Fake News Detection: Truth Is in the Eye of the Beholder

Alice Lin
Simon Fraser University, Burnaby, Canada
all13@sfu.ca

***Abstract*—The rise of social media has made the spread of information almost instantaneous nowadays. Manual fact-checking simply cannot compete with the ease of clicking a "like" or "share" button. Consequently, false or misleading news propagates freely through social media networks, negatively impacting individuals and society. Detecting "fake news" has become a prominent area of study in natural language processing (NLP). This paper investigates various techniques for identifying fake news and compares their performance on a dataset prepared as part of the Leaders Prize^{TM}: Fact or Fake News? Competition.**

***Keywords*—fake news, misinformation, detection, social media, text classification, natural language processing, machine learning**

I. Introduction

According to a 2019 Pew Research Center survey, Americans rate "made-up news" as a bigger problem than climate change, racism or terrorism [1]. This result would be surprising if not for how ubiquitous the term "fake news" has become in recent years. Everything from deliberately false and misleading articles to unfavourable media coverage has been deemed fake news [2]. However, a study by Tandoc Jr. et al. identified six distinct definitions of fake news: satire, parody, fabrication, manipulation, propaganda, and advertising [3].

Automated fake news detection has been studied from the perspectives of machine learning (ML), data mining (DM), and natural language processing (NLP) [4]. Oshikawa et al. surveyed NLP techniques for automated fake news detection [4]. According to their study, fake news detection has been formulated as a classification problem, a regression problem, or a clustering problem. Common techniques for fake news detection include non-neural network models such as Naive Bayes Classifiers (NBC), Support Vector Machines (SVM), Logistic Regression (LR), and Random Forest Classifiers (RFC). Popular neural network models include Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN). Neural network performance is often enhanced by incorporating attention mechanisms.

For this paper, we study the problem of fake news detection by participating in the Leaders Prize^{TM}: Fact or Fake News? Competition [5]. We first analyze the claims metadata for potential signifiers of a claim's truthfulness. We then advance onto traditional machine learning models and deep learning models.

II. Problem Description

The goal of the Leaders Prize^{TM} Competition is to develop an artificial intelligence algorithm to correctly label a given claim as *true*, *partly true* or *false*. A claim is considered *true* if it is factually accurate, *partly true* if important details have been left out or verified facts are taken out of context, and *false* if it is incorrect [5].

There are two phases to the competition. In phase 1, participants create an algorithm to fact-check claims by producing a truth rating for each claim. Programs take as input a metadata file with information about a series of claims and an empty *label* field for each claim. The output is a predictions file with a rating of *true*, *partly true*, or *false* in the *label* field for each claim. Participants submit their program online at most once per day to be evaluated against the validation set [5].

At the end of phase 1, the most recent submission for each participant is evaluated against the test set. The top 10 teams with the highest score on the test set will be invited to continue with phase 2 of the competition, which involves providing evidence to support the outputted truth rating. As submissions are currently being evaluated against the test set, this paper is concerned only with phase 1 of the competition [5].

A. Data Description

The dataset contains two parts: metadata for each claim in the training set and text files of related articles for all the claims in the training set. An example of a claim is shown below:

```
"claim": "\"We have $91 billion going to Puerto Rico. We have $29 billion to Texas and $12 billion to Florida for the hurricanes.\"",

"claimant": "Donald Trump",

"date": "2019-03-28",

"label": 1,

"related_articles": [60922, 41290, 28742],

 "id": 123
```

where "claim" is the statement to be evaluated on truthfulness, "claimant" is the entity who made the claim, "date" is when the claim was made, "label" is the truth rating of the claim (0: *false*, 1: *partly true*, 2: *true*), "related_articles" is a list of article IDs that point to the names of text files containing the articles, and "id" is the unique identifier for each claim [6].

In total, there were 15,555 claims and 64,974 articles in the training set. Of the 15,555 claims, 7,408 were *false*, 6,451 were *partly true*, and 1,696 were *true*.

