Automatic Online Fake News Identification by Stance Detection

A Thesis Submitted

In Fulfillment of the Requirements for the

Degree of Bachelor of Science in Computer Science and Engineering

Md. Alkemy Hossain

ID: 16103248

Md. Sarwar Jahan Shuvo

ID: 16103254

to the

Department of Computer Science and Engineering

College of Engineering and Technology (CEAT)

IUBAT-International University of Business Agriculture and Technology

August, 2019

CERTIFICATION

This thesis paper titled **"Automatic Online Fake News Identification by Stance Detection"** submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering in 26th August 2019.

Group Members:

Md. Alkemy Hossain

Md. Sarwar Jahan Shuvo

Supervisor:

_____

Nusrath Tabassum
Lecturer
Department of Computer Science and Engineering
International University of Business Agriculture and Technology

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis paper, titled, **"Automatic Online Fake News Identification by Stance Detection"**, is the outcome of the investigation and research carried out by the following students under the supervision of Nusrath Tabassum, Lecturer, Department of Computer Science and Engineering, International University of Business Agriculture and Technology.

_____

Md. Alkemy Hossain
ID: 16103248

_____

Md. Sarwar Jahan Shuvo
ID: 16103254

ACKNOWLEDGEMENT

At first, we would like to thank Almighty for His blessing. Without His concern nothing can not be possible.

We would like to express our heartiest gratitude to our honorable supervisor, Nusrath Tabassum**,** Lecturer, Department of Computer Science and Engineering, International University of Business Agriculture and Technology, for his guidance, encouragement, motivation and support to prepare this thesis paper by spending his valuable time to review and evaluate this paper.

We are very grateful to the Department of Computer Science and Engineering (CSE) of IUBAT—International University of Business Agriculture and Technology for providing their all-out support during the thesis work. Specially we would like to thank to our Chair Prof Dr. Md. Abdul Haque, Department of Computer Science and Engineering and also Coordination Dr. Utpal Kanti das, Department of Computer Science and Engineering.

Finally, we express our gratitude to our parents and classmates for always being motivating and supportive.

Dhaka

August, 2019

Md. Alkemy Hossain

Md. Sarwar Jahan Shuvo

ABSTRACT

The large use of social media has tremendous impact on our society, culture, business with potentially positive and negative effects. Now-a-days, due to the increase in use of online social networks, the fake news for various commercial and political purposes has been emerging in large numbers and widely spread in the online world. The proliferation and rapid diffusion of fake news on the Internet highlight the need of automatic fake news detection systems. In the context of social networks, machine learning (ML) methods can be used for this purpose. Fake news detection strategies are traditionally either based on content analysis (i.e. analyzing the content of the news) or - more recently - on social context models, such as mapping the news' diffusion pattern. The proliferation of misleading information in everyday access media outlets such as social media feeds, news blogs, and online newspapers have made it challenging to identify trustworthy news sources, thus increasing the need for computational tools able to provide insights into the reliability of online content. In this paper, we focus on the automatic identification of fake content in online news. First, we introduce datasets for the task of fake news detection. We describe the collection, annotation, and validation process and present several exploratory analyses on the identification of linguistic differences in fake and authorized news content. Second, we conduct a set of learning experiments to build accurate fake news detectors. In addition, we provide comparative analyses of the automatic identification of fake news. We are building a classifier that can predict whether a piece of news is fake based on data sources, thereby approaching the problem from a purely NLP perspective. Our goal is to develop a reliable model that classifies a given news article as either fake or true.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1. Introduction

With the advent of technology, information is free for everyone. This is an advancement in human history, but at the same time it blurs the line between true media and maliciously fabricated generated media. A freely available tool to verify the trustworthiness of a news is needed to filter the information we receive every day. With the advent of fake news being used to influence many things, the identification of false information has become an important task. Governments, newspapers and social media platforms are working hard on distinguishing credible news from fake news. The goal of the Fake News Challenge is to automate the process of identifying fake news by using machine learning and natural language processing. This process can be broken down into several stages. Recent development of machine learning provides a possible solution to automate this process. However, accurately and repeatedly identifying fake news is still proven difficult due to the complex nature of human language. With the popularity of online media and detrimental effect of fake news on many aspects of our society, developing a reliable machine learning model for fake news identification becomes very important.

1.1 Fake news and stance detection

Stance is one of the most important indications of news authenticity. In this project, we have studied the stance of the news article headlines on their body texts by predicting the relevance between the headlines and body texts, and if relevant, further identifying the opinion of the title on its body. This is an important part of fake news identification process. A first helpful step towards the identification of fake news is to understand what other news sources are saying about the same topic. That is why the fake news challenge initially focuses on stance detection. Stance detection comprises the estimation of the relative perspectives of two different text pieces on the same topic as described by. Specifically, the task is to estimate the stance of a news headline, relative to the contents of a news article which can but does not have to address the same topic. Thus, the relative stance of each headline-article pair must be classified as either unrelated, discuss, agree or disagree. The discovery of a disagreeing headline-article pair does not necessarily correspond to the discovery of a fake article, but it is an automated first step which could make human reviewers aware of a discrepancy.

Human reviewers or specialized algorithms can then ultimately decide which articles are fake.

1.2 Natural Language Processing (NLP)

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation. Natural language processing (NLP) plays an important role as a communication medium. It is an empirical field of work for any language. To build a program that understands spoken language, we need all facilities of a written language understand as well as enough additional knowledge to handle all kinds of ambiguities. Natural language processing includes understanding and generation, as well as other tasks.

1.3 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

1.4 Motivation

Our goal was to attempt to tackle the growing issue of fake news, which has been exacerbated by the wide-spread use of social media. For example, many believe fake news on social media to be a large contributing factor to results. We wanted to create an easy-to-use system to detect the credibility of a user's claim or article. For this reason, we are intended to work on Automatic Online Fake News Identification.

1.5 Objective

Now a days so many news are generated. Among all the news it is very tough to identify which one is real and which one is fake. Now it has become a challenge. So we are conducted a research on 'Automatic Online Fake News Identification by Stance Detection' that can help people to identify real news and stop spreading rumors. As a result, automating fake news detection is essential to maintain robust online media and social network.

1.5 Contributions of the Work

Now a day's so many news is generated. Among all the news it is very tough to identify which one is real and which one is fake. The main prospects of our proposed system are listed below

- To remove people confusion by reading fake news.
- To stop rumor spreading by fake news.
- To stop getting wrong information by reading fake news.

1.6 Naive Bayes

Using our Naive Bayes algorithm, we identified the top-k tokens that were found to be the most indicative on the classification of the example. This was computed by finding the k/2 tokens which have the highest posterior probability of being in fake news, and the k/2 tokens with the lowest posterior probability of being in fake news. The following expression was used to rank the tokens by Their indication of fake news: Token Rank= $\exp(\varphi j|y=1)/\exp(\varphi j|y=0)$

The k/2 most indicative tokens for each class was used to form a new feature space for our Logistic Regression model. These tokens were also examined heuristically to

ensure they pass the eye-test given our team's knowledge of contemporaneous fake news.

1.7 Support Vector Machine (SVM)

Due to its robustness, a support vector machine (SVM) was used as the second algorithm in our Average-Hypothesis model. The SVM algorithm used uses a hinge loss that seeks to maximize the margin between the two classes of data. The SVM algorithm uses a second-order Gauss kernel that operates on the full 5078 token feature space.

