Towards Automated Fact-Checking with Methods from Information Retrieval, Data Mining and Machine Learning

Anonymous (Student Number: 18143060)

ABSTRACT

The ever-increasing spread of misinformation in the internet and other media outlets is believed to have had a significant impact on various important areas of life, such as distorting elections, votes, and the stock market. Accordingly, there is increased interest in developing automated fact-checking (AFC) systems, which help keep the spread of such information at bay and support human fact-checkers in their work. While research in this area is very active and has yielded several methods and implementations, there are many open questions still. The NP-hard problem of misinformation containment remains challenging to solve without human intervention.

In this study, the core components of an AFC system are discussed and implemented. To this end, the FEVER (Fact Extraction and VERification) dataset, a large-scale collection of manually fact-checked claims, is utilised. First, relevant documents w.r.t. a given claim are retrieved. Next, relevant sections in these documents are identified. Finally, a conclusion about the claim's veracity is drawn. To implement all these steps, methods from information retrieval, data mining and machine learning are applied.

1 INTRODUCTION

In recent years, the internet's ubiquity has greatly enhanced the amount of available information and changed the way information is published and consumed. This enhanced ability of reaching audiences can be used to spread both true and false information. Recent studies have shown that false information reaches greater audiences [65], and there is strong evidence that it has significantly influenced public votes, presidential elections, the stock market and other important areas of life [45, 59].

These developments have led to increased demand for fact-checking, and given the quantity of data, the only feasible approach is to automate the process as much as possible. Journalists, policy-makers, and technology companies alike have an interest in finding effective, large-scale responses to misinformation. Meanwhile, the scientific community is constantly exploring ways to satisfy this increased demand and conquer the inherent challenges [14].

In this study, the core components of a system for AFC are discussed and implemented. To that end, the FEVER (Fact Extraction and VERification) dataset, a large-scale collection of manually fact-checked claims, was used.

The remainder of the report is organised as follows: Section 2 summarises related work. Section 3 introduces the document corpus and analyses latent text statistics. Section 4 presents the problem statement and baselines. Next, the three main aspects of the fact-checking system are discussed: Section 5 covers relevant document retrieval, Section 6 deals with the selection of relevant evidence sentences, and Section 7 addresses the prediction of claim veracity. Finally, Section 8 concludes the report.

2 RELATED WORK

2.1 Determining Veracity in Big Data

Pendyala [45] gives a thorough overview of aspects and methods of determining the truth of large-scale datasets. Many factors play into veracity, including missing values, uncertainty, falsity, and bias. Pendyala shows that there is sufficient evidence that the problem of misinformation containment is NP-hard. Consequently, approximate methods are used, including machine learning (ML), change detection, optimisation techniques, natural language processing (NLP), formal methods, fuzzy logic, information retrieval (IR) techniques, collaborative filtering, and the blockchain.

Change detection techniques build on the idea that untruthfulness is a change from the norm, and can therefore be considered noise in an otherwise truthful environment. Hence, models like CUSUM or the Kalman Filter capture the representative state of a time series and detect deviations.

Learning to distinguish truth from lies can also be regarded as a ML problem, where algorithms are either supervised, using labeled training data, or unsupervised. An essential element of such models is parameter estimation: the model parameters need to be optimised in terms of algorithmic accuracy. The eventual goal is to partition the solution space into two classes, truthful and not truthful. Commonly used ML techniques include logistic regression (LR), neural networks (NN), support vector machines, and vector space similarity.

Lastly, Pendyala discusses blockchain as a promising technology, as it enables secure transactions in a trustless environment. This may also improve the problem of veracity in big data to a great extent.

2.2 Automated Fact-Checking

Fact-checking is the task of assessing the truthfulness of a claim. With the increased focus on misinformation, research in this area has recently been very active. One focus of this research is automation, which is located in the intersection of NLP, ML, knowledge representation, databases, and journalism [59]. End-to-end AFC systems aim to address three elements: identification (e.g. identifying factual statements), verification (e.g. checking against authoritative sources), and correction (e.g. providing contextual data). Recently, several programming competitions inspired research progress and real-world AFC initiatives received a wave of funding [14]. However, while the first proposals to automate online fact-checking surfaced nearly a decade ago, the task remains challenging, as the conclusions reached by professional fact-checking organisations often require the ability to understand context, exercise judgement, and synthesise evidence from multiple sources. There is a widespread view among researchers that, for the time being, expectations should remain modest and it will take AFC systems another 5–10 years to mimic the advanced things human fact-checkers are

able to do. Hence, the most valuable immediate impact of this research is automating some of the simpler tasks, thereby helping human fact-checkers to work more efficiently [14].

There are different lines of research. One approach to AFC, as has been proposed by Vlachos and Riedel [64] and implemented by multiple real-world AFC systems, is matching statements to previous fact-checks, which reduces the task to sentence-level textual similarity. While this is the most effective approach to date, it shifts the responsibility to human-made results, and is limited to repeated or paraphrased claims that have been verified before. In addition, it often requires additional world knowledge, typically not provided alongside the claim itself [59]. Also, Graves [14] argues that this approach entails a trade-off between recall and precision, and what's more, even subtle changes in the wording, timing, or context of a claim can make it ambiguous [14].

Some authors analyse whether the language used is indicative of factual texts or subjective statements [39]. While this can be a helpful insight, it is not sufficient, as credible sounding sources might still contain untrue information [59].

Another approach includes checking claims against an authoritative source. This requires AFC systems to recognise the required evidence and retrieve it in a machine-readable format. Again, some uncertainty remains as even reliable sources make mistakes [14].

Some work uses speaker profiling. This has been shown to be effective, but the solutions are tightly coupled to the specific domains they were developed for and often have some important ethical implications [59].

Other approaches exploit network structures, e.g. using knowledge graphs [4]. These approaches are limited as it is not feasible to capture and store every conceivable fact in the graph in advance of knowing the claim [59]. Similar work analyses social cues such as Facebook likes [58] or replies to articles [70]. These approaches are typically tailored tightly to fit the specific platforms used in the respective studies.

The majority of all methods make use of supervised ML, i.e. they obtain a text classifier from some labeled training data. Many different neural architectures and variations thereof have been used, most importantly multilayer perceptrons (MLP) [30], convolutional neural networks (CNN) [12, 24, 30, 53], recurrent neural networks (RNN) [54], and Long short-term memory (LSTM) networks [13, 20, 53, 70].

2.3 Related Fields

There are several academic fields closely related to AFC, including fake news detection (which many researchers do not even distinguish from fact-checking), stance detection, sentiment analysis [42], verification, common sense reasoning, subjectivity and emotive language, deceptive language detection, rumour detection, speaker profiling, and click bait detection [59]. Another closely related field is recognising textual entailment (RTE), also referred to as natural language inference (NLI), which is the task of recognising relationships between sentences. It determines whether a given text (the premise) is for, against, or observing a second text (the hypothesis) [6]. RTE has been employed in AFC systems to great success. However, it operates under the assumption that the textual evidence to check a claim is given. Thus it is inapplicable in cases where such is not provided, or at least needs to be accompanied by other components which first identify the relevant evidence [59, 60].

3 DOCUMENT CORPUS STATISTICS

The ground truth for this study is the June 2017 English Wikipedia dump that is part of the FEVER dataset. More specifically, 5,416,536 pages (7.09 GB) are considered. Each entry is indexed by the article title and contains the introductory section as a whole, as single lines, and auxiliary information about which other pages are referenced.

To gain a better understanding of the corpus, the documents were pre-processed and analysed. First, documents with a text of less than 20 characters in length were permanently filtered out, as many documents were either completely empty or contained no useful information. After this step, 5,391,645 documents remained.

