Dominic Afuwape
ucapdaf@ucl.ac.uk

Automated Fact Checking

ABSTRACT

An interesting arising problem is the increasing amount of online misinformation, sometimes known as "fake news". This has lead to the research and development of automatic fact checking methods. In this project, a publicly available dataset | Fact Extraction and VerïiňĄcation (FEVER) | containing 185,445 manually verified claims is analysed using machine learning methods in order to classify claims as Supported, Refuted or NotEnoughInfo.

CCS CONCEPTS

• **Computing methodologies** → *Information extraction*; *Classification and regression trees*; *Neural networks*.

KEYWORDS

datasets, neural networks, information retrieval, text tagging,

ACM Reference Format:
Dominic Afuwape. 2019. Automated Fact Checking. In ,. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In an increasingly interconnected world where information travels at extraordinary rates, the proliferation of so called " fake news" has become a significant issue. The term fake news is a neologism which tends to refer to the misinformation that is stated as fact and presented as legitimate news. The term itself gained particular prominence during the 2016 U.S. presidential election. However, this results in an interesting problem. How can consumers of online information determine the truthfulness of online statements. Recently we have seen the rise of fact checking websites such as factcheck.org and fullfact.org where claims as researched by independent fact-checkers. For a scalable solution is makes sense to makes an automated fact checking system. This project tries to propose a novel way to implement such a system. The underlying data-set for the system which is developed is the publicly available Fact Extraction and Verification (FEVER) dataset[4] containing over 2 million wiki-pages with attached sentences and 185,445 manually verified claims which have been validated by humans and tags as either Supported, Refuted and NotEnoughInfo. Each claim, if not tagged as "NotEnoughInfo" also contains an information about the title of the wikipedia pages and location of sentences which can verify or invalidate the claim. This project is seperated in to 3 distinct sections. **1.** An examination of different techniques for document retrieval: Vector Space Document Retrieval where a TF-IDF representation of the claims and document are calculated and compute a cosine similarity and Probabilistic Document Retrieval where we establish a query-likelihood unigram language model and apply Laplace Smoothing, Jelinek-Mercer Smoothing and Dirichlet Smoothing **2.** Using the top retrieved documents, a concatenation of the claim and the sentences in the documents retrieved for that claim are then used as an input into a logistic regression model to determine whether the sentences are relevant. Metrics for evaluating the model performance are also implemented. **3.** A neural network is then trained to attempt to classify the sentences as either Supporting or Refuting the claim. For this section of the project, an architecture consisting of Dense Layers that encode a representation of the vector difference of the average word embedding for both the claim and the sentence. The pipeline for this project is then improved in the final stage.

2 SUBTASK 1: TEXT STATISTICS

Zipfs' law, first proposed by American linguist George Kingsley Zipf is an empirical law that states that with a sufficiently large corpus of text, the frequency of occurrence of a word times that word's rank is approximately equal to a constant.This implies a heavily skewed word probability.

$$rank \cdot frequency = constantK$$

$$rank \cdot wordprobability = constantC$$

Empirical studies on the Zipf's constants of various languages have been conducted. Data suggests that this constant C is approximately equal to 0.1 for the English Language. Testing Zipf's law for a given corpus allows some confirmation of the lexicographical similarity with a given language. In order to test Zipf's law for the collection of documents in the FEVER dataset, each document is opened and it's text added to a dictionary. Each text is split and parsed, from this a count of each word is then calculated .

From this figure, the power law suggested by Zipf is apparent as the data shows an approximately straight line on LOG-LOG axes and therefor the freqeuncy msut be inversely proportional to it's rank.

To further verify Zipf's law an average of the rank times word probability was taken for the top 1000 words was taken finding an average C value of 0.09219 very close to previously observed empirical data.