1.8 Logistic Regression

Due to its simplicity and elegance, Logistic Regression (LR) was used as the third algorithm within the Average Hypothesis model. The LR model uses gradient descent to converge onto the optimal set of weights (θ) for the training set. Where J is the loss function and alpha are the learning rate. For our model, the hypothesis used is the sigmoid function.

1.9 Neural Network

A one-layered neural network model was used on the 80 tokens identified to be most causal to a source's classification. The hidden layer neurons use sigmoid activation function and, the output layer uses the SoftMax activation. Also, ReLU and tanh function were tested for the activation function of the hidden layer. Although the results from sigmoid are not good enough to be used as compared to other models discussed above, it was better than ReLU and tanh activation function.

1.10 Datasets

They validated their approach using three different datasets. The first one is the same used in: this allows to easily compare the accuracy of our method with the accuracy of

a purely social-based method. The dataset consists of the public posts and posts' likes of a list of Facebook pages (selection based on) belonging in two categories: scientific news sources vs. conspiracy news sources. The resulting dataset is composed of 15,500 posts, coming from 32 pages (14 conspiracy pages, 18 scientific pages), with more than 2,300,00 likes by 900,000+ users. 8,923 (57.6%) posts are hoaxes and 6,577 (42.4%) are non-hoaxes. Additional details about the dataset are provided by. The second and third datasets come from the Fake Newsnet dataset, recently published by ; They used both the PolitiFact and BuzzFeed news sets they provide: the former contains a ground truth of 240 news (half labeled as fake, half labeled as real by the well-recognized fact-checking website PolitiFact – http://www.politifact.com/subjects/), the latter a ground truth of 182 news (half labeled as fake, half labeled as real by expert opinion of journalists from BuzzFeed – https://www.buzzfeed.com). Both datasets provide, for each news, the text content of the news and the anonymized IDs of the users who posted/spread the news on Twitter (among other information).

1.11 Content-based method

For the Facebook Data dataset, they produced, for each Facebook post, a text corpus joining the actual text content of the post (retrieved using the Facebook Graph APIs – https://developers.facebook.com/docs/graph-api) and, if the post shared a link, the title and text preview of the link (as provided by the Facebook Graph APIs) together with the actual content of the shared webpage. To retrieve the content of a webpage, they applied some simple heuristics: they removed the CSS and JavaScript content from the page, then we extracted the text contained in the remaining HTML tags and, in order to discard useless content (such as menu items), we kept only the lines having more than n words. In this work, we fixed n = 7. Each word of the corpus has then been stemmed and each post has been represented as a vector of TF-IDF frequencies on the stem's vocabulary. Note that we used Python snow ball stemmer (https://pypi.python.org/pypi/snowballstemmer), setting the language to Italian since all the text content of the pages was in Italian Finally, they performed the post classification using a logistic regression model. As for the PolitiFact Data and BuzzFeed Data datasets the content was already available, they used only the text value as provided in and they applied the same classification method, only changing the

stemmer, since the text content of all the news was in English. They used the Porter Stemmer (available at http://www.nltk.org/) in this case.

1.12 Combining social and content signals

There intuition, as discussed in the Introduction, is that social based methods – and in particular the Boolean crowdsourcing algorithms presented in – work extremely well (even with very limited training sets) when they have to classify a post whose number of social interactions is above a certain threshold, while their performance might get worse when only little information about social interactions is available. In these cases, content-based methods can complement them. They therefore defined a threshold λ and classified the posts combining content-based and social-based approaches. In particular, we combined each of the two social-based methods proposed in with the content-based method introduced in the previous section using a simple rule: likes $< \lambda$: use the content-based classifier likes $\geq \lambda$: use the social-based classifier Where likes is the number of users who like a post or, more generally, the number of social interactions collected by the post. The model is intentionally simple, yet it captures the different contributions of the two (alternative) approaches, it guarantees a simple implementation and, as we will show in the results section, its accuracy is higher than the one provided by more sophisticated models. They then evaluated the performances of the combined method using accuracy (the same metric used in) plus some additional metrics that make easier the comparison with other methods in the literature: F1 score, precision, recall. They also carried out a sensitivity analysis to study how the accuracy of our classifier is affected by changes in the threshold λ.

1.13 Fake News Datasets

As highlighted earlier, the datasets used in previous work have either relied on satirical news (e.g., “The Onion”), which also have confounded such as humor or irony; or used fact-checking websites (e.g., “7olitiFact” or “Snopes”), which are typically focused on only one domain (generally politics). They thus decided to construct two new datasets of fake news that cover several news domains and specifically model the deceptive property of fake news without major confounds. One dataset is collected via

crowdsourcing and covers six news domains; the second dataset is obtained directly from the web, and covers celebrity fake news. Guidelines for a Fake News Corpus. In building a fake news dataset, they adhered to the nine requirements of a fake news corpus proposed by (Rubin et al., 2016). Specifically, the authors suggested that such a corpus should (1) include both fake and real news items, (2) contain text-only news items, (3) have a verifiable ground-truth, (4) be homogeneous in length and (5) writing style, (6) contain news from a predefined time frame, (7) be delivered in the same manner and for the same purpose (e.g. humor, breaking news) for fake and real cases, (8) be made publicly available, and (9) should take language and cultural differences into account. In our work, to the extent possible, they aimed to address all of the above guidelines. As outlined in the following, the ground-truth remains challenging since they cannot verify with absolute certainty whether all the content of real news items is in fact true.

1.14 Organization of the Thesis

This thesis is divided into five chapters. This chapter briefly discussed a general overview of topic related to this thesis. In addition, the motivation, objectives and some basic term related to this thesis are presented. In the next chapter 2, an overview of our project related terminologies related to the project and contains brief discussion on previous works that is already implemented with their limitations. Chapter 3 describes elaborately the working procedure of our proposed system with appropriate figure and tables. We also explain authorship detection mechanism with appropriate iteration and figure. In Chapter 4, we have illustrated our implementation of the project and explain the implementation step by step. The graphical representation, abstract view of the system is explained here with necessary figures. In this chapter we also specify the system requirements of the proposed model. Chapter 6. This thesis contains one appendix intended for persons who wish to explore certain topics in greater depth. Appendix A contains the source code of the full project with comments which will be helpful.

Chapter 2
Literature Review

2 Literature Review

2.1 Fake News Identification

Their goal is to develop a reliable model that classifies a given news article as either fake or true. Their model is designed to emulate the functionality of the BS Detector, a popular extension for Chrome that automatically flags articles, websites and content as BS. Their accuracy percentage is 83%.

They used python, Machine Learning and numerous binary algorithm. They use Python and MATLAB to create models.

First they convert their words into numbers using NLP (python) and got dataset then they pre -processed and clean those dataset using Naive Bayes, SVM & logistic regression. They used a 1 layer deep neural network, and 2 layer deep neural network for fake news identification.

Related Work

- Text Processing: Tokenization to contextual clustering with time.
- Machine Learning Tools : Classification algorithms

Dataset and Features

They took data from website called *kaggle.com* with the help of BS detector .They have a total of 12,165 samples which we distributed to train.