TABLE I. TRUTH RATINGS FROM SOURCE WEBSITES MAPPED TO CONTEST LABELS

	Label Mappings		
Website	***False***	***Partly True***	***True***
Politifact	'Pants on Fire!', 'False'	'Mostly False', 'Half-True', 'Mostly True'	'True'
Snopes	'false', 'fiction', 'legend', 'scam', 'probably false'	'mostly false', 'mixture', 'mixture of true and false information', 'partly true', 'mostly true'	'true'
Washington Post	'four pinocchios', 'false', 'not true', 'wrong', 'fake news'	'mostly false', 'spinning the facts', 'not exactly', 'three pinocchios', 'two pinocchios', 'one pinocchio', 'not the whole story', 'needs context', 'lacks context', 'mis-leading', 'spins the facts', 'not quite right', 'twists the facts', 'disputed data', 'not quite'	'correct', 'geppetto checkmark', 'true'
Weekly Standard	'FALSE'	'MIXED'	'TRUE'
Africa Check	'false', 'incorrect', 'hoax', 'fake', 'false headline'	'mixture', 'understated/ exaggerated', 'misleading', 'mostly correct'	'correct', 'checked', 'true'
FactsCan	'farcical', 'false'	'misleading'	'true'
AFP	'faux', 'falso', 'false', 'fake'	'misrepresentation', 'misleading', 'mixed'	'vrai'
Polygraph	'false'	'misleading', 'unclear', 'partly false', 'likely false', 'mostly false', 'partially false', 'partially false and misleading', 'unclear but likely partially true', 'unclear and partially true', 'questionable', 'partially true', 'likely true'	'true'

B. Data Origin

The dataset consists of claims and associated metadata downloaded from 9 fact-checking websites: politifact.com, snopes.com, washingtonpost.com, weeklystandard.com, africacheck.org, factscan.ca, factcheck.afp.com, polygraph.info, factcheck.org. On the websites, each claim was published with a "truth rating" along with an explanation of the rating and links to related articles [6].

Since each website uses different metrics, the truth ratings were mapped to the labels 0 (*false*), 1 (*partly true*) and 2 (*true*) as shown in Table I.

For each claim, the related articles linked to by the fact-checking website were downloaded. Each claim was provided with at least two related articles. The related articles could either be source articles that contain the claim (albeit differently phrased) or supporting articles that provide evidence of the claim's truth rating. Source and supporting articles are not differentiated in the dataset [6].

C. Evaluation

Submission scores were calculated using the macro average F1 score of the outputted truth ratings. The formula is defined as follows:

$$score = \frac{2 * P * R}{P + R}$$

where P is precision and R is recall, defined as:

$$P = \frac{P_{true} + P_{partly} + P_{false}}{3}$$

$$R = \frac{R_{true} + R_{partly} + R_{false}}{3}$$

The precision and recall for each class are defined as:

$$P_{class} = \frac{TP_{class}}{TP_{class} + FP_{class}}$$

$$R_{class} = \frac{TP_{class}}{TP_{class} + FN_{class}}$$

TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives [6].

III. SOLUTION TECHNIQUES

Since contest submissions were limited to one submission per day, testing was done locally to develop the program. The training data was split into a training set and a holdout set at a ratio of 80:20. There were 12,444 claims in the training set and 3,111 claims in the holdout set.

A validation score was used to evaluate performance on the holdout set. The validation score was based on the `f1_score` metric from the `sci-kit learn` library. The average parameter was set to `macro` to reflect the macro F1 score used by the contest evaluation (the "test score").

A. *Random Guessing*

The contest provided a sample submission which uses random guessing to classify claims. The random guesser assigns a random integer from the set {0, 1, 2}, corresponding to the labels *false*, *partly true*, and *true*, to each claim. In one experiment, random guessing achieved a validation score of 0.309249 and a test score of 0.250182. This result confirmed the validation score was reflective of the contest evaluation score and could be used for local testing. The random guesser served as the baseline system.

With the uniform random guesser, each class has an equal probability of being outputted. However, the truth ratings in the training set are unevenly distributed, with 47.85% of claims being *false*, 41.12% being *partly true*, and 11.03% being *true*. A more representative guesser would attribute a greater weight to the classes that occur more frequently in the training set. We used a weighted random guesser which outputted the label *false* 50% of the time, *partly true* 40% of the time, and *true* 10% of the time. The frequency of each class was rounded for simplicity. In one experiment, the weighted random guesser achieved a validation score of 0.335557 and a test score of 0.251016.

B. *Metadata*

Additional classifiers were developed by analyzing the claims metadata. We looked at whether the length of a claim could be relevant to its truth rating. For example, *true* claims could be shorter because they contain verifiable facts and *false* claims could be longer because they contain rhetoric or irrelevant information.