3 DOCUMENT RETRIEVAL

In order to retrieve documents from the datasets based on a claim, a representation of both the documents and claims must be developed which can accurately match a document to a claim. Two popular ways of solving this issue (Vector Space Model and Unigram Language Model) were explored. For this task, for the sake of computational time, only the top 5 documents for the first ten claims from the train data-set were retrieved. To further increase the speed of retrieval, an inverted index of the location of words present in the claims was generated. This resulted in a dramatic

UCL'19, Automated Fact Checking

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

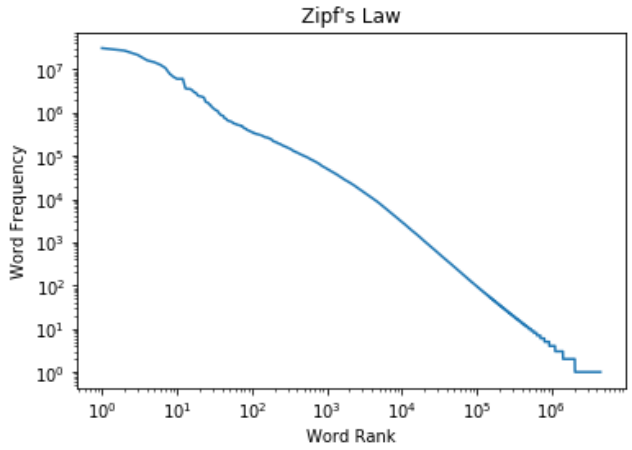

Figure 1: Zipf's Law. Plot of Word Frequency against Word Rank on LogLog Axes.

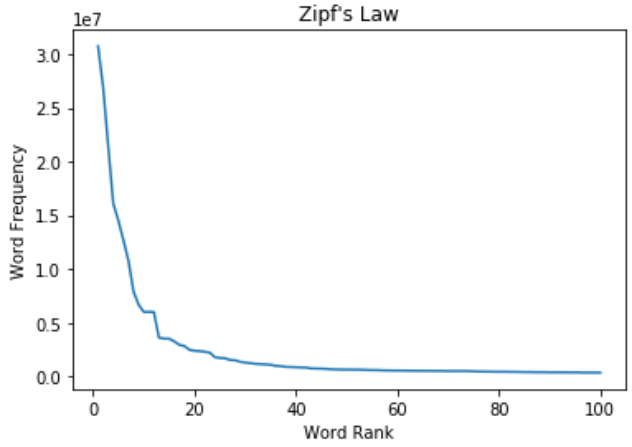

Figure 2: Zipf's Law. Plot of Word Frequency against Word Rank for Top 100 words.

speed of for generating the top 5 documents for each claim from over 1 and a half hours to a matter of minutes.

3.1 Subtask 2: Vector Space

The Vector Space document retrieval method involves representing a collection of words as a vector

$$\mathbb{R}^{\|V\|}$$

where $\|V\|$ is the vocabulary size of the collection. with each term in the collection representing a different direction. For example with two sentences A :"I am a boy " and B:"I am a girl" with each term representing a direction we can generate two

$$\mathbb{R}^5$$

vectors A = [1 1 1 1 0] and B = [1 1 1 0 1] respectively. It is apparent that the closeness of the two statements can then be compared by taking a cosine similarity.

$$\frac{A \cdot B}{\|A\| \, \|B\|}$$

If a word appears multiple times then the magnitude of the vector in that word's direction is increased. Problems can arise with common terms such as "the" "a" or "and" because these common terms would contribute greatly to the vector's direction and magnitude. To solve this, a Term Frequency - Inverse Document Frequency representation $TF \cdot IDF$ is used. A term's Term Frequency TF is defined as the sum of the number of times the term in the query occurs in document. The Inverse Document Frequency $IDF$is defined as

$$log(\frac{N}{n_t})$$

where N is the number of documents and n_t is the number of documents where a particular term appears. Punctuation is removed from the term's and all terms are represented in lowercase so that no difference is made in the retrieval between The and the for example. This therefore discounts the importance of words that commonly appear in many documents. For the sake of computational time, the size of the vocabulary in this experiment is taken to be the size of all the unique words in the first ten claims that are not stop words and at alpha numeric. Using a smaller inverted index with only these terms the retrieval process for the first top 5 documents of each ten claims takes 38 minutes. Examples of documents retrieved by this method are for the claim "The Ten Commandments is an epic film"-"The Ten Commandments -LRB-1923 film-RRB-" or for the claim "The Boston Celtics play their home games at TD Garden."-"TD Garden". One thing to notice is that for some claims we are returned documents which contain information that could verify a claim but are not relevant to the claim according to the tags provided. These results typically lack the key words from the claim in the returned document's title which is something that could be explored to augment the document retrieval.