These documents were then further processed for the text statistics analysis: Some inherent artifacts of the FEVER data extraction were removed, the text was transformed to lowercase and split into tokens with a simple tokenising mechanism. Lastly, 179 common stop words[1] were removed to mitigate later steps (see Appendix A. No stemming or further processing was applied. After these steps, the corpus consisted of 271,036,237 total words, utilising a vocabulary of 2,697,407 unique terms.

A commonly used model to describe the distribution of terms across documents is Zipf's law, which states that the product of a word's rank r and its frequency f is approximately a constant c [32]. One possible approach often used to verify Zipf's law on a corpus is plotting the data points in a log-log scaled plot and visually confirming the distribution. To increase the confidence in the result, I additionally fitted a straight line through the points using weighted least-squares fit. This resulted in the function $f_i = 9.12 - 1.44 \cdot r_i$ (in logarithmic space). The R^2-value is reasonably high (0.98), making the line a good fit to describe the data (see Figure 1). However, this approach is often criticised for various reasons [5, 62, 63]. Therefore, a second method was applied in addition: Assuming that $r_i \cdot P_i \approx c$, where P_i is the probability of word occurrence, one can compute the product for each term and obtain the mean. For the given corpus, this results in $c \approx 0.012$ with a standard deviation of $\sigma = 0.012$.

4 PROBLEM STATEMENT AND BASELINES

This study makes use of the FEVER dataset, a recent large-scale collection of 185,445 textual claims that have been manually verified against the introductory sections of Wikipedia pages and classified as Supported, Refuted or NotEnoughInfo. [60], where the term "claim" implies short statements in natural language (mean length of 9.4 tokens). Verifiable claims, i.e. either Supported or Refuted, contain references to one or more pieces of evidence in the Wikipedia dump. In some cases, evidence from multiple pages has to be considered to obtain a sufficient argument.

The claims are partitioned into a training subset of 145,449 and development and test subsets of 19,998 items each. To mitigate the computational requirements, some parts of this study only consider a small proportion of each subset (see Table 1). FEVER differs from most comparable datasets in that the evidence is not given, but must be retrieved from the large document corpus [59]. When confronted with a claim, classification systems are supposed to find suitable

[1] https://gist.github.com/sebleier/554280

evidence at sentence level, and then reason correctly with respect to the claim. This study aims to develop the key components of such a system.

In their baseline system, the dataset's authors achieved to predict accurate labels for 50.91% correctly, indicating that the task of verifying a textual claim against a large textual corpus is challenging, but feasible. In a subsequent shared task, they challenged participants to implement their own fact-checking systems [61]. The task received entries from 23 competing teams with the best performing system achieving 68.21% label accuracy. Most submissions implemented a three-fold pipeline structure, breaking the problem down into the sub-problems document selection, sentence selection and natural language inference. The study at hand will follow the same general approach.

5 RELEVANT DOCUMENT RETRIEVAL

5.1 Objective

Retrieving relevant documents is the first of three steps towards automatically classifying a given claim as true or false w.r.t. a given ground truth. The system has to parse the claim and retrieve a small number of relevant documents from the complete corpus.

5.2 Methodology

5.2.1 Inverted Index. Given the significant size of the document corpus and the relatively expensive computations to be carried out for every potential candidate for each claim, it was essential to initially construct a converted index and pre-compute as many values as possible. This issue is further elaborated in Appendix A.

5.2.2 Vector Space Retrieval. In order to assign a score to measure the match between a claim c and a document d, a weight gets assigned to each term, taking into account its term frequency tf as well as its inverse document frequency idf. By then multiplying tf and idf, I obtained a measure for how discerning a term is w.r.t. its commonness in the whole collection. To retrieve documents for a claim, I considered the union of all candidate documents, i.e. documents that are included in the inverted index entries for one or more claim terms. The claim and each candidate document were then represented as vectors, with the corresponding elements for each term given by the tf-idf value. This is a "bag of word" (BOW) model, as the exact positions of words does not get preserved. Finally, the similarity between d and c (or any two documents) can be computed as the dot product: $sim(d_1, d_2) = \vec{d_1} \cdot \vec{d_2}$ [32]. Ranked by this score, the $k = 5$ most relevant documents were retrieved.

5.2.3 Probabilistic Document Retrieval. Furthermore, a second approach for retrieving documents was implemented: The intuition behind language modelling in IR is asking how likely the language used in a given document is to generate a second document—e.g. a claim, or more general, a query. I implemented a Query Likelihood Model (QLM), which is the most commonly used of various different language modelling approaches. It builds a probabilistic model M_d for each document d, and then, using a maximum likelihood estimation (MLE), ranks documents based on the probability of the model generating the query: $P(q|M_d)$ [32]. Again, only candidate documents according to the inverted index were considered.

In this study, I used the simplest version of this model, a unigram model, which is insensitive to context and estimates the probability for each term independently.

5.2.4 Smoothing. The classic problem of probabilistic models is that some words can be absent from the document, but included in the information need expressed in a query or claim. However, a basic QLM entails strict conjunctive semantics, i.e. documents will only ascribe a query non-zero probability if all of the query terms appear in the document [32]. To mitigate this problem, I applied smoothing, i.e. all terms absent from a document but part of the vocabulary got assigned a small, non-zero probability of occurrence. At the same time, I decreased the estimated probability of seen events accordingly. First, I implemented Laplace Smoothing, where a number α is added to the count of each seen and unseen word. Next, two variations of smoothing with background were implemented; in both cases, the probability estimate for a word present in the document combines a discounted MLE and a fraction of the estimate of its prevalence in the whole collection, while for words not present in a document, the estimate is completely based on the prevalence of the word in the whole collection [32]. While Jelinek-Mercer (JM) smoothing uses an optimised, but constant value λ for the proportions, Dirichlet smoothing takes document length into account, and adaptively gives the background probability less weight for longer documents.

5.3 Exploiting Structural Features

The approach described above considers the given dataset as unstructured. However, the documents should rather be regarded as semi-structured, as they include some additional structure and information. Namely, the IDs of Wikipedia articles are equivalent to their titles, offering concise information on the subject. Furthermore, the documents' lines are accompanied by links to other articles that are referenced. This is not uncommon; in fact, most digital documents have such additional structure encodings in their metadata [32]. Such additional data have been shown to improve document ranking w.r.t a query as well as general retrieval effectiveness. However, while different field-based models have been developed, there is no formal justification of field weighting [23, 69].

It is striking how closely related the article titles often are to the corresponding claims in the FEVER dataset, even on a purely lexical level. E.g., for the claim "Nikolaj Coster-Waldau worked with the Fox Broadcasting Company", the relevant documents are `Nikolaj Coster-Waldau` and `Fox Broadcasting Company`. To leverage this correlation, I implemented a variation of the retrieval techniques described above, where the score of a document d is the weighted mean of the score of the text and the title individually:

$$score'(d) = \lambda \cdot score(d_{\text{text}}) + (1 - \lambda) \cdot score(d_{\text{title}}) \qquad (1)$$

5.4 Results and Discussion

All techniques described above were used to retrieve documents for the first ten verifiable training claims. While the results should be taken with a grain of salt due to the small sample size, Table 2 shows that the best recall was obtained by the QLM with Laplace smoothing and a Lindstone correction (i.e. instead of adding 1, adding $\alpha = 0.1$), which included 9 out of the 12 expected evidence

documents. The QLM variations with no smoothing or basic Laplace smoothing also performed surprisingly well. However, this is a lucky chance as the sample contained multiple claims that were reproducing the corresponding evidence sentences almost word-by-word, which does not hold for the complete dataset, neither for a real-life AFC system. The tf-idf retrieval performed worst, which confirms other studies' findings that also found the significant superiority of probabilistic methods over tf-idf [32, 48].

All models failed to retrieve documents which exclusively included synonyms of important claim terms. E.g., for the relevant document to be retrieved for the claim "Roman Atwood is a content creator", the system would have to translate that to "YouTube personality, comedian, vlogger and pranker".

