




University
of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Computer Science	Spring semester, 2018 Open / Confidential
Author: Erlend Ådnanes Rekve	 (signature of author)
Programme coordinator: Vinay Jayarama Setty Supervisor(s): Vinay Jayarama Setty	
Title of Master's Thesis: Automated false claims detection using deep neural networks	
Credits: 30 ECTS	
Keywords: Deep learning • Neural networks • False claims detection • Fake news • Text classification • Natural language processing	Number of pages: 56 Stavanger, June 15, 2018



Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Automated false claims detection using deep neural networks

Master's Thesis in Computer Science

by

Erlend Ådnanes Rekve

Internal Supervisor

Vinay Jayarama Setty

June 15, 2018

Abstract

Recently false claims and misinformation have become rampant in the web, affecting election outcomes, stock markets, and various other societal issues. Consequently, fact-checking and claim verification websites such as snopes.com are becoming increasingly popular and are also being integrated into news search engines such as Google news. However, these websites require expert analysis which is slow and not scalable. Many recent papers have proposed machine learning methods using handpicked linguistic and source-based cues to automate the claim verification process. In this thesis, we propose deep neural models which avoid tedious feature engineering and strong assumptions and yet detect false claims with high accuracy. To achieve this, we propose a hybrid model which combines textual content of the news articles as well as the reactions they receive in social media forums such as Reddit. Using large-scale manually curated data from fact-checking websites such as snopes.com, politifact.com and emergent.info we perform extensive experiments to show that our models outperform the state-of-the-art CRF-based models.

Acknowledgements

I would like to thank my supervisor Vinay Jayarama Setty for suggesting this thesis and giving good guidance, ideas, and encouragement during its duration.

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Motivation	1
1.2 Problem statement	2
1.3 Usecase	2
1.4 Challenges	4
1.5 Contributions	5
1.6 Outline	5
2 Background	7
2.1 Technical background	7
2.1.1 Neural networks	7
2.1.2 Neural networks for text classification	11
2.1.3 Introduction To performance measures	13
2.2 Related work	14
3 Solution Approach	17
3.1 News article model	18
3.2 Social media discussion model	19
3.3 Joined model	21
4 Experimental Setup and Data Set	23
4.1 Environment	23
4.2 Dataset	24
4.2.1 Text preprocessing	25
4.2.2 Data analysis	26
5 Experimental Results	33
5.1 News article model	33
5.2 Social media model	37

5.3	Joined model	38
6	Conclusion and Future Directions	41
6.1	Future directions	41
6.2	Conclusion	42
	Bibliography	43

Chapter 1

Introduction

1.1 Motivation

Online news sites and social networks have become a major source of news, information, and knowledge for a great number of people. Hundreds of thousands of news articles, tweets, blogs, and social network posts are published, shared, and constantly discussed online. Unfortunately, this also helps misinformation and false claims to spread faster and deeper in social networks and other web sources than ever before [1]. To address this issue, many popular social networks and news aggregators such as Facebook and Google news are leveraging either crowd or fact-checking services such as snopes.com, poltificat.com, and emergent.info. While these solutions are effective and important, they rely on expert analysis and manual effort. Due to the number of claims that need to be verified, the manual labor required results in significant time constraints and limits the scale of these services. After working with Facebook for a year, politifact.com stated that their biggest weakness was not being capable of fact checking all the claims appearing on Facebook¹. An automated detection tool of false claims could greatly aid this issue. Such a tool would significantly reduce the effort required to verify these claims. While inevitably not being as accurate as manual fact checking, the tool may help human readers to be more skeptical of claims floating around on the internet and encourage them to verify a claim from other sources.

¹<http://www.politifact.com/truth-o-meter/article/2017/dec/15/we-started-fact-checking-partnership-facebook-year/>

1.2 Problem statement

Given a claim c represented as a sentence, our goal is to automatically detect whether it is ‘true’ or ‘false.’ For this purpose, we consider news articles which mention the claim N_c and corresponding social media responses S_c . For example, given a claim that “the European Scientific Journal, a peer-reviewed academic publication, concluded that the collapse of the Twin Towers and World Trade Center Building 7 on 11 September 2001 was the result of a controlled demolition”², we have a corresponding news article reporting “Scientific Study: Towers Collapsed Due To Controlled Demolition”³ and discussion thread of reactions it received in social media. While many in online forms tend to believe this article, some comments disagree and provide proof for it. For example, one Reddit comments say “This isn’t a study of any kind. It’s just an article that highlights some of Jones’ arguments...”, see Figure 1.1 for a full example of this claim. Our goal is to decide whether claim c is true or false automatically. Our goal is to model a neural network which only relies on the textual content of the news articles and the online reaction relating to claim. We believe that it is a simple yet powerful way to capture the essential features and patterns necessary to determine if a claim is valid.

1.3 Usecase

Imagine a user browsing the web is presented with a news article stating the above-mentioned claim. Our model can display results to help the user estimate the trustworthiness of this article. Which is done by feeding other news articles reporting this claim and their social media reactions into the model. We hope this will aid users to be more skeptical of claims they read on the web and make it harder for outlets to deceive and confuse. Another use of this model can be a type of search engine where a user can search for a dubious claim. The model will then gather articles and social media reactions mentioning that claim, and produce a confidence score for the trustworthiness of that claim.

²<https://www.snopes.com/fact-check/journal-endorses-911-conspiracy-theory/>

³<https://yournewswire.com/scientific-study-towers-collapsed-due-to-controlled-demolition/>


Claim

The European Scientific Journal, a peer-reviewed academic publication, concluded that the collapse of the Twin Towers and World Trade Center Building 7 on 11 September 2001 was the result of a controlled demolition.

News Articles

Scientific Study: Towers Collapsed Due To Controlled Demolition

© September 3, 2016 • Sean Adig Tabatabai | 5h News US | 214



Physicists & Engineers Publish Another 9/11 Study: All THREE Buildings Were “Destroyed By Controlled Demolition”


Published 2 years ago on September 11, 2016
By Arjun Wala

WIND EXCLUSIVE
9/11 CONSPIRACY GETS SUPPORT FROM PHYSICISTS’ STUDY
Europhysics magazine report finds Twin Towers brought down by ‘controlled demolition’
Published: 09/31/2016 at 10:06 PM

Share on Facebook
Share on Twitter
Email
Print

15 YEARS LATER: ON THE PHYSICS OF HIGH-RISE BUILDING COLLAPSES

Steven Jones, Robert Korol, Anthony Szamboti and Ted Walter
*Brighton Young University (orig. writer) *Hohmann University (emritica) *Industrial design engineer in the aerospace industry - *Architects & Engineers for 9/11 Truth - DOI: http://dx.doi.org/10.1021/epw.201640c



Social Media Reactions

[\[-\] pignie](#) 6 points 1 year ago

Pretty obvious to anyone with functioning synapses. Nice to have even further evidence.

[permalink](#) [embed](#) [save](#)

[\[-\] Sebas94](#) 6 points 1 year ago

Will the truth ever be accepted?

This isn't a study of any kind. It's just an article that highlights some of Jones' arguments. I have no idea why this was published in what appears to be a respectable scientific publication, but it begins with this:

NOTE FROM THE EDITORS

This feature is somewhat different from our usual purely scientific articles, in that it contains some speculation. However, given the timing and the importance of the issue, we consider that this feature is sufficiently technical and interesting to merit publication for our readers. Obviously, the content of this article is the responsibility of the authors.

So make of it what you will.

Figure 1.1: Example of a claim circulating on the web and its associated news articles and social media reactions

1.4 Challenges

Many automated claim verification and fake news detection techniques have been proposed in the literature which mainly rely on manually crafted linguistic features such as lexicon of bias, sentiment, and subjectivities [2–4]. However, according to some studies, misinformation is very difficult to detect even for critical human readers [5]. Moreover, it is not hard to write high-quality news articles conveying false facts. For example, a professionally written news article claiming that “The Queen was threatening to abdicate should Britain leave the EU” was published in “Yournewswire.com” and shared on Facebook over 23,000 times even though it is a false claim⁴. Since these articles are written by humans, it is hard to come up with a specific set of features to verify their truthfulness. Another widely used feature for detecting false news is source-based features such as page rank of the news website or other reliability scores based on the verified truthfulness of past articles [2]. While the source-based features boost the recall of detecting false claims, they result in poor precision since every news article from a certain website tends to be classified as true or false. For example, A news article from frobes.com which has a high reputation and page rank reports that “That Scientific Global Warming Consensus...Not!”⁵ which is debunked by politio.com. Moreover, it is not difficult to mask the source of the article using blogs in reliable domains such as “wordpress.com”. Finally, these features are combined using some linear models [3] or more sophisticated models such as Conditional Random Fields (CRFs) to integrate various features have also been proposed [2].

In addition to linguistic and source-based features, the response these articles receive online in social media forums such as Reddit and Twitter is often crucial. There are recent works which consider the temporal patterns of the response received for news articles and model them using deep neural networks such as LSTMs [6, 7]. However, these works do not consider the news article contents rather only focus on the textual content of the reactions. To the best of our knowledge, there is no existing work which holistically considers the textual content of the news article as well as the social media reactions to detect the false claims.

⁴<http://www.bbc.com/news/av/world-us-canada-38794905/fake-news-this-is-a-war-on-alternative-media>

⁵<https://www.forbes.com/sites/larrybell/2012/07/17/that-scientific-global%2Dwarming-consensus-not/#2d60d3b83bb3>

For a neural network to be adequately trained, a significant amount of labeled data is required. A huge challenge is that there exists very little labeled data for the task of false claims detection. Some efforts have been made, for example, [8] which presents a dataset of 12.8K manually labeled short statements collected from politifact.com.