3.2 Subtask 3: Unigram Language Model

A language model is a model of the possible probability distribution of a series of terms. A language model that assess each term in the joint probability distribution independently without context of the other terms is called a unigram language model.

$$P(t_1, t_2) = P(t_1)P(t_2)$$

Where in a query likelihood unigram language model, $P(t|D)_{MLE}$ is the number of times a query term occurs in a document over the number of words in the document $\|D\|$. A problem occurs because when $P(t_i|D) = 0$ the total joint probability distribution of the query becomes equal to zero. The basic unigram model performed very poorly because as mentioned if any of the terms in the query are not in the document we return a probability of 0 and so essentially random documents are retrieved. A few queries were able to return results due to my dictionary of word counts only containing words which are in the first ten claims. For example "Homeland -LRB-TV series-RRB" is the only non random document retrieved for the claim "Homeland is an American television spy thriller based on the Israeli television series Prisoners of War." It should be noted that the speed for retrieval for the unigram model

is dramatically faster(1.136 minutes) due to not needing to calculate the tf-idf cosine similarity

3.2.1 Laplace smoothing . To solve the problem of zero joint probability distribution, methods of smoothing are applied. One such method of smoothing is the Laplace smoothing method where the new term probability is

$$\frac{P(t|D)+1}{\|D\|+\|V\|}$$

this leads to all terms in the query having some weight. We see very good performance for the Laplace smoothing model as many of the documents containing evidence for the claim are retrieved in the top 5. The time taken for retrival was 1.269 minutes, one again much faster than td-idf.

3.2.2 Jelinek-Mercer Smoothing. As the Laplace smoothing gives large weights to unseen terms one way of dealing with this is to also apply a weight related to the probability of the unseen word in the whole corpus. $P(t|C)$ is the number of times a query term occurs in the whole corpus over the number of words in the corpus $\|C\|$.

$$P(t|D) = 1 - \beta P(t|D)_{MLE} + \beta P(t|C)$$

The speed for retrieval for the unigram Jelinek-Mercer model is also reasonably fast (1.39 minutes). There is a need for the setting of the parameter Beta which is between 0-1. It is clear that long queries should require high values of Beta and vice versa. For simplicity sake, the value of Beta was set to 0.4 but the use of variable Beta depending on query length could be explored.

3.2.3 Dirichlet Smoothing.

$$\frac{\|D\|}{\|D\|+\mu}\frac{tf_{document}}{\|D\|+\mu} + \frac{\mu}{\|D\|+\mu}P(t|C)$$

The Dirichlet Smoothing method requires tuning of the parameter μ. As this mu essentially is similar to a representation of the amount of words in the document, a good starting place is taking an average word per document which is approximately.

4 SUBTASK 4: SENTENCE RELEVANCE

Using the unigram Dirichlet method explained in section 3, each sentences for each of the documents retrieved and each of the claims/queries are transformed formed into an embedding space. An embedding space allows for words with similar meanings and contexts to given similar vector representations. A full text can then represented as the average of the embeddings for the words in the text. In this project the word2vec method of word embedding has been used with the pretrained 100-Dimensional Wikipedia2Vec English embeddings which extends the skip-gram model for words to resolve vector representations of entities. This embedding was chosen as it has been shown to outperform some popular methods due to it's attempt to solve problems of Named Entity Disambiguation an example given being "the capital of the US, the actor Denzel Washington, the first US president George Washington". By concatenating the average word embedding for the claim and sentences a 200 dimension vector is formed which can be used as an input to a logistic regression model which trains the input vector to match an output of 1 when the sentence is "relevant" to the claim and can be used to verify the claim. By processing all of the sentences from the documents retrieved by the Unigram Dirichlet method 5659 sentences are retrieved and of these 5659 sentences just 6 of them are relevant sentences.This leads to a highly class unbalanced data-set which would possibly lead to majority class domination in the classification. In order to properly train the model it made sense to augment the data set for logistic regression with known relevant claim-sentence pairs from the train dataset. Due to the Unigram retrieval not retrieving relevant documents for all claims instead documents listed in the train dataset which had relevant sentences where used and ALL sentences from this documents fed into the logistic regression model. This led to a selection of 20699 sentences being used in the logistic regression training. The thetas were when found using full gradient descent with 1000 iterations with value of alpha = 0.8. From observation of the values of theta, it can be seen that the values for the first 100 features(The claim represented in the embedding space) are all very close to one. In order to evaluate the performance of this logistic regression model, once again as the unigram method does not retrieve relevant documents for all claims as documents relating to claims in the development dataset were hand selected. To evaluate the performance of the model, the precision metric was used.

$$Precision = \frac{No.of RelevantRetrievedDocuments}{No.of RetrievedDocuments}$$

Strangely this results in 100 percent precision as the logistic regression only predicts one claim-sentence pair as relevant(Claim:Andrew Kevin Walker is only Chinese. Document:Andrew Kevin Walker Line:0). However, as will be examined in the next section, this high precision has significant trade-offs with the models recall.