A further limitation is that while 18% of the claims require combining evidence from multiple articles, this is often impossible with my approach. It can work out if both entities are explicitly named in the claim, as in the example referenced in Section 5.3, and indeed works out for all claims in the small subset. But there are counter-examples, e.g. for verifying the claim "Deepika Padukone has been in at least one Indian film", one has to combine "She starred roles in Yeh Jawaani Hai Deewani" and "Yeh Jawaani Hai Deewani is an Indian film". Using just the claim to retrieve documents, as I did, makes it impossible to retrieve the second document in that example (see also [28]).

The inclusion of the title has not fulfilled the expectations. Out of four tested weightings ($\lambda = 0/0.25/0.5/0.75$), none was capable of finding more than 33% of the evidence. However, it has thus far only been tested with the tf-idf retrieval. Further research may determine the optimal value for λ. Despite the bad result, utilising the titles holds great potential, when combined with advanced NLP concepts: Many teams in the original FEVER challenge successfully utilised some combination of named entities, noun phrases and capitalised expressions from the claims [61].

With all the pre-computations and optimisations in place (see Appendix A), the retrieval run-times were sustainable. However, they strongly depend on the number of candidate documents for a claim. To demonstrate the scalability of the system, documents for the first 10,000 training and development claims were retrieved, using the QLM with Laplace Smoothing ($\alpha = 0.1$), as it had performed best on the small sample.

6 EVIDENCE SENTENCE SELECTION

6.1 Objective

The second part of the pipeline receives the claim c and the five documents $\{d_1, ..., d_5\}$ retrieved in the previous step, and is then responsible for determining which lines are relevant, i.e. are part of the corresponding evidence to support or refute the claim.

6.2 Methodology

6.2.1 Logistic Regression. As in Section 5, I focused on the first ten verifiable claims. This also implied that for all considered claims, there was at least one relevant sentence. To discern relevant sentences from irrelevant sentences, I implemented a LR. LR is a commonly implemented supervised learning algorithm, and lent itself to this type of classification problem: Claim-sentence pairs were either part of the claim's evidence or not. Given the labeled training data, the LR tried to classify them. A cross-entropy cost function then determined the difference between the predicted label and the actual label and consequently caused the weights to be adjusted through gradient descent (GD) such that the cost was minimised.

Each of the five candidate documents d was decomposed into its lines $\{l_1 ... l_n\}$, and each pair (c, l) was supplied as input to the LR. The corresponding output was zero, unless this particular sentence belonged to the claims evidence set. There were two problems: The amount of irrelevant input was orders of magnitude higher than its counterpart, and depending on the performance of the document retrieval, the correct evidence might not even have been among them. To de-couple this sub-task from the first step, I therefore enhanced the set of candidate documents with any missing evidence documents. While this guaranteed that at least some positive samples were part of the training, it effectively aggravated the first problem, as the total number of sentences per claim increased. The training set consisted of 7,351 lines, out of which 15 were relevant (0.2%) and the development set of 14,839 lines with 10 being relevant (0.07%).

6.2.2 Gradient Descent. In the training phase, a batch GD was implemented, i.e. the entirety of 7,351 pairs was considered at every step. While being more computationally expensive and taking longer to converge, this approach has the benefit of being more precise, as every parameter update is influenced by more samples [40].

6.2.3 Word Embeddings. In order to represent natural language in a machine-readable way, I had to express words as numbers. For the previous sub-task, I represented documents as data points in a $|V|$-dimensional vector space, i.e. every term of the vocabulary resulted in one dimension, and respectively each word could be expressed as a one-hot vector. However, as discussed in Section 5.4, this model performed badly at capturing semantic meaning, e.g. synonyms. In this sub-task, I used low-dimensional word embeddings instead, as they are capable of capturing fine-grained semantic and syntactic patterns. The initial word2vec method based on skip-gram with negative sampling [34] had a large impact on the field and is still among the most used embeddings. However, many alternative training approaches have emerged and a rich supply of pre-trained vectors is available for download [38, 55]. One of the alternatives is GloVe, which utilises a regression model which combines the advantages of the two major model families in the literature: global matrix factorisation and local context window methods. It outperforms word2vec in time and accuracy [46] and has been embraced by many recent studies in the fields of IR and RTE [3, 41, 43, 66]. For the study at hand, I used the 300-dimensional model that was pre-trained on 840 billion tokens.

6.2.4 Out-of-Vocabulary Words. When using pre-trained word embeddings, tokens that were absent from the learning corpus but included in the documents to be encoded resulted in out-of-vocabulary (OOV) words. Interestingly, there is no consent in the literature: Different authors handle the issue heterogeneously, e.g. completely omitting unknown words [12], adding a dedicated unknown word token [55], assigning it a zero vector [49], assigning to one of a set of random embeddings [43], assigning it to a randomly initialised vector and subsequently optimise during training [52], taking the sum of the context words [17], or generating embeddings

on-the-fly [47]. For this study, I adhered to the suggestion of GloVe author Pennington[2] to take the average of all or a subset of the word vectors as the unknown vector.

6.2.5 Sentence Representation. Next, I needed a combined value, consolidating all words or a claim or document sentence. There are different approaches, and simple averaging [9, 11, 22] or summation [3] of word vectors have been shown to work reasonably well, especially for small units of text such as a query. De Boom et al. [9] proposed a more sophisticated method that also performs well for very short fragments of text. However, without adjustments, their method requires texts of fixed length. As their best performing baseline approach, Min/Max, performed only slightly worse, I utilised that instead: Considering all word vectors $\vec{v_1}, ..., \vec{v_n} \in \mathbb{R}^d$ of a sentence, the coordinate-wise minimum and maximum vectors are computed. These two vectors are concatenated, leading to a feature vector in $\mathbb{R}^{2d}$ space. Given the 300-dimensional word embeddings provided by GloVe, the Min/Max approach thus yields a 600-dimensional vector.

6.2.6 Feature Vector Generation. Lastly, I had to unite the vectors obtained from the claim and the document sentence, such that I obtained one input for the LR. As with the previous steps, there are different approaches to this task. Simply taking the difference of two sentence vectors has been shown to work reasonably well to capture the semantic relationship [44, 71]. Hence, the final feature vector representing the (c, l) pair is still in $\mathbb{R}^{2d}$ space.

6.3 Class Imbalance Correction

As mentioned above, the dataset was extremely skewed, with the minority class constituting less than 1% of the total population. It is well known that such an imbalance causes a major problem for the LR, as the low event rate exacerbates the bias. However, there is no commonly agreed solution among applied researchers [27, 29, 56]

King and Zeng proposed methods to correct the bias and collect data more efficiently [25]. The easiest method is conducting a case-control sampled LR, i.e. selecting the input on the dependent variable, which allows modifying the class distribution as needed. Upon running the LR, they correct the bias as follows:

$$\beta_0 = \hat{\beta}_0 - \ln\left[\left(\frac{1-\tau}{\tau}\right)\left(\frac{\bar{y}}{1-\bar{y}}\right)\right] \tag{2}$$

where τ is the fraction of events in the whole population and $\bar{y}$ is the fraction of events in the sample. Their work has been mentioned frequently and appeals for its ease of use. However, other authors have commented that the approach might not be ideal and potentially over-correct the bias [27, 68].

Allison [1] argues that the problem is not specifically the rarity of events, but rather the possibility of a small number of cases on the rarer of the two outcomes. This objection is in line with the "one in ten" rule of thumb, which states that one predictive variable can be studied for every ten events, where the number of events is the size of the smallest of the outcome categories [16]. Accordingly, to predict the 600 dimensions of the input vector, 6,000 events would be needed, while the training data contained merely 15.