Analysis of claim length showed that *false* claims had an average length of 142.34 characters, *partly true* claims had an average length of 140.85 characters, and *true* claims had an average length of 124.96 characters. We used a claim length classifier which classified a claim as *false* if the claim length ≥ (142 + 140) / 2, *partly true* if the claim length ≥ (140 + 124) / 2, and *true* otherwise. The validation score was 0.203872.

Similar analysis was done for the word count of the claims. The average number of words per claim was 23.4 for *false* claims, 23.1 for *partly true* claims, and 20.6 for *true* claims. Again, there was only a marginal difference between *false* claims and *partly true* claims, and a small difference between *false* and *partly true* claims and *true* claims. The average word count per claim was 5.26 for *false* claims, 4.84 for *partly true* claims, and 4.40 for *true* claims. We used a word count classifier which classified a claim as *false* if the word count of the claim ≥ (5.26 + 4.84) / 2, *partly true* if the word count ≥ (4.84 + 4.40) / 2, and *true* otherwise. The validation score was 0.217368.

The competition documentation stated that at least two “related articles” were provided for each claim [6]. We investigated whether there is any relation between the number of articles cited by each claim and the truth rating of the claim. Perhaps *true* claims would reference more articles to provide sources supporting their claim. The analysis showed that the maximum number of related articles was 66 for *false* claims, 41 for *partly true* claims, and 27 for *true* claims. The average number of related articles was 5.26 for *false* claims, 4.84 for *partly true* claims, and 4.399 for *false* claims. We used a related article count classifier to classify a claim as *false* if the number of related articles ≥ (5.26 + 4.84) / 2, *partly true* if the number of related articles ≥ (4.84 + 4.40) / 2, and *true* otherwise. The validation score was 0.258209.

C. *Related Articles*

The exact relationship between the related articles and the claims is unclear. However, since the articles were provided as part of the dataset, they must have some relevance to the claims which could be useful for predicting the truth ratings.

For each claim in the training set, we looked at the article IDs of the related_articles. Each article ID was then associated with the label of each claim that references it. For example, the article #171 is associated with two *false* claims and 1 *partly true* claim: `171: [0, 0, 1]`.

After obtaining the truth ratings for all the claims associated with each article ID, we took the average of the associated truth ratings and assigned the rounded average to the article. For example, article #171 would be assigned the truth rating *false* since it is most commonly cited by *false* claims. A dictionary was created in which the key is the article ID and the value is the truth rating associated with the article.

During evaluation against the holdout set, the related article IDs for each claim was cross-referenced with the dictionary. A list of truth ratings for the related articles of each claim was returned. The average of the truth ratings was taken and rounded to the nearest integer. A claim whose related articles were associated with a truth rating of *false* would be given a *false* label. If the claim referenced a related article whose ID was not in the dictionary, the article would be given a truth rating determined by the weighted random guesser discussed above. The validation score was 0.345475 and the test score was 0.26116.

D. *Claimant*

Sitaula et al. showed that an author’s history of association with fake news can be used to detect fake news [7]. The credibility of a claimant can therefore be used to predict the truth rating of the associated claim.

TABLE II. TOP CLAIMANTS

Claimant	Total Claims	False Claims	Partly True Claims	True Claims
(None)	3,987	**2,441**	1,026	520
Donald Trump	987	**660**	308	19
Bloggers	304	**257**	44	3
Barack Obama	192	35	**119**	38
Hillary Clinton	172	41	**98**	33
Viral image	110	**101**	9	0
Ted Cruz	86	32	**50**	4
Facebook posts	81	**61**	16	4
Various websites	80	**80**	0	0
Marco Rubio	76	19	**49**	8

In the training set, there are 8,457 unique claimants. Table II shows the top 10 claimants who made the most claims in the training set. Not surprisingly, the top claimant is “no claimant”.

Each claimant was associated with a list of the truth ratings of the claims that they made. For example, one claimant might be represented as `‘Elizabeth Warren’: [0, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1]`. The average of the truth ratings for each claimant was taken. Each claimant was then associated with the rounded average of their truth ratings. For example, `‘Elizabeth Warren’` would be associated with the truth rating `‘1’`, reflecting the fact that most of the claims she made were *partly true*. The results were stored in a dictionary where the key is the claimant’s name and the value is the truth rating associated with the claimant.