- Tokenize the body and headline with the Punkt statement tokenizer from the NLTK NLP library.
- Tokenize words with our algorithm, and take care of lemmatization.
- Tag each sample with the tokens obtained from entire headline set, and body set.

We kept only the tokens that had a frequency more than 10 over the entire title dataset, and for body, we kept only the tokens that had a frequency of more than 200 over the entire dataset. This leaves us with a total of 5261 tokens.

We used **Naive Bayes** to obtain the tokens with high posterior probability, which we then used for deep learning and logistic regression.

Methods

- Average-Hypothesis model (naive bayes with laplace smoothing, SVM, logistic regression)
- Neural Network

Results

Accuracy of 73%

2.2. Towards Automatic identification of Fake News

First they have used SVM (support vector machine) to TF-IDF features to discern whether a headline-article pairing is related or unrelated. Then they employ various neural network architectures on top of LSTMs(Long-Short-Term-Memory Models) to determine agree, disagree, or discuss. They scored .8658 according to FNC-1(fake news challenge 1) performance metric.

Provided dataset:

Stance	Description	% of Provided Data
Agree	article agrees with headline	7.36
Disagree	article disagrees with headline	1.68
Discuss	article discusses same topic as headline (no position)	17.83
Unrelated	article unrelated to headline	73.13

In this paper they suggested two part solution. First suggest a linear classifier to classify headline-article pairs as related or unrelated. Second, we suggest several

neural network architectures built upon Recurrent Neural Network Models (RNNs) to classify related pairings as agree, disagree, or discuss.

FNC-1 Dataset & Scoring Metrics: The dataset consists of 1648 distinct headlines, 1683 distinct articles, and 49972 distinct headline-article pairings. The headlines had various lengths ranging from 10 to 220 words, while articles had lengths ranging from 25 to 5000 words. Metrics to performance on the task is:

$$S_1 = A_{cc}\ \text{Related,unrelated}$$
$$S_2 = A_{cc}\ \text{Agree, disagree, discuss}$$
$$S_{FNG} = .25S1 + .75S2$$

Baseline modes:

Baseline Model	$S_{F\ NC}$
Lexicalized Classifier	.7860
BOW MLP	.7787
LSTM with Concatenated Input	.4005

Methods:

- Split into Two Classification Problems
- Subproblem 1: Related vs Unrelated via Linear Classifier
- LSTM Attention Architectures
    - Conditionally Encoded (CE) LSTMs
    - Adding Global Attention
    - Adding Word-by-Word Attention
    - Bidirectional Global Attention
    - Bidirectional Conditional LSTM with Bidirectional Global Attention
    - Bilateral Matching with Multiple Perspectives
    - Attention Layers for Bilateral Multi-Perspective Matching Model

Results:

Accuracy level is high and it is 0.97 out of 1.

2.3 Fake review detection in yelp

Datasets

The dataset is collected from Yelp.com and firstly used by Rayana and Akoglu and it includes product and user information, timestamp, ratings, and a plaintext review. In their project, they randomly choose equal-sized fake and non-fake reviews from the dataset. They used a total of 16282 reviews and split it into 0.7 training set, 0.2 dev set, and 0.1 test set.

Features

They extract two types of features: review-centric features and reviewer-centric features.

Review-centric features:

1. Structural features : length of the review, average word length, number of sentences, average sentence length, percentage of numerals, percentage of capitalized words.

2. POS percentages

3. Semantic features: We calculate the percentages of positive and negative opinion-bearing words in each review.

4. Unigram features: We extract 100 unigram features from the reviews. More about feature selection in later parts.

5. Bigram features: We extract 100 bigram features.

Reviewer-centric features

1. Maximum number of reviews in a day

2. Percentage of reviews with positive / negative ratings

3. Average review length

4. Standard deviation of ratings of the reviewer's reviews

Their average hypothesis model combines the hypotheses obtained from Nave Bayes, Logistic Regression and SVM by averaging the output probabilities obtained from each model. The aim of averaging is to obtain a model that is less susceptible to over-fitting compared to a model that only uses one of the constituent methods. Given our large feature set consisting of 5,078 features, certain judgment calls were used and validated to integrate this model. Within the Average-Hypothesis model, the Nave Bayes algorithm (which includes Laplace smoothing) and SVM algorithm was run using all 5,078 tokens, while Logistic Regression was performed using only the 20 tokens that were determined to be most indicative to a sample's classification. The following sections delineates the theory used in our implementations of these three learning algorithms.

2.4 Stance Detection For Fake News Identification

The goal of this project is to identify whether given headline-article pairs: **agree, disagree, discuss** the same topic, or are **not related** at all, as described in. Our method feeds the headline-article pairs into a bidirectional LSTM which first analyzes the article and then uses the acquired article representation to analyze the headline. On top of the output of the conditioned bidirectional LSTM, we concatenate global statistical features extracted from the headline-article pairs. We report a 9.7% improvement in the Fake News Challenge evaluation metric and a 22.7% improvement in mean F1 compared to the highest scoring baseline.

Stance detection datasets

The dataset consists of about 50,000 headline-article pairs each labeled with either unrelated, discuss, agree or disagree. 49,972 pairs among them 73.13% are unrelated, 17.83% Discuss, 7.36% agree, and 1.68% Disagree.

Methods:

- Long short term memory (LSTMS)
- Bag of Words
- Convolutional neural networks for n-grams
- Attention mechanisms
- Two-step classification

Collecting Legitimate News. They started by collecting a dataset of legitimate news belonging to six different domains (sports, business, entertainment, politics, technology, and education). The news was obtained from a variety of mainstream news websites (predominantly in the US) such as the ABC News, CNN, USA Today, New York Times, Fox News, Bloomberg, and CNET among others. To ensure the veracity of the news, they conducted manual fact-checking on the news content, which included verifying the news source and cross-referencing information among several sources. Using this approach, they collected 40 news in each of the six domains, for a total of 240 legitimate news. Collecting Fake News using Crowdsourcing. To generate fake versions of the news in the legitimate news dataset, they make use of crowdsourcing via Amazon Mechanical Turk, which has been successfully used in the past for collecting deception data on several domains, including opinion reviews (Ott et al., 2011b), and controversial topics such as abortion and death penalty (P´erez-Rosas and Mihalcea, 2015). However, collecting deceptive data via AMT poses additional challenges on the news domain. First, the reporting language used by journalists might differ from AMT workers language (e.g., journalistic vs. informal style). Second, journalistic articles are usually lengthier than consumer reviews and opinions, thus increasing the difficulty of the task for AMT workers as they would be required to read a full news article and create a fake version from it. To address the former, they asked the workers to the extent possible to emulate a journalistic style in their writing. This decision was motivated by the 5th point of the fake news corpus guidelines described in section 3, which suggests obtaining news with homogeneous writing style. To address the latter, they opted to working with smaller information units. Their approach consists of manually selecting a news excerpt that briefly describes the news article. Thus, from the legitimate news dataset collected earlier, they manually extracted 240

news excerpts. The final dataset consists of 33,378 words. Each news excerpt has on average 139 words and approximately 5 sentences.