[2] https://groups.google.com/forum/#!searchin/globalvectors/unk/globalvectors/9w8ZADXJclA/X6f0FgxUnMgJ

As a result of these considerations, an alternative solution was developed. The complete training set was considered, and all events, i.e. 8,884 relevant (c, l) pairs, were extracted. For each pair, an irrelevant pair was added, where the sentence was sampled from the same document if possible, or a random document otherwise. This procedure generated an evenly balanced input of 17,768 pairs. The complete development dataset was pre-processed identically. The trained classifier was then evaluated against imbalanced data, balanced data, and bias-corrected data.

6.3.1 Evaluation Metrics. To analyse the model, I implemented four commonly used evaluation metrics: accuracy, precision, recall, and F1-score. The respective formulae are well-known, they can e.g. be found in [36]. Additionally, I used an existing library to visualise ROC curves and Precision-Recall curves.

6.4 Results and Discussion

To optimise the classifier, different combinations of parameters were empirically tested and evaluated. Regardless of other parameters, $n = 100,000$ iterations achieved the best results. I analysed the effect of different learning rates; as it is often recommended [50], I focused on powers of ten (0.001, 0.01, 0.1, 1.0). As Figure 2 shows, the cost decreased in every iteration, as expected. Also, smaller learning rates significantly slowed the gradient descent, and for $\alpha \leq 0.01$, failed to adequately minimise the cost function. For $\alpha = 1$, the cost plummeted very quickly; however, this model performed very badly when it was evaluated against unseen data, i.e. it failed to pick up the underlying patterns it was supposed to discover. Also, it is known that larger learning rates can cause the algorithm to fail to converge, or even diverge [50].

Next, the classifier was evaluated with the development set, to see how well it performed with unseen data. As Table 3 shows, the LR achieved very high accuracy, and higher learning rates correlate with better accuracy. However, this metric is deceptive for imbalanced data sets, as it rewards correctly predicted non-events on par with correctly predicted events. In other words: Given the skewed training data, the model is prone to just predict all input as being irrelevant—even more so with the course-grained $LR_{\alpha=1.0}$ model. As the development set also contains 99.93% irrelevant sentences, the model is correct most of the time. However, in a classification task like the one at hand, it is of much higher interest to correctly classify the ones than the zeros.

This problem of imbalanced datasets is well known. The evaluation measures precision, recall and F1 are therefore typically used to complement accuracy, as they do not take true negatives into account; they are only concerned with the correct prediction of the minority class. [31, 70]. Indeed, while both $LR_{\alpha=1.0}$ and a base rate model which always predicts zero (B_{zeros}) achieve extremely high accuracy, they score zero for precision, recall and F1, as they fail to identify a single events. Instead, when ranking by F1 score, $LR_{\alpha=0.1}$ is clearly superior, outperforming all other models by an order of magnitude.

The Receiver Operating Characteristic (ROC) curve, which plots the true positive rate (or sensitivity) as a function of the false positive rate (or specificity) for different decision thresholds, also suffers from the imbalance problem, presenting an overly optimistic view of an algorithm's performance (see Figure 3). Therefore, to account

for the class imbalance, Precision-Recall (PR) curves are often used, especially in IR [7, 57]. Figure 4 demonstrates that $LR_{\alpha=0.1}$, while achieving fairly low precision, still generates a curve that is permanently above a base rate model which randomly predicts zero or one (B_{random}). Hence, the model in fact succeeded to pick up some patterns that define relevant (c, l) pairs despite the small and skewed training set.

After implementing the improvements discussed in Section 6.3 and re-training the model (only with $\alpha = 0.1$), the ensuing results were analysed and are summarised in Table 4. First, evaluation was conducted against the same development set as before, i.e. with the imbalanced original distribution. Even though the validation set and the training set came from different distributions class ratios, this model shows a significant improvement of another order of magnitude compared to $LR_{\alpha=0.1}$. Next, the bias correction was applied. This, unfortunately, deteriorated the results and was therefore found to be the wrong approach for the given dataset. Lastly, the model was tested against the balanced projection of the development set. This model performed best, achieving an F1 score of 0.66. The PR curve looks quite different too, as the baseline is now lifted to 0.5 and the difference between the base rate model and the LR is even more obvious (see Figure 5). However, it provides the least real-world value, as new unseen data usually can not be selected on the dependent variable. In general, some authors criticise this approach of undersampling the majority class to get an artificial 50/50 balance, as such classifiers might not apply to a population with a much different prevalence of events [15].

7 CLAIM VERACITY PREDICTION

7.1 Objective

The third and final part of the study is concerned with discriminating SUPPORTED from REFUTED claims. I made the assumption that all accurate evidence is known and also discarded non-verifiable claims, rendering it a two-way-classification problem.

7.2 Methodology

7.2.1 Neural Network Architecture. The overview of related work in Section 2.2 showed that there are many different neural architectures that have successfully been applied. Most of them are out of the scope of this study to implement or improve upon. However, the most basic approach, a simple MLP, has often been shown to achieve remarkably good results [49].

For their baseline system, the FEVER authors compare two RTE models [60]: They implement a simple but well-performing MLP with a single hidden layer, which uses term frequencies and tf-idf cosine similarity between the claim and evidence as features [51]. In addition, they use a decomposable attention (DA) model [43], as it was, at the time of writing, the highest scoring publicly available system for the Stanford Natural Language Inference (SNLI) task [3]. This model has since been superseded by other publicly available neural architectures, such as the bilateral multi-perspective matching (BiMPM) model. Given two sentences, this model first encodes them with a Bidirectional LSTM (BiLSTM) encoder. Next, the two encoded sentences are matched in two directions. In each direction, each time step of one sentence is matched against all time steps of the other from multiple perspectives. Then, another BiLSTM layer is utilised to aggregate the matching results into a fixed-length matching vector. Finally, based on the matching vector, a decision is made through a fully connected layer [67].

For this study, I first implemented a simple MLP, which I optimised in terms of the hyperparameters and the input format. Additionally, I evaluated how the BiMPM model, which represents the new state-of-the-art in RTE, performed with the FEVER dataset.

7.2.2 Input Representation. The representation of textual data at the input layer is a key consideration when designing neural networks. There are several common solutions: Single characters can be represented as one-hot vectors, which can then be concatenated or summed to generate longer text. Alternatively, whole terms can have a one-hot or a local representation. Often, pre-trained embeddings are used, which can optionally be further tuned during the training phase [36].

In this study, I further explored the feature vectors developed in Section 6.2.6, i.e. combining GloVe word embeddings to sentences using the Min/Max technique and then capturing the relationship between two texts by subtracting the respective vectors. The complete training and development sets were processed.

As a first step, the pairs were assembled as in the previous subtask, i.e. one (c, l) pair for each relevant line. Then, another representation was obtained, in which all relevant sentences for a claim were concatenated to one string, which was then encoded to a feature vector as described above. Such a united representation was also used in the FEVER baseline system [60] as well as by the highest scoring team in the FEVER shared task [61]. Intuitively, it seems necessary, as a human fact-checker would also require all pieces of the evidence to make an educated decision on the veracity of a claim.

7.2.3 Advanced Feature Engineering. Feature engineering, i.e. the process of selecting and designing good features for ML with the help of domain knowledge, is one of the most important factors for effective ML models [10, 33]. In the ML discipline Learning to Rank (LTR), which concerns itself with ranking documents for IR, features are often hand-crafted and exploit insights gained through analysis. The features can be categorised as query-independent, query-dependent and query-level features, and shift the focus away from the purely textual content. The resulting parameter space is then referred to as bag-of-features (BOF) rather than BOW [21, 35]. Previous work has shown that adding lexicalised and unlexicalised features can significantly improve a classifier's accuracy [2, 3, 8]. In the spirit of these findings, I crafted seven additional features, including the number of references to other articles in an evidence sentence, the position of the sentence in the document, and the number of overlapping terms between the sentence and the claim as well as between document title and claim. For the variation employing concatenated sentences, the additional features were adjusted accordingly (e.g. mean of the considered sentences' positions). The additional features where then concatenated to the existing feature vector, increasing the dimensionality to 607.

