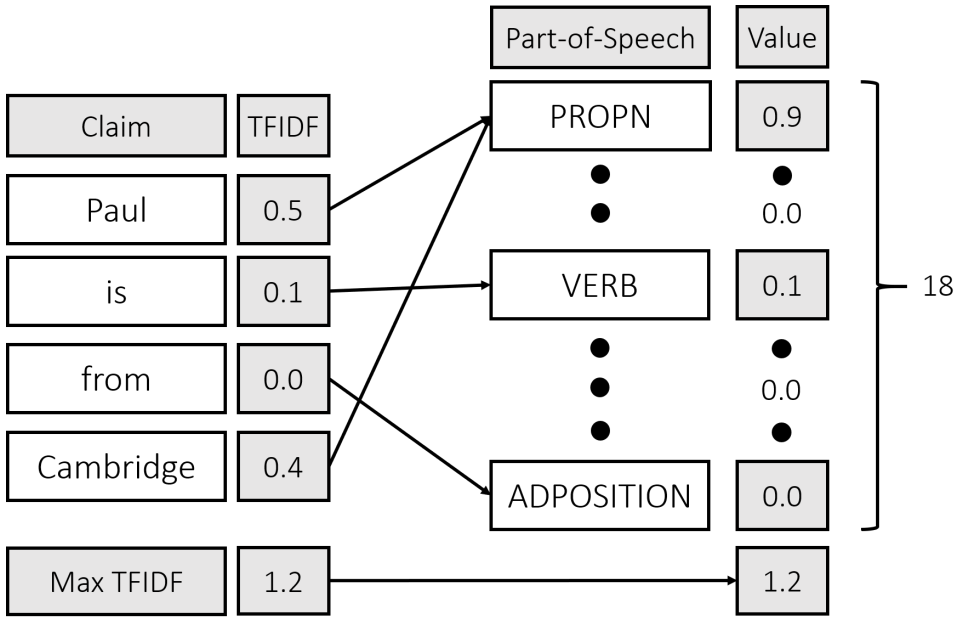


Figure 3: Feature extraction using TFIDF from claim. This is a fictitious example, where the document is a single sentence: 'Paul is living in Cambridge'. The word 'from' does not occur in the document and therefore the TFIDF score is zero. If it did occur, its TFIDF would be 0.2. The maximum TFIDF is 0.5+0.1+0.2+0.4=1.2.

Concatenating Feature Vectors The previous example generates features for one specific configuration of TFIDF (e.g. TFIDF with titles and bigrams). However, TFIDF with titles captures different dependencies than TFIDF with text and it is useful to have the information of both settings. I therefore concatenate the feature vectors of four different TFIDF settings. The setup of the feature vector is shown in Table 6. Titles mostly consist of proper nouns or nouns. As titles are short, most of the features derived from title-based approaches are zero. Documents have much more words and words with different part-of-speech tags. I expect that more part-of-speech specific information can be derived from documents than from titles.

Table 6: Feature List

This table provides an overview of the number of input features for the second system of the two-stage document retrieval system. There are four different TFIDF experiments used to construct the feature vectors. For the title with unigrams, title normalisation is used with regular tokenisation. For the other experiments, regular tokenisation is used with a constant factor as term frequency.

Claim		
n-gram	**Type**	**Nr features**
unigrams	POS count	18
unigrams	nr words	1
TFIDF		
n-gram	**Source**	**Nr features**
unigram	title	19
bigram	title	19
unigram	text	19
bigram	text	19
		+95

Models I use the following three models to rank the documents from the feature vector: (1) a logistic regression, (2) a multilayer perceptron (MLP) and (3) a residual network. Yoneda et al. (2018) use a logistic regression, but I believe that is too simplistic for this many input variables. Therefore, I also investigate the performance of an MLP and a residual network (He et al. 2016). The configuration of the neural networks and the accuracy/loss of the training set and validation set are shown in Appendix B.

4.2 Stage 2: Sentence Retrieval

Most of the leading FEVER 1.0 systems use adapted versions of the Enhanced Sequential Inference Model (ESIM) (Chen et al. 2016). In the literature review (section 2), I indicated that there are more advanced systems, like BERT and XLNET, which have a much better performance on NLI tasks. Unfortunately, these systems are hard to implement in an academic setting with limited time.

ESIM is a relatively simple model with good results and there are well-documented implementations on Github[14][15][16]. As a result, I use ESIM as a baseline. The performance gap between ESIM and BERT/ XLNET is large and it is unlikely that an improvement of ESIM with LSTMs will beat transformer implementation of BERT. In order to contribute to the current literature, I propose an alternative way of dealing will OOV words by augmenting the input embeddings with *Exact Word Match Embeddings* (see section 4.2.2) and *Part-of-Speech Embeddings* (see section 4.2.3). I use ESIM to explore the impact, but this methodology can be applied to all existing NLI models.

4.2.1 Enhanced Sequential Inference Method (ESIM)

This NLI model is the baseline of many of the FEVER systems. ESIM uses 300 dimensional Glove word embeddings as input features and predicts whether a sentence contains

[14]https://github.com/coetaur0/ESIM
[15]https://github.com/HsiaoYetGun/ESIM
[16]https://github.com/doudoucao/ESIM-1

the proof or not (single output feature). The architecture is shown in Figure 4 and consists of four stages: (1) input encoding, (2) local inference modeling, (3) inference composition and (4) prediction.

Input encoding The first layer uses bidirectional LSTMs to encode the hypothesis (evidence sentence) and premise (claim) using the following formula:

$$\begin{aligned} \bar{\boldsymbol{a}}_i &= \text{BiLSTM}(\boldsymbol{a}, i), \forall i \in [1, ..., l_a] \\ \bar{\boldsymbol{b}}_i &= \text{BiLSTM}(\boldsymbol{b}, i), \forall i \in [1, ..., l_b], \end{aligned} \tag{7}$$

where $\bar{\boldsymbol{a}}_i$ is the hidden state at time i over the input sequence $\boldsymbol{a}$, l_a is the number of words in the hypothesis, $\bar{\boldsymbol{b}}_i$ is the hidden state at time i over the input sequence $\boldsymbol{b}$ and l_b is the number of words of the premise. $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are the word embeddings of the i^{th} word in the hypothesis and premise respectively.

Local Inference Modelling The model uses a soft alignment layer which computes the attention weights as the similarity between the hidden states. The similarity is defined as

$$e_{ij} = \bar{\boldsymbol{a}}_i^T \bar{\boldsymbol{b}}_j \tag{8}$$

The attention weights are used to compute the relevant parts from the premise and hypothesis with

$$\begin{aligned} \tilde{\boldsymbol{a}}_i &= \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_b} \exp(e_{ik})} \bar{\boldsymbol{b}}_j, \forall i \in [1, ..., l_a], \\ \tilde{\boldsymbol{b}}_j &= \sum_{i=1}^{l_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_a} \exp(e_{ik})} \bar{\boldsymbol{a}}_i, \forall i \in [1, ..., l_b], \end{aligned} \tag{9}$$

where $\tilde{\boldsymbol{a}}_{\boldsymbol{i}}$ is a weighted summation of $\{\bar{\boldsymbol{b}}_j\}_{j=1}^{l_b}$. Intuitively, $\tilde{\boldsymbol{a}}_{\boldsymbol{i}}$ encodes the content of $\{\bar{\boldsymbol{b}}_j\}_{j=1}^{l_b}$ that is relevant to that position i. The local inference information is enhanced by computing the difference and element-wise product of $\langle \bar{\boldsymbol{a}}, \tilde{\boldsymbol{a}} \rangle$ and $\langle \bar{\boldsymbol{b}}, \tilde{\boldsymbol{b}} \rangle$. All features are concatenated to get the following two vectors:

$$\begin{aligned} \boldsymbol{m}_a &= [\bar{\boldsymbol{a}}; \tilde{\boldsymbol{a}}; \bar{\boldsymbol{a}} - \tilde{\boldsymbol{a}}, \bar{\boldsymbol{a}} \odot \tilde{\boldsymbol{a}}], \\ \boldsymbol{m}_b &= [\bar{\boldsymbol{b}}; \tilde{\boldsymbol{b}}; \bar{\boldsymbol{b}} - \tilde{\boldsymbol{b}}, \bar{\boldsymbol{b}} \odot \tilde{\boldsymbol{b}}], \end{aligned} \tag{10}$$

where $\odot$ is the element-wise multiplication of two vectors.