They set up an AMT task that asked workers to generate a fake version of the provided news. Each hit included the legitimate news headline and its corresponding body. They instructed workers to produce both a fake headline and a fake news body within the same topic and length as the original news. Workers were also requested to avoid unrealistic content and to keep the names mentioned in the news. The fake news was produced by unique authors, as they allowed only a single submission per worker. They restricted the submission to workers located in the US as they might be more familiar with news published in the US media. In addition, they restricted participation to workers who maintained an approval rate of at least 95% to reduce potential spam contributions. It took approximately five days to collect 240 fake news. Each hit was manually checked for spam and to make sure workers followed the provided guidelines. In general, they received few spam responses and most of the workers followed instructions satisfactorily; the only exceptions were a few cases where they provided only the headline or included unrealistic content. Interestingly, they observed that AMT workers succeeded in mimicking the reporting style from the original news, which may be partly explained by typical verbal mirroring behaviors with drive individuals to produce utterances that match the grammatical structure of sentences they have recently read (Ireland and Pennebaker, 2010). This partially addresses our initial concern of authors reporting style being a source of noise while analyzing news generated by journalists and AMT workers. The final set of fake news consists of 31,990 words. Each fake news has on average 132 words and approximately 5 sentences. Table 1 shows a sample fake news, along with its legitimate version, in the technology domain. Throughout the rest of the paper, we refer to this crowdsourced dataset as FakeNews AMT.

Chapter 3
Methodology

3. Methodology

There are many ways one could attempt to detect fake or biased news on the internet. However, we feel our implementation based on stance detection offers the greatest flexibility and reliability without having to get into the weeds of labeling individual claims as true or false. Rather we aim for a more general approach classifying articles from unknown sources as generally agreeing or generally disagreeing with sources of known (high and low) credibility. Moreover, our implementation is particularly compelling because we can accept user input as either a link to an article OR as any arbitrary claim to be fact checked like (Obama is not a US citizen). In this way our program acts as a fact-finding search engine and returns links to relevant articles along with the article's stance (agree/disagree/is-neutral) on that topic! Our program offers tremendous research and discovery potential to users as well as simply checking claims. We wanted to create an easy-to-use system to detect the credibility of a user's claim or article, based on the concept of *stance detection*. Fake news is tough to identify. Many 'facts' are highly complex and difficult to check, or exist on a 'continuum of truth' or are compound sentences with fact and fiction overlapping. The best way to attack this problem is not through fact checking, but by comparing how reputable sources feel about a claim.

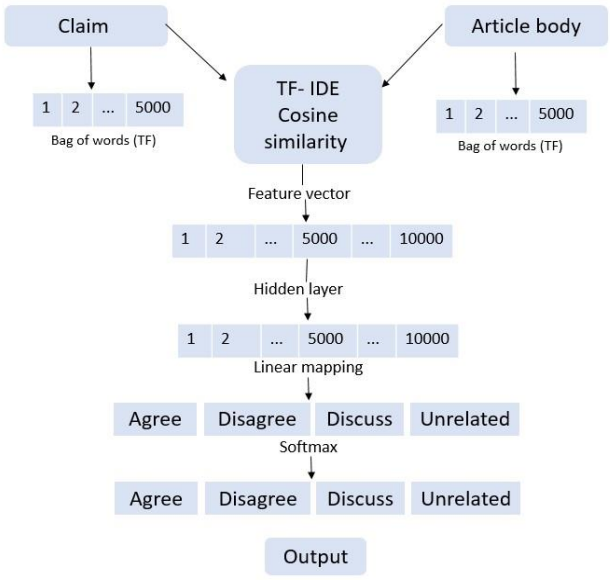

Figure 1 Methodology

We created and implemented a machine learning model in Tensorflow that's based off of several research papers in the field of stance detection. Our model uses a combination of Bag-of-Words, Google's word-2-vec, TF, TF-IDF (Term Frequency, Inverse Document Frequency), and 'stopwords' inside Scikit-learn to vectorize our input. That is run through a single hidden layer with ReLU activation, a fully connected layer and a softmax activation function to produce one of 4 outputs. We are comparing an arbitrary body of text to an arbitrary claim. So our ML outputs whether or not our body of text is 'related' or 'unrelated' to the claim. If it's related, then it outputs if the body 'agrees', 'disagrees' or 'is neutral towards' our claim. Our model achieved 82% accuracy on our test data (for pure stance detection. Not necessarily 'fake news' detection).

One challenge we faced was the relatively long running time of our program. Focusing in on the machine learning, initializing our model took roughly 10 seconds on a recent laptop. To help mitigate wait time, we began loading our model as soon as a visitor entered the site. Therefore, it was usually available before the user even submitted a search request. Once the articles were retrieved, the model only took about 5 seconds to compare our typical thousands of articles to the user's claim before returning the results. These results, how the user's claim compared (agree/disagree) to our reference articles were then passed into our source reputability engine to compute a final score: fake news or not. All subsequent searches from the same user would be conducted on the same session of that model, so no further computation or wait time is required.

3.2 Building a Web Dataset

They collected a second dataset of fake news from web sources following similar guidelines as in the previous dataset. However, this time, they aimed to identify fake content that naturally occurs on the web. They opted for collecting news from public figures as they are frequently targeted by rumors, hoaxes, and fake reports. They focused mainly on celebrities (actors, singers, socialites, and politicians) and their sources include online magazines such as Entertainment Weekly, People Magazine, Radar Online, among other tabloid and entertainment-oriented publications. The data were collected in pairs, with one article being legitimate and the other fake. In order to determine if a given celebrity news was legitimate or not, the claims made in the article

were evaluated using gossip-checking sites such as ”GossipCop.com”, and were cross-referenced with information from other sources. During the initial stages of the data collection, they noticed that celebrity news tends to center on sensational topics that sources believe readers want to read about, such as divorces, pregnancies, and fights. Consequently, celebrity news tends to follow certain celebrities more than others further leading to an inherent lack in topic diversity in celebrity news. To address this issue, they evaluated several sources to make sure we obtain a diversified pool of celebrities and topics. Upon beginning the data collection procedure using these guidelines, another characteristic surfaced: several pairs contained nearly the same information with similar lexicon and reporting style, with differences being as simple as just negating the false news. For example, the following headlines correspond to a news pair where the legitimate version only negates the fake version: “Aniston gets into fight with husband” (fake) and “Aniston did NOT get into fight with husband” (legitimate). To address this issue, they sought to identify related news that still followed the fake-legitimate pair property while being sufficiently diverse in lexicon and tone. In the former example, the fake news was paired with an article titled “Aniston and Husband enjoy dinner” that was published on the date of the alleged fight. Using this approach, they collected 100 fake news articles and 100 legitimate news articles in the celebrity domain. The final fake news set has an average of 399 words and 17 sentences per article, for a total of 39,940 words. The corresponding legitimate news set has an average of 709 words and 33 sentences per article, for a total of 70,975 words. 2 shows an example of an article pairing in the dataset. Throughout the rest of the paper, we refer to this web dataset as Celebrity.

3.3 Model description

Four different types of classification models, including support vector machines (SVM) (linear and nonlinear), softmax, multinomial Naive Bayes, and multilayer perceptron classifier (MLP) are leveraged for this task. A combination of these models are also tested in order to further improve the accuracy of prediction. Using scikit learn , these models are implemented to learn from the training data using k-fold (k=10) cross-validation, and then predict using the test sets. Bidirectional LSTMs have been successfully used across different natural language processing tasks.