7.2.4 Hyperparameter Tuning. The size of the input layer was given by the dimensionality of the input vectors and the output size two, as the network was performing a binary classification. The

other hyperparameters of the MLP were determined empirically. Using 20 epochs resulted in the best overarching performance in initial experiments, so for better comparability, all models were trained for 20 epochs. To decrease the computational cost, mini-batches of size 100 were used. Initially, the network contained one hidden layer of 100 neurons, used a Sigmoid activation function and implemented Stochastic GD (SGD) optimisation. The following changes were then made: The Sigmoid non-linearity was replaced by ReLu, as it avoids and rectifies the vanishing gradient problem and is less computationally expensive [19]. The SGD was replaced by the Adam algorithm, as it is well suited for problems that involve lots of data or many parameters while being computationally efficient and known to outperform other optimisation methods [26, 50]. The hidden dimension was then scaled up to 500, as previous work has shown that the output can be made more accurate by introducing more neurons between the input and output layer [45]. Lastly, the number of hidden layers was increased to three, as deeper networks can split the input space into more non-independent linear regions than shallow networks [36, 37]. Each configuration was evaluated with four different variations of the input format: input_1 denotes the feature vectors as they were used for the LR; input_2 denotes the variation where all evidence sentences for a claim were concatenated; input_3 denotes the original representation, but enhanced with the additional features described in Section 7.2.3; finally, input_4 denotes the concatenated variation with the additional features.

7.3 Evaluation of a State-of-the-Art RTE Model

To establish how well the state-of-the-art RTE model BiMPM classifies the FEVER claims, I obtained the original implementation by the paper's authors. As a small error rendered the code non-operational, I forked it and fixed the issue[3]. Next, I transformed the training and development datasets into the expected format[4] and trained the model for 20 epochs.

7.4 Results and Discussion

Unlike in Section 6, the datasets for this sub-task were fairly balanced: After discarding NotEnoughInfo, 80,035 of the 109,810 training claims (72.88%) and 6,666 of the 13,332 development claims (50%) are Supported. Therefore, the network performance was primarily optimised for accuracy on the development dataset.

As a baseline, the best performing LR model from the previous sub-task was trained for this task. The accuracy on the development set was no better than random guessing (0.50677). The MLP performance varied strongly depending on the configuration (Table 5 lists the results for all combinations). The initial settings performed poorly, whether using Sigmoid or ReLu activation. A significant improvement of more than 15% was made by utilising the Adam optimisation. The increase of the hidden dimension caused another small improvement, while the addition of more hidden layers failed to increase the accuracy. The improvement of concatenated sentences over separate ones and the addition of features was mixed; overall, they seemed to slightly improve the results. Also, they were both applied in the combination which performed best: The network with one hidden layer, 500 neurons and the concatenated, enhanced feature vectors achieved 69.6% accuracy. Figure 6 illustrates how the cross-entropy loss decreased for this particular combination. It is also apparent that the curve is noisier, not as smooth as the LR curve, as mini-batching was applied.

While this result is surprisingly good for such a simple model, the BiMPM easily outperformed it, achieving an 83.64% accuracy rate on the development set. Both models superseded their equivalents in the FEVER baseline, in which the MLP achieved less than 65% and the DA less than 80% accuracy [60]. This result underlines the importance of interdisciplinary exchange between AFC and RTE, while also demonstrating that simple neural architectures with carefully selected features can be a worthy alternative.

8 CONCLUSION

The ever-increasing spread of misinformation in the internet and other media outlets is believed to have had a significant impact on various important areas of life, leading to an increased interest in AFC systems to keep the spread of such information at bay and support human fact-checkers.

While methods from IR, DM and ML have been applied with some success, this study operated under many assumptions, mostly to de-couple the sub-tasks and address document retrieval, evidence selection and veracity prediction independently. A real-world AFC system has to succeed at all these problems, as the outcome of one component influences the next, and the system will only perform as well as its weakest component. The results of this study reiterate that implementing satisfying solutions for AFC is challenging but feasible, and existing state-of-the-art approaches of RTE and neighbouring research fields can be employed to great effect.

REFERENCES

[1] Paul Allison. 2012. Logistic Regression for Rare Events. https://statisticalhorizons.com/logistic-regression-for-rare-events. (02 2012).

[2] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python* (1st ed.). O'Reilly Media, Inc.

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 632–642.

[4] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M. Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational Fact Checking from Knowledge Networks. *PLOS ONE* 10, 6 (06 2015), 1–13.

[5] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review* 51, 4 (2009), 661–703.

[6] Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing Textual Entailment: Rational, Evaluation And Approaches. *Journal of Natural Language Engineering* 4 (January 2010). https://www.microsoft.com/en-us/research/publication/recognizing-textual-entailment-rational-evaluation-and-approaches/

[7] Jesse Davis and Mark Goadrich. 2006. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*. ACM, 233–240.

[8] Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. 2017. What is the Essence of a Claim? Cross-Domain Claim Identification. (04 2017).

[9] Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *arXiv e-prints* (Jul 2016), arXiv:1607.00570.

[10] Pedro Domingos. 2012. A Few Useful Things to Know About Machine Learning. *Commun. ACM* 55, 10 (Oct. 2012), 78–87.

[11] Nadbor Drozd. 2016. Text Classification With Word2Vec. http://nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/. (05 2016).

[12] Marek Galovic. 2017. Convolutional Attention Model for Natural Language Inference. https://towardsdatascience.com/convolutional-attention-model-for-natural-language-inference. (06 2017).

[3] <Link removed due to anonymisation.>

[4] <Link removed due to anonymisation.>

[13] Souvick Ghosh and Chirag Shah. 2018. Towards automatic fake news classification. *Proceedings of the Association for Information Science and Technology* 55 (01 2018), 805–807.

[14] Lucas Graves. 2018. Understanding the Promise and Limits of Automated Fact-Checking.

[15] Frank Harrell. 2017. Damage Caused by Classification Accuracy and Other Discontinuous Improper Accuracy Scoring Rules. http://www.fharrell.com/post/class-damage/. (03 2017).

[16] Frank E. Harrell Jr., Kerry L. Lee, Robert M. Califf, David B. Pryor, and Robert A. Rosati. 1984. Regression modelling strategies for improved prognostic prediction. *Statistics in Medicine* 3, 2 (1984), 143–152.

[17] Aurelie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. *arXiv e-prints* (Jul 2017), arXiv:1707.06556.

[18] Paul Hsieh. 2014. Hash functions. http://www.azillionmonkeys.com/qed/hash.html. (2014).

[19] Rohan Kapur. 2016. The vanishing gradient problem. https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b. (03 2016).

[20] Georgi Karadzhov, Preslav Nakov, Lluis Marquez, Alberto Barron-Cedeno, and Ivan Koychev. 2017. Fully Automated Fact Checking Using External Sources. *arXiv e-prints* (Oct 2017), arXiv:1710.00341.

[21] Tom Kenter, Alexey Borisov, Christophe Van Gysel, Mostafa Dehghani, Maarten de Rijke, and Bhaskar Mitra. 2017. Neural Networks for Information Retrieval 2017. In *SIGIR 2017*. ACM, 1403–1406.

[22] Tom Kenter, Alexey Borisov, Christophe Van Gysel, Mostafa Dehghani, Maarten de Rijke, and Bhaskar Mitra. 2018. Neural Networks for Information Retrieval 2018. In *WSDM 2018*. ACM.

[23] Jin Young Kim and W. Bruce Croft. 2012. A Field Relevance Model for Structured Document Retrieval. In *Proceedings of the 34th ECIR (ECIR'12)*. Springer-Verlag, 97–108.

[24] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1746–1751.

[25] Gary King and Langche Zeng. 2001. Logistic Regression in Rare Events Data. *Political Analysis* 9 (2001), 137–163. Issue 2.