Inference Composition A bidirectional LSTM layer is used to find the enhanced local inference information of $\boldsymbol{m}_a$ and $\boldsymbol{m}_b$. The local inference is computed with

$$\begin{aligned} \boldsymbol{v}_{a,i} &= \text{BiLSTM}(F(\boldsymbol{m}_a), i), \ \forall i \in [1, ..., l_a], \\ \boldsymbol{v}_{b,i} &= \text{BiLSTM}(F(\boldsymbol{m}_b), i), \ \forall i \in [1, ..., l_b], \end{aligned} \tag{11}$$

where $F()$ is a 1-layer feedforward neural network with ReLU activation to control model complexity of this layer. The model complexity needs to be controlled, because the feature vectors $\boldsymbol{m}_a$ and $\boldsymbol{m}_b$ are four times longer than $\bar{\boldsymbol{a}}_i$ and $\bar{\boldsymbol{b}}_i$.

Prediction The variable length vectors $\boldsymbol{v}_{a,i}$ and $\boldsymbol{v}_{b,i}$ are reduced to a fixed size vector $\boldsymbol{v}$ using average and max pooling with

$$\begin{aligned}
\boldsymbol{v}_{a,ave} &= \sum_{i=1}^{l_a} \frac{\boldsymbol{v}_{a,i}}{l_a}, \\
\boldsymbol{v}_{a,max} &= \max\{\boldsymbol{v}_{a,i}\}_{i=1}^{l_a}, \\
\boldsymbol{v}_{b,ave} &= \sum_{i=1}^{l_b} \frac{\boldsymbol{v}_{b,i}}{l_b}, \\
\boldsymbol{v}_{b,max} &= \max\{\boldsymbol{v}_{b,i}\}_{i=1}^{l_b}, \\
\boldsymbol{v} &= [\boldsymbol{v}_{a,ave}; \boldsymbol{v}_{a,max}; \boldsymbol{v}_{b,ave}; \boldsymbol{v}_{b,max}]
\end{aligned} \tag{12}$$

Finally, the fixed length vector $\boldsymbol{v}$ is fed into the MLP classifier which has a hidden layer with *tanh* activation and a *softmax* output layer.

ESIM & **Sentence Retrieval** The original ESIM system is designed for the SNLI challenge (see section 2.2.3) and has three output labels. For the sentence retrieval stage, the number of output labels is reduced to two, but the rest of the system remains the same. The two labels indicate if a sentence contains evidence or not. Appendix B.4 shows the training settings of ESIM.

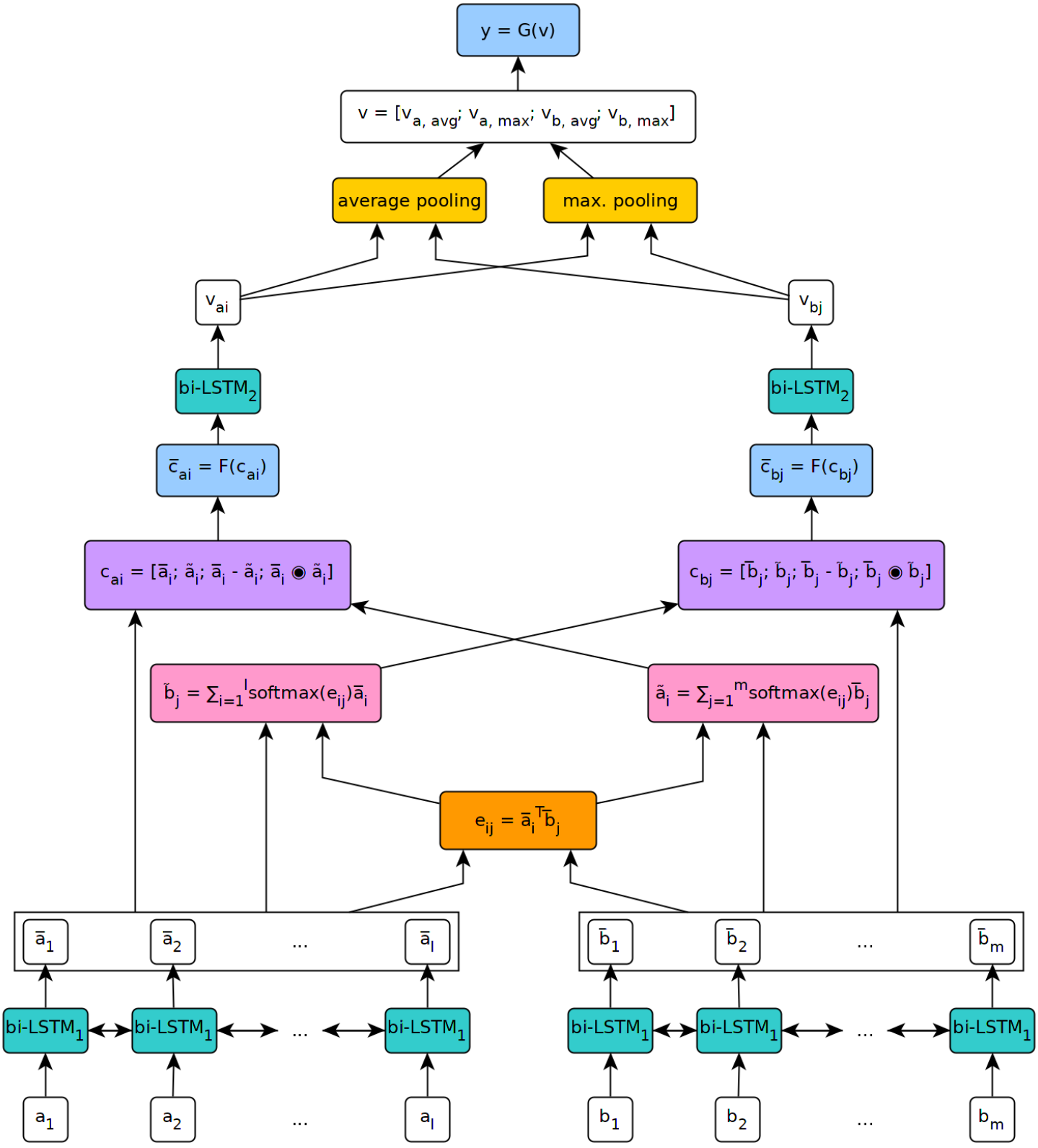


Figure 4: ESIM architecture overview[17]

[17]Source image: `https://github.com/coetaur0/ESIM`

4.2.2 Extension I: Adding Exact Word Match Embeddings

Malon (2019) note that ESIM has a single token for OOV words and that consequently certain entities in the corpus cannot be distinguished. For the SNLI challenge, there are already 4000 OOV words in the database and the genres are less diverse compared to the MultiNLI dataset. Fact checking databases, such the Wikipedia Corpus, consist of a wide range of specific topics and the systems should be able to handle OOV words without losing the ability to distinguish between them.

OOV words are often proper nouns such as cities, countries, people or companies. One does not need to know the meaning of every proper noun to predict if a claim is correct or not. Take for example the claim 'Paul lives in Cambridge' and the evidence sentence 'Paul is living in London'. Even if you have never heard of London and Cambridge, the fact that they are different words should suggest that the claim is false. The solution of the problem is therefore to create flags which indicate if a word in the claim matches a word in the evidence set exactly. The embeddings are created with the following steps:

1. Map all words in the claim to an index. Words that feature multiple times should have the same index.

2. Map the words of the evidence sentence which also feature in the claim to the same index.

3. Map the words in the evidence sentence which do not feature in the claim to a common index (e.g. 65).

4. Create a one hot encoded vector from the index for every word in the claim/evidence sentence.

Figure 5 shows an example of how indices are assigned. For every claim-evidence pair, the word-index combinations are reset. 'Paul' has the index 1 with this claim, but it can have another index with another claim. The length of the one hot encoded vector is set to the maximum number of unique words in a claim plus one. In the claim database there are some long claims, so the length is 65.

Figure 5: Exact Match Embedding Indices

The indices of the one hot encoded words from the claim (premise) and evidence sentence (hypothesis). In this example, the index for words that do not occur in the claim is 65.

Claim	Index	Evidence	Index
Paul	1	Paul	1
lives	2	is	65
in	3	living	65
Cambridge	4	in	3
		London	65

I concatenate the one hot encoded vector to the Glove embeddings, which increases the total size of the vector with approximately 20%. This increase should not be an issue, because other implementations such as Nie et al. (2019) use additional embeddings which are much longer. For simplicity, Table 5 shows an implementation where the indices have the values 1-4. The designed system randomly assigns indices to the words of the claim

and 'Paul' could have had the index 48, Cambridge the index 32, etc. The fact that the indices are randomised, forces the system the recognise OOV similarities from all the indices.