Thus, we adopt a bidirectional LSTM for fake news challenge stance detection task. Our model is depicted in Each headline-article pair is processed as follows.

Let (xH 1 ; xH 2 ; :::; xH n) denote the sequence of word vectors corresponding to words in the headline and (xA 1 ; xA 2 ; :::; xA m) denote the same for words in the article. Each word is represented by a D dimensional word embedding that was pre trained using GloVe. Using a bidirectional LSTM we first encode the article as: at = biLSTMA(at−1; xA t ; xA m−t+1), where we initialize the LSTMs with a zero state. The biLSTM consists of 2 LSTMs. In step t one takes in xA t and the other one takes in xA m−t+1. We define the article encoding as A = [a1; :::; am] 2 R2d×m. Similar to , we then initialize a second bidirectional LSTM biLSTMH with the last state of the first LSTM am and extract the headline encoding as ht = biLSTMH(ht−1; xH t ; xH m−t+1). We define the headline encoding as H = [h1; :::; hn] 2 R2d×n. The concatenation of the last states of the forward and backward pass of the article LSTM am 2 R2d and the headline LSTM hn 2 R2d encode the local word embeddings of the headline and article of our model. As the articles tend to be quite long, we condition the headlines on the articles and not vice versa in order to avoid gradients vanishing before they reach the first LSTM. In this regard, it also helps to feed in the outputs of the first LSTM directly into the hidden layer. To extract global features, we follow the baseline method of [1] and include the features described in Table 6 on page 11, which will be denoted as g 2 Rf, where f is the number of features. The global and the local features are then concatenated to form a vector that is multiplied by W 2 R4×4d+f in the softmax layer to produce the classification output c of our model as follows:

c = softmax (W[hn; am; g] + b) 2 R4 (1) R4 corresponds to the dimension of our labels agree, disagree, discuss, and unrelated. Given c we optimize over a cross entropy loss and train the model with the Adam optimization method [10].

Support vector machines are learning algorithms which convert features to points in high dimensional space and divide points from different categories by a gap as wide as possible.

Specifically, SVM solves the following optimization problem:

With kernels, SVM can also perform nonlinear classification. As a multinomial generalization of the logistic regression, softmax is a classical method for classification when there are more than two categories. It is intuitive to implement softmax in this problem, when considering the relatedness and the stances of the news as outputs. With a certain parameter θ,

the probability of the output classified to class naïve is listed as the following:

Multinomial naïve Bayes defines a generative process for the data set and assumes that for features, p(x1|y=c), p(x2|y=c), ..., p(xn|y=c) are independent given a specific category label y=c. Therefore, the joint probability of all features conditioned on y=c is the product of each feature conditioned on y=c.

Then the maximum likelihood and predictions of new data could be calculated through Bayes theorem. Due to these properties, naïve Bayes classifier is often used in text classification problems in which the order of words does not matter.

Consisting of an input layer, an output layer, multiple hidden layers each with multiple neurons, neural network is a very powerful tool for text stance classification as it relies less on accuracy of feature extraction and can work on some crude features. As one of the neural network models, multi-layer perceptron algorithm takes all the features as the input, return classification as output and use backpropagation for training. In this project, ReLU function g(z) = max(z, 0) is used as activation for each neuron of the neural network.

3.4 Tools

To implement the system we need system requirements. The main objective of the proposed system is to identify fake news of a given claim or news body. This is a software which perform this operational task. As the software run in an operating system so need to specify the requirements to run this software. There are three type of requirements need to run this project hardware specification, necessary software tools and the font type. The system requirements for the proposed model is listed below:

We chose to use papersapce to train and run our machine learning model primarially because of how quick and easy it was to get a paperspace machine up and running for machine learning. Therefore, speed was of the essence and the MLL-in-a-box preset saved us a significant amount of time while trying to get our TensorFlow moel up and running.

We used Paper space because it is:

1. Fast to setup

   ML in a box climates the nightmare that setting up CUDA can become.

   Graphical mode is very useful when first setting up a computer

2. Easy to use

   With Windows app terminals are dead simple to access

3. Plentiful V-RAM for a bargain

Our model required roughly 12 GB of V-RAM which made simply the quantity of V-RAM required our biggest limiting factor when choosing a GPU.

As of posting this article they're way cheaper than the competition (AWS, Azure) for more V-ram and faster cards.

