Understanding Fake News and Convolutional Neural Networks

Team 4 (Red)

Ashish Vinodkumar, Charlie Xie, Jasmine Young, Leon Zhang

Abstract

Fake News has recently become a buzzword describing lies and false information spread through news outlets and social media. Many have sought to find ways to classify news articles and headlines as either fake news or accurate information. This paper aims to explore Convolutional Neural Networks as a classification tool for the Liar dataset created by William Yang Wang. To test our dataset we used word embeddings on the text of our labeled dataset. Both our regression and neural network are multi-class models because our data is labeled with one of six categories from "completely false" to "completely true". We were able to achieve similar results as reported by William Yang Wang in the original Liar dataset paper. Our Convolutional Neural network achieved 21% accuracy and our Logistic Regression achieved 25% accuracy. Our results demonstrate the importance of model comparison, the strength of logistic regressions, and the difficulty of multi-class models.

Introduction

In this short paper, we will try to utilize the Logistic Regression (LR) model and the Convolutional neural networks (CNN) model to replicate the results in the article written by Wang(2017) by using the LIAR dataset. Specifically, this paper is organized as follows: the background section examines the literature in the field of fake news detection and also introduces the LIAR dataset. The methodology section describes the data processing approach, the LR model, and the CNN model. In the results section, we compare and interpret the metrics from those two models. Finally, current limitations of models' application and possible directions for further improvements are discussed at the last Section.

Background

Detecting fake news can be time consuming and expensive. The conventional way to conduct it is to reach out to professionals for verifying claims based on facts. However, applications of Natural Language Processing(NLP) can reduce the human time and effort for detecting fake news. In this section, we will go over past research efforts in terms of claims evaluation.

NLP researchers were trying to create datasets with high quality for detecting fake news. The very first dataset including 221 claims collected from POLITIFACT and CHANNEL4.COM was released by Vlachos and Riedel (2014). However, it is hard to utilize this one in machine learning predictions due to its small size. Based on the previous work, William Wang included 12,836 claims in his LIAR dataset, and each one is labeled with six-grade truthfulness(Wang, 2017). Furthermore, Thorne and his colleagues(2018) published a dataset called Fever, containing 185,445 claims gathered from Wikipedia, with each claim labeled with Supported, Refuted, or Not Enough Info. We will focus on the LIAR dataset in our paper.

Regarding the Non-Neural Network Models, Support Vector Machine (SVM) and Naive Bayes Classifier (NBC) are usually utilized as baseline models (Conroy et al., 2015; Khurana and Intelligentie, 2017; Shu et al., 2018). Moreover, Logistic regression (LR) (Khurana and Intelligentie, 2017; Bhattacharjee et al., 2017) and Random Forest Classifier (RFC) (Hassan et al., 2017) are also employed. Furthermore, for the Neural Network Models, the Recurrent Neural Network (RNN) prevails, especially for the Long Short-Term Memory (LSTM), which is able to capture the longer-term dependencies(Oshikawa et al, 2020). Moreover, Wang(2017) leveraged the Convolutional neural networks (CNN) model which is usually used since they succeed in text analysis. In addition, a Multi-source Multi-class Fake news Detection framework (MMFD) was released by Karimi et al. (2018) where they took advantage of CNN to analyze local patterns of each text in a claim and also used LSTM for analyzing temporal dependencies in the entire text. Last but not least, Long et al. (2017) proposed an attention model which includes the name of speaker and the topic of claims to attend to features first, then feeded the weighted vectors with an LSTM. Kirilin and Strube(2018) used memory networks, an attention-based neural network, that echoed the idea from Pham (2018).

In this paper, we are going to replicate Wang's work by using Logistic regression(LR) as our baseline model and then utilizing a combination of CNN and LSTM models to approach the LIAR dataset. Within the LIAR dataset, the distribution of labels and political parties within the dataset is well balanced. The dataset relies on six labels of truthfulness: pants-fire, false, barely-true, half-true, mostly true, and true. We see 4,150 Democratic affiliations to match 5,687 republican affiliations, leaving 2,185 unknown affiliations (i.e. Facebook posts). The dataset also includes information about the speaker such as their name, party affiliation, and their history of accurate or fake statements. We will also compare the results from different models and discuss the pros and cons of these models in this circumstance.

Approach/Methodology: Baseline Model (Regression)

In order to assess the efficacy of our neural network model, we aimed to create a baseline multi-class logistic regression model that contains 6 labels ranging from Pants on Fire (Completely false) to Completely True. We first created a set of common stop-words using the nltk library, and further augmented it with unicode characters. We then parsed through each statement in our dataset, to identify these stopwords and remove them, to create a new column containing the cleaned statements in our dataset. We then proceeded to perform a 80/20 split, where we accounted for 80% of the data as part of our training set, and 20% of our data for testing. Splitting the data allowed us to train and validate the model accuracy not just in terms of tuning the logistic model, but also to compare metrics across models between logistic and the neural network.