1.5 Contributions

We propose neural network models to represent the textual content of the news articles and social media comments holistically to detect false claims. We avoid handcrafting any features but instead rely on the neural networks to learn the necessary features automatically which are often complex and difficult to identify. One of the challenges in dealing with data from varied sources such as news and social media is that the language and vocabulary used them tends to be fundamentally different. At the same time, we also recognize that integrating these two data sources is essential. To address this issue, we propose an elegant way to integrate two different models representing news articles and social media comments by jointly learning them in a single neural network. Using extensive experiments, we show that our neural network models outperform both standard classifiers such as Support Vector Machines (SVM) and Naive Bayes, as well as more sophisticated conditional random fields (CRF) models using linguistic and source-based features.

1.6 Outline

The rest of the thesis is structured like this:

Chapter 2, Background, presents the technical background required for this thesis and discusses related work in the field of automated false claim detection.

Chapter 3, Solution Approach, presents a detailed explanation of models we propose.

Chapter 4, Experimental Setup and Dataset, presents the setup and datasets used to run the experiments.

Chapter 5, Experimental Evaluation and Discussion, presents and discusses the experiments and results of the various models experimented with

Chapter 6, Conclusion And Further Directions. Concludes the thesis and suggests future directions.

Chapter 2

Background

2.1 Technical background

2.1.1 Neural networks

Machine learning is a type of artificial intelligence, where the goal is for the computer to learn from prior experience. It is split into two parts, supervised- and unsupervised learning. Where unsupervised learning is the task of grouping similar data without labels and supervised learning is the task of grouping already labeled data.

Neural networks are a type of machine learning that has layers of neurons and loosely resembles the brain. It can be thought of as a function that requires a specific set of inputs and produces a specific set of outputs[9]. The network consists of multiple layers of neurons that have a weighted connection between each other. The layers are split into input-, hidden -, and output layers. If a network has multiple hidden layers its called Deep learning and gains its strength by having simpler abstractions of a problem to come to an end solution. Some important features of a neural network are

Neurons. A neural network consists of multiple neurons. These are functions which have multiple weighted inputs and produces an output using an activation function. A neuron has a weighted connection to a subset of other neurons in a network.

The *weights* are the connection between two neurons. These weights decide how much influence node A has on the output of Node B. The weights get updated during training, and its what makes the network learn.

Bias is an additional neuron in each layer that has no inputs and is an extra input to each neuron in the following layer.

The *activation function* computes the weighted sum for every weight and bias in a neuron's input.

Training a neural network requires a vast amount of data. The weights of the network are at the start is randomized. When data is sent through the network the neurons activate and at the end produces some output. The output gets compared to the expected label of the data, and an error is calculated depending on how well the network performed. The error is then propagated back through the network, and the weights and biases are updated. This process is called backpropagation. The error can also be saved in mini batches, and after a set amount backpropagated through the network which results in more stable learning. How the network learns is decided by the learning algorithm, the most common one is stochastic gradient descent. This algorithm tries to find some local minima of the error. How fast it converges is decided by a learning rate. Once the network has been trained, data can be feed through the network in the same process to predict the outputs.

The goal of a supervised learning algorithm is to approximate a function f given its input x and output y such that $y = f(x)$.^[10] We want the approximated function to generalize well, such that the model performs well on data not seen during training. Two big causes of poor performance in a machine learning algorithm are the concept of over- and underfitting. Overfitting refers to a model that learns training data too well such that it has a negative impact on the performance on unseen data. Underfitting refers to a model that can neither generalize the training data nor the unseen data. The goal is for the model to find a good fit and perform well on seen and unseen data. These concepts are illustrated in Figure 2.1

There are several different neural networks architectures. The most common ones are feedforward-, convolutional-, and recurrent neural networks. A small introduction to each of these architectures follows.

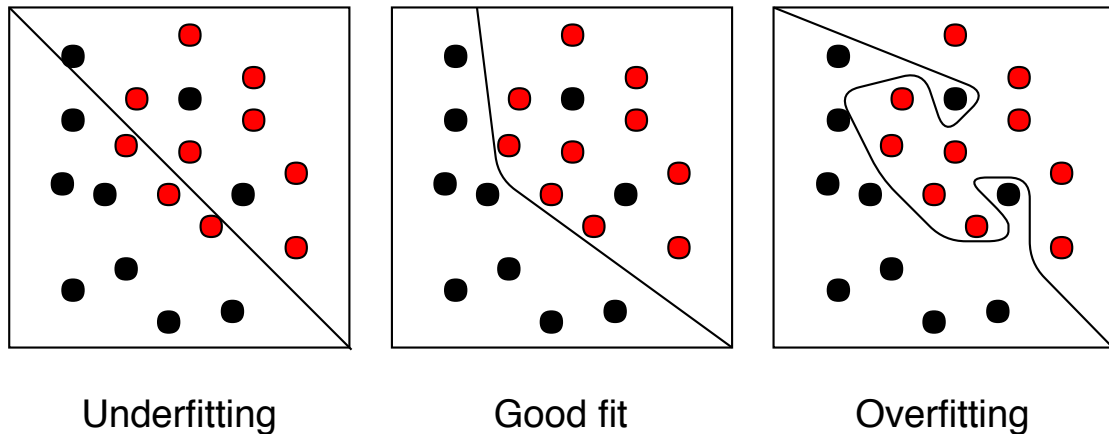


Figure 2.1: Illustration of Overfitting, Good fit and underfitting in machine learning algorithms

Feedforward neural networks

Feedforward neural networks, also called multilayer perceptrons (MLP) is the quintessential deep learning model [9]. The network is called feedforward because the information flows only one way through the network.

Figure 2.2 shows an illustration of a multilayer perceptron. Each circle is a neuron, and each connection is the weights connecting it to other neurons in the network. The input layer is responsible for representing the input data, the hidden layers draw abstractions of the problem, while the output layer produces the results of the problem at hand.

As an example take a set of black and white 28×28 images of handwritten digits ranging from 0-9, where the objective is to classify which digit appears in the image. The input layer would be $28 \times 28 = 784$ set of neurons representing the gray scale value of each pixel in the image. The hidden layers may learn representations of what constitutes each digit, such as detecting edges of circles and lines. The output layer would have ten neurons each representing a digit. In the case of the output layer having a softmax activation function, each neuron would have a confidence value on how likely it is to be that particular digit.

Convolutional neural networks

Convolutional neural networks (CNN's) is a special kind of feedforward network. CNN's are used for problems where the input data has grid-like topology, and gets

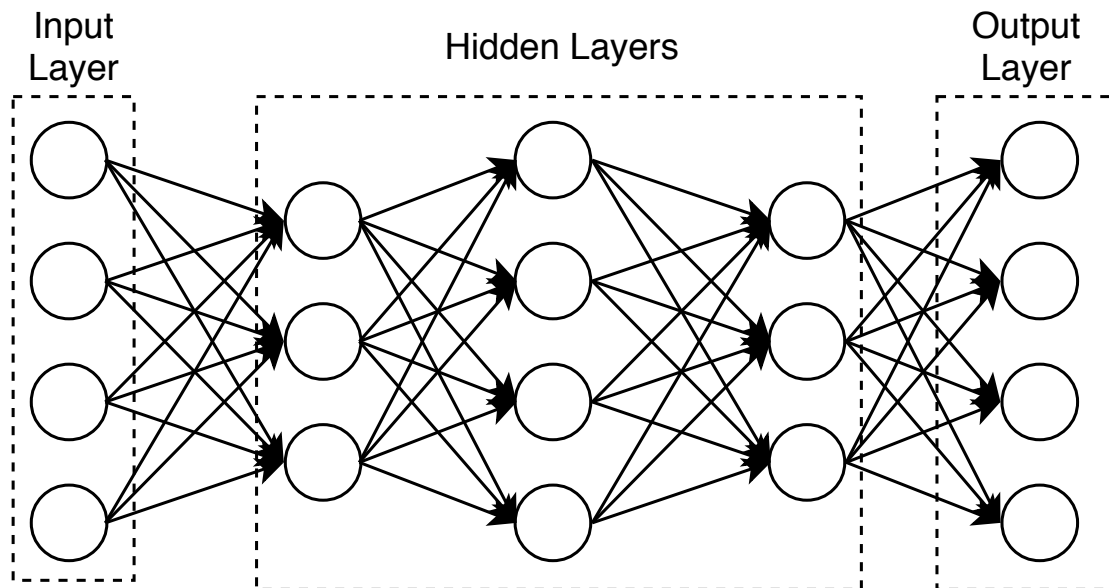


Figure 2.2: Illustration of Multilayer perceptron (MLP) network

its name from the mathematical operation called convolutions, and is simply put an operation done on two functions [9]. A CNN gains its strength by requiring fewer weights than a fully connected network, by preserving the spatial relationship in the grid and using small parts of the input to draw features.

A CNN consists of 3 types of layers. Convolutional layers, pooling layers, and fully connected layers. The convolutional layers draw features from the proceeding layers, by segmenting small parts of the input data. Parameters of the convolutional layer are Filter size, kernel size, and stride. The filter size is how many neurons there is in the layer, the kernel size is how big the feature mapping is and stride how many places the feature mapping should move at one time. The pooling layer downsamples the features found in in the convolutional layer and keeps the best performing features. In the end, there is a fully connected layer that flattens the structure, and with its weights and biases produces a final output. An illustration of this process is shown in Figure 2.3

Continuing with the example of the digit classification problem, the input to the network will be a 2-d matrix with a dimension of 28x28. Each cell is representing a pixel. The convolutional layer then looks at parts of the image each time and outputs a value for each stride. The pooling layer then takes the best performing strides and sends them to the fully connected layer for final classification.