During evaluation against the holdout set, the claimant for each claim was cross-referenced with the dictionary and the associated truth rating for the claimant would be returned as the predicted label for the claim. If the claim had no claimant or if the claimant did not exist in the dictionary, a truth rating of *false* was returned. The rationale is that the probability of a claim being *false* is higher than the probability of the claim being *true* or *partly true*. The validation score was 0.438962 and the test score was 0.382788.

E. Ensemble

We studied whether building an ensemble classifier would improve the performance over each of the classifiers on their own. A weighted voting classifier was built. The list of classifiers included claim length, claim word count, related article count, related article IDs, and claimant. Weights for each of the classifiers was determined by the classifier’s Matthews correlation coefficient (MCC), which is defined as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. The weights were chosen using the MCC instead of the F1 score because MCC considers all four classes (TP, TN, FP, FN) in computing its final score [8]. By contrast, the F1 score only considers TP, FP, and FN.

We set the weights to zero for classifiers with negative MCC scores. For each claim in the training set, we obtained a truth rating prediction from each of the classifiers. The prediction from each classifier would be multiplied by the classifier’s weight and added to the tally of *false*, *partly true*, or *true* counts. The label with the highest count would be outputted as the claim’s truth rating. The validation score was 0.3768601 and the test score was 0.293169.

TABLE III. ENSEMBLE CLASSIFIER WEIGHTS

Classifier	**MCC**
Claim Length	-0.008654
Claim Word Count	-0.006189
Number of Related Articles	0.035123
Related Article IDs	0.956777
Claimant	0.489670

The ensemble classifier performed worse than the individual classifiers. Since most of the classifiers are weak predictors, their combination likely eroded the performance of the strongest classifier (claimant), thereby worsening the result.

F. Sentiment Analysis

Many research works have suggested the use of sentiment analysis to detect fake news. Rubin et al. used satirical cues to detect misleading news [9]. For this study, we expected *false* claims to be more polarizing, i.e. very negative or very positive.

To study sentiment analysis on the dataset, we used the `TextBlob` library and its built-in sentiment function. The sentiment function returns a named tuple of the form `Sentiment(polarity, subjectivity)`. The polarity score is a float within the range [-1.0, 1.0] where -1.0 is very negative and 1.0 is very positive. The subjectivity score is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective [10]. The expectation is that *true* claims would be more neutral (polarity around 0.0) and more subjective (subjectively around 0.0).

Histogram plots of the polarity and subjectivity scores for *false*, *partly true*, and *true* claims are shown in Figures 1 and 2. The score distributions are approximately the same for all three classes. The polarity scores for the three classes are grouped together around 0.0, meaning that the claims are all generally neutral. Subjectivity is also near 0.0 for all three classes, meaning that the claims are all generally objective. The average polarity and subjectivity scores for the claims are shown in Table IV.

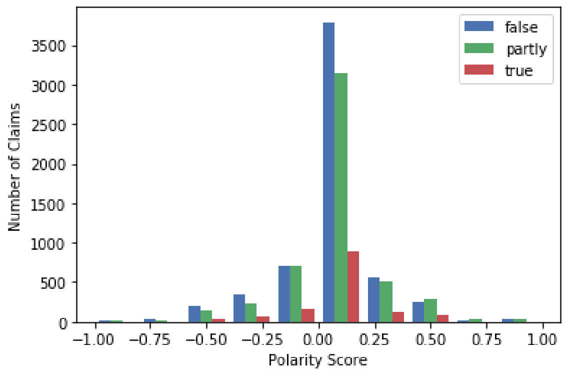

Figure 1: Distribution of polarity scores for the claims

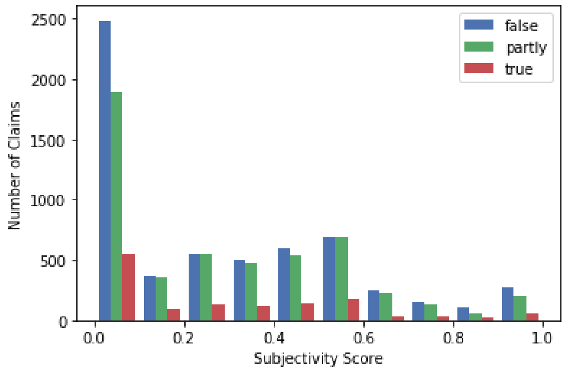

Figure 2: Distribution of subjectivity scores for the claims

TABLE IV. AVERAGE CLAIM POLARITY AND SUBJECTIVITY SCORES

Class	Average Polarity	Average Subjectivity
False claims	0.028367	0.271642
Partly true claims	0.045805	0.285879
True claims	0.050810	0.267652