[26] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv e-prints* (Dec 2014), arXiv:1412.6980.

[27] Heinz LeitgÃűb. 2013. The Problem of Rare Events in ML-based Logistic Regression. https://www.europeansurveyresearch.org/conf/uploads/494/678/167/PresentationLeitg_b.pdf. (2013).

[28] Sizhen Li, Shuai Zhao, Bo Cheng, and Hao Yang. 2018. An End-to-End Multi-task Learning Model for Fact Checking. In *Proceedings of the First Workshop on FEVER*. Association for Computational Linguistics, 138–144.

[29] Yazhe Li, Niall Adams, and Tony Bellotti. 2017. Issues using Logistic Regression for Highly Imbalanced Data. https://crc.business-school.ed.ac.uk/wp-content/uploads/sites/55/2018/01/28-Yazhe_Li.pdf. (08 2017).

[30] Wenjun Liao. 2018. *Stance Detection in Fake News: An Approach based on Deep Ensemble Learning*. Ph.D. Dissertation.

[31] Aldo Lipani, Mihai Lupu, Evangelos Kanoulas, and Allan Hanbury. 2016. The Solitude of Relevant Documents in the Pool. In *Proceedings of the 25th ACM CIKM (CIKM '16)*. ACM, 1989–1992.

[32] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[33] Donald Metzler and W. Bruce Croft. 2007. Linear Feature-based Models for Information Retrieval. *Inf. Retr.* 10, 3 (June 2007), 257–274.

[34] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at ICLR* (01 2013).

[35] Bhaskar Mitra. 2017. Neural Models for Information Retrieval. https://de.slideshare.net/BhaskarMitra3/neural-models-for-information-retrieval. (10 2017).

[36] Bhaskar Mitra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval. *Foundations and TrendsÂő in Information Retrieval* 13, 1 (2018), 1–126.

[37] Bhaskar Mitra, Nick Craswell, Emine Yilmaz, and Daniel Campos. 2019. Deep Learning for Search. https://de.slideshare.net/BhaskarMitra3/deep-learning-for-search. (01 2019).

[38] Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. 2017. Comparative study of word embedding methods in topic segmentation. *Procedia Computer Science* 112 (2017), 340 – 349.

[39] Ndapandula Nakashole and Tom M. Mitchell. 2014. Language-Aware Truth Assessment of Fact Candidates. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference* 1, 1009–1019.

[40] Andrew Ng. 2018. Supervised Learning. http://cs229.stanford.edu/notes/cs229-notes1.pdf. (2018).

[41] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten de Rijke, and Matthew Lease. 2018. Neural information retrieval: at the end of the early years. *Information Retrieval Journal* 21, 2 (Jun 2018), 111–182.

[42] Ray Oshikawa, Jing Qian, and William Yang Wang. 2018. A Survey on Natural Language Processing for Fake News Detection. *arXiv e-prints* (Nov 2018), arXiv:1811.00770.

[43] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 EMNLP*. Association for Computational Linguistics, 2249–2255.

[44] Fabian Pedregosa. 2012. Learning to rank with scikit-learn: the pairwise transform. http://fa.bianp.net/blog/2012/learning-to-rank-with-scikit-learn-the-pairwise-transform/. (10 2012).

[45] Vishnu Pendyala. 2018. *Veracity of Big Data: Machine Learning and Other Approaches to Verifying Truthfulness*. Apress.

[46] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[47] Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking Word Embeddings using Subword RNNs. *arXiv e-prints* (Jul 2017), arXiv:1707.06961.

[48] Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *Proceedings of the 21st Annual International ACM SIGIR (SIGIR '98)*. ACM, 275–281.

[49] Neel Rakholia. 2017. Is it true? Deep Learning for Stance Detection in News.

[50] Shashank Ramesh. 2018. A guide to an efficient way to build neural network architectures- Part I: Hyper-parameter selection and tuning for Dense Networks using Hyperas on Fashion-MNIST. https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures. (05 2018).

[51] Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *arXiv e-prints* (Jul 2017), arXiv:1707.03264.

[52] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. In *International Conference on Learning Representations (ICLR)*.

[53] Arjun Roy, Kingshuk Basak, Asif Ekbal, and Pushpak Bhattacharyya. 2018. A Deep Ensemble Framework for Fake News Detection and Classification. *arXiv e-prints* (Nov 2018), arXiv:1811.04670.

[54] Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. CSI: A Hybrid Deep Model for Fake News Detection. *arXiv e-prints* (Mar 2017), arXiv:1703.06959.

[55] Sebastian Ruder. 2017. Word embeddings in 2017: Trends and future directions. http://ruder.io/word-embeddings-2017/. (10 2017).

[56] Anne Ruiz and Nathalie Villa. 2008. Storms prediction : Logistic regression vs random forest for unbalanced data. *arXiv e-prints* (Apr 2008), arXiv:0804.0650.

[57] Takaya Saito and Marc Rehmsmeier. 2015. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. In *PloS one*.

[58] Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some Like it Hoax: Automated Fake News Detection in Social Networks. *arXiv e-prints* (Apr 2017), arXiv:1704.07506.

[59] James Thorne and Andreas Vlachos. 2018. Automated Fact Checking: Task Formulations, Methods and Future Directions. In *Proceedings of the 27th CICLing*. Association for Computational Linguistics, 3346–3359.

[60] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *NAACL-HLT*.

[61] James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The Fact Extraction and VERification (FEVER) Shared Task. In *Proceedings of the First Workshop on FEVER*.

[62] Carlos M. Urzua. 2011. Testing for Zipf's law: A common pitfall. *Economics Letters* 112, 3 (2011), 254 – 255.

[63] Carlos M UrzÃża. 2000. A simple and efficient test for Zipf's law. *Economics Letters* 66, 3 (2000), 257 – 260.

[64] Andreas Vlachos and Sebastian Riedel. 2014. Fact Checking: Task definition and dataset construction. 18–22.

[65] Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science* 359, 6380 (2018), 1146–1151.

[66] Shuohang Wang and Jing Jiang. 2016. Learning Natural Language Inference with LSTM. In *HLT-NAACL*.

[67] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. (Feb 2017), arXiv:1702.03814.

[68] Richard Williams. 2018. Analyzing Rare Events with Logistic Regression. https://www3.nd.edu/~rwilliam/stats3/rareevents.pdf. (05 2018).

[69] Chenyan Xiong and Jamie Callan. 2015. EsdRank: Connecting Query and Documents Through External Semi-Structured Data. In *Proceedings of the 24th ACM CIKM (CIKM '15)*. ACM, 951–960.

[70] Qiang Zhang, Aldo Lipani, Shangsong Liang, and Emine Yilmaz. 2019. Reply-Aided Detection of Misinformation via Bayesian Deep Learning.

[71] Xin Zhao, Yinan Yu, Yongzhen Huang, Kaiqi Huang, and Tieniu Tan. 2012. Feature coding via vector difference for image classification. In *19th IEEE ICIP*. 3121–3124.

A STRATEGIES FOR COMPUTATIONAL EFFICIENCY

A.1 Indexed Data Access and Pre-Computation

As mentioned in Section 5.2.1, the significant size of the document corpus and the relatively expensive computations to be carried out for every potential candidate for each claim required indexing the data access.

Several properties were pre-computed (length of all documents, vector norm of all documents, idf for all terms, tf-idf values for each term in each document, number of occurrences of each term in each document). Using these data, an inverted index was created. The retrieval process was significantly sped up, as many computations could be skipped. In fact, for the td-idf retrieval, the document would not need to be retrieved at all, as the added metadata was sufficient. Also, the candidate documents to consider in all retrieval techniques could be limited to the union of the documents included in any of the claim terms' index entry.

Because the index would, in turn, become rather large itself, and long read times for the pre-computed values would defeat its purpose, it was split up into multiple so-called shards. In practice, $k = 10,000$ provided a reasonable trade-off between shard file size and I/O overhead.