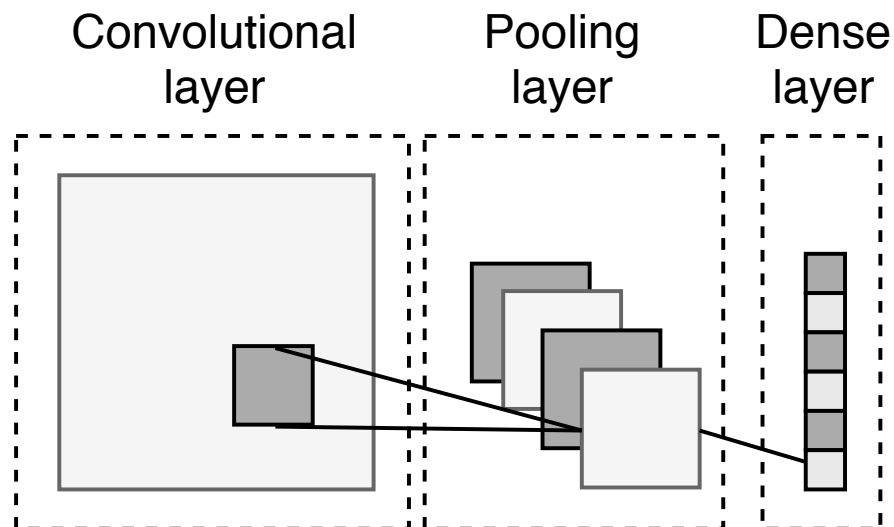


Figure 2.3: Illustration of a convolutional neural network (CNN)

Recurrent neural networks

RNN's is a family of neural networks for processing sequential data. In contrast, to feed-forward networks, RNN's have connections that go backward adding memory and feedback to the network. Which allows the network to remember over a sequence of inputs. As this thesis utilizes RNN's solely for comparison, the specifics will not be discussed here, for further insight into RNN's [9] is an excellent resource.

2.1.2 Neural networks for text classification

We chose to use neural networks because in literature it has gained good results for text classification [11–13]. To employ neural networks for natural language tasks the text data has to be converted to real-valued vectors. There are multiple strategies to vectorize text documents. The methods experimented with in this thesis are:

Tf-IDF vectorizer stands for term frequency-inverse document frequency. Often used in information retrieval, and is intended to reflect how important a word is to a document in a corpus. Term frequency reflect how many times a term occurs in a document and its simplest form is defined as

$$tf_{t,d} = f_{t,d}$$

where $f_{t,d}$ is the number of occurrences of term k in document d . Inverse document frequency reflects the importance of a term in the entire corpus of documents, the more a term shows up in the corpus, the less significant it is. It is defined as

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents that contain term t . \log is used to dampen the effect of IDF. Combined TF-IDF is defined as

$$tfidf_{t,d} = tf_{t,d} * idf_t$$

Term frequencies measure the importance of a term in the document. Inverse document frequency measures its importance in the corpus. To use TF-IDF in machine learning, the TF-IDF is calculated over all the training data, and each data sample calculates a vector based on the TF-IDF.

Word Embeddings A Word embedding is a learned vector representation of words. Every word in a training corpus is mapped to a multidimensional vector and is learned based on the usage of words. This method allows for words with similar meaning to be given similar vector representations. A textbook example of a possible representation of words in an embedding is that $king - man + woman = queen$ capturing the analogy king is to a queen like a man is to a woman[14]. Without going into the math on how these embeddings are learned two popular embedding techniques are Word2vec[15] and GLOVE[16]. To use word embedding in a neural network, an embedding layer is used. This layer is trained jointly with the neural network and requires that each word is encoded to a real number, the embedding layer is at the front end of the network and is fit in a supervised way through the backpropagation. A common practice is to use a pre-trained word embedding to initialize the embedding layer. Common ones are GLOVE¹ pre-trained on Wikipedia and word2vec² pre-trained on google news. During training, these embeddings can be updated to fit the training data better.

A popular method of using neural networks for document classification is using CNN's. The model presented in [11] is a CNN model designed for sentence classification trained on top of pre-trained word vectors. It shows that a CNN with

¹<https://nlp.stanford.edu/projects/glove/>

²<https://code.google.com/archive/p/word2vec/>

little hyperparameter tuning and static vector achieves excellent results on multiple benchmarks. Updating the word vectors through training further enhances the model. [17] Provides a sensitivity analysis of using CNN's for text classification and gives a beginners guide on how to set hyperparameters for text classification tasks using CNN's.

2.1.3 Introduction To performance measures

This section introduces the performance metrics used to evaluate the performance of our models. Figure 2.4 shows a confusion matrix, a popular tool for classification problems. Using our task as an example, the positives is when a claim is false, and negatives are when a claim is when a claim is true.

True Positives (TP) is the cases when the actual and predicted label is a false claim.

True Negatives(TP) when the actual and predicted label is a true claim.

False positives(FP) when the prediction is a false claim but the actual claim is true.

False negatives(FN) is when the actual label is a false claim but the predicted one is true.

There are multiple metrics which can be calculated out of the confusion matrix. The most common ones are:

Precision tells us the proportion of claims that was predicted as false actually was false.

$$Precision = \frac{TP}{TP + FP}$$

Recall tells us the proportion of claims that actually was false was predicted as false by the model.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score is the harmonic mean between precision and Recall. The harmonic mean is a mean where the value is closest to the smaller number if they differ.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

		Actual	
		Positives	Negatives
Predicted	Positives	TP	FP
	Negatives	FN	TN

Figure 2.4: Figure of a confusion matrix used for performance evaluation of classifiers

Micro average Accuracy is the number of correct predictions made overall predictions made. Regarding the confusion matrix, its calculated like this

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Macro average accuracy is the mean of the true claims- and false claim recall.

$$MacroAccuracy = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FN}}{2}$$

2.2 Related work

Detecting false claims is equivalent to other tasks in the literature such as fake news detection, news credibility analysis, rumor detection, etc. We avoid the loaded term “fake news” as it is misused recently by many politicians.

Linguistic cues from the news articles are the most natural features for detecting false claims. It has been shown by the NLP community that language of deception could provide good cues for detecting fake news articles [4]. There are also other works which consider manually engineered linguistic and source-based features [2, 3, 18]. However, identifying the specific linguistic cues that are decisive for fake news is not yet fully understood. There are also efforts to address some sub-problems

of detecting false claims such as stance detection to check if the news articles are click-baits by verifying if the claims in their headline and the body match [19]. However, as we show in our experiments, news article contents alone are not always sufficient, and none of these approaches consider additional information such as social media discussions to detect false claims.

When it comes to social media, most approaches in the literature manually hand-craft features such as Facebook likes, number of shares, and user demographics to train classifiers for detecting rumors [20, 21]. Modeling rumor cascades in social networks as Recurrent Neural Networks (RNNs) have been proposed recently [7, 22]. Hybrid models which consider the temporal sequence of the textual content of social media comments have also been proposed [6]. However, these models do not consider the news article text and social media comments holistically.

In this thesis, to the best of our knowledge, we make a first attempt to jointly train representations for news article content as well as social media comments using neural networks with the goal of detecting false claims.

As a state of the art baseline we consider the Conditional Random Field (CRF) model presented in [2]. This model classifies claims on the web by considering a set of articles referring to the claim. The model captures interplay between the language in the articles, the reliability of the source, and the stance of the article towards the claim. The paper reports good results in classifying false claims and the model gained 80% macro average accuracy in their experiments.

Chapter 3

Solution Approach

As both linguistic cues from articles and social media reactions have shown promising results in false claims detection, our models build on using both these features to classify a claim. Shown in Figure 3.1, the model classifying a claim c will retrieve news articles which mention the claim N_c , and associated social media responses S_c . This data is then fed into the model consisting of two main parts.

1. *News Article Model*. Responsible for learning a representation of the news articles. The content of the articles are represented as a k -dimensional vector and fed into a CNN.
2. *Social Media Model*. Responsible for learning a representation of the social media comments for a given news article. All the comments and their form of origin are combined. The comment text is represented as a TF-IDF vector, and the form of origin as a one hot encoded vector. These inputs are then fed into an MLP network.

The final layer of both models are concatenated and fed into a series of dense hidden layers, and finally labeled as true or false using a softmax output layer. To get the final classification of a claim, using late fusion the confidence scores of all the associated articles is averaged to produce a final label.

We experimented with many different variations of these two models. The architectures presented in this chapter was chosen because they provided the best results during extensive testing. The experiments on different models and their performance will be discussed in Chapter 5

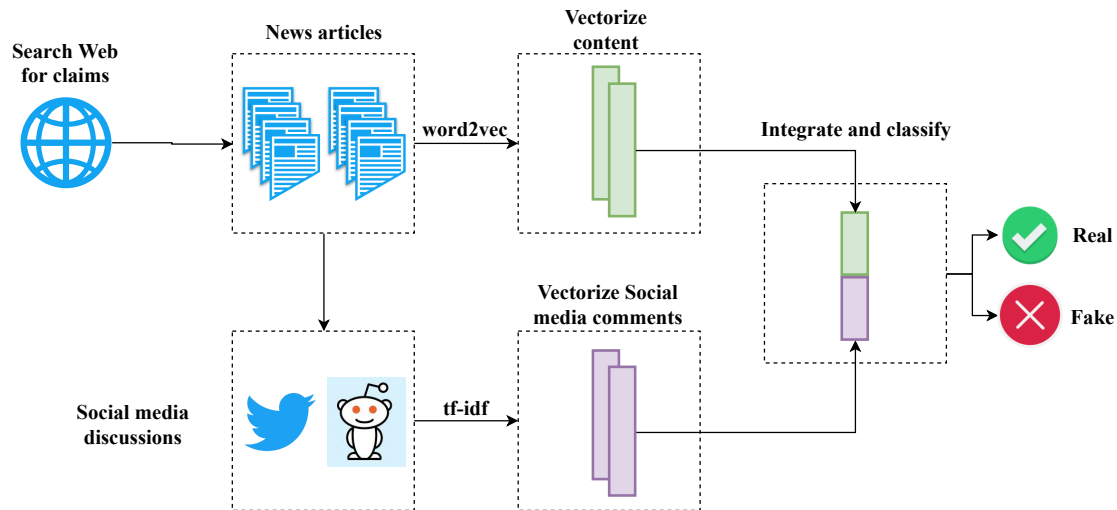


Figure 3.1: Overview of proposed model. Given a claim the model retrieves news articles and social media comments. Vectorize the inputs and integrates them into the submodels for classification of claim.