Sentiment analysis was also conducted on the claims using `VADER` (Valence Aware Dictionary and sEntiment Reasoner) which is a lexicon and rule-based sentiment analysis tool based on social media text [11]. `VADER` returns a dictionary of the form: `{'pos': 0.317, 'compound': 0.5249, 'neu': 0.556, 'neg': 0.127}` where `'pos'` is the positive sentiment score, `'neu'` is the neutral sentiment score and `'neg'` is the negative sentiment score, all summing up to 1. The `'compound'` score is a normalized, weighted composite score ranging from -1 (most extreme negative) to +1 (most extreme positive).

Figure 3 plots the compound scores for the claims. Again, the score distributions are approximately the same for all the classes. Most of the claims have a compound score of 0.0, meaning that the claim sentiments are generally neutral. This corroborates the findings using `TextBlob`. Sentiment analysis did not seem to be effective for differentiating between the different classes so sentiment classifiers were not implemented.

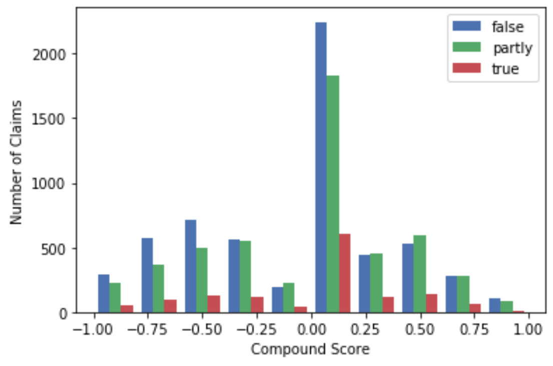

Figure 3: Distribution of compound scores for the claims

G. Machine Learning Models

We trained a machine learning model to predict a claim's truth rating based on the content of the claim. The claims in the training set were converted to word count vectors with `sklearn`'s `CountVectorizer`. The claims in the training set were tokenized and used to build a vocabulary. Each word in the vocabulary was then encoded as a vector. We used a multinomial Naive Bayes classifier to train the model. Naive Bayes was chosen as the classifier because it outperformed Support Vector Machine (SVM), Logistic Regression (LR), and Random Forest Classifier (RFC).

Tested on the holdout set, the model had a validation accuracy of 60.40% (calculated as the number of correct predictions divided by the number of claims) and a validation score of 0.441848. This was a slight improvement over the claimant classifier. However, the normalized confusion matrix in Figure 4 shows that the Naive Bayes model is poor at predicting the *true* class. The model's weak performance in this area is confirmed by the test score of 0.376152, which is lower than the claimant classifier.

The result is not surprising when we consider the counts of each class in the training set. There are 5,954 *false* claims, 5,117 *partly true* claims, and 1,373 *true* claims. The *true* class is severely under-represented, making up only 11.03% of the samples in the training set. To rectify this problem, we used over-sampling to generate new samples in the classes which are under-represented. Specifically, we use the Synthetic Minority Oversampling Technique (SMOTE) to oversample the minority classes [12]. This balanced out the training samples so there were 5,954 samples from each class.