The advantage of using the exact word match embeddings is that the length of the embedding is independent of the evidence sentence. This algorithm is therefore also applicable for systems where the evidence text consists of multiple sentences. The settings for training are shown in Appendix B.4.

4.2.3 Extension II: Adding Part-of-Speech Tags

The Glove embeddings do not take the semantic context of words into consideration. The second place ranked system of SNLI by Zhang et al. (2018) achieves a higher performance with BERT by adding a semantic role labeler. Instead of semantic role labelling, I investigate if adding part-of-speech tags results in a higher performance. The part of speech tags are estimated with the spaCy package[18]. It consists of a statistical model which predicts the speech tag based on the context of the word. Table 7 shows the different part-of-speech tags. The part-of-speech embedding is an one hot encoded vector of size 19.

Table 7: SpaCy Part-of-Speech Tags

SpaCy [19]universal position tags of words. Note that the numbers include Roman numbers like MXIV or VI and that symbols include emoticons.

POS	Description	Examples	POS	Description	Examples
ADJ	adjective	big, green, first	PART	particle	's, not,
ADP	adposition	in, to, during	PRON	pronoun	he, somebody
ADV	adverb	very, tomorrow	PROPN	proper noun	John, London, NATO
AUX	auxiliary	is, has (done)	PUNCT	punctuation	., (,), ?
CONJ	conjunction	and, or, but	SCONJ	subordinating conjunction	if, while, that
CCONJ	coordinating conjunction	and, or, but	SYM	symbol	$, %
DET	determiner	a, an, the	VERB	verb	runs, ate
INTJ	interjection	psst, ouch	X	other	sfpksdpsxmsa
NOUN	noun	cat, tree	SPACE	space	
NUM	numeral	7, thirty-one, IV			

4.2.4 Training Data

The model is trained on 90% of the training data set and validated on 10% of the training data set. For every claim that is labelled as correct or refuted, there is a list of evidence sentences provided by annotators that form the proof. Each sentence in the list is treated as an observations with label 1 (contains the proof). A sentence in the Wikipedia corpus which does not feature in this list has label 0 (does not contain the proof). ESIM is

[18]`https://spacy.io`

[19]https://spacy.io/api/annotation#pos-tagging

trained best if it contains sentences with label 0 which are similar to the claim, but are not the proof. A sentence which is totally unrelated to the claim does not help the model much. For every evidence sentence in the list, I look for another sentence in the same document which has the highest cosine similarity with the claim. This sentence is added to the training set and has label 0.

4.3 Stage 3: Label Prediction

The objective of the label prediction stage is to predict if the label of the claim is *CORRECT*, *REFUTED* or *NOT ENOUGH INFO*. The leading FEVER systems use adapted versions of systems from the SNLI challenge for the label prediction stage. The SNLI challenge is very similar to the objective of the label prediction stage, because both tasks have the same output labels. The difference is that there are five evidence sentences with FEVER instead of one evidence sentence with SNLI.

4.3.1 Label Prediction Model

As a first attempt, I concatenated the five sentences in the evidence set to a single hypothesis and applied ESIM to predict the output label. The system had an accuracy of 33% and kept predicting the same label. The bad performance could be the result of the architecture of ESIM. With multiple sentences, a single bi-LSTM layer is probably not enough. To link information between sentences, it probably helps to stack multiple layers. Another reason could be that the average pooling and max pooling layer work less well if there are too many sentences with sparse overlapping information. To conclude, finding an adapted version of ESIM that is able to handle five concatenated sentences is hard.

ESIM is designed for scoring individual sentence-pairs. Therefore, ESIM can best be used for that task. The proposed system uses ESIM to individually predict the output label probabilities for every sentence in the evidence set with respect to the claim. The final output label is computed using the rule based score combination of Malon (2019). The training settings are shown in Appendix B.4.

4.3.2 Training Data

The model is trained on 90% of the claims of the training set and validated on 10% of the training data set. The data generation process is similar to the one used with sentence retrieval. The difference is that the evidence set consists of five sentences instead of one and that there are three output labels instead of two. The observations for *CORRECT* and *REFUTED* claims are created as follows. A separate observation is created for each annotator of claim. The observation consists of an evidence set of five sentences and its corresponding label. First, I add all the sentences which form the proof of the claim to the evidence set. Secondly, I randomly add sentences from the documents in which the proofs occur to the evidence set until the evidence set consists of five sentences. Observations for claims with *NOT ENOUGH INFO* are not provided by the FEVER challenge. I use the claims with the label *CORRECT* or *REFUTED* to generate observations. I look in the documents with the proof and select five sentences that are closest to the claim in terms of cosine similarity but are not the proof. These observations will still be relevant and about the same topic, but do not contain proof to either support or refute the claim.

4.4 Final Model

The purpose of creating an integrated system is to see if improvements in a particular stage also translate to improvements in the overall system. Due to limited time, I am only able to investigate the impact of the number of documents selected in the first stage and the effect of using exact matching embeddings and part-of-speech embeddings in stages two and three.

The final model has the following settings: The document retrieval stage of the system uses TFIDF with unigram title normalisation and term frequency as TF. The sentence retrieval stage uses ESIM with the exact word match embeddings and the part-of-speech embeddings. The label prediction stage uses the methodology described in section 4.3 to predict the output label from the evidence set. The ESIM uses the exact word match embeddings and the part-of-speech embeddings.

4.5 Score Metrics

This section describes the score metrics used to evaluate the performance of FEVER systems. I also introduce a new score metric called the E-SCORE which reflects the average number of evidence sentences that are recovered correctly by the system and describe how to implement label specific scoring. Even though the F1 score and the accuracy are shown in the leaderboard, only the FEVER score is used for ranking.

FEVER Score (F-SCORE) The FEVER Challenge labels a prediction as correct if the predicted label is correct and if the evidence of at least one annotator is fully captured by the predicted evidence set. The FEVER score (F-SCORE) is computed with

$$\text{F-SCORE} = \frac{1}{C}\sum_{c=1}^{C} min\Big(\sum_{a=1}^{A_c} I[E_{c,a} \in PE_c], 1\Big) I[L_{c,p} = L_c], \tag{13}$$

where C is the number of claims, A_c is the number of annotators for claim c, $E_{c,a}$ is the evidence from annotator a, PE_c is the predicted evidence set, $L_{c,p}$ is the predicted label of claim c and L_c is the claim's true label.

Evidence Score (E-SCORE) An effective fact checking algorithm should find as many evidence sentences as possible for every claim. The FEVER Score only measures if at least one evidence set is predicted correctly. The E-SCORE reflects the average number of evidences recovered correctly. In order to calculate the evidence score, one first computes for every claim and annotator the percentage of evidence sentences which feature in the predicted evidence set. After that, the evidence score is computed by averaging over the claims and annotators. The formula is:

$$\text{E-SCORE} = \frac{1}{C}\sum_{c=1}^{C} I[L_{c,p} = L_c] \frac{1}{A_c}\sum_{a=1}^{A_c} \frac{1}{M_{c,a}} \sum_{i=1}^{M_{c,a}} I[E_{c,a,i} \in PE_c], \tag{14}$$

where C is the number of claims, A_c is the number of annotators for claim c, $M_{c,a}$ is the number of evidence sentences for claim c and annotator a. PE_c is the predicted evidence set for claim c and $E_{c,a,i}$ is the i^{th} evidence sentence of annotator a for claim c. $L_{c,p}$ is the predicted label of claim c and L_c is the claim's label.

Evidence F1 The evidence F1 is the F1 score (Chinchor 1992) of the predicted evidence set. It is defined as the harmonic mean of the precision and recall and can be written as

$$F1 = \left(\frac{\frac{1}{P} + \frac{1}{R}}{2} \right)^{-1} = \frac{2PR}{P+R}, \tag{15}$$

where the precision P and recall R are defined as

$$\begin{aligned} P &= \frac{tp}{tp+fp} \\ R &= \frac{tp}{tp+fn}, \end{aligned} \tag{16}$$

where tp is the number of true positives, fp is the number of false positives and fn is the number of false negatives.

Document Retrieval Scoring To identify the effectiveness of the document retrieval stage, the F-SCORE and E-SCORE are used as if the predicted evidence set contains all the sentences from the selected documents. The F-SCORE of the document retrieval stage is therefore an upper bound of the F-SCORE of the entire system. The same applies to the E-SCORE. Thorne et al. (2018) and Yoneda et al. (2018) note that retrieving evidence for refuted claims is harder than retrieving evidence for supported claims. In order to quantify this phenomenon, I calculate the scoring metrics for each label separately.