3.1 News article model

As shown in Figure 3.1 our first model is responsible for learning a representation of the content of the news articles. This model is an implementation of the CNN for sentence classification proposed by Yoon Kim in [11]. The body of the news articles are represented as a sequence of k -dimensional vectors $x_i \in \mathbb{R}^k$ of length n . The sequences are padded if the body is of length less than n . The weights for these vectors are initialized using the weights from a pre-trained word2vec model. Any missing words are initialized with random weights.

As shown in Figure 3.2, these sequences are fed into a fully connected embeddings layer so that their weights are also trainable. Concatenating the n word vectors of k dimensions, forms a $n \times k$ matrix as an input. Then further features are generated using three parallel one-dimensional convolutional operations with varying kernel sizes. Followed by a max-pool layer that further downsamples the input space. The features generated from the convolutional layers are then merged into one feature and fed into two fully connected (dense) layers that produce a final classification.

Dropout [23] is added in between the computational layers, as a regularization technique that randomly “drop out” neurons during training. With a dropout of 0.5, only half the weights of a layer will be updated during training. Preventing overfitting by making the network not rely too heavily on specific nodes.

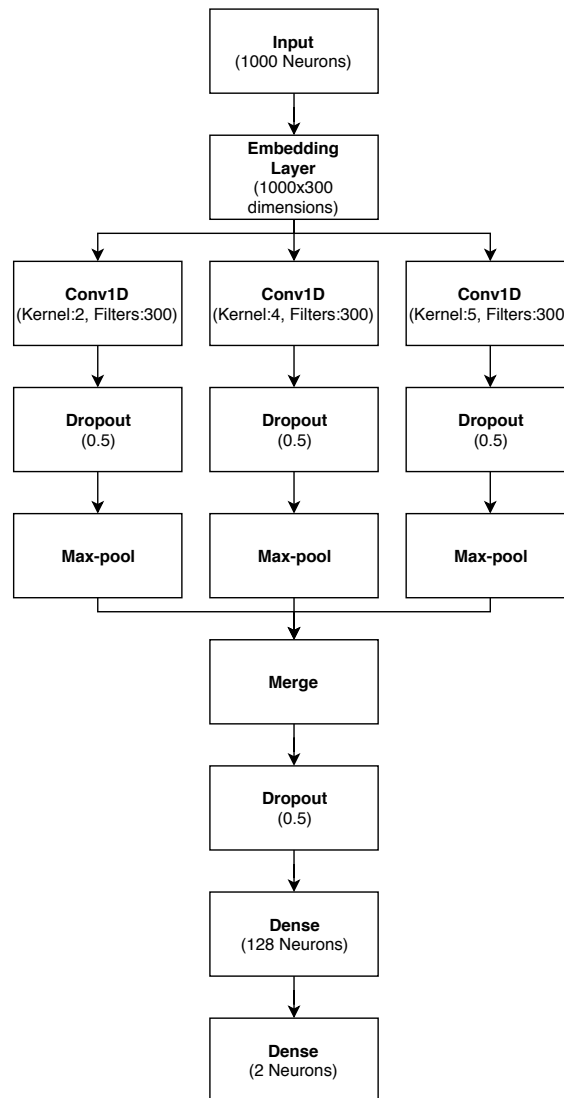


Figure 3.2: Block diagram of the news article model

The final layer, also known as the output layer. Has the softmax activation function, which produces a probability of the input belonging to either the false or real claim class. All the other computational layers have the activation function rectified linear units (ReLU)[24].

3.2 Social media discussion model

This model is responsible for learning a representation of the social media reaction an article receives. The amount of social media messages received per article varies greatly. Also, social messages are often known to contain informal language (slang) and emojis. Therefore instead of relying on pre-trained word vectors, the comments

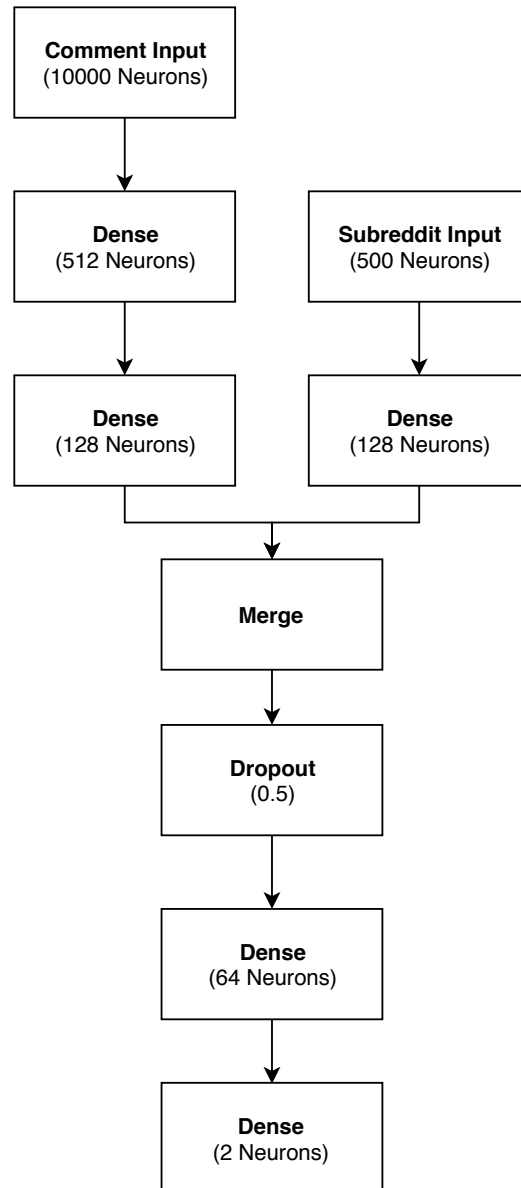


Figure 3.3: Block diagram of the social media model

an article receives are represented as a TF-IDF vector of length n . In addition, the quality and reliability of the comments also rely on the forum in which they are published. For example, comments from specific Reddit subforums (subreddits) such as “conspiracy” have poor reliability. Therefore, we also include an additional feature representing the form the comment originated as a one-hot encoded vector. As seen in Figure 3.3 the data is fed into the network, and in a fully connected fashion classifies the inputs.

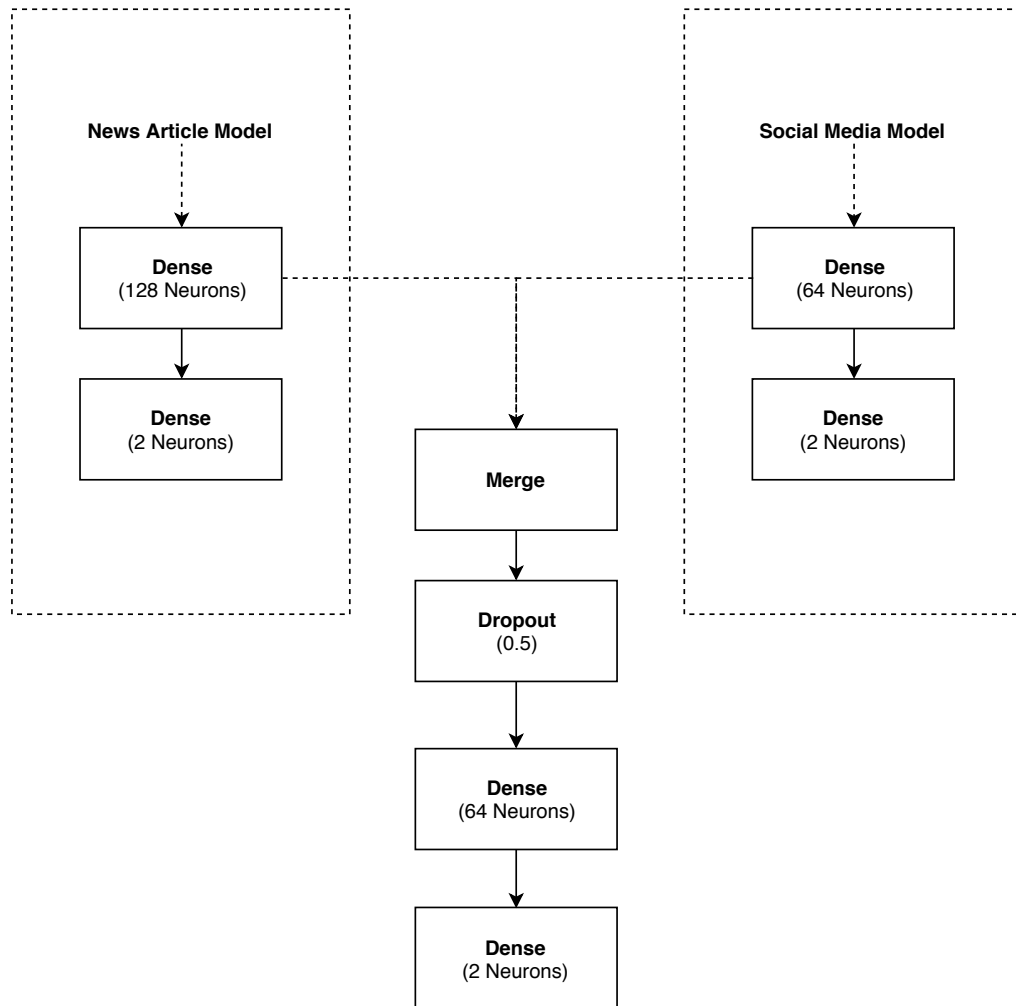


Figure 3.4: Block diagram of the joined model

3.3 Joined model

This model is responsible for using the features from both the news article and the social media model to produce a final classification. As seen in Figure 3.4 the penultimate layer of both models are merged and fed into a series of dense layers with an ReLU activation function, and finally produces a prediction using a softmax activation function. Both the news article and social media model are pre-trained to account for their uneven relationship of data. For example, not all news articles have associated social media comments. Finally, the joined model is trained on data that contain both news articles and social media data.