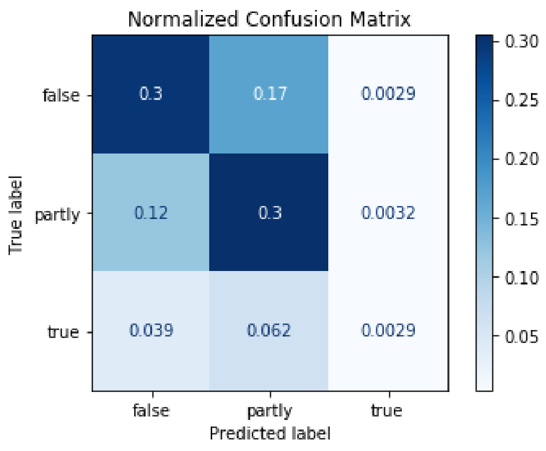

Figure 4: Normalized confusion matrix for multinomial Naive Bayes classifier

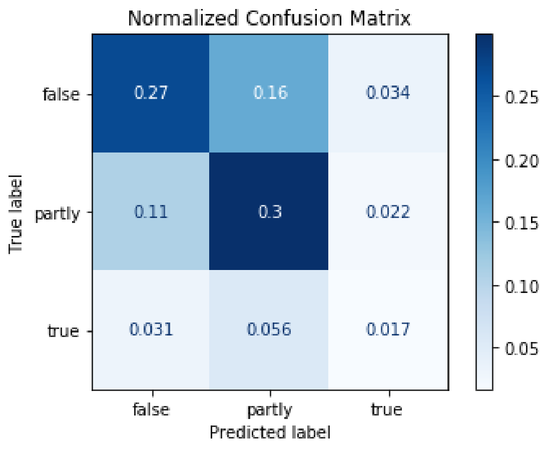

Figure 5: Normalized confusion matrix for multinomial Naive Bayes classifier with SMOTE

Figure 5 shows the confusion matrix for the Naive Bayes classifier using SMOTE during training. We can see a slight increase in the prediction frequency of the *true* class, although prediction is still quite weak in this area. The validation accuracy decreased slightly to 58.47% but the validation score increased to 0.482802. The test score was 0.428168.

Since the claimant classifier using a dictionary performed well, we tried implementing a machine learning model to predict the class based on the claimant. Using a Naive Bayes model and SMOTE oversampling as above, we achieved a validation accuracy of 53.97% and a validation score of 0.440491.

We attempted to enhance the performance of the Naive Bayes claim model by combining it with the Naive Bayes claimant model. For each claim in the training set, the claimant was concatenated to the end of the claim statement. The training samples were balanced using SMOTE and a machine learning model was trained using a Naive Bayes classifier. The validation accuracy was 57.89%, the validation score was 0.478622, and the test score was 0.43149.

H. *Deep Learning Models*

Convolutional neural networks (CNN) trained on top of pre-trained word vectors have been successful in many text classification tasks [13]. We trained a simple 1D CNN to predict truth ratings based on the claim statement. We first pre-processed the data for input to the CNN by tokenizing the claims, converting the tokens to lowercase, and removing punctuation. Oversampling was done using SMOTE to balance the classes. The claims in the training set were encoded into vectors using pre-trained 300-dimensional GloVe embeddings [14]. The truth ratings for the claims were encoded as one-hot vectors.

The embedding vectors were fed to a convolutional layer with a ReLU activation. A standard max-pooling operation was performed on the hidden layer. The max-pooled text representations were then fed into another convolutional layer with a ReLU activation, followed by another max-pooling operation, before being fed to a fully connected layer with a softmax activation function to output the final prediction. Dropout and batch normalization layers were employed for regularization to mitigate against overfitting.

The model was trained for 10 epochs. The training accuracy was 0.643 and the validation accuracy was 0.605, while the test score was only 0.409578. The model accuracy plateaued after 10 epochs. It seemed there were not enough training samples for the CNN to learn effectively.

The training set was augmented with the LIAR dataset which contains 12.8k labelled claims for fake news detection [15]. The LIAR dataset contains six labels for truth ratings: *pants-fire, false, barely-true, half-true, mostly-true, and true*. The LIAR truth ratings were mapped to the contest labels as shown in Table V. The LIAR dataset (including training, validation, and testing sets) was concatenated to the training set to increase the total number of claims in the training set to 25,235.

TABLE V. TRUTH RATINGS FROM LIAR DATASET MAPPED TO CONTEST LABELS

	Label Mappings		
Dataset	***False***	***Partly True***	***True***
LIAR	‘pants-fire’, ‘false’	‘barely-true’, ‘half-true’, ‘mostly true’	‘true’

After data augmentation, the classes were still unbalanced, so oversampling was again done with SMOTE. The model was trained for 100 epochs and achieved a validation loss of 0.8075 and a validation accuracy of 0.6798. The test score was 0.435576.