The model training loss is evaluated using the following equation at each iteration.Called the cross entropy

$$\frac{1}{N}\sum(-ylog(\hat{y}) - (1-y)log(1-\hat{y}))$$

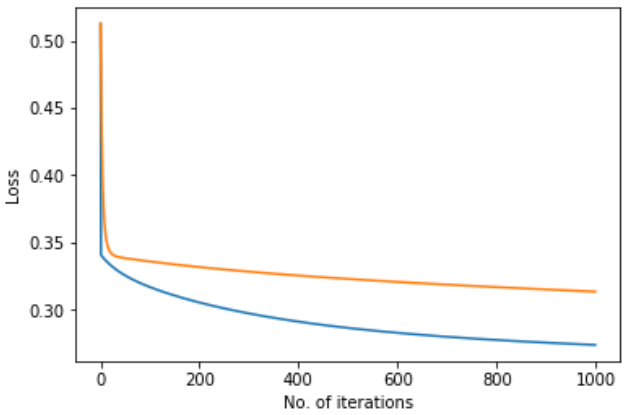

Figure 3: loss Function. Plot of Loss against Iteration for Alpha = 0.1 (Orange) and Alpha = 0.8 (Blue).

It can be seen that there is a dramatic speed up in the rate of loss decrease by increasing alpha within a suitable confine.

5 SUBTASK 5: RELEVANCE EVALUATION

To evaluate the model's performance various metrics are implemented. Recall calculates what proportion of relevant documents were retrieved by the method

$$\frac{No.of RelevantRetrievedDocuments}{No.of RelevantDocuments}$$

Precision calculates what proportion of the retrieved documents are in fact relevant

$$\frac{No.of RelevantRetrievedDocuments}{No.of RetrievedDocuments}$$

Both methods of evaluation have The F1 score is a composite of both Recall and Precision

$$2 \cdot \frac{Precision \cdot Recall}{precision + Recall}$$

Precision	100%
Recall	16.666 %
F1 Score	0.2826

As mentioned in the previous section, claims from the development data subset and the documents relevant to these claims were used to select suitable sentences for input. It is clear that the data set is highly unbalanced which is the likely cause of the poor recall. We see a typical trade off between accuracy and recall. By examination of the F1 score we can conclude that this is generally a poor method. Furthermore, for the majority of the claims, a relevant sentence was not retrieved with the documents retrieved in the document retrieval process, indicating this is not a suitable method for the task. 1/1 sentences retrieved were relevant sentences. 1/6 sentences of the relevant sentences were correctly retrieved.

6 SUBTASK 6: TRUTHFULNESS OF CLAIM

Within the data-set each claim is either "SUPPORTED", "REFUTED" or "NOT ENOUGH INFO". In order to try an classify each claim a neural network architecture can be created. For this task, the data was first simplified to only attempt to classify between "SUPPORTED" and "REFUTED" claims. The idea for the architecture that was devised is that both claim and the sentence which can support or refute a claim can both be represented in an embedding space. This is done using the standard procedure of taking the average word embedding for both. These embedding should contain a sufficient presentation of both items. If a neural network is built on top of a subtraction of the embedding from each other, this should be able to represent the difference between the claim and the sentence verifying the claim in the embedding space. Therefore, the full architecture that was chosen was originally a dual input neural network which feeds into 2 Dense Neural Net each with 150 hidden layers and then a subtraction layer on top of this followed by a dropout layer.Experimentation was conducted into the activation functions which should be used for the inputs into the subtraction layer and testing a LeakyRELU with alpha of 0.9 and 0.1 yielded negligible improvements in performance over normal Relu activation.The Relu activation function did however show noticeable increases in the training speed reaching 90 percent accuracy on the training data with over 200 less epochs than using sigmoid or tanh activation functions.These is then feed in to another Dense neural net with 300 hidden layers followed by a dropout layer, once again to add regularisation to the model. Finally, the output activation layer is a sigmoid activation layer. The model is then optimized using a binary cross entropy loss function and the adam optimizer. Considerations were made to add l1 regularisation to the first 2 DNNs however testing showed a decrease in the development data set test accuracy from 58 percent to 55 percent.