Chapter 4

Experimental Setup and Data Set

4.1 Environment

Due to its ease of development and large machine learning community, all models and scripts are written in python. To aid the development, we used several python packages and the most important ones are.

- *Keras* [25] A high-level neural network API built on top of popular machine learning frameworks. Chosen due to its ease of implementation and fast prototyping.
- *Tensorflow* [26] The machine learning framework used as a back-end for Keras.
- *NLTK* [27] As a toolkit for working with human language in python.
- *Scikit-learn* [28]. Used for evaluation of the model's prediction results. It is also used for implementing the linear models used as baselines.

All the experiments were run on a Tesla P-100 12gb GPU on the University of Stavanger's Unix system Gorina4 machine.

4.2 Dataset

The data used in our experiments are a combination of different data sets and consist of:

Social Media Analysis for Fake News (SMAFN) A data set collected using the official Twitter API. During the period 21.08.2017 to 12.12.2017, we collected tweets mentioning articles from a set of news publisher domains that is labeled as either trusted or untrusted. The trusted domains were gathered from the study [29], that looks at which news agencies are the most trusted in the United States across different media audiences. Domains labeled as untrusted are known publishers of fake news, the list is curated from Wikipedia’s list of fake news websites¹. All the articles from the trusted domains are labeled as real, and all the articles from the untrusted domains are labeled as false. The dataset contains the most discussed news articles from this set of news domains. In total the dataset contains 10,182 news articles with a distribution of 44% news articles label as false and 190,677 tweets.

Rumor Dataset This dataset contains claims labeled by Snopes.com, politifact.com, and emergent.com. To get relevant articles, similar to [2] we use a search engine to search for articles related to a claim. We also included the search results from [2] into our own data. This method results in a lot of un-relevant data, i.e., articles that do not relate to the claim searched after. To reduce the noise, we filtered out weblinks from well known fact-checking websites and popular social media networks. In addition, we used a pre-trained stance detection model[30] to filter out articles not relevant to a claim. This model was created during the fake news challenge² to determine if an article is unrelated, for, against or discussing its headline.

Reddit Data set, This dataset contains comments from the popular news aggregation forum Reddit. Where users can share and discuss almost any topic, a user can create a post that includes a title and a link to other web content. Other users can then make comments and discuss the various topics that are linked. Posts are organized into “Subreddits” which are smaller communities that discuss topics of interest. One such forum is “r/news“ which

¹https://en.wikipedia.org/wiki/List_of_fake_news_websites

²<http://fakenewschallenge.com>

Labeled by	Articles	Claims	Reddit comments	Article class balance
Snopes	36,271	3096	352,708	73% false claims
Politifact	837	101	28,145	80% false claims
Emergent	250	48	20,073	36% false claims
SMAFN	10182	9352	261,279	44% false claims
Total	47,540	12,597	662,205	66 % false claims

Table 4.1: detailed statistics of dataset

discusses current news headlines. To collect comments discussing articles from the above datasets we used the official Reddit API and queried for posts discussing articles in our dataset.

All the datasets are then merged and used as training data for our models. All the claims are linked to multiple articles and associated Reddit comments. As seen in Table 4.1 the dataset is unbalanced towards fake claims and is a consequence of the fact-checking services nature to research claims that seem likely to be false.

4.2.1 Text preprocessing

All the textual data follows the same preprocessing step. To lose as little data as possible the data cleaning is kept to a minimum. Tomas Mikolov, the creator of word2vec, says this about preprocessing when using word embeddings:

“There is no universal answer. It all depends on what you plan to use the vectors for. In my experience, it is usually good to disconnect (or remove) punctuation from words, and sometimes also convert all characters to lowercase. One can also replace all numbers (possibly greater than some constant) with some single token such as.

All these pre-processing steps aim to reduce the vocabulary size without removing any important content (which in some cases may not be true when you lowercase certain words, ie. Bush is different than Bush, while Another usually has the same sense as another). The smaller the vocabulary is, the lower is the memory complexity, and the more robustly are the parameters for the words estimated. You also have to pre-process the test data in the same way.”³

³<https://groups.google.com/forum/#!msg/word2vec-toolkit/jPfyP6FoB94/tGzZxSc00GsJ>

The preprocessing of both the news articles and Reddit comments text data follows the steps listed below:

1. Lowercase letters
2. Remove web-links in the text.
3. Remove punctuation
4. Remove English stop words, defined by the python package NLTK's stopwords list.

4.2.2 Data analysis

This section provides analysis of the statistics and the textual content of both the news articles and Reddit comments.

News articles

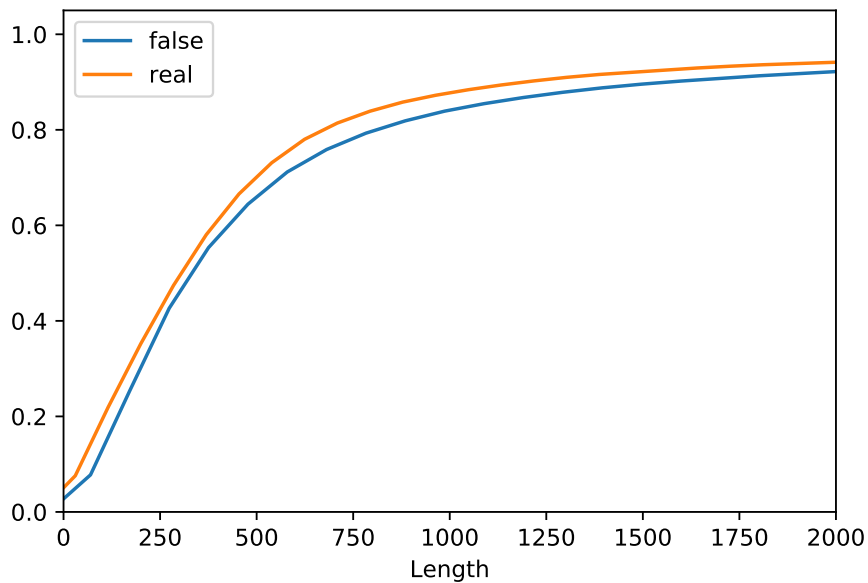


Figure 4.1: Distribution of article length, x axis is the number of words in each article. y-axis is the likelihood of an article containing less than x

Figure 4.1 shows a CDF-plot of the length of the news articles discussing either false or real labels. It can be seen that the average length of an article is around 350 words, and articles containing real claims is a bit longer than its false counterpart.

As 1000 words are enough to get almost all the articles, we set the length of the input sequences to the news article model to be padded to the first 1000 words of an article.

To illustrate what is written in these articles Figure 4.2 shows the top 15 most used words for news articles mentioning false or real claims. To account for the dataset having more articles containing false claims the word occurrences is min-max normalized on the form

$$y_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

where y_i is the normalized word and $x = \{x_1, x_2, \dots, x_i\}$. The words used for each claim is quite similar, and it can be seen from the occurrences of American politicians that both real and false claims have an emphasis on political news. To get a better understanding of what is different between articles mentioning real claims against false claims, Figure 4.2 shows the occurrences of words used by subtracting the occurrences of words with each other. Similar to the overall word frequency it can also be seen here that the articles have an emphasis on political news.