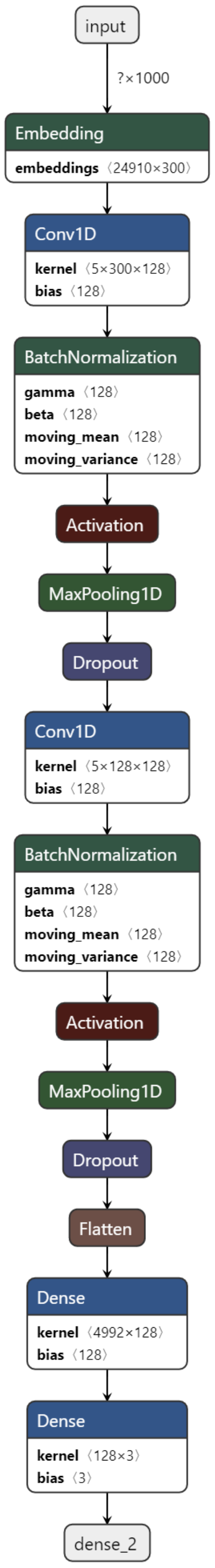

Figure 6: An illustration of the architecture of our CNN

IV. Discussion

Table VI summarizes the performance of our different approaches for fake news detection. The CNN model performed the best after augmenting the training dataset with the LIAR dataset. The additional training samples seemed to reduce the CNN's vulnerability to overfitting and improved its performance. But even then, the CNN model performed only marginally better than the Naive Bayes model. Given the computational overhead and time complexity of neural networks, a Naive Bayes model with similar performance would be the preferred choice. Indeed, Khan et al. showed that a Naive Bayes model (with n-gram) could attain analogous results to neural network-based models on a dataset with fewer than 100k news articles [16].

Figures 7 and 8 show the training/validation loss and accuracy of the CNN + LIAR model. Validation loss plateaued around 80% and validation accuracy plateaued around 65%. Performance of the previously implemented CNN was improved by augmenting the training dataset with the LIAR dataset. This suggests that additional data augmentation could generate further performance gains.

TABLE VI. Results

Classifier	**Validation Score**	**Test Score**
Uniform Random Guesser	0.309249	0.250182
Weighted Random Guesser	0.335557	0.251016
Claim Length	0.203872	-
Word Count	0.217368	-
Related Article Count	0.258209	-
Related Article ID	0.345475	0.26116
Claimant	0.438962	0.382788
Ensemble	0.3768601	0.293169
Naive Bayes (Claim)	0.441848	0.376152
Naive Bayes (Claim) + SMOTE	0.482802	0.428168
Naive Bayes (Claimant) + SMOTE	0.440491	-
Naive Bayes (Claim + Claimant) + SMOTE	0.478622	0.43149
CNN	0.605	0.409578
CNN + LIAR	**0.6798**	**0.435576**

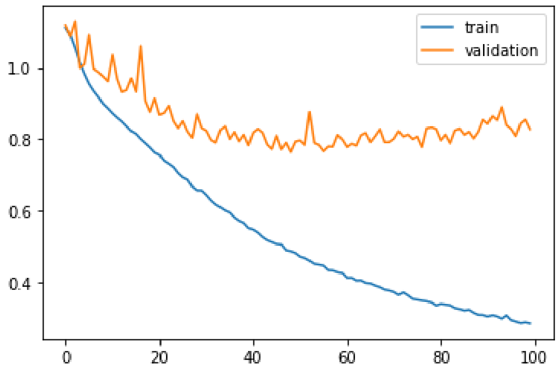

Figure 7: CNN + LIAR Training/Validation Loss

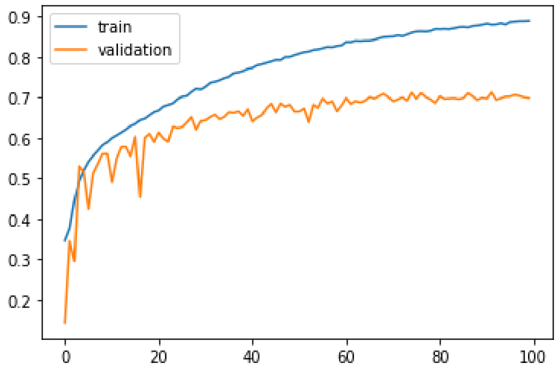

Figure 8: CNN + LIAR Training/Validation Accuracy

Performance could also be improved if the competition parameters were more transparent. It was not entirely clear how the training and test sets were collected and whether they were produced from different sources. It was also not clear how the related articles were relevant to the claims. The documentation describes the articles as being either "source" or "supporting" articles, but the distinction was deliberately obfuscated. Finally, the evaluation function was changed partway through the competition with little explanation, even though this would have an obvious impact on the algorithm design.