"SuperFastHash" [18], a fast, non-cryptographic hashing function was used to assign a key to a shard, in the hope of creating shards of roughly equal size. The major problem hampering this approach was the skewed term frequencies; this was substantially improved by pruning stop words.

Similarly, an index was constructed to access the Wikipedia documents, which the dataset provided in 109 large, semi-sorted[5] batch files. A look-up table allowed retrieval scripts to only read the one relevant line of the relevant batch to retrieve a document. Retrieving a document thereafter took only a few milliseconds.

A.2 Cloud Computing and Multiprocessing

To process the given amounts of data in a feasible time, Google's Compute Engine[6] was utilised. It offers a free trial in the equivalent value of $300, which was enough to cover this study. I set up a virtual machine that could be dynamically scaled; for the heaviest computational tasks, I used 96 vCPUs, 624 GB RAM, 4 NVIDIA Tesla P100 GPUs, and 100 GB SSD storage.

In order to actually make use of this increased computing capability, the code had to be written to use multiprocessing. For different steps in the sub-tasks, opportunities for parallel processing were identified and solutions were implemented to then merge the partial results.

Furthermore, the code of later sub-tasks detects whether a GPU is available in the current context, and accordingly shifts computations to it.

A.3 Optimised Data Structures and Serialisation

Where possible and allowed, I used existing libraries with highly optimised data structures, such as Numpy or Pandas. Where I created my own structures, I tried to keep them to the necessary minimum. For the vector space retrieval in Section 5.2.2 e.g., it was sufficient to use $|c|$-dimensional vectors, where $|c|$ is the number of unique terms in the query, as the resulting values in the dot product would result in zero for any terms missing from the claim.

Furthermore, I split the tasks into multiple scripts and used serialisation to transfer intermediate results between scripts. This ensured that the same computation would not have to be conducted repeatedly, and errors would not propagate far.

[5] At first glance, the batches appear to be sorted, which would make assigning documents easier. However, the sorting is not persistent and thus couldn't be relied on.
[6] https://cloud.google.com/compute

List of Figures

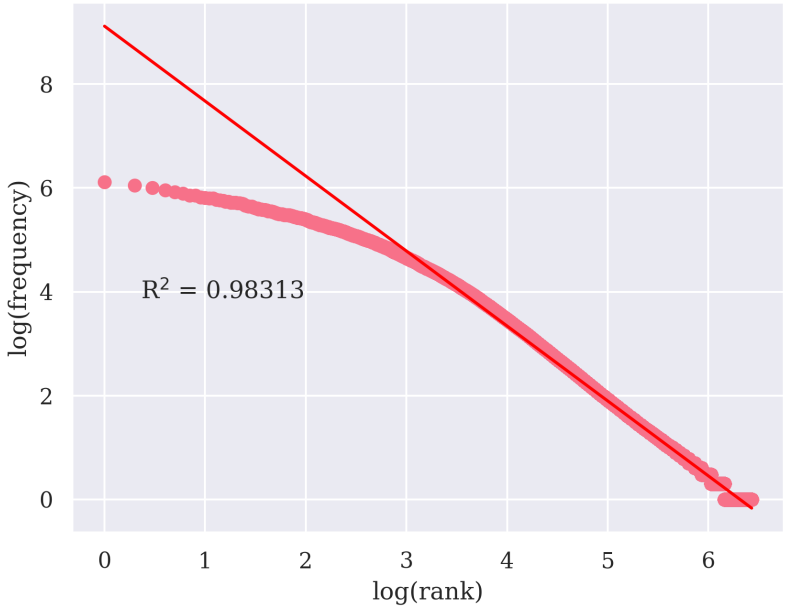

Figure 1: Frequency-rank distribution for terms in the Wikipedia document corpus. The line shows the approximation of the distribution by Zipf's law.

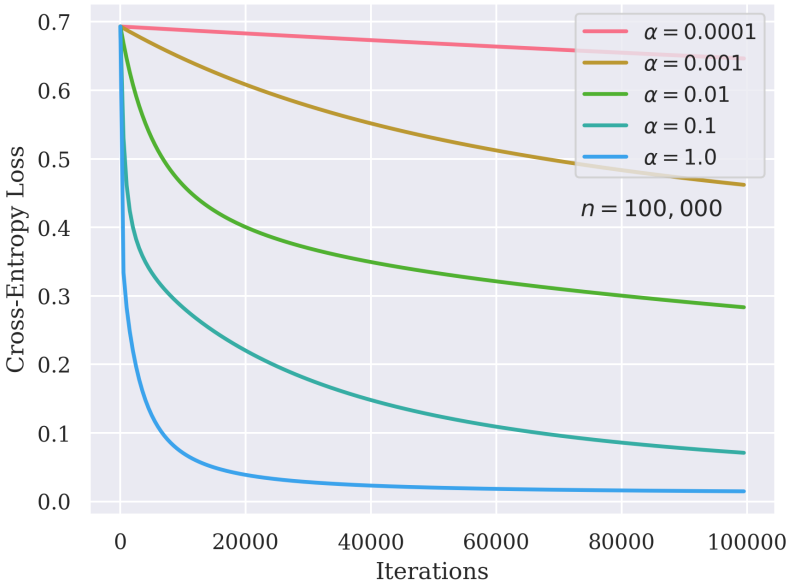

Figure 2: Comparison of different learning rates and how they affect the loss function

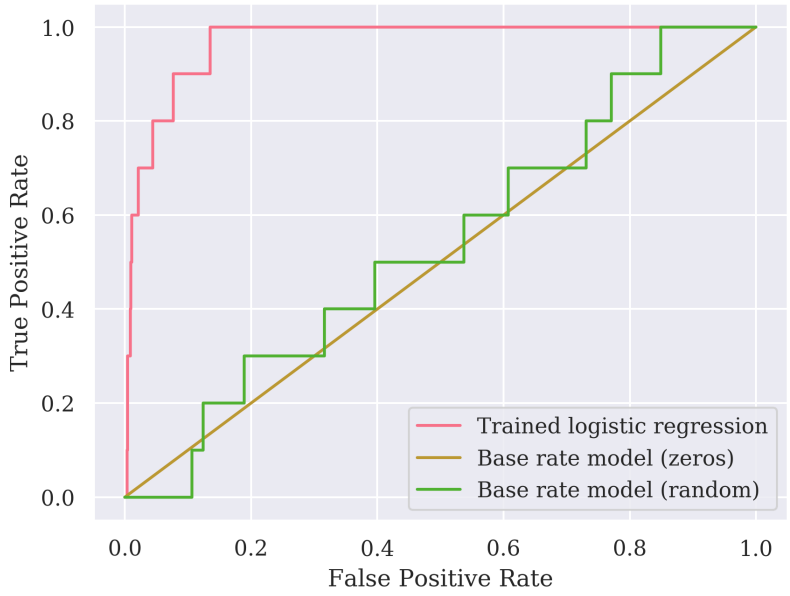

Figure 3: ROC Curves of $LR_{\alpha} = 0.1$ and two base rate models for comparison