4.6 Implementation

At the start of the project, I attempted to implement the methodology and code from Yoneda et al. (2018) and Nie et al. (2019). The neural network of Yoneda et al. (2018) did not train properly and the approach of Nie et al. (2019) required much more RAM than was available. To obtain a better understanding of the three stages, I decided to implement the three stages from scratch.

Document Retrieval requires the TFIDF values to be stored for every document. For the system to run on my computer, I decided to use SQLite[20] to avoid RAM limitations. SQLite is an SQL database engine which runs locally and it used by computers and mobile phones. The query time is the fastest[21] one of the database packages that I found (MySQL[22] and PostgreSQL[23]). For practical reasons, I used sqlitedict 1.6.0, a Python package which wraps around Pythons sqlite3 database. With SQLite, I had some issues with retrieving special characters, such as é or ö, because they were stored in a different utf format, but that was not the case with sqlitedict.

The spaCy package for Python is used to tokenise the claim, title and text and predict the part-of-speech tags. I did not use the popular natural language toolkit (NLTK) to perform tokenisation, because it does not take into consideration the context of a word to

[20] `https://www.sqlite.org/index.html`
[21] `https://www.sqlite.org/speed.html`
[22] https://www.mysql.com/
[23] https://www.postgresql.org/

predict the part-of-speech tag. The word 'work' can be both a noun and a verb and NLTK labels it as a noun by default. Another advantage of the spaCy package with respect to NLTK is that it separates tokens from numbers and words. '$100' is separated into '$'and '100' and '(movies)' is separated into '(', 'movies', and ')'.

Sentence Retrieval and Label Prediction Both the second and the third system use ESIM as the baseline. The used code[24] for ESIM is designed for SNLI and MultiNLI and the breaking NLI dataset BNLI (Glockner et al. 2018). For the sentence retrieval stage and label prediction stage, I adapted the code to add part-of-speech embeddings and exact match embeddings.

[24]https://github.com/coetaur0/ESIM

5 Results

In this section, the overall results of the end-to-end system are presented first. After that, the results of the individual systems are presented.

5.1 Proposed System

The designed end-to-end system is tested with and without the extra embeddings. Table 8 shows the performance of the proposed systems without extra embeddings. The accuracy of 50.7% is high, considering that the document retrieval stage puts an upper bound on the maximum accuracy of the entire system of 68%. The designed system performs better than the FEVER baseline but much worse than many of the leading models.

Table 8: Performance of Proposed System

The score metrics of the proposed system without the exact word match embeddings and part-of-speech embeddings.

Precision	Recall	Evidence F1	Accuracy	FEVER score
0.126	0.508	0.202	0.507	0.284

Table 9 shows that all performance metrics (except the accuracy) drop as a result of the embeddings. Reasons for a decrease in performance are given in section 5.3.

Table 9: Performance of Proposed System with Extra Embeddings.

The score metrics of the proposed system with the exact word match embeddings and part-of-speech embeddings.

Precision	Recall	Evidence F1	Accuracy	FEVER score
0.119	0.481	0.191	0.508	0.282

5.2 Document Retrieval

In this section, I present the results of the different systems designed for document retrieval.

5.2.1 Vocabulary Size

The tokenisation method impacts the number of unique words in the Wikipedia corpus. Sometimes, lowercasing and/or lemmatisation are used to reduce the vocabulary size and thereby the computational complexity. In order to relate the vocabulary size to the performance in the next sections, Table 10 shows the vocabulary size for different methods of tokenisation.

The number of unique bigrams in the titles is less than the number of unique unigrams. The first reason is that titles have an average length of 2.8 words and are short. A title of three words has two bigrams and three unigrams. As a result, there are significantly fewer bigrams that occur. The second reason is that I do not store all the bigrams. Only bigrams for which both the part-of-speech tags are nouns, proper nouns, etc. are included, but not the ones which contain verbs, pronouns, etc.

Applying lemmatisation and including the part-of-speech for unigrams in the title, results in a slightly larger vocabulary than using regular tokenisation. Even though lemmatisation should reduce the vocabulary size, including the part-of-speech tag increases it.

Table 10: Vocabulary Size

Number of unique words in Wikipedia corpus for different methods of tokenisation. The number of words are presented in millions (M).

source	n-gram	tokenisation	nr words
title	unigram	tokenise	1.6 M
title	unigram	lemma** + pos*	1.7 M
title	unigram	lowercase	1.5 M
title	bigram	tokenise	3.4 M
text	unigram	tokenise	4.1 M
text	unigram	lowercase	3.7 M
text	bigram	tokenise	17.7 M

(*) lemmatisation
(**) part-of-speech

5.2.2 TFIDF

In this section, I show how the performance of TFIDF is impacted by the source (title or document), n-gram, method of tokenisation and TF method. Table 11 shows the performance of TFIDF for various settings. The best performance is reached with the unigrams of titles with regular tokenisation. If the TF is 'constant', the TFIDF scorer has a bias for titles with more words. If the TF is 'term frequency', the bias is removed, and that translates to a higher performance ($\approx 12\%$ absolute difference for $K = 5$). For $K = 100$ the performance difference is not visible, because the scores of both approaches are roughly similar. The IDF part of the TFIDF remains the same, and the titles mostly consist of two or three words, so that impact is limited as well.

With text as a source, 'constant' performs better than 'term frequency'. The reason is that proper nouns have a relatively large weight with TFIDF, but are not mentioned linearly with the document length. Proper nouns can be people, companies, etc. and do not occur in many documents. Their IDF is larger than for other part-of-speech tags in general. In the Wikipedia pages, people can be referred to as 'he' or 'she' and a company as 'business', 'company', 'venture', etc. These words occur in many documents and have a low IDF value. To conclude, for large text, there is no linear relationship between how much a proper noun is mentioned and the length of the document. As a result, one does need to normalise by the length of the document.

Converting text to lowercase is a common thing to do (e.g. Yoneda et al. (2018) do it with HEXAF). However, I find that converting the titles to lowercase does not have a big impact on the vocabulary size, but degrades the performance a lot. The reason is that proper nouns are often capitalised, but consist of common words. For example, enterprises such as 'Apple' and 'General Motors' are infrequent words, which become frequent occurring words if they are lowercased. For example, the company 'Apple' is confused with the fruit 'apple'.

Table 11: TFIDF with Unigrams and Bigrams

This table shows the performance of TF-IDF for unigrams and bigrams using titles or text. The TFIDF Method indicates the TF method used to compute the TFIDF. 'Lemma' means lemmatisation and 'pos' means adding the part-of-speech tags. K is the number of top ranked documents that are selected.

				$K = 5$		$K = 100$	
Source	**n-gram**	**Tokenisation**	**TF Method**	**E-SCORE**	**F-SCORE**	**E-SCORE**	**F-SCORE**
title	unigram	tokenize	term frequency	0.675	0.711	0.820	0.859
title	unigram	tokenize	constant	0.552	0.583	0.820	0.860
title	unigram	lemma	term frequency	0.434	0.459	0.709	0.743
title	unigram	lemma + pos	term frequency	0.498	0.526	0.675	0.708
title	unigram	lowercase	term frequency	0.442	0.467	0.732	0.768
title	bigram	tokenize	term frequency	0.528	0.557	0.579	0.610
title	bigram	tokenize	constant	0.444	0.469	0.571	0.600
text	unigram	tokenize	term frequency	0.019	0.021	0.158	0.167
text	unigram	tokenize	constant	0.460	0.483	0.791	0.818
text	unigram	lowercase	constant	0.219	0.229	0.476	0.486
text	bigram	tokenize	term frequency	0.088	0.094	0.367	0.385
text	bigram	tokenize	constant	0.244	0.259	0.499	0.524

5.2.3 Title Normalisation

Table 12 shows the effect of title normalisation for different methods of tokenisation. For all tokenisation methods, title normalisation reduces the performance for $K = 5$, but increases the performance for $K = 100$. In short, it becomes worse at ranking the closest documents to the claim, but it becomes better at making a general selection. One would expect that a scoring method should be better or worse in both because the reason that it performs better or worse for $K = 100$ should also apply for $K = 5$. This should not necessarily be the case, because some models are good at extracting simple dependencies from the data that apply to all observations, but lack the capability of extracting more precise dependencies which only apply to a subset of the observations.

The reason for implementing title normalisation could also be the reason why the performance with five selected documents is a bit worse. With title normalisation, an equal score is assigned to documents that exactly feature in the dataset. However, we can argue that having an exact match with a 3-word title is better than having an exact match with a 2-word title. As a result, there can be a bias towards shorter titles, because they are more likely to occur in the claim entirely.