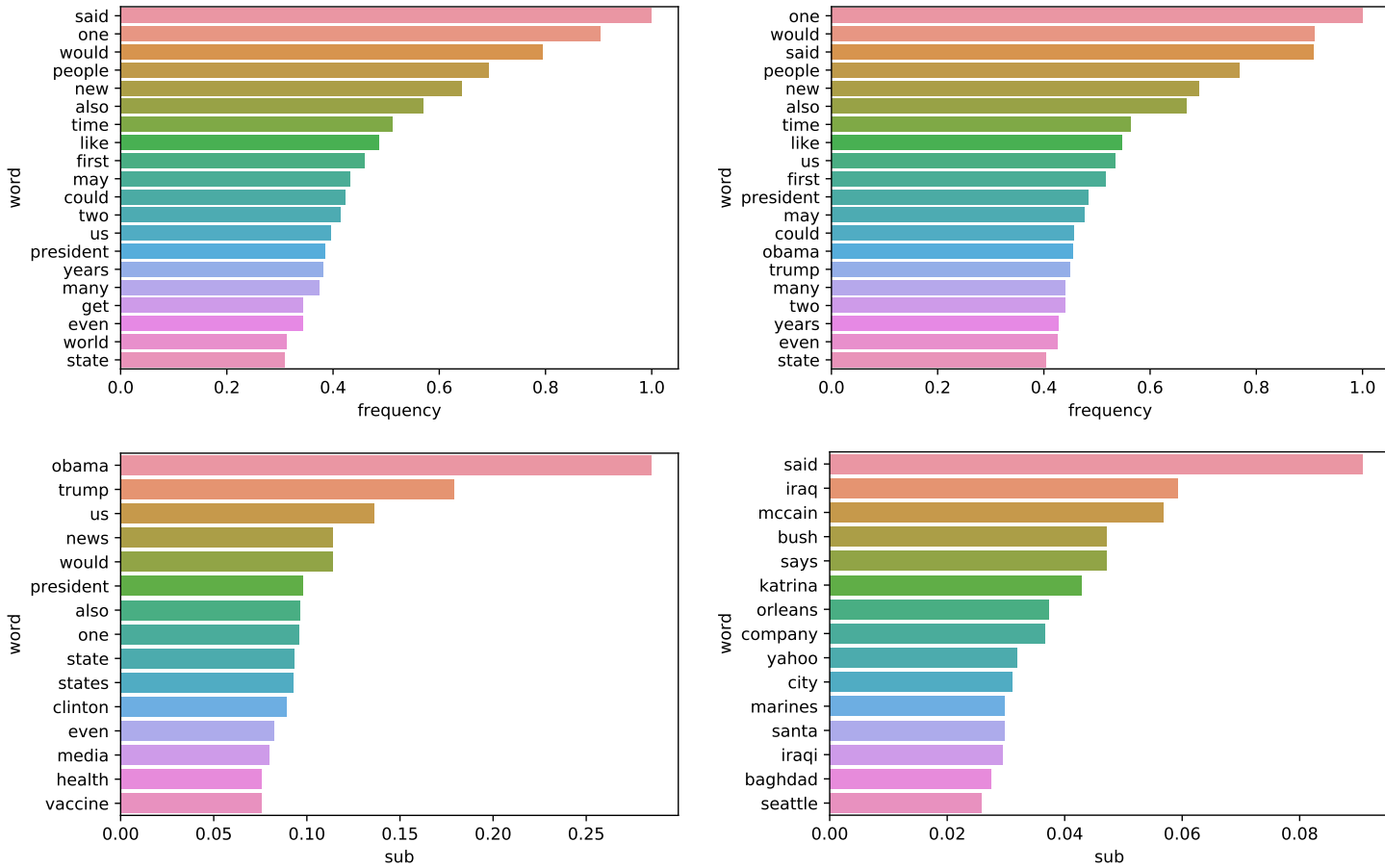


Figure 4.2: Most popular words in used in the news articles. Top Left: Occurrences of words in articles mentioning real claims, Top Right: Occurrences of words in articles mentioning false claims, Bottom Left: Occurrences of words in articles mentioning false claims subtracted by words used in articles mentioning real claims, Bottom Right: Occurrences of words in articles mentioning false claims subtracted by words used in articles mentioning real claims

Social media comments

Figure 4.3 shows an overview on which form (subreddit) the discussion on the claims ends up. Each entry is a post which contains multiple comments, in total the dataset contains 12,930 posts with 6683 discussing false claims and 6247 discussing real claims. As mentioned in Chapter 3 we use this as an additional feature to determine how reliable a comment is. Subtracting the occurrences of each claim Figure 4.3 shows on which subreddit false or real claims is more frequent. For false claims, it can be seen that forms such as “the_donald” and “conspiracy” has a high percentage of claims ending up being false. While for claims that end up to be true, “worldnews” end up coming high.

Figure 4.2 shows an overview of the most used words in the Reddit comments. The frequencies are quite similar but if we subtract the occurrence by the false claims. It can be seen from Figure 4.4 that words such as evidence, Russia, and intelligence have high occurrence over its real claims counterpart. For the real claims, it can be seen from the Figure 4.4 that Minecraft and Microsoft have a high occurrence, this is due to a rumor in our dataset discussing the acquisition of the popular computer game Minecraft by Microsoft. This is a restriction of our model, all rumors containing these words will likely be labeled as real claims. An future improvement of the data can be to clean the data further and remove proper nouns from the textual data.

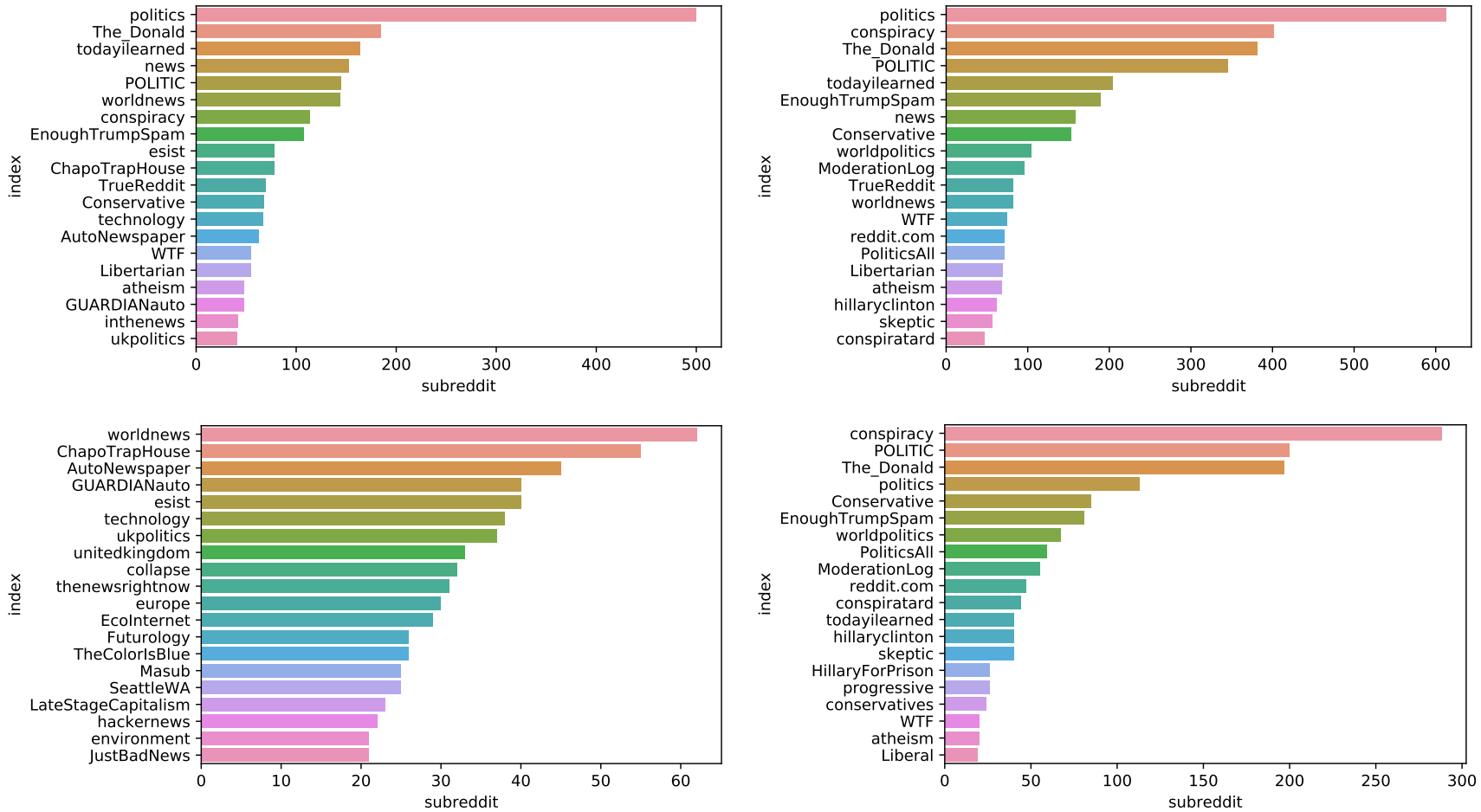


Figure 4.3: Appearances of claims on various forms on Reddit. Top Left: Occurrences of real claims, Top Right: Occurrences of false claims, Bottom Left: Occurrences of real claims subtracted by false claims, Bottom Right: Occurrences of false claims subtracted by real claims