3.5 Interface

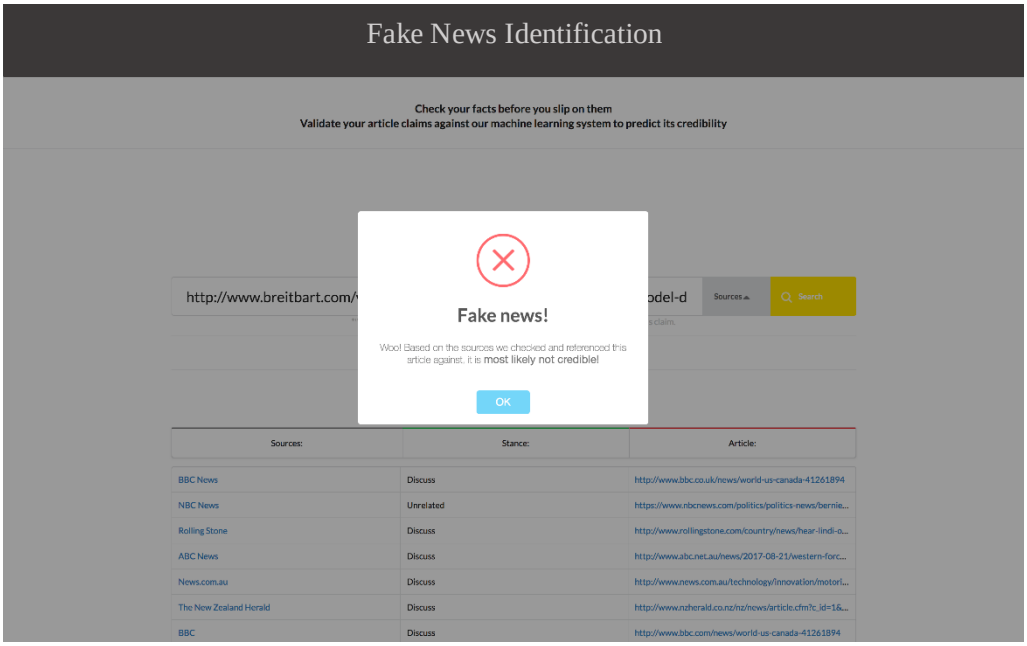

Figure 2: Project UI

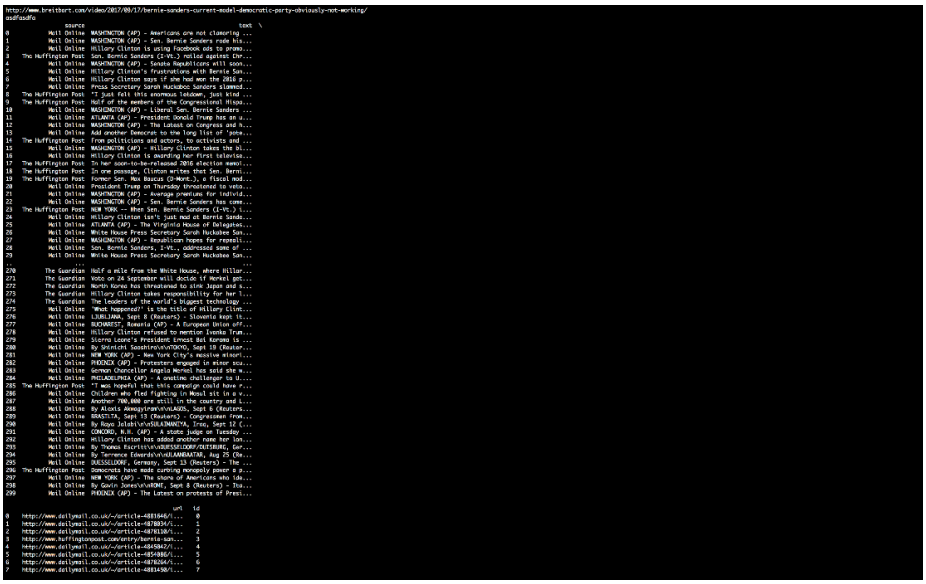

Figure 3: Output in Terminal

3.6 Program pipeline

● Users input a claim like “Obama is not a US citizen”

● Our program will search Event Registry’s database for thousands of articles related to the keywords.

● We run those articles through our home-grown stance detection machine learning model which will determine each article’s relevance to the claim and it’s stance on it. We determine if an article agrees/disagrees/is-neutral or is unrelated to the input claim.

● We then access our ever-evolving database of source reputability. If lots of reputable sources all agree with your claim, then it’s probably true!

● Then we cite our sources so our users can click through and read more about that topic!

3.7 Bag of Words

Some of our experiments were based on a completely different approach based on bag of words (BoW). Here we describe the most successful model of this kind. A diagram of this model can be seen in Appendix A (Figure 5 on page 11). For word representation, we used a 50-dimensional version of the pre-trained GloVe vectors [3] used in our other models. For each headline-body pair, stop words are removed from both the headline and the body. The body is split up into sentences and the average word vector is calculated for each sentence. A corresponding vector is calculated for the headline. We then calculate the cosine similarity of the headline vector to each body sentence vector and pick the 3 with the highest similarity. Those vectors as well as the headline vector are then concatenated to create the input vector for our classifier. Optionally we concatenated the global features to the input vector as well. The input vector is then fed into a neural network with a single 100-unit ReLU hidden layer and a softmax output layer. The BoW model performs surprisingly well given its simplicity and only performs slightly worse than our full model, see Table 2 on the preceding page. A confusion matrix can be seen on Figure 4 on page 10 in Appendix A. The model seems to capture similar information as the global features because adding them only gives a small boost to the performance unlike with the LSTM.

3.8 Parsing input and fetching articles

Given a user URL or claim, we used Microsoft's Azure Cognitive and IBM's Natural Language Processing to parse the article or claim and perform keyword extraction. We then used combinations of the keywords to collect up to a few thousand articles from Event Registry's database to pass on to the machine learning model. Here we aired on the side of collecting more rather than fewer articles because the machine learning will accurately determine relevance further in the pipeline.

After combing through numerous newspaper and natural language processing APIs, we discovered that the best way to find related articles is by searching for keywords. The challenge was implementing a natural language processing algorithm that extracted the most relevant keywords that were searchable, and to extract just the right number of keywords. Many algorithms were simply summarizers, and would return well over 50 keywords, which would be too many to search with. On top of that, many algorithms were resource exhaustive and would sometimes take up to a minute to parse a given text. In the end, we implemented both Microsoft's Azure and IBM's Watson to process, parse, and extract keywords given the URL to a news article or a claim. We passed the extracted keywords to Event Registry's incredible database of almost 200 million articles to find as many related articles as possible.

With more time, we would love to implement Event Registry's data visualization capabilities which include generating tag clouds and graphs showing top news publishers given a topic.

3.9 Source Reputability Database

In order for our application to work, we needed to be able to compare new stances to our ever-improving database of source reputability. We wrote a python script to keep track of all encountered sources along with a reputation score of calculated weight. As a start, we hard-coded reputations based off nationwide research studies, and then every time we ran our algorithm, we added any new encountered sources to our database. In order to do this, we calculated a reputation score for each new article by comparing its stance towards the input claim with the stances of sources with known reputation and averaging the result. In the future we hope to incorporate more accurate data-science

techniques to improve our database. As a smaller project, we also hope to figure out a more streamlined approach than keeping track of the database with. csv's by having a copy of the database exist outside of a single run of the application.

After setting up the baseline model of [1], we first use a model similar to the model of [5]. First, we encode the headline using a bidirectional LSTM. Then, we encode the article with another bidirectional LSTM conditioned on the output of the headline LSTM. To our surprise, processing the article first and conditioning the headline on the article encoding worked better than vice versa for our dataset. Just by switching the order in which the article and headline are processed, we were able to increase our performance from a 57.8% score and 41.1% mean F1 score to a 65.3% score and 50.2% mean F1 score. Further fine-tuning, such as optimizing the number of hidden units and the dimension of word embeddings, balancing the dataset during training, and truncating the articles as mentioned in the implementation details maxes out the performance of our bidirectional LSTMs at 70.5% score and 51.4% mean F1 score. On the other hand, our previous analysis of the baseline of [1] has shown that its features are especially useful for discriminating related from unrelated articles, but not the other classes, as can be seen in Table 3. To leverage the baseline's classification accuracy of related and unrelated articles, we include the features of [1] in our model by concatenating them to the LSTM features before the softmax computation. As can be seen in Table 2 our final conditioned bidirectional LSTM model with global features outperforms all other baselines and models with an overall score of 87.4% and a mean F1 score of 69.5%. Figure 2a shows that the concatenation of global features with the bidirectional LSTM features effectively reduces the number of false positives for the unrelated category compared to using the bidirectional LSTM only. At the same time, the LSTM is now able to better focus on the discrimination of the agree, disagree and discuss categories instead of having to deal with the related/unrelated discrimination. Yet, most confusions happen in the agree and disagree categories, which is expected given that these two categories have the lowest number of examples in the dataset, as can be seen in Table 1. As a result, these two categories are biased towards being classified as discuss and confusions between agree and disagree are frequent as well

Chapter 4
Result and Discussion

4 Result

The metric is a weighted accuracy score, with 25% weight on correctly classifying "related" stances, which includes "agree", "disagree" and "discuss", and "unrelated" stances, and 75% weight on correctly classifying three "related" stances.