Table 12: Title Normalisation

This table shows the performance of TF-IDF for unigrams and bigrams using titles or text. K is the number of top ranked documents that are selected.

				$K = 5$		$K = 100$	
n-gram	**Tokenisation**	**TF Method**	**TN**	**E-SCORE**	**F-SCORE**	**E-SCORE**	**F-SCORE**
unigram	tokenize	term frequency	False	0.675	0.711	0.820	0.859
unigram	tokenize	term frequency	True	0.648	0.684	0.902	0.942
unigram	lemma	term frequency	False	0.434	0.459	0.709	0.743
unigram	lemma	term frequency	True	0.201	0.213	0.773	0.811
unigram	lemma + pos	term frequency	False	0.498	0.526	0.675	0.708
unigram	lemma + pos	term frequency	True	0.291	0.308	0.766	0.803
unigram	lowercase	term frequency	False	0.442	0.467	0.732	0.768
unigram	lowercase	term frequency	True	0.048	0.051	0.589	0.617

5.2.4 Selecting More Documents

Increasing the number of selected documents K will increase the probability that at least one of the proofs is contained in the document set. However, it also forces the second stage to process more sentences, which in return could reduce the performance. Table 13 shows the effect of selecting more documents per claim. The absolute performance difference between $K = 5$ and $K = 100$ is $\approx 25 - 30\%$ and is substantial. It is not practical to select 100 documents, but this setup could definitely be used for the two-stage system. The minimum accuracy of that system should be $\frac{0.685}{0.952} = 0.72$.

Table 13: Number of Selected Documents

This table shows the effect of changing the number of selected documents. The experiment uses tokenisation of unigrams with title normalisation and 'term frequency' (TF method). K is the number of top ranked documents that are selected.

K	E-SCORE	F-SCORE
5	0.648	0.685
10	0.804	0.846
20	0.865	0.883
50	0.883	0.925
100	0.902	0.952

5.2.5 Label Specific Scoring

Thorne et al. (2018) and Yoneda et al. (2018) note that it is harder to retrieve documents with evidence for claims that are supported than from claims that are refuted. Most document retrieval methods are based on the similarity between the claim and the document. The proof of refuted claims is in nature more dissimilar to the claim than the proof of a supported claim. As a result, the document retrieval methods should have more difficulties with retrieving proof from refuted statements. Table 14 show the label specific score for TFIDF with titles and text. With titles as a source, the F-score is roughly the same for each label. It is not harder to verify refuted claims with respect to correct claims. With text as a source, the F-SCORE for correct claims is between 5 and 15% higher than refuted claims. From this, we can conclude that it is harder to find documents with proof for refuted claims with document-based TFIDF methods.

Table 14: Label Specific Scoring

This table shows the F-SCORE for correct and refuted claims. There is no title normalisation. K is the number of top ranked documents that are selected.

				$K = 5$		$K = 100$	
Source	**n-gram**	**Tokenisation**	**TFIDF Method**	**CORRECT F-SCORE**	**REFUTED F-SCORE**	**CORRECT F-SCORE**	**REFUTED F-SCORE**
title	unigram	tokenize	term frequency	0.716	0.706	0.858	0.861
title	unigram	tokenize	constant	0.581	0.585	0.855	0.865
title	bigram	tokenize	constant	0.563	0.551	0.613	0.606
text	unigram	tokenize	constant	0.548	0.417	0.856	0.781
text	bigram	tokenize	term frequency	0.291	0.227	0.555	0.492

5.2.6 Two Stage Document Retrieval

Table 15 shows the performance of the two-stage document retrieval system. The difference between a one-stage system and a two-stage system is that the one-stage system creates a SCORE which has no upper bound, whereas the two-stage system calculates a probability directly. The performance of all proposed systems is worse than the baseline. The neural network is trained to make predictions directly from features of a claim and a document. It might be that the features of other documents with respect to the same claim can provide contextual information to make a better prediction. A suggestion for potential research is to incorporate the context by including input features from other documents as well. Another reason why the performance of the two-stage system is worse than the one-stage system is that the training data set was small. Due to time constraints, I could only create a dataset of 30,000 observations.

Table 15: Performance Two-Stage Document

This table shows the performance of using a two-stage document retrieval process with different neural networks. The baseline is to take the top 5 documents directly from the rankings of the first stage of the two-stage document retrieval system.

Method	**F-SCORE**
One Stage Baseline	0.684
Logistic Regression	0.190
MLP	0.059
Residual Network	0.212

5.2.7 Setting a Threshold

The output of the two-stage system is the probability that a document contains the evidence. A threshold-based model selects a document if the predicted probability is above a certain threshold TR. That means that some claims select more documents than others. Table 16 shows the effect of the threshold on the F-SCORE. The number of documents is not changed a lot by the threshold. That means that he model either signs a very high probability to the document or a very small one. The accuracy of the model is not very high, so that means that it has overfitted a lot and was unable to pick up enough information to make an accurate prediction of the probability. Probably the neural network has performed clustering and within the clusters, all observations have the

same (high or low) probability. More data or a Bayesian neural network should hopefully yield better results.

The best performing model is the residual neural network, because it has a higher F-SCORE and less documents per selection. The logistic regression works well as well, considering that it only has a small number of parameters and is the most simple one.

Table 16: Performance Two-Stage Document Retrieval

This Table shows the impact of the threshold on the number of documents per selection (D/S) for three different models.

	Logistic Regression		MLP		Residual Network	
Threshold	**F-SCORE**	**D/S**	**F-SCORE**	**D/S**	**F-SCORE**	**D/S**
0.25	0.377	31.9	0.262	27.8	0.379	24.7
0.5	0.364	29.1	0.235	24.7	0.375	24.3
0.75	0.351	27.1	0.212	22.3	0.370	24.0

5.3 Sentence Retrieval

The training results of ESIM with different embeddings is shown in Table 17. Even though the validation accuracy slightly increases with including embeddings, the actual validation loss is higher. I, therefore, conclude that the performance is worse with exact match embeddings and part-of-speech embeddings. The extra embeddings seem to encourage overfitting: the training loss decreases and the validation loss increases. A reason can be the increased size of the embeddings. However, other systems (e.g. Nie et al. (2019)) use embeddings, which are much longer and these embeddings only increase the embedding size with approximately 20%.

The generated training dataset consists of approximately 108000 observations, which is five times smaller than the SNLI dataset. Yoneda et al. (2018) pre-train their neural network with the SNLI dataset. Future work can investigate if pre-training with the SNLI dataset can be done to improve performance.

Another reason that the exact matching embeddings do not yield a better performance could be the fact that the number of hidden units of the LSTMs was constant (300) for all experiments. The 300-dimensional Glove word embeddings have the same size as the number of hidden units. Without the additional word embeddings, the hidden states can be a transformation of the word embeddings. With the additional word embeddings, the input embedding needs to be compressed to a hidden vector and that might be hard for the network. Future research can investigate if setting the number of hidden units to the length of the input embeddings improves the performance.

Table 17: Sentence Retrieval: Training Results

This table shows the effect of adding exact matching embeddings (EM Emb) and part-of-speech embeddings (POS Emb) for sentence retrieval. The epoch number indicates the number of epochs it took to train the model with the lowest validation accuracy.

EM Emb	**POS Emb**	**epoch**	**train loss**	**val loss**	**val acc**
		9	0.124	0.227	0.927
✓		14	0.094	0.241	0.927
✓	✓	19	0.087	0.247	0.928

5.4 Label Prediction

Table 18 shows the training of ESIM for label prediction. Adding the embeddings results in a slightly higher validation accuracy but a higher validation loss. Therefore, we cannot conclude that the extra embeddings improve the system. These observations are in line with the training results from sentence retrieval in the previous section.

Table 18: Label Prediction: Training Results

This table shows the effect of adding exact matching embeddings (EM Emb) and part-of-speech embeddings (POS Emb) for label retrieval. The epoch number indicates the number of epochs it took to train the model with the lowest validation accuracy.

EM Emb	POS Emb	epoch	train loss	val loss	val acc
		19	0.239	0.469	0.857
✓	✓	16	0.173	0.484	0.867

6 Conclusion & Discussion

There is a joint consensus between industry, politics and society that 'Fake News' is a serious problem that needs to be addressed. The FEVER challenge encourages the development of automated fact-checking systems, but these systems do not provide solutions to the inherent challenges of fact-checking. In this thesis, I aimed to identify and quantify these challenges and provide solutions that are scalable and contribute towards the long term goal: fact-checking claims from unstructured data.