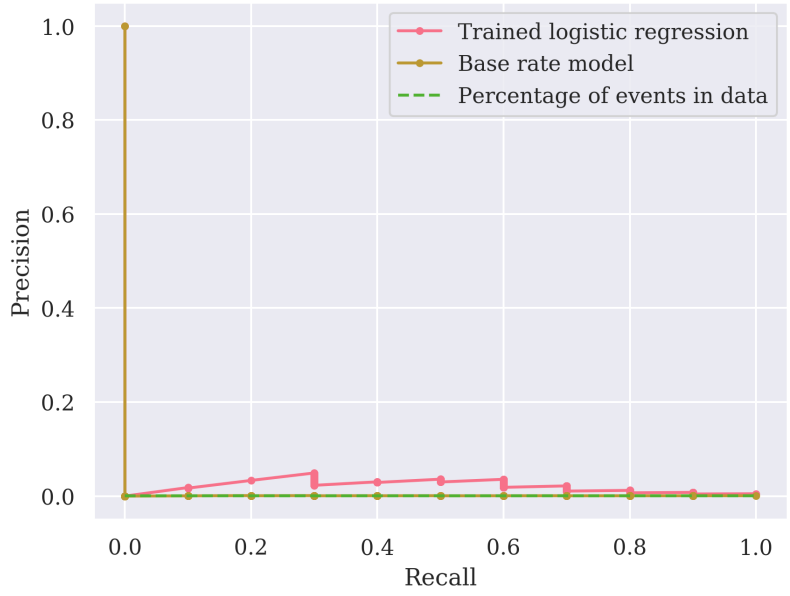

Figure 4: Precision-Recall curve of the LR trained and evaluated with the imbalanced dataset

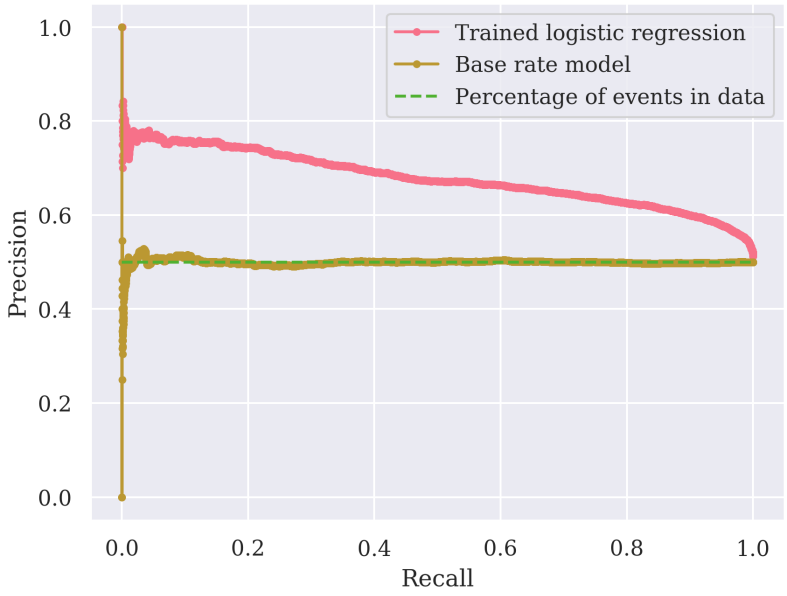

Figure 5: Precision-Recall curve of the LR trained and evaluated with the artificially balanced dataset

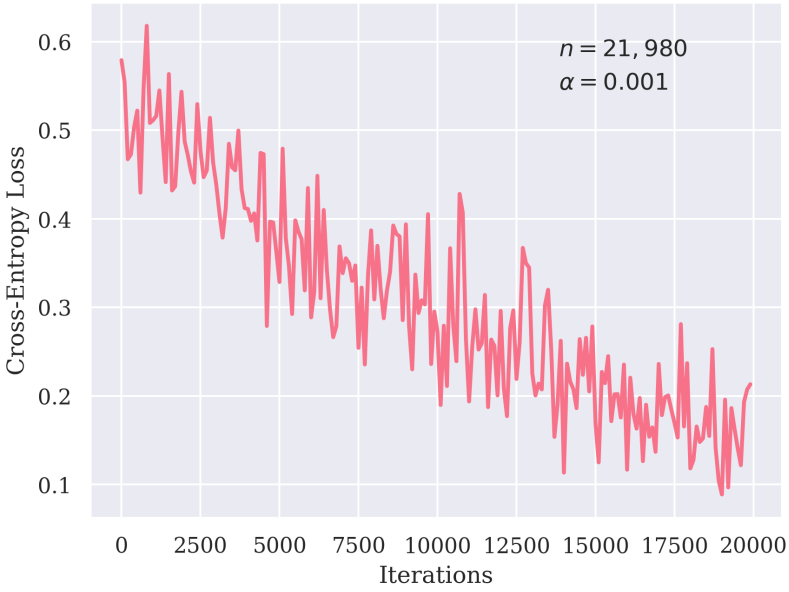

Figure 6: Cross-Entropy Loss for the MLP (20 epochs)

List of Tables

Table 1: Sample of the first 15 claims in the training subset. To decrease computational complexity, some parts of this study are limited to this subset, and furthermore often only verifiable claims are of interest.

ID	Claim	Label
75397	Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.	Supports
150448	Roman Atwood is a content creator.	Supports
214861	History of art includes architecture, dance, sculpture, music, painting, poetry literature, theatre, narrative, film, photography and graphic arts.	Supports
156709	Adrienne Bailon is an accountant.	Refutes
83235	System of a Down briefly disbanded in limbo.	NotEnoughInfo
129629	Homeland is an American television spy thriller based on the Israeli television series Prisoners of War.	Supports
149579	Beautiful reached number two on the Billboard Hot 100 in 2003.	NotEnoughInfo
229289	Neal Schon was named in 1954.	NotEnoughInfo
33078	The Boston Celtics play their home games at TD Garden.	Supports
6744	The Ten Commandments is an epic film.	Supports
226034	Tetris has sold millions of physical copies.	Supports
40190	Cyndi Lauper won the Best New Artist award at the 27th Grammy Awards in 1985.	Supports
76253	There is a movie called The Hunger Games.	Supports
188923	Ryan Gosling has been to a country in Africa.	Supports
138503	Stranger Things is set in Bloomington, Indiana.	Refutes

Table 2: Recall values for the first ten verifiable claims. A: tf-idf; B: tf-idf with $\lambda = 0.25$; C: QLM without smoothing; D: Laplace Smoothing, $\alpha = 1$; E: Laplace Smoothing, $\alpha = 0.1$; F: Laplace Smoothing, $\alpha = 0.01$; G: Jelinek-Mercer Smoothing; H: Dirichlet Smoothing

	A	B	C	D	E	F	G	H
Recall	0.25	0.33	0.42	0.67	**0.75**	0.67	0.58	0.67

Table 3: Evaluation of LR performance on the development subset. The results of the model, trained with different learning rates and $n = 100,000$ iterations each, are contrasted with base rate models which randomly predict 0/1, always predict 0, or predict 1 respectively.

	$LR_{\alpha=0.0001}$	$LR_{\alpha=0.001}$	$LR_{\alpha=0.01}$	$LR_{\alpha=0.1}$	$LR_{\alpha=1.0}$	B_{random}	B_{zeros}	B_{ones}
Accuracy	0.85309	0.78745	0.81232	0.97581	0.99879	0.49282	0.99933	0.00067
Precision	0.00311	0.00316	0.00358	0.01928	0	0.00066	0	0.00067
Recall	1	1	1	0.7	0	0.5	0	1
F1	0.0062	0.0063	0.00713	**0.03753**	0	0.00133	0	0.00135

Table 4: Evaluation of LR performance after the the majority class in the training data has been downsampled and the sample size was significantly increased. A: Evaluated against unbalanced development set; B: Additionally using bias correction; C: Evaluated against balanced development set

	A	B	C
Accuracy	0.95795	0.05917	0.65830
Precision	0.0127	0.00067	0.5
Recall	0.8	1	1
F1	0.25	0.00135	0.6667

Table 5: Accuracy values of the MLP using different combinations of hyperparameters and input formats as described in Section 7.2.4 when tested against the development subset

	input_1	input_2	input_3	input_4
Initial parameters	0.550	0.500	0.505	0.500
Sigmoid → ReLu	0.509	0.502	0.509	0.504
SGD → Adam	0.668	0.677	0.654	0.672
hidden_dim = 100 → 500	0.680	0.683	0.685	**0.696**
hidden_layers = 1 → 3	0.679	0.688	0.683	0.690