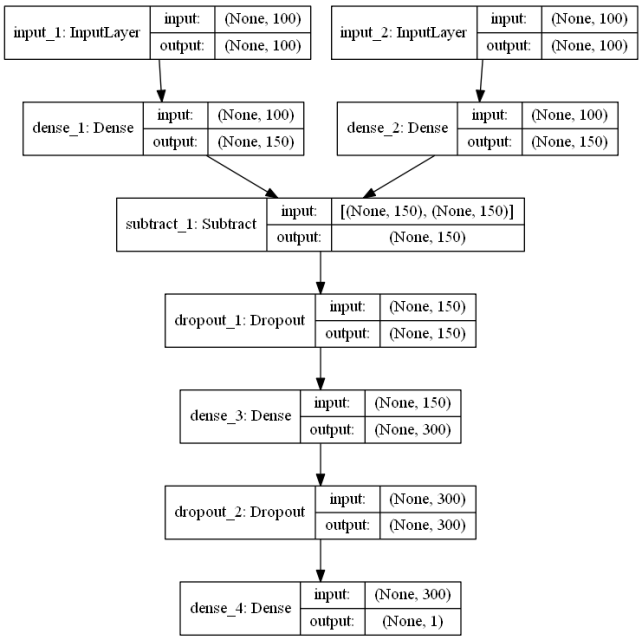

Figure 4: Neural Net with input and output shapes.

To fit the model 8546 verifiable claim-sentence pairs from the train.jsonl file were used for training the data and for evaluating the accuracy of the model 7456 verifiable claim-sentence pairs from the dev.jsonl file were used. After training 400 epochs with a batch size of 1000 the network returned a model which performed with 58.5 percent accuracy on the test data. While this shows there is some predictive power to this method, it is far from consistent, leading to the belief that is is insufficient for the given task. However, one limiting factor may be the fact that only a 100 dimensional average word embedding is used.

7 SUBTASK 7: LITERATURE REVIEW

The FEVER dataset has been worked on extensively. A number of strong solutions have been put forward in the FEVER CodaLab challenge. Common themes in solving this problem involve the use of Keywords or Entities for document retrieval,logistic regression for sentence retrieval and attention and LSTMs for the language inference.

7.1 Document Retrevial

The baseline approach to the document retrieval used a term frequency - inverse document frequency approach to finding relevant

documents, producing a retrieval score of 66.1 percent. One simple way that NEC Laboratories America found to quickly improve the performance of the document retrieval was to simply add the title(Document ID) of the document to the term frequency - inverse document frequency representation of the document. [2] NEC Laboratories America found an extremely successful approach for document retrieval. By retrieving pages based on named entity recognition,specifically phrases tagged as named entities by SpaCy,they were able to increase document retrieval to 80.8 percent with only a small decrease in performance(less than double) in comparison with the FEVER baseline performance of 66.1 percent. Named entity recognition allows,for example, for "Kate Bush" for be tagged as a person and "Google" to be tagged as an organisation. Similarly other approaches used keyword matching with superior results(88.63 percent retrieval).[2]

Similarly, the UKP-Athene:Research Training Group AIPHES also focused on an entity linking framework for the document retrieval, however as standard Named entity recognition only focused on main types of entities (Location, Organization, Person) they used a constituency parser to consider every noun phrase as a potential entity. It was noted that some entities were composed of other types of word such as adjectives and verbs and to solve this all words before the main where used to form and entity. Using this method which was augmented by the use of the MediaWiki API, this team was able to reach extremely high accuracy of 92.6 percent for top 3 result retrieval. [1]

UCL Machine Reading Group focused on building a dictionary of article titles which are used to produce an initial list of potential articles and then a logistic regression using an amalgamation of features such as whether there are stop words and word matching between the claim and first line of the document. They were able to find that the topped ranked retrieved document contained the full relevant evidence in 74.7 percent of cases. Analysis of the various approaches taken shows that the most effective solutions involved the use of entity discovery and matching. Typically these systems performed at a minimum 80 percent accuracy. The other key to successful document retrieval appears to be the use of title information which intuitively makes sense as titles typically contain the most high level concepts referred to in the document.[5]