4.1.1 Effect of Feature Extraction on Performance

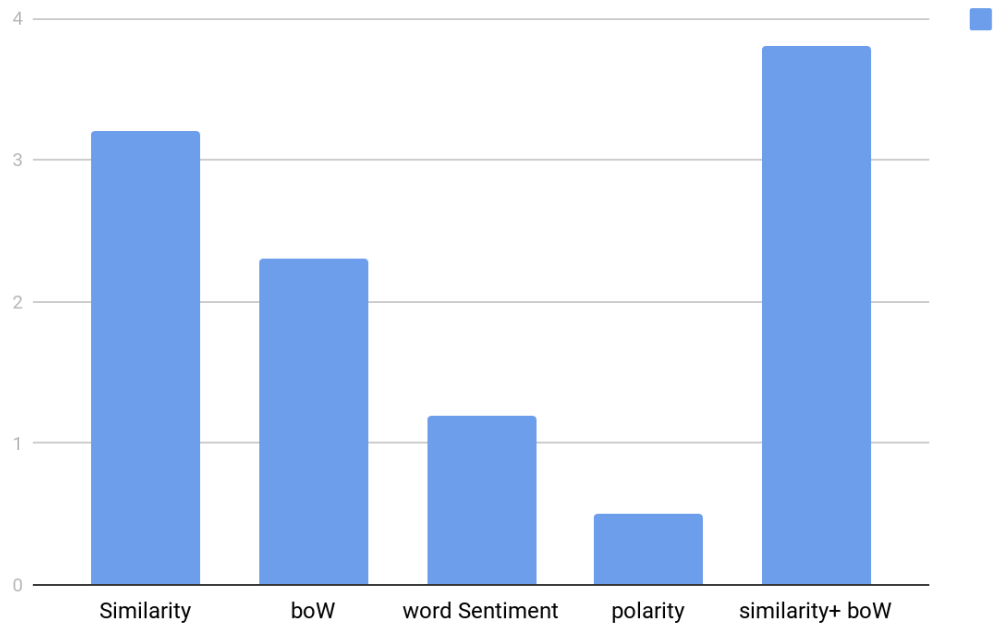

Figure 4: Performance improvement after adding the following features: similarity, bag-of-word (BOW), and sentiment features.

The performance improvement percentage from different features is shown in Figure 1. It is found that the "similarity" and "bow" features better describe the stances of the headlines towards the bodies, than the word "sentiments" and "polarity features". The addition of both "similarity" and "bow" features improve the performance better than adding a single type of features.

4.2 Discussion

4.2.1 Model Performance on Each Category

A summary of all models is shown in Table 1. Overall, all models have above 90% accuracy on prediction of unrelated stance and below 5% accuracy on disagree stance. Moreover, all models have around 80% accuracy rate on the discussed stance. This discrepancy could be explained by a) the difference in the number of test instances, b) feature extraction, and c) model parameters. For 25413 test instances, 18349 has stance unrelated and 7064 related. Related instances contain 1903 agree, 697 disagree, and 4464 discussed. Data with disagree stance are significantly less than data with unrelated stance. Lack of training data might contribute to the low accuracy rate of disagree stance, and agree stance. Also, extracted features such as overlapped words or cosine similarity focus more on relevance between each pair of headline and body rather than positive and negative attitudes. Finally, model parameters, such as the numbers of layers and nodes in the MLP model, are yet to be optimized to improve the accuracy of classification. The accuracy rate for all models in test set is shown in Table 2. Across different models, MLP Classifier has the overall best performance. Softmax and linear SVM have better performance on unrelated stance compared with non-linear models, such as SVM with RBF kernel or MLP classifier. Among related stances (agree, disagree, and discussed), MLP works best compared with other models. Multinomial naive Bayes has comparatively better performance on agree stance and disagree stance, and MLP Classifier has the best performance on discussed stance.

Table 1: Test set labels output by multilayer perceptron (MLP), softmax(SF), multinomial naive Bayes (MNB) and support vector machine (SVM)

A\P	**Agree**	**Disagree**	**Discuss**	**Unrelated**
Agree	**147** (MLP) **116** (SF) **378** (MNB) **97** (SVM)	0 4 39 0	1528 1446 1246 1513	228 337 240 293
Disagree	34 27 62 12	**0** **0** **11** **0**	444 381 436 420	219 289 188 265
Discuss	198 122 625 86	0 0 38 0	**3761** **3556** **3168** **3691**	505 786 633 687
Unrelated	0 7 70 6	0 0 0 0	304 168 1096 202	**18045** **18174** **17183** **18141**

Table 2: Accuracy rate of each stance for all models. Here, the percentage for each category is the percentage accuracy for the test set. The total scores

Accuracy Rate	**Linear SVM**	**Softmax**	**Multinomial Naive Bayes**	**MLP Classifier**
Unrelated	**99%**	**99%**	94%	98%
Related	82%	80%	85%	**87%**
Agree	5%	6%	**20%**	8%
Disagree	0%	0%	**2%**	0%
Discussed	83%	80%	71%	**84%**
Total Score	75.89%	74.76%	72.48%	**77.74%**

4.2.2 Performance of model combination

Single models alone have below 20% accuracy on classification of “agree” stances and “disagree” stances and different models show advantages and disadvantages on classification of different stances. We proposed the reason behind the bad performance of the model to be the disproportionate number of the “agree”, “disagree” news instances versus the “unrelated” instances. Therefore, the algorithms tend to predict more “test” newspaper headlines to be “unrelated” to its bodies. To handle this problem, we decided to use a sub-category classification here. The idea is to use a method to classify “related” from “unrelated” first, and use other methods to do further classifications. Two types of model combinations are proposed to specify the classification process in more details. The two-model combination splits the stance detection task into two classification subtasks and each subtask is completed by a classification model. The first subtask is to classify all headline-body pairs into unrelated and related stances. Related stances include “agree”, “disagree”, and “discuss” stances. For headline-body pairs with related stances, the second subtask

further classifies “agree”, “disagree”, and “discuss” stances. The three-model combination splits the task into three subtasks, each completed by a classification model. The first model classifies “related” and “unrelated” stances. For “related” stances, the second model classifies whether a stance is neutral(“discuss” stance) or not. For non-neural stances, the third model classifies whether a stance is “agree” or “disagree”. Models in the combination could use different feature sets. For example, in the 3-model combination, the BOW feature is helpful for the first two classification subtasks. When the BOW feature is applied to the classification of “agree” and “disagree” stances, overfitting is observed and damages the overall performance. Overall, two-model combinations and three-model combinations achieved above 75% score on the test set. In the three-model combination, when the third task is completed by multinomial naive bayes, classification of “agree” and “disagree” stances will be significantly improved and the same for the overall performance. The overall performance for SVM+MLP+SVM combination has achieved the highest score of 78.46%

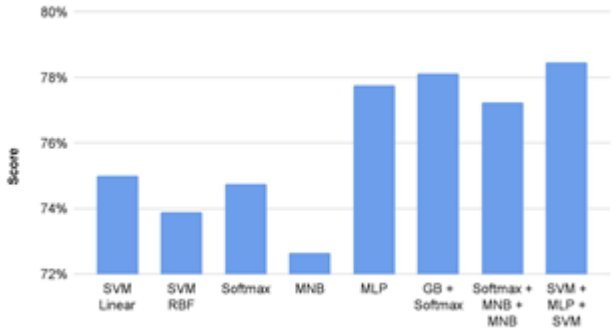

Figure 5: Comparison of test set score generated from different models GB = Gradient Boosting Classifier. LR = Logistic Regression (when softmax is applied to binary classification). MNB = Multinomial Naive Bayes. M1+M2(+M3) = 2(3)-model combination.

.

.

Chapter 5

Conclusion and Future Work

5. Conclusion

In this paper, we provide a comprehensive repository which contains information from news content, social context, and spatiotemporal information. We propose a principled strategy to collect relevant data from different sources. Compared with single model, splitting the stance detection task into two or three subtasks and utilizing combination of models improved the overall performance. In most cases, we obtained a range of accuracy values between 80% and 82%. Among different features, combination of cosine similarity and bow features significantly improved the performance.