Current leading FEVER systems, such as Nie et al. (2019), Hanselowski et al. (2018) and Malon (2019), use rule-based title matching algorithms which are sensitive to stop words, punctuation and other symbols. To deal with this problem, they use a large list of operations to pre-process the titles. My contribution is a new method called *Title Normalisation*, which selects the same documents as the rule-based systems, but is robust to stop words, punctuation, etc. and can be applied to other datasets. I find that 94% of the claims have at least one of the annotator's proofs in the top 100 ranked documents (see Table 12).

An unaddressed issue in fact-checking is that the number of retrieved documents should depend on the claim. Current systems select the top K ranked documents for every claim, but they should ideally select more document for longer claims than for shorter claims because there is more information to verify. I created a system which takes the top 100 ranked documents using title normalisation and predicts the probability that the document contains the proof using additional features. All documents with a predicted probability over a certain threshold were selected. Even though the two-stage approach performs much worse than the initial ranking (see Table 15), I think that introducing more features and including more data can improve the performance.

Fact-checking systems have databases with a wide range of topics. If the word embeddings are not trained on the same dataset, then there can be a lot of OOV words. I proposed an *Exact Match Embedding* which adds flags to the input embeddings that indicate which words in the claim and premise are exactly the same. For sentence retrieval and label prediction, the exact match embeddings decrease the performance. However, several suggestions have been proposed, such as pre-training the model with SNLI and setting the number of hidden units equal to the size of the embedding.

Thorne et al. (2018) and Yoneda et al. (2018) note that retrieving documents for refuted claims is harder than retrieving documents for supported claims. The reason is that the proof of supported claims is more similar to the claim. There are more overlapping words which can be used to match it with the claim. I created label specific scores and quantified the effect. In line with the hypothesis, I find that document-based TFIDF approaches have an F-SCORE, which is 15% higher for supported claims compared to refuted claims. However, title-based TFIDF approaches retrieve supported and refuted claims equally well and do not show this phenomenon. Titles mostly consist of proper nouns, nouns and numbers. I therefore conclude that refuted claims do not have many name changes.

Given the results of the report, I argue that the document retrieval stage should be interpreted as an *entity linking problem*. First of all, the experiments which mostly used proper nouns (entities) for document retrieval generally perform better. For example, title-based TFIDF methods have a higher performance than document-based TFIDF methods and titles mainly consist of proper nouns (entities). Secondly, experiments which mainly used proper nouns (entities) are better at retrieving refuted claims than experi-

ments which used the entire text. For example, title based approaches retrieve refuted and supported claims equally well, and text-based approaches have a harder time retrieving the evidence of refuted claims. Thirdly, it is more practical only to consider proper nouns or nouns. It reduces the vocabulary and therefore, the computational complexity. Furthermore, one can generally only refer to entities (e.g. 'France') with one word, which is not the case with verbs that come in many tenses. This makes it easy to search for them compared to verbs, which occur in many tenses.

7 Future Research: Unresolved Issues in Fact Checking

In the methodology (section 4) and results (section 5), I have provided multiple suggestions for future research. In this section, I will not repeat these remarks. Instead, I introduce two issues which are relevant to fact-checking and should be investigated further.

Even for humans, it can sometimes be difficult to label a claim as correct or incorrect with conflicting information. For example, is the following claim true: 'Trump is a real estate investor'? Currently, he is the president, but in the past he did real estate investing. It also depends on the set of potential evidence sentences. If there is just one sentence which says 'Trump is the president of the US', one refutes the claim, but what do you do if another sentence says 'Trump is the director of a real estate company'? These two sentences are conflicting and it is currently unclear what the right way is to deal with this problem. Future research should investigate if these claims should be flagged as *CONFLICTING INFO*. This problem has not been discussed much, because it does not exist with other challenges, such as SNLI that only have a single evidence sentence.

Fact checking systems in the future should be able to add information from additional databases to the existing database. Therefore, future research should focus on creating its own database structure with a different entity on every page. Loosely speaking, the objective is make your own Wikipedia corpus. An easy approach would be to make a separate page for every entity and to copy all lines that mention the entity to it's respective page. Hopefully, this should improve the document retrieval process as well. The advantage of this approach is that you only need to retrieve one document per entity in the claim and that you have all the sentences about that entity at your disposal to perform sentence retrieval and label prediction.

References

Berant, J., Chou, A., Frostig, R. & Liang, P. (2013), Semantic parsing on freebase from question-answer pairs, *in* 'Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing', pp. 1533–1544.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003), 'Latent dirichlet allocation', *Journal of machine Learning research* **3**(Jan), 993–1022.

Bowman, S. R., Angeli, G., Potts, C. & Manning, C. D. (2015), 'A large annotated corpus for learning natural language inference', *arXiv preprint arXiv:1508.05326* .

Chen, D., Fisch, A., Weston, J. & Bordes, A. (2017), 'Reading wikipedia to answer open-domain questions', *arXiv preprint arXiv:1704.00051* .

Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H. & Inkpen, D. (2016), 'Enhanced lstm for natural language inference', *arXiv preprint arXiv:1609.06038* .

Chinchor, N. (1992), Muc-4 evaluation metrics, *in* 'Proceedings of the 4th conference on Message understanding', Association for Computational Linguistics, pp. 22–29.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805* .

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N., Peters, M., Schmitz, M. & Zettlemoyer, L. (2018), 'Allennlp: A deep semantic natural language processing platform', *arXiv preprint arXiv:1803.07640* .

Glockner, M., Shwartz, V. & Goldberg, Y. (2018), 'Breaking nli systems with sentences that require simple lexical inferences', *arXiv preprint arXiv:1805.02266* .

Hanselowski, A., Zhang, H., Li, Z., Sorokin, D., Schiller, B., Schulz, C. & Gurevych, I. (2018), 'Ukp-athene: Multi-sentence textual entailment for claim verification', *arXiv preprint arXiv:1809.01479* .

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L. & Levy, O. (2019), 'Spanbert: Improving pre-training by representing and predicting spans', *arXiv preprint arXiv:1907.10529* .

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H. & Mikolov, T. (2016), 'Fasttext. zip: Compressing text classification models', *arXiv preprint arXiv:1612.03651* .

Khalid, M. A., Jijkoun, V. & De Rijke, M. (2008), The impact of named entity normalization on information retrieval for question answering, *in* 'European Conference on Information Retrieval', Springer, pp. 705–710.

Liu, X., He, P., Chen, W. & Gao, J. (2019), 'Multi-task deep neural networks for natural language understanding', *arXiv preprint arXiv:1901.11504* .

Malon, C. (2019), 'Team papelo: Transformer networks at fever', *arXiv preprint arXiv:1901.02534* .

Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A. & Weston, J. (2016), 'Key-value memory networks for directly reading documents', *arXiv preprint arXiv:1606.03126* .

Miller, G. A. (1995), 'Wordnet: a lexical database for english', *Communications of the ACM* **38**(11), 39–41.

Nie, Y., Chen, H. & Bansal, M. (2019), Combining fact extraction and verification with neural semantic matching networks, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 33, pp. 6859–6866.

Pennington, J., Socher, R. & Manning, C. (2014), Glove: Global vectors for word representation, *in* 'Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)', pp. 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018), 'Deep contextualized word representations', *arXiv preprint arXiv:1802.05365* .

Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. (2018), 'Improving language understanding by generative pre-training', *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf* .

Rajpurkar, P., Jia, R. & Liang, P. (2018), 'Know what you don't know: Unanswerable questions for squad', *arXiv preprint arXiv:1806.03822* .

Riedel, B., Augenstein, I., Spithourakis, G. P. & Riedel, S. (2017), 'A simple but tough-to-beat baseline for the fake news challenge stance detection task', *arXiv preprint arXiv:1707.03264* .

Salton, G., Wong, A. & Yang, C.-S. (1975), 'A vector space model for automatic indexing', *Communications of the ACM* **18**(11), 613–620.

Thorne, J., Vlachos, A., Christodoulopoulos, C. & Mittal, A. (2018), 'Fever: a large-scale dataset for fact extraction and verification', *arXiv preprint arXiv:1803.05355* .

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017), Attention is all you need, *in* 'Advances in neural information processing systems', pp. 5998–6008.

Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. (2008), Extracting and composing robust features with denoising autoencoders, *in* 'Proceedings of the 25th international conference on Machine learning', ACM, pp. 1096–1103.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. & Bowman, S. R. (2018), 'Glue: A multi-task benchmark and analysis platform for natural language understanding', *arXiv preprint arXiv:1804.07461* .

Weinberger, K., Dasgupta, A., Attenberg, J., Langford, J. & Smola, A. (2009), 'Feature hashing for large scale multitask learning', *arXiv preprint arXiv:0902.2206* .