7.2 Sentence Retrieval

Similar to the process they used for document retrieval, one attempt by the UCL Machine Reading Group involved using a simple logistic regression model evaluating the relevance of the sentence. The feature selection was tailored to include various features such as article length. They also noted that evidence sentences appear at the start of an article and that the article title was often mentioned. When the evidence recall was examined based on the top 5 predicted evidences when averaged across the various aggregation methods, an accuracy of approximately 84.2 percent is seen.[5]

Many groups examined in the use of the bidirectional LSTM in their Automated Fact Checking schemes which used external sources. Analysis of their results indicate the bidirectional LSTM is promising as because while similarly to a simple Recurrent neural network it keeps a memory, it is better able to learn from distant inputs in the sequence as well as considering previous and future inputs as it performances and forward and backward LSTM on the sentences sequences . These methods do add complexity as they require the inputting of each word in a sentence as a sequence is to the neural network but provide often times noticeable performance increases. These papers heavily suggests the use of an word embedding of each word in a sentence fed into the bidirectional LSTM as the basis for the truthfulness stage. Furthermore, many papers mention the use of the ESIM (Enhanced Sequential Inference Model).This model was originally proposed by the Stanford NLP Group for the purpose of sentence pair entailment. The ESIM can be used to encode both the claim and the sentences related to that claim. It consists of bidirectional LSTM a fundamental block that learns a representation of words which as typically in a pre-trained word embedding form which feeds into layers for local inference modelling and inference classification which is another Bidirectional LSTM. While use of the ESIM is almost ubiquitous in the top performing sentence retrieval methods, many solutions built some variation on top of the basic ESIM model.

The UKP-Athene:Research Training Group's modified architecture,for example instead extended the model to be able to provide a ranking score of all the documents on top of this these used a hinge loss function during training to maximise the difference between a positive claim sentence pair and a random negative claim sentence pair. This model was particularly successful as it managed to provide 85.37 percent sentence recall with only the top 3 search results. [1]

7.3 Textual entailment

The task of textual entailment and sentence relevance has been shown to be a similar type of problem which can be solved with any of the standard methods for encoding sequential data. As the previous section has shwon, the ESIM model is a very potent solution for these types of problems. Many researchers simply extended their ESIM models used in the sentence relevance portion. One challenge within the automated fact checking task for classifying the claim as Supported, Refuted or NotEnoughInfo come from needing to correctly aggregate the different possible evidences for a claim be that through pooling attention or sentence concatenation.

Improvements from the ESIM model seem to be possible. NEC Laboratories America's novel high performing solution used the idea of a "Transformer network" developed by Google AI which is based on attention mechanisms alone rather than recurrence. This method also has particular benefits which may not be relevant to all, in that it is highly parallelisable and so can useful for those with access to high performance computing components. By adding the title of the document containing the relevant sentence to the evidence and categorising the claims using the transformer network top results were achieved, when placed into direct comparison with the ESIM model they used for comparison, the transformer network achieved 95.8 percent accuracy compared to the ESIM's 84.6 percent accuracy. [2]

8 SUBTASK 8: MODEL IMPROVEMENT

In order to improve the models that have been shown previously in this report, one way might be by increasing the quality of features used in the models. For the truthfulness of claim classification

there is also the possibility that by using the polarities of the sentences i.e not, no, doesn't, all having a negative polarity these have be represented as features into a neural network. Finally, by implementing a bidirectional LSTM which has proven to be highly successful in previous reports, on top of this polarity feature there is a possibility of improved performance. By using a regular Fully connected network in Subtask 6 rather than a recurrent neural network, the network lacks the ability to apply entailment based on the relationship between a sequence of words as instead an average embedding of those words has been used. By using the Bidirectional LSTM which has been mentioned in previous sections this problem may be solved. For both claim and sentence, a separate LSTM is used to encode the sequences. Each word in the claim and sentence as converted to a 100 Dimensional word embedding form as done previously in this project based on a word2vec wikipedia corpus. In order to train these LSTMs, 5000 verifiable claim sentence pairs from the train dataset were chosen split 4000/1000 train and validation.