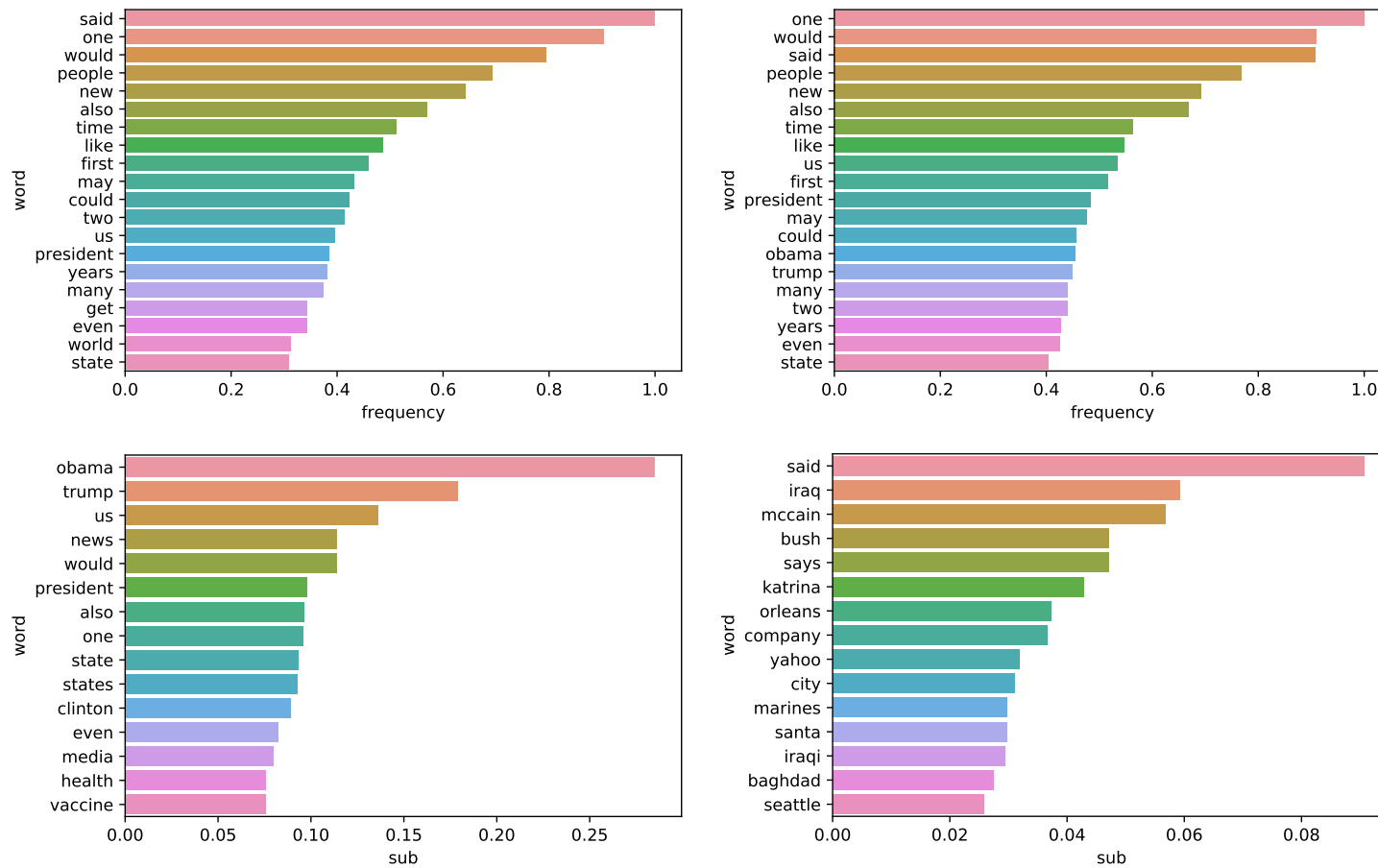


Figure 4.4: Most popular words in used in the Reddit comments. Top Left: Occurrences of words in posts discussing real claims, Top Right: Occurrences of words in posts discussing g false claims, Bottom Left: Occurrences of words in posts discussing false claims subtracted by words used in posts discussing real claims, Bottom Right: Occurrences of words in posts discussing false claims subtracted by words used in posts discussing real claims

Chapter 5

Experimental Results

This chapter presents and discusses the experimental results. Firstly the results and steps done to improve the news article model will be discussed. Secondly, we will present the results of the social media model. Finally, we will present the results of these two models combined and give concrete examples on the model classifying various claims. All the results shown in this chapter is run with 10-Fold cross-validation.

5.1 News article model

This model is responsible for learning a representation of the content of the news articles linked to a claim. The primary task of this model is to get an input vector representing the textual information of a news article and classifying it as either True or False.

There is a plethora of different neural models proposed for classifying a document. Initially, we experimented with two different CNN architectures, one MLP, and one RNN. A short description of their configuration follows.

1. *CNN 1, Yoon-kim.* An Implementation based on Yoon-Kim's CNN [11]. The model is a CNN designed for sentence classification, see Chapter 3 for further details on the model.

2. *CNN 2, Chollet*¹. An implementation of a CNN designed for document classification. Its created by Francois Chollet, author of Keras[25]. Illustrated in 5.1, The network is a 3-layer sequential convolutional neural network followed by a dense layer into a softmax classifier.

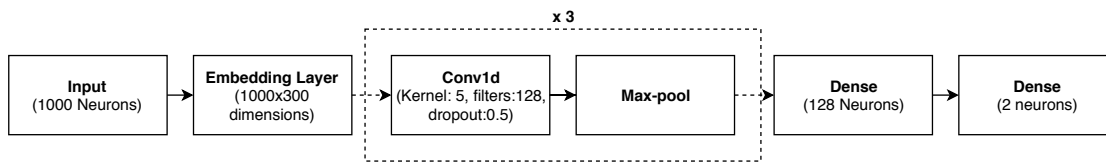


Figure 5.1: Illustration of Chollet’s CNN network tested in news article model

3. *MLP*. Is an implementation of a layer multilayer perceptron for text classification. Illustrated in Figure 5.2, the input layer is a TF-IDF representation of the document, followed by three hidden layers with 512,256, and 128 neurons respectively, with a dropout of 0.5 on all the hidden layers.

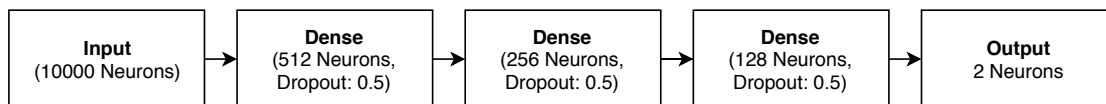


Figure 5.2: Illustration of the MLP network tested in news article model

4. *RNN*. Is an implementation of long short-term memory (LSTM)[31] network. Illustrated in Figure 5.3 the network starts with an input layer which consists of a padded sequence of words of length 1000, followed by an embedding layer, a 16 unit LSTM, and a fully connected layer.

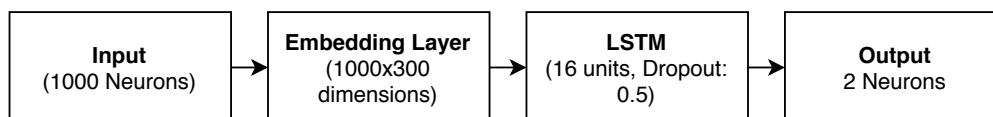


Figure 5.3: Illustration of the RNN network tested in news article model

To better evaluate the performance of the neural models, we also implemented the two linear models support vector machine (SVM) and Naive Bayes.

As shown in Chapter 4 the labels in the dataset is imbalanced towards false claims. To prevent the models to not classify every claim as false the data is undersampled such that the class distribution is even.

¹<https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html>

All the neural models are run for 100 training cycles (epochs) with a mini-batch size of 128. To prevent overfitting the models, they are equipped with early stopping. Which is a mechanism that stops the training if the validation loss does not increase after ten epochs. All the models use the optimizer function Adadelta [32], a per-dimension learning rate method for gradient descent. This method dynamically adapts over time and therefore requires no manual tuning of learning rate and momentum. The models that use an embedding layer are preloaded with the weights from Google’s pre-trained word2vec model ²

The training times of the models vary, per epoch, the Yoon-Kim model uses 13 seconds, Chollet 8 seconds, the MLP 1 second, and the LSTM 142 seconds. The differences between the CNN models and the MLP is negligible, but the considerable training time of the LSTM network is a limiting factor.

To predict a claim, we take the average of the prediction of all the articles relating to a claim. It can be seen from Table 5.1 that the linear models and the neural models’ performance are similar. However, without parameter tuning, Yoon-Kim’s model is performing just ahead of the rest. Therefore we decided to use Yoon-Kim’s models as the base of our news article model. In the following sections, we explore ways to improve the model to get a better performance in the overall news article model.

Configuration	Accuracy		Fake claims			Real claims			AUC
	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	
Yoon-kim	0.802	0.798	0.777	0.873	0.822	0.838	0.724	0.777	0.885
Chollet	0.791	0.791	0.809	0.787	0.798	0.773	0.795	0.784	0.876
SVM	0.793	0.796	0.839	0.748	0.791	0.754	0.843	0.796	0.884
MLP	0.790	0.786	0.768	0.859	0.811	0.822	0.714	0.764	0.872
Naive Bayes	0.769	0.761	0.715	0.927	0.807	0.882	0.595	0.711	0.890
LSTM	0.735	0.726	0.690	0.895	0.779	0.829	0.558	0.667	0.835

Table 5.1: Performance evaluation News Article model

Tuning hyperparameters

To tune the model’s hyperparameters we employed the typical strategy of Grid search[9]. Which is a hyperparameter optimization technique that does an exhaustive search through a manually specified subset of the parameters. This strategy requires some knowledge of the model and its hyperparameters so we used the

²<https://code.google.com/archive/p/word2vec/>

parameters suggested by the paper [17] and explored parameters close to that range. The parameters tested where

- **Weight initialization** Normal, uniform, zero
- **Dropout regularization** 0.3,0.5,0.7
- **Neurons in hidden layer** 64,128,256
- **Filter size** 100, 150,300
- **Kernel size**[[2, 3, 4], [2, 4, 5], [3, 4, 5]]

Due to our optimization algorithm being Adadelta there is no need to search for learning rate and momentum[32]. After grid search the best parameters where

- **Weight initialization** Normal
- **Drop out regularization** 0.5
- **Neurons in hidden layer** 64
- **Filter Size** 300
- **Kernel size**[2, 4, 5]

Configuration	Accuracy		Fake claims			Real claims			AUC
	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	
Tuning	0.823	0.822	0.834	0.831	0.833	0.810	0.814	0.812	0.907
No Tuning	0.802	0.798	0.777	0.873	0.822	0.838	0.724	0.777	0.885

Table 5.2: Performance evaluation after of news article model, after hyper parameter tuning

As shown in Table 5.2 the overall accuracy increased by 2,1 % after hyperparameter optimization.