Williams, A., Nangia, N. & Bowman, S. R. (2017), 'A broad-coverage challenge corpus for sentence understanding through inference', *arXiv preprint arXiv:1704.05426* .

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. & Le, Q. V. (2019), 'Xlnet: Generalized autoregressive pretraining for language understanding', *arXiv preprint arXiv:1906.08237* .

Yinhan Liu, Myle Ott Naman Goyal Jingfei Du Mandar Josh, D. C. O. L. M. L. L. Z. V. S. (2019), 'Roberta: A robustly optimized bert pretraining approach', *arXiv preprint arXiv:1907.11692* .

Yoneda, T., Mitchell, J., Welbl, J., Stenetorp, P. & Riedel, S. (2018), Ucl machine reading group: Four factor framework for fact finding (hexaf), *in* 'Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)', pp. 97–102.

Zhang, Z., Wu, Y., Li, Z., He, S., Zhao, H., Zhou, X. & Zhou, X. (2018), 'I know what you want: Semantic learning for text comprehension', *arXiv preprint arXiv:1809.02794* .

Appendix

A Data

A.1 Claim with large evidence.

Code:

```
Example of a cl
{'id': 88173,
 'verifiable': 'VERIFIABLE',
 'label': 'SUPPORTS',
 'claim': 'Bonnie Hunt was in a movie.',
 'evidence': [[[105393, 118858, 'Bonnie Hunt', 1]],
  [[105393, 118859, 'Bonnie Hunt', 4],
   [105393, 118859, 'Zootopia', 8],
   [105393, 118859, 'Zootopia', 6],
   [105393, 118859, 'Zootopia', 0],
   [105393, 118859, 'Zootopia', 1],
   [105393, 118859, 'Zootopia', 2],
   [105393, 118859, 'Zootopia', 9],
   [105393, 118859, 'Zootopia', 7],
   [105393, 118859, "A Bug's Life", 12],
   [105393, 118859, "A Bug's Life", 14],
   [105393, 118859, "A Bug's Life", 8],
   [105393, 118859, "A Bug's Life", 0],
   [105393, 118859, "A Bug's Life", 1],
   [105393, 118859, "A Bug's Life", 5],
   [105393, 118859, "A Bug's Life", 2],
   [105393, 118859, "A Bug's Life", 9],
   [105393, 118859, 'Monsters, Inc.', 13],
   [105393, 118859, 'Monsters, Inc.', 14],
   [105393, 118859, 'Monsters, Inc.', 8],
   [105393, 118859, 'Monsters, Inc.', 6],
   [105393, 118859, 'Monsters, Inc.', 0],
   [105393, 118859, 'Monsters, Inc.', 1],
   [105393, 118859, 'Monsters, Inc.', 2],
   [105393, 118859, 'Monsters, Inc.', 9],
   [105393, 118859, 'Toy Story 3', 12],
   [105393, 118859, 'Toy Story 3', 13],
   [105393, 118859, 'Toy Story 3', 14],
   [105393, 118859, 'Toy Story 3', 8],
   [105393, 118859, 'Toy Story 3', 0],
   [105393, 118859, 'Toy Story 3', 4],
   [105393, 118859, 'Toy Story 3', 3],
   [105393, 118859, 'Monsters University', 13],
   [105393, 118859, 'Monsters University', 2],
   [105393, 118859, 'Monsters University', 19],
   [105393, 118859, 'Monsters University', 7],
   [105393, 118859, 'Monsters University', 0],
   [105393, 118859, 'Cars (film)', 11],
   [105393, 118859, 'Cars (film)', 8],
   [105393, 118859, 'Cars (film)', 9],
   [105393, 118859, 'Cars (film)', 0],
   [105393, 118859, 'Cars (film)', 10]]]}
```

B Training Settings of Neural Networks

B.1 Logistic Regression

Table 19: Training Settings Logistic Regression

This Table shows the training settings of the logistic regression for document retrieval.

Variable	Value
Optimizer	Adam
Loss function	BCEWithLogitsLoss
Learning Rate	0.001
Max nr epochs	200
Batch size	128

B.2 MLP

Table 20: Training Settings MLP

This Table shows the training settings of the MLP for document retrieval.

Variable	Value
Optimizer	Adam
Loss function	BCEWithLogitsLoss
Learning Rate	0.001
Max nr epochs	30
Width	1000
Depth	2
Batch size	64

B.3 Residual Network

Table 21: Training Settings Residual Network

This Table shows the training settings of the residual neural network for document retrieval.

Variable	Value
Optimizer	Adam
Loss function	BCEWithLogitsLoss
Learning Rate	0.001
Nr epochs	30
Width	2000
Depth	2
Batch size	64

B.4 ESIM

Table 22: Training Settings ESIM

This Table shows the training settings of ESIM for sentence retrieval and label prediction.

Variable	Value
Optimizer	Adam
Loss Function	CrossEntropyLoss
Learning Rate	0.0004
Max nr epochs	64
Hidden dimension	300
Dropout	0.5
Batch size	32

C Training Neural Networks

C.1 Logistic Regression

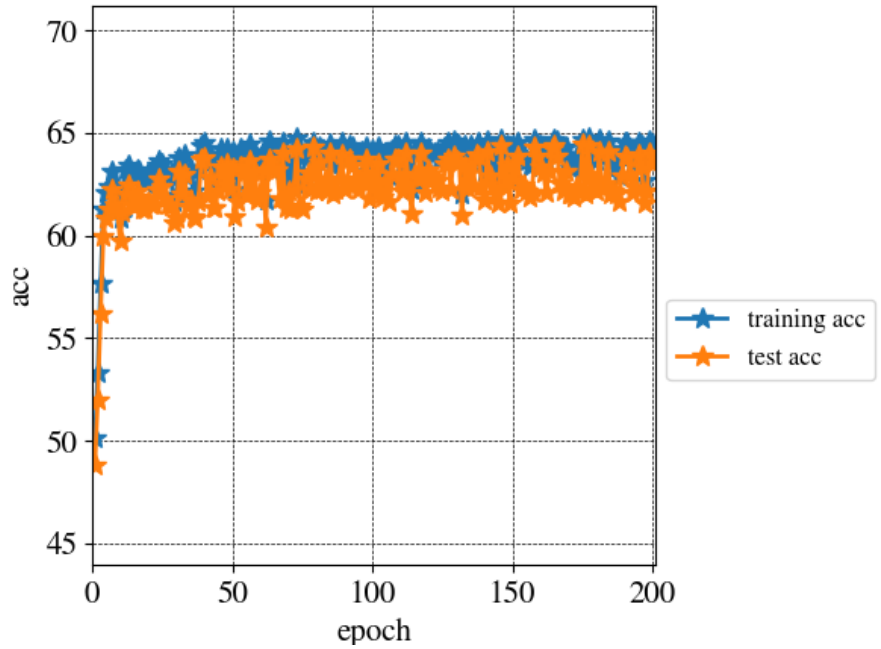


Figure 6: Training and validation accuracy versus the number of epochs for the logistic regression model.

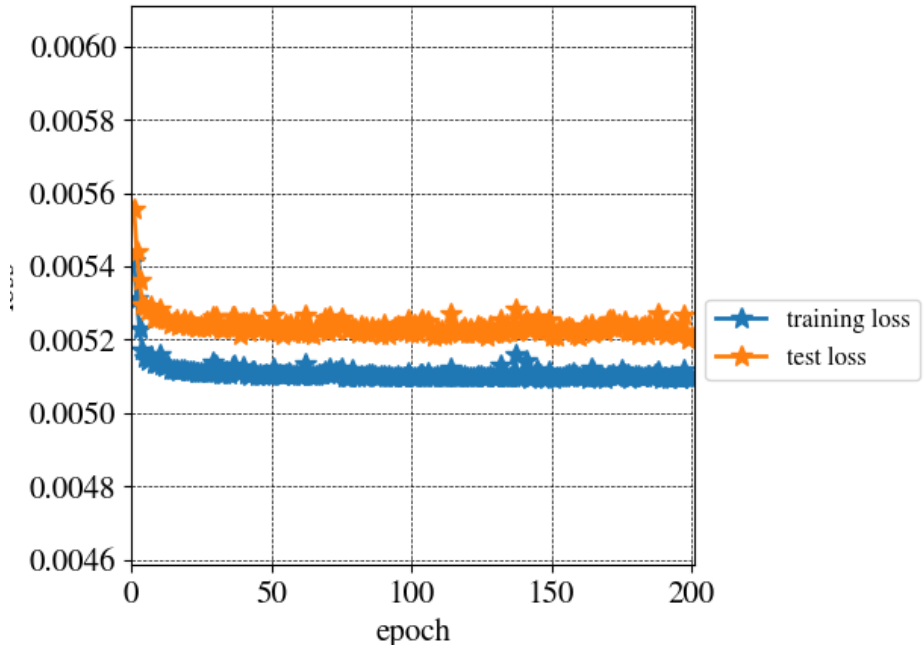


Figure 7: Training and validation loss versus the number of epochs for the logistic regression model.