The neural network has 3 distinct inputs, the 2-D dimensional embedding of the claims which has been zero padded to have a length of 10 for up to 10 words in the claim, the 2-D dimensional embedding of the sentence which has been zero padded to have a length of 35 for up to 35 words in the sentence, and a 4 dimensional vector reflecting [subjectivity of claim,subjectivity of sentence,polarity of claim,polarity of sentence]. The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective. These scores are generated in two possible ways "PatternAnalyzer (based on the pattern library) and NaiveBayesAnalyzer (an NLTK classifier trained on a movie reviews corpus)."[3] A comparison of these two methods can be made and is something that could be explored in the future. The sentiment vector is then put through a Dense neural network with a sigmoid activation. The claim and sentence embeddings are put through their respective LSTM's with an output shape of 50 and a 0.5 dropout layer is applied to both for the sake of regularisation. Increasing the output shape and dropout seemed to produce increases in training accuracy at given epochs,however the computational load ha to be limited. The 3 outputs are concatenated and put through final dropout layer before a Sigmoid activation layer.For the sake of minimising the complexity of the problem, this model only classifies verifiable and non-verifiable. Using binary cross entropy and the "adam" optimizer, the model is then trained with 10 epochs. In these 10 epochs the model achieves a training accuracy of 98.2 percent. In order to further evaluate the predictive power of the model, it is then used to evaluate 5000 sentences retrieved from the development data set. On this data set there is an accuracy score of 68 percent. The results are aggregated across multiple evidences by taking the average output from the various evidences related to a claim.

To see if the use of sentence sentiment has a positive effect on classification, a comparison is made with a similar network without the sentiment input. It is trained on exactly the same data but without the sentiment input vector branch. It is also evaluated on the same input.

The results show that the addition of the sentiment vector does provide some additional predictive capability as the the accuracy of the model without the sentiment vector reduces by 3.5 percent

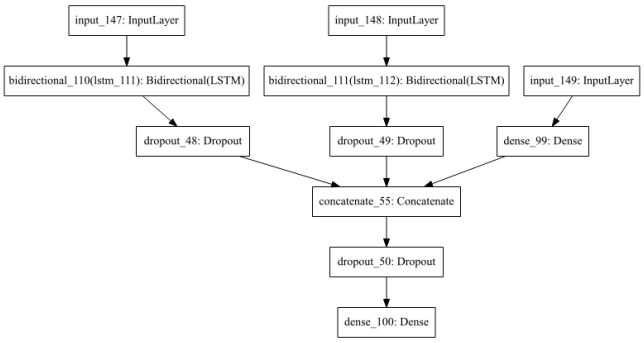

Figure 5: Second Neural Net with input and output shapes.

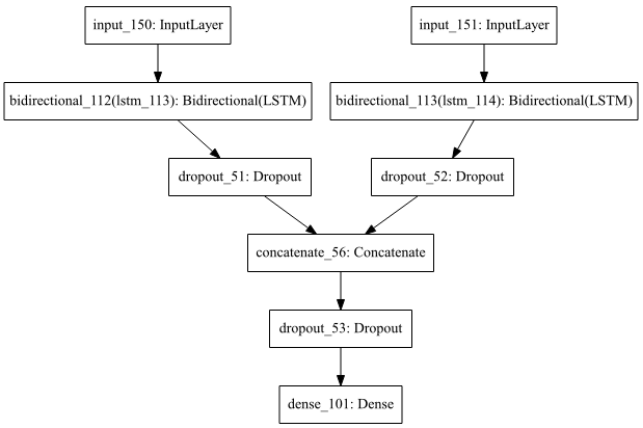

Figure 6: Second Neural Net without sentiment input for comparison.

on the test dataset even though it also reaches above 97 percent accuracy on the train validation dataset. Overall, this feature augmentation could be added to a other network, which are more high performing than the Bidirectional LSTM such as the ESIM or transformer model and could lead to even greater classification accuracy.

REFERENCES

[1] Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. *UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification.* Technical Report. 103–108 pages. https://www.ukp.tu-darmstadt.de/

[2] Christopher Malon. 2018. *Team Papelo: Transformer Networks at FEVER.* Technical Report. 109–113 pages. https://aclweb.org/anthology/W18-5517

[3] TextBlob. [n. d.]. API Reference ĂćâĈňâĂĬ TextBlob 0.15.2 documentation. https://textblob.readthedocs.io/en/dev/api{_}reference.html{#}textblob.blob.TextBlob.sentiment

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

[5] Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. UCL Machine Reading Group: Four Factor Framework For Fact Finding (HexaF) Takuma. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER).*