Configuration	Accuracy		Fake claims			Real claims			AUC
	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	
Class weights	0.815	0.817	0.849	0.793	0.820	0.782	0.841	0.811	0.908
SMOTE	0.819	0.813	0.781	0.914	0.842	0.880	0.712	0.787	0.907
Under sampling	0.823	0.822	0.834	0.831	0.833	0.810	0.814	0.812	0.907

Table 5.3: Performance evaluation of various class balancing techniques

Imbalanced classes

As shown in Chapter 4 the dataset has a bias towards false claims, a problem with class unbalance in neural networks is that it tends to only classify the majority class. There are different strategies to this problem, but the most common ones are undersampling or oversampling the dataset, and a penalized cost to the loss function. The under fitting is done by randomly removing the majority class so that the class balance is even. For oversampling, we employed the well know SMOTE [33] technique, which is a combination of under-sampling the majority class and oversampling the minority class. The class weights are set to be a distribution of the training data, in our case 0.34 for the false claims and 0.66 for the real claims. Seen from Table 5.3 the SMOTE technique over priorities the false claims class while penalizing the class weights over-prioritizes the real claims. Undersampling seems to be a good fit for this data set, however, acquiring more data would be a more optimal solution.

5.2 Social media model

This model is responsible for learning a representation of the social media reactions an article receives. As many social networks such as Facebook and Twitter have restrictions on data scraping, the model is trained on data gathered from Reddit.

Similarly to the news article model we experimented with a few different configurations of only the textual content of the comments. We tested one MLP and one CNN aswell as the two baselines Naive Bayes and SVM. The CNN model is the same as the Yoon-Kim model in the news article model. The MLP structure is presented in chapter 3. It can be seen from Table 5.4 that the models perform on par with each other. However, the TF-IDF MLP network gives us the best overall accuracy hence its the base of the social media model.

Configuration	Accuracy		Fake claims			Real claims			AUC
	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	
MLP	0.657	0.647	0.651	0.740	0.692	0.667	0.567	0.613	0.686
Yoon-kim	0.646	0.647	0.673	0.625	0.648	0.621	0.668	0.644	0.686
Naive Bayes	0.606	0.605	0.652	0.674	0.663	0.647	0.624	0.635	0.661
SVM	0.606	0.605	0.608	0.643	0.625	0.603	0.567	0.584	0.592

Table 5.4: Performance evaluation of Reddit comments

As for hyperparameter optimization on this model, the time ran out for this project. Due to grid search being computationally expensive, this is suggested as future improvements of this thesis.

Additional features

To increase the social media models performance, which form the comments originated from is added as an additional feature. The input for this feature is a one hot encoded array of the 500 most popular subreddits in the training data. As seen by Table 5.5, including the form as an additional feature further enhances the model.

Configuration	Accuracy		Fake claims			Real claims			AUC
	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	
Additoinal	0.670	0.671	0.697	0.637	0.665	0.645	0.705	0.674	0.739
Comments	0.657	0.647	0.651	0.740	0.692	0.667	0.567	0.613	0.686

Table 5.5: Performance evaluation of Social Media model with additional features

5.3 Joined model

To combine the news article model and social media model into one. The output of the penultimate layer of each model is concatenated and fed into a series of dense hidden layers, before going into a softmax classifier. As a baseline we use the CRF model presented in [2], with premission from the author we ran a pre-trained model on our data. It can be seen from Table 5.6 that there is a performance gain by concatenating the two models, a reason for it being so small is that we do not have Reddit comments from all the articles either by there not existing

Configuration	Accuracy		Fake claims			Real claims			AUC
	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	
Joined	0.826	0.825	0.837	0.834	0.836	0.814	0.817	0.812	0.909
Content	0.823	0.822	0.834	0.831	0.833	0.810	0.814	0.812	0.907
Reddit	0.657	0.647	0.651	0.740	0.692	0.667	0.567	0.613	0.686
CRF-Model	0.563	0.552	0.569	0.727	0.638	0.551	0.378	0.449	0.509

Table 5.6: Performance evaluation Joined model

or limitations of the Reddit API. If we take away all the articles not containing Reddit comments, the performance of the joint model over the news article model increases by 2.5%. We can also see that the joined model outperforms the CRF model, and proves that neural networks can be an essential part of detecting false claims on the web. [2] reports an macro-accuracy of 80%, a reason for the CRF model performing badly in our experiments is that we only tested on one fold of their trained models. Our data is also differently pre-processed and may cause a performance hit on their model.

Example

To see how the model is evaluating a specific claim, let’s look at the claim presented in the introduction, “The European Scientific Journal, a peer-reviewed academic publication, concluded that the collapse of the Twin Towers and World Trade Center Building 7 on 11 September 2001 was the result of a controlled demolition”.³ Table 5.7 shows the confidence scores of this claim from the various models. It can be seen that news article model labels this claim as a true. The social media model labels this claim as false. The comments appear in subreddits such as “conspiracy” and “conspiratard” which shown in Section 4.2.2 occurs more often in false claims.

News Article Model	Reddit Model	Joined Model	CRF Model	Label
18% False	99% False	58% False	71 % False	False

Table 5.7: Confidence scores of example claim

³<https://www.snopes.com/fact-check/journal-endorses-911-conspiracy-theory/>

Chapter 6

Conclusion and Future Directions

6.1 Future directions

New Models

This thesis has experimented with a range of various architectures in both the news article and social media model. It is not certain that these structures are optimal, and experimentation with different structures or hyperparameters could provide better results.

Data

A neural model is often limited by its training data. This thesis used data from search engines and Twitter. Although it provided good results, further improvements can be made. The data is still noisy even after cleaning, and more claims can be gathered to provide a more robust model. Social media reactions from other sources such as Twitter and Facebook can also be considered.

Additional features

We experimented with adding additional features to our models. Among these features are

- Title as an additional feature in the news article model, which decreased the models performance. It was implemented as an additional CNN running in parallel with the feature extraction of the body of the article.
- Form of origin in the social media model, which increased the models performance.

Experimenting with more features could help increase the model's performance. The list below contains suggestions for further research.

- Adding domain info as additional features.
- Experimenting with images as an extra feature for the news articles.
- Sequential modeling of Reddit comments.

Web application

Develop a web application that functions as a search engine for false claims verification. The application could provide a simple user interface that lets a user enter a claim they are unsure about. Following that, fetch articles and social media comments discussing the rumor and use our model to label the claim.

6.2 Conclusion

This thesis proposes a neural network for classifying false claims on the web. The model jointly trains two neural network models to represent the textual content of the news articles and social media comments holistically to detect false claims. A set of experiments was conducted to find a optimal structure and hyperparameters of the model.

Using extensive experiments with large-scale manually curated data from fact-checking websites such as snopes.com, politifact.com and, emergent.info we show that our neural network models outperform both standard classifiers such as Support Vector Machines (SVM), Naive Bayes as well as more sophisticated CRF models using linguistic and source-based features. With our best result being 82.6% accuracy.

Bibliography

- [1] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018. ISSN 0036-8075. doi: 10.1126/science.aap9559. URL <http://science.sciencemag.org/content/359/6380/1146>.
- [2] Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *WWW*, pages 1003–1012, 2017. doi: 10.1145/3041021.3055133. URL <http://doi.acm.org/10.1145/3041021.3055133>.
- [3] Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. Credibility assessment of textual claims on the web. In *CIKM*, pages 2173–2178, 2016. doi: 10.1145/2983323.2983661. URL <http://doi.acm.org/10.1145/2983323.2983661>.
- [4] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news. *CoRR*, abs/1708.07104, 2017. URL <http://arxiv.org/abs/1708.07104>.
- [5] www.npr.org/sections/thetwo-way/2016/11/23/503129818/. [Online; accessed 25-Apr-2018].
- [6] Natali Ruchansky, Sungyong Seo, and Yan Liu. CSI: A hybrid deep model for fake news detection. In *CIKM*, pages 797–806, 2017. doi: 10.1145/3132847.3132877. URL <http://doi.acm.org/10.1145/3132847.3132877>.
- [7] Liang Wu and Huan Liu. Tracing fake-news footprints: Characterizing social media messages by how they propagate. In *WSDM*, pages 637–645, 2018. doi: 10.1145/3159652.3159677. URL <http://doi.acm.org/10.1145/3159652.3159677>.

- [8] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.
- [9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [10] Overfitting and Underfitting With Machine Learning Algorithms. <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning/> [Online; accessed 30-May-2018].
- [11] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
- [12] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116, 2017.
- [13] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.
- [14] What Are Word Embeddings? <https://machinelearningmastery.com/what-are-word-embeddings/> [Online; accessed 30-May-2018].
- [15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [16] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [17] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.
- [18] Benjamin D Horne and Sibel Adali. This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *arXiv preprint arXiv:1703.09398*, 2017.

- [19] Fake news challenge. <http://www.fakenewschallenge.org/>. [Online; accessed 25-Apr-2018].
- [20] Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. Detect rumors using time series of social context information on microblogging websites. In *CIKM*, pages 1751–1754, 2015. doi: 10.1145/2806416.2806607. URL <http://doi.acm.org/10.1145/2806416.2806607>.
- [21] Zhe Zhao, Paul Resnick, and Qiaozhu Mei. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *WWW*, pages 1395–1405, 2015. doi: 10.1145/2736277.2741637. URL <http://doi.acm.org/10.1145/2736277.2741637>.
- [22] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824, 2016. URL <http://www.ijcai.org/Abstract/16/537>.
- [23] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- [24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [25] François Chollet et al. Keras. <https://keras.io>, 2015.
- [26] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.

- [27] Steven Bird and Edward Loper. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics, 2004.
- [28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.
- [29] Michael W. Kearney. Trusting news project report 2017. <https://www.rjionline.org/reporthtml.html>, 2017.
- [30] Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *CoRR*, abs/1707.03264, 2017. URL <http://arxiv.org/abs/1707.03264>.
- [31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [32] Matthew D Zeiler. Adadelata: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [33] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.