5.1 Future Improvement

As future work, we would like to extend our approach. Prediction Improvement on Distinguishing Agree vs. Disagree Categories.

The current project did not include domain knowledge related features, such as entity relationships. Future studies could extract name entities from each pair of news headline and news body and analyze their relationships through a knowledge base. News articles contain a lot of named entities that can result in unknown words, a pointer method similar to could help resolve unknown words and better link the headline to the article body.

And lastly, we would like to extend in Bangla news articles.

References

[1] Mrowca, D., Wang, E., & Kosson, A. (2017). Stance detection for fake news identification.

[2] Davis, Richard, and Chris Proctor. "Fake News, Real Consequences: Recruiting Neural Networks for the Fight Against Fake News." [2] Fake News Detection Challenge: FNC-1. <http://www.fakenewschallenge.org>

[3] Mohammad, Saif M., and Peter D. Turney.

"Emotions evoked by common words and phrases:

[4] Using Mechanical Turk to create an emotion lexicon." *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*. Association for Computational Linguistics, 2010.

[5] Shu, K., Mahudeswaran, D., Wang, S., Lee, D., & Liu, H. (2018). Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*.

[6] Rubin, V., Conroy, N., Chen, Y., & Cornwell, S. (2016, June). Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the second workshop on computational approaches to deception detection* (pp. 7-17).

[7] Mone, S., Choudhary, D., & Singhania, A. FAKE NEWS IDENTIFICATION CS 229: MACHINE LEARNING: GROUP 621.

[8] Della Vedova, M. L., Tacchini, E., Moret, S., Ballarin, G., DiPierro, M., & de Alfaro, L. (2018, May). Automatic online fake news detection combining content and social signals. In *2018 22nd Conference of Open Innovations Association (FRUCT)* (pp. 272-279). IEEE.

[9] Pérez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2017). Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Appendix

```
# import numpy as np
import pandas as pd
import random
import tensorflow as tf
import time
# import local packages
# import rep
# import webscraper
from ml import ourModel
from ml import util


print("Pipeline running...")


####################################################
################### ML INIT CODE ###################
####################################################
# Set file names
file_train_instances = "ml/train_stances.csv"
file_train_bodies = "ml/train_bodies.csv"
file_test_instances = "ml/test_stances_unlabeled.csv"
file_test_bodies = "ml/test_bodies.csv"


file_predictions = 'ml/ML_predictions.csv'




# Initialise hyperparameters
r = random.Random()
lim_unigram = 5000
target_size = 4
hidden_size = 100
train_keep_prob = 0.6
l2_alpha = 0.00001
```

```
learn_rate = 0.01
clip_ratio = 5
batch_size_train = 500
epochs = 90



# Load data sets
raw_train = util.FNCData(file_train_instances,
file_train_bodies)
raw_test = util.FNCData(file_test_instances,
file_test_bodies)
# n_train = len(raw_train.instances)



# TODO OH DUDE JUST LET THIS THING DO IT'S SHIT IN THE
INITILIZATION!!! Use the test and train sets provided then
just use the vectors created!


# Process data sets - THIS TAKES 17 SECONDS!
train_set, train_stances, bow_vectorizer, tfreq_vectorizer,
tfidf_vectorizer = util.pipeline_train(raw_train, raw_test,
lim_unigram=lim_unigram)
# feature_size = len(train_set[0])
# fix feature_size at 10001
feature_size = 10001


# Define model


# Create placeholders
features_pl = tf.placeholder(tf.float32, [None,
feature_size], 'features')
stances_pl = tf.placeholder(tf.int64, [None], 'stances')
keep_prob_pl = tf.placeholder(tf.float32)


# Infer batch size
```

```
batch_size = tf.shape(features_pl)[0]


# Define multi-layer perceptron
hidden_layer =
tf.nn.dropout(tf.nn.relu(tf.contrib.layers.linear(features_pl
, hidden_size)), keep_prob=keep_prob_pl)
logits_flat =
tf.nn.dropout(tf.contrib.layers.linear(hidden_layer,
target_size), keep_prob=keep_prob_pl)
logits = tf.reshape(logits_flat, [batch_size, target_size])


# Define L2 loss
tf_vars = tf.trainable_variables()
l2_loss = tf.add_n([tf.nn.l2_loss(v) for v in tf_vars if
'bias' not in v.name]) * l2_alpha


# Define overall loss
loss =
tf.reduce_sum(tf.nn.sparse_softmax_cross_entropy_with_logits(
logits, stances_pl) + l2_loss)


# Define prediction
softmaxed_logits = tf.nn.softmax(logits)
predict = tf.arg_max(softmaxed_logits, 1)
sess = tf.Session()
util.load_model(sess)
    # return sess, test_set, keep_prob_pl, predict,
features_pl
####################################################
####### END ML INIT CODE #######
####################################################


url = 'http://abcnews.go.com/US/wireStory/hurricanes-teach-
us-ap-finds-fast-coastal-growth-49893843'
# webscraper.web_scrape(url)
# webscraper.web_scrape(url)
# example call to python2 file:
```

```
# result = call_python_version("2.7", "module(folder_name)",
"filename.py", "function_name", ["param1", "param2"])


######################
## MACHINE LEARNING ##
######################
def runModel(sess, keep_prob_pl, predict, features_pl,
bow_vectorizer, tfreq_vectorizer, tfidf_vectorizer):
    start_time = time.time()
    print("Now running predictions...")


    # THIS is the info from Henry
    userClaims = "ml/claims2.csv"
    userBodies = "ml/bodies.csv"
    # parse that info
    raw_test = util.FNCData(userClaims, userBodies)
    # need more stuff for this
    test_set = util.pipeline_test(raw_test, bow_vectorizer,
tfreq_vectorizer, tfidf_vectorizer)
    # idk what this does really
    test_feed_dict = {features_pl: test_set, keep_prob_pl:
1.0}
    # run predictions
    test_pred = sess.run(predict, feed_dict=test_feed_dict)
    # timing
    print("generate test_set--- %s seconds ---" %
(time.time() - start_time))
    print("Preditions complete.")
    return test_pred




stances = runModel(sess, keep_prob_pl, predict, features_pl,
bow_vectorizer, tfreq_vectorizer, tfidf_vectorizer)
print(stances)
```

```
################ lots of returns from loadML()
################
# sess, test_set, keep_prob_pl, predict, features_pl =
ourModel.loadML()
# stances = ourModel.runModel(sess, test_set, keep_prob_pl,
predict, features_pl )


# stances = [1,2,3,2,3,3,2,2,3,1,0,0,2,3]
# bodyID = range(len(stances))
# sourceNames = range(len(stances))
# urls = range(len(stances))


# ml_output = pd.DataFrame(
#     {'BodyID': bodyID,
#      'Stances': stances,
#      'SourceName': sourceNames,
#      'URL': urls
#     })


# print(ml_output)


# print(ml_output.loc[0,'Stances'])
# print(ml_output.loc[1,'Stances'])









########################
## REPUTATION SYSTEMS ##
########################
# rep.loadDefaultReputations()
# rep.mlToOut(ml_output)




print("Pipeline complete")
```