V. Conclusion and Future Work

In this paper, we present a performance analysis of various approaches to fake news detection. We show that a traditional machine learning model such as Naive Bayes has promising results for small datasets. CNNs also prove promising, although they require more training data to achieve similar results to the Naive Bayes model. In this regard, the scarcity of fake news datasets poses a problem. One solution may be to use pre-trained models such as Google's state-of-the-art BERT model. Other deep learning models such as recurrent neural networks (RNNs) or bi-directional long short-term memory networks (Bi-LSTMs) may also be useful.

Another consideration is whether the structure of standard fake news datasets is appropriate or sufficient for the task. Most datasets contain information on the claim, the author, and the truth rating. However, context seems to be important for fake news detection. The same statement made by a different person, or at a different time and place, could affect the statement's truthfulness. Intent also seems to be meaningful. Is a small, malicious lie more "fake" than a big but honest mistake? Perhaps like beauty, truth is in the eye of the beholder.

Acknowledgements

We thank Dr. Toby Donaldson for his valuable guidance and encouragement during the planning and development of this project. The work presented in this paper is a research project conducted as part of the author's undergraduate degree in Computing Science at Simon Fraser University.

References

1. A. Mitchell, J. Gottfried, G. Stocking, M. Walker and S. Fedeli, "Many Americans Say Made-Up News Is a Critical Problem That Needs To Be Fixed," *Pew Research Center*, Jun. 5, 2019. [Online]. Available:

https://www.journalism.org/2019/06/05/many-americans-say-made-up-news-is-a-critical-problem-that-needs-to-be-fixed/. [Accessed: Dec. 15, 2019]

2. D.A. Graham, "Some Real News About Fake News," *The Atlantic*, Jun. 7, 2019. [Online]. Available: https://www.theatlantic.com/ideas/archive/2019/06/fake-news-republicans-democrats/591211/. [Accessed: Dec. 15, 2019].
3. E.C. Tandoc Jr., Z.W. Lim and R. Ling, "Defining 'Fake News'," *Digital Journalism*, 6(2), pp.137-153, Aug. 2017.
4. R. Oshikawa, J. Qian, and W.Y. Yang, "A Survey on Natural Language Processing for Fake News Detection," arXiv:1811.00770 [cs.CL], Nov. 2018.
5. "Leaders Prize: Fact or Fake News?". [Online]. Available: https://datacup.ca/competitions/leadersprize2019. [Accessed: Dec 22, 2019].
6. *Documentation for the Data*, Leaders Prize^{TM} Competition.
7. N. Sitaula, C.K. Mohan, J. Grygiel, X. Zhou and R. Zafarani, "Credibility-based Fake News Detection", arXiv:1911.00643 [cs.CL], Nov. 2019.
8. D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, 10, 25, 2017.
9. V. Rubin, N. Conroy, Y. Chen, S. Cornwell, "Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News.", DOI:10.18653/v1/W16-0802, June 2016.
10. "Tutorial: Quickstart — TextBlob 0.15.2 documentation," *textblob.readthedocs.io*. [Online]. Available: https://textblob.readthedocs.io/en/dev/quickstart.html. [Accessed: Oct. 28, 2019].
11. "VADER Sentiment Analysis," *GitHub.com*. [Online]. Available: https://github.com/cjhutto/vaderSentiment. [Accessed: Oct. 29, 2019].
12. N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, 16, pp. 321-357, 2002.
13. Y. Kim, "Convolutional Neural Networks for Sentence Classification," arXiv:1408.5882 [cs.CL], Sep. 2014.
14. J. Pennington, R. Socher, C. D. Manning, "GloVe: Global Vectors for Word Representation," *nlp.stanford.edu*. [Online]. Available: https://nlp.stanford.edu/projects/glove/. [Accessed: Nov. 30, 2019].
15. W.Y. Yang, "'Liar, Liar Pants on Fire': A New Benchmark Dataset for Fake News Detection", arXiv:1705.00648 [cs.CL], May 2017.
16. J.Y. Khan, Md. T.I. Khondaker, A. Iqbal and S. Afroz, "A Benchmark Study on Machine Learning Methods for Fake News Detection," arXiv:1905.04749 [cs.CL], May 2019.