C.2 MLP

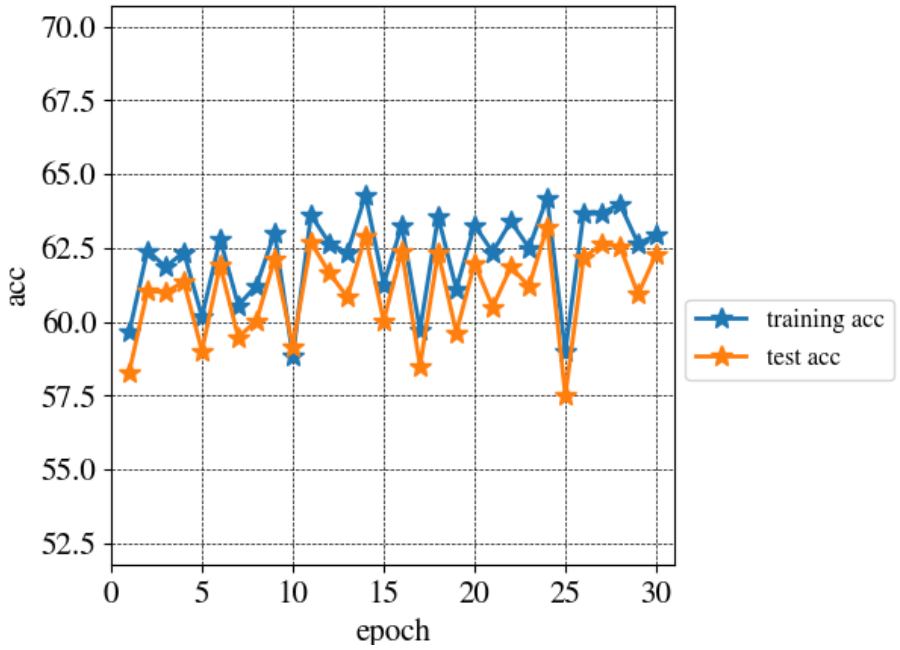


Figure 8: Training and validation accuracy versus the number of epochs for the MLP.

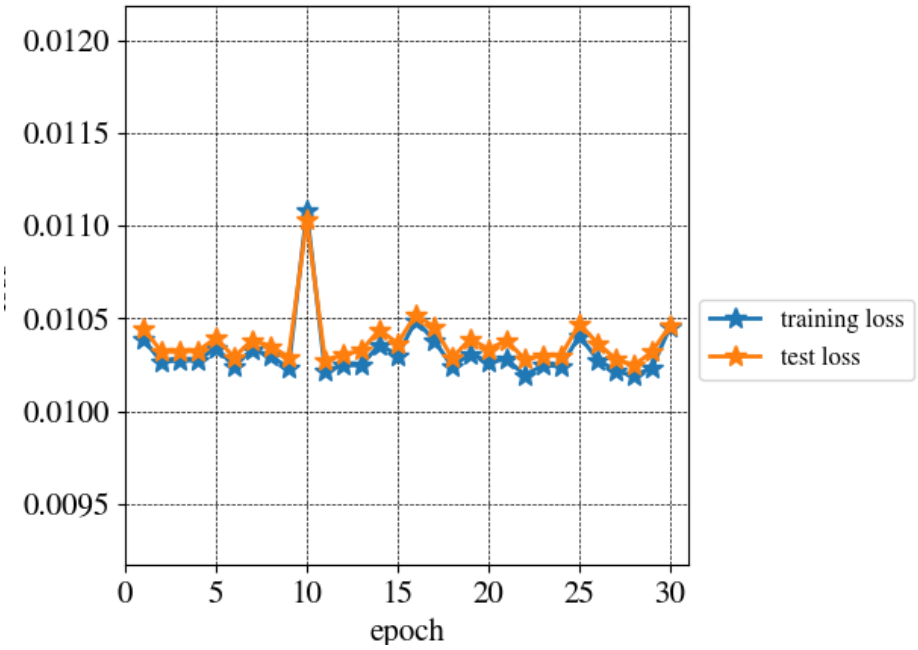


Figure 9: Training and validation loss versus the number of epochs for the MLP.

C.3 Residual Network

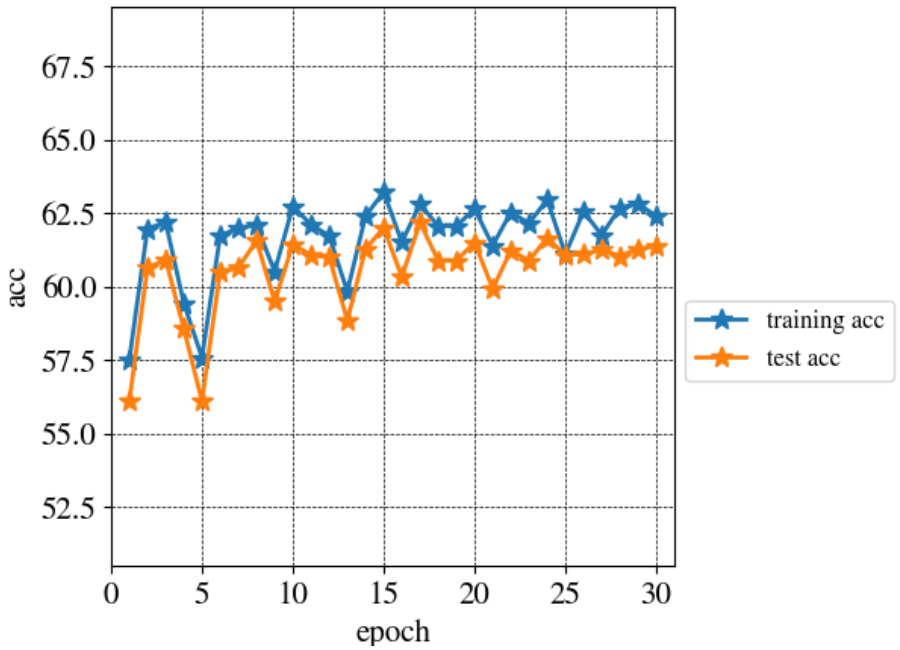


Figure 10: Training and validation accuracy versus the number of epochs for the residual neural network.

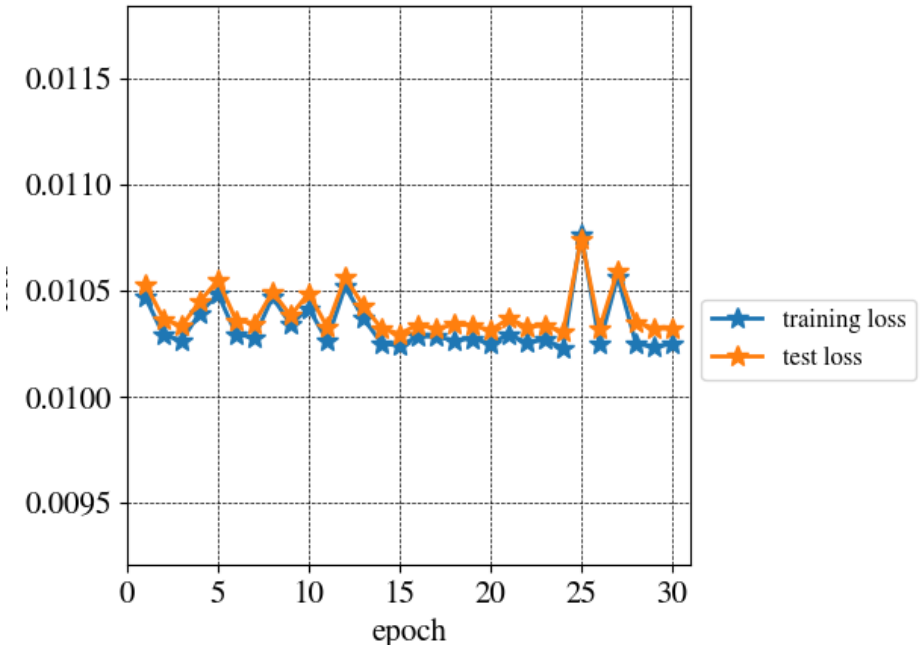


Figure 11: Training and validation loss versus the number of epochs for the residual neural network.