We then proceeded to create a Term Frequency - Inverse Document Frequency (TF-IDF) embedding for our training dataset. With Term-Frequency (TF), we represent each word in our dataset by the frequency (count) of the number of times a word appeared in the corpus. With Inverse Document Frequency (IDF), we take the log of the number of documents divided by the number of documents with that given word. This is particularly effective, as if a word appears across all the documents, then it has less significance in characterizing the document. More importantly, the log(1) is 0, and this results in collapsing all words to 0 that have no significance, thus highlighting unique, descriptive, and high importance words across each statement in the corpus.

Having created TF-IDF embeddings, we proceeded to fit a Multi-Class Logistic Regression model with the TF-IDF embeddings detailed above as the input. We tuned the multi-class logistic regression model by setting the max iterations to 1000. We then proceeded to calculate the precision, recall, accuracy, and f1 score of our model.

Approach/Methodology: Word Embedding

We applied word embedding techniques to the speakers' statements. Word embedding encodes similar words that have similar contexts. This approach of representing words is considered one of the biggest advances in Natural Language Processing. The greatest advantage in using word embedding over other naive text processing techniques such as bag of words and TF-IDF is that word embedding is able to capture the meaning of words and preserve a lot more information than other techniques.

Word embedding transforms words into real-valued vectors in a predefined vector space. We used the GloVe (global vectors for word representation) library for embedding our statements. In GloVe, words are represented in a size of a hundred dimensions. The size of the vector is the feature space of embedded words. The more features we include, the more meaning the embedding can capture. However, we need to strike a balance between quality of embedding and computation efficiency. We believe the 100-feature representations for each word in GloVe are granular enough to capture the information we need. To embed the words, we parsed the statements into separate vocabularies and mapped the word to the embeddings from the GloVe library. To ensure our input is of the same size, we made our input the size of the longest statement and padded the shorter statement with zeros.

Approach/Methodology: Neural Network

We built a neural network for performing multi level classification in detecting fake news. The network takes in the word embedding input, which is capped at 100 - the average length of our statements, and goes through a single layer of convolutional network with max pooling. After

max pooling, the values are passed into two layers of LSTM nodes. Following the LSTM layers, we built another two layers of dense nodes and the final output is calculated using softmax for finding which class it belongs to. We will discuss the details of these layers in the coming paragraphs. Below is the diagram of our neutral network and layer specifications:

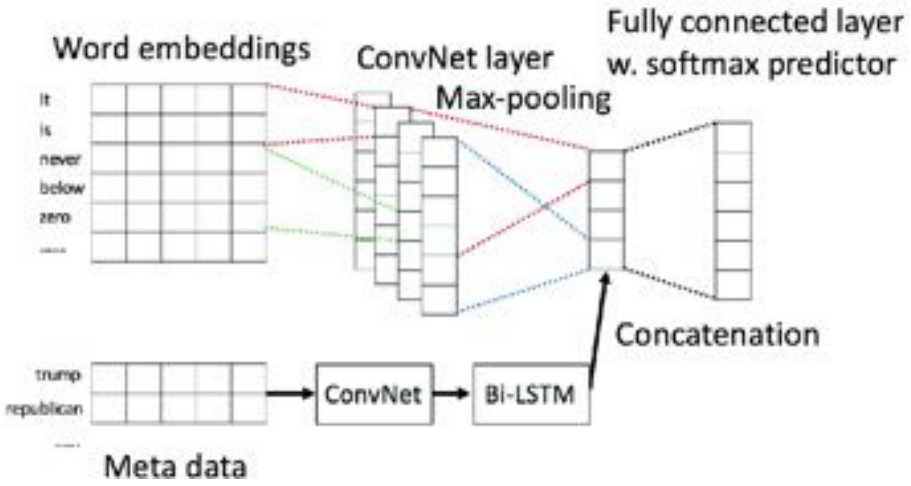

Figure 1. Architecture of our Neural Network

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 100)         1238400
_________________________________________________________________
dropout (Dropout)            (None, None, 100)         0
_________________________________________________________________
conv1d (Conv1D)              (None, None, 64)          32064
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, None, 64)          0
_________________________________________________________________
lstm (LSTM)                  (None, None, 20)          6800
_________________________________________________________________
lstm_1 (LSTM)                (None, 20)                3280
_________________________________________________________________
dropout_1 (Dropout)          (None, 20)                0
_________________________________________________________________
dense (Dense)                (None, 512)               10752
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 1542
=================================================================
Total params: 1,424,166
Trainable params: 185,766
Non-trainable params: 1,238,400
_________________________________________________________________
```

Figure 2. Specifications of our Neural Network

Convolutional neural network has found tremendous success in recent years. It has gained recognition, especially in the computer vision field, due to its ability to capture spatial and temporal dependencies in images. It also has the benefit of reducing the number of parameters, so it is often placed at the front of a neural network architecture to make training of later layers more efficient. It has been found that convolutional neural networks are also useful in tackling text classification problems. NLP problems can benefit from the convolution layer because it helps to aggregate spatial and temporal dependencies within words and sentences. For our project, we used a single convolutional layer with input size of 64 and a max pool size of 4, meaning a 4X4 window will scan over 8 input at a time and return the maximum input.

LSTM layers are followed by convolutional networks. LSTM stands for long short term memory layer, it is widely used in the NLP field and is known for its ability in having longer memory than typical recurrent neural networks. LSTM gates can learn which data in a sequence of work is important to keep or throw away. Therefore, it could pass down useful information down the long chain of sequence to make classifications. The LSTM layer we created has 20 units, resulting in a vector of size 20.

Finally two dense layers are built after the LSTM layer to format the output for the softmax function to make predictions. The softmax function is a generalization of the logistic function to multiple dimensions. It is given by the formula:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Equation 1. Softmax function

Where K is number of classes and z are in input. Since our label has 6 levels, our softmax function will output a vector of size 6 representing the prediction probability of 6 different levels. To convert into prediction, we will return the level that has the highest probability.

In addition we used a dropoff of 0.2 in LSTM and dense layers to prevent overfitting. Dropoff works by randomly disabling a portion of nodes in training which regularize the model. We also specified callback functions such as ReduceOnPlateau to reduce learning rate when the validation accuracy stops improving and early stopping when it sees overfit.

Results: Baseline Model (Regression)

We will first discuss the results for our logistic regression model. We noticed that the overall model accuracy for the multi-class logistic regression model was 25%. The precision score was 26%, recall score was 22.6%, along with an F1 score of 22.1%.

Interestingly, having a multi-class response variable reduces the model accuracy as the words that identify completely true are similar to the words that identify half-true. This is true not just for the true label statements, but also for the false label statements. This overlap of words results in no clear delineation between two levels, thus severely reducing our model accuracy, precision and recall.

To further test this hypothesis, we proceeded to bucket the 6 labels into broader labels of simply consisting of False and True. We immediately noticed that the overall model accuracy increased to 62.8%. The precision score increased to 65%, recall score to 75.1%, and F1 score increased to 69.7%.

Here is a plot of the ROC curve that showcases an AUC of 65.1% when we have a binary response label instead of a multi-class label.

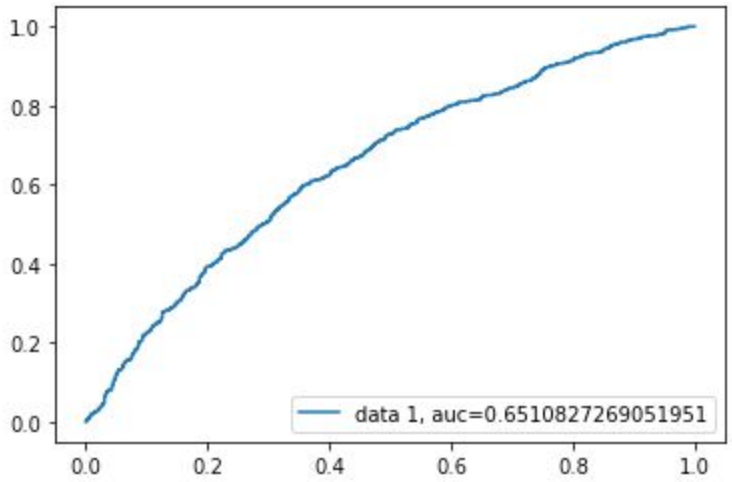

Figure 3. ROC Curve for the binary logistic model.

In order to model our analysis as close to the original research paper as possible, we proceeded to retain the initial 6 labels in the raw dataset. As a result, we compared a model accuracy of 26% with the logistic regression model, to the model accuracy for that of the neural network model.

Results: Neural Network

We outline the results of our Convolutional Neural Network below. Our Convolutional Neural Network performed with an accuracy of 21% on our separated test data. In the figures below we see graphs of the model's accuracy and the model's loss. Our model reached epoch 6 before it began overfitting.

Though the LIAR dataset is larger than any other fake news classification dataset, it may still not be large enough to expect fine tuned results from a Convolutional Neural Network. A larger dataset would improve our accuracy and ability to classify fake news. We also note that our model struggled to accurately detect all 6 categories of Fake News. Bucketing the six categories of truthfulness into "True" and "False" may also improve the accuracy and performance of our model, as it did with our Logistic Regression. However, the number of labels could be dependent on the use case.

Overall, our Convolutional Neural Network's accuracy was comparable to the original paper's CNN accuracy of 27%. It is important to remember that our inaccurate predictions may have been the result of a one-category mis-step (for example, the difference between "half-true" and "completely true"), while maintaining the overall sentiment of a text. We are confident that adding other layers, feeding the model more predictors, or reducing the classification task would improve our model's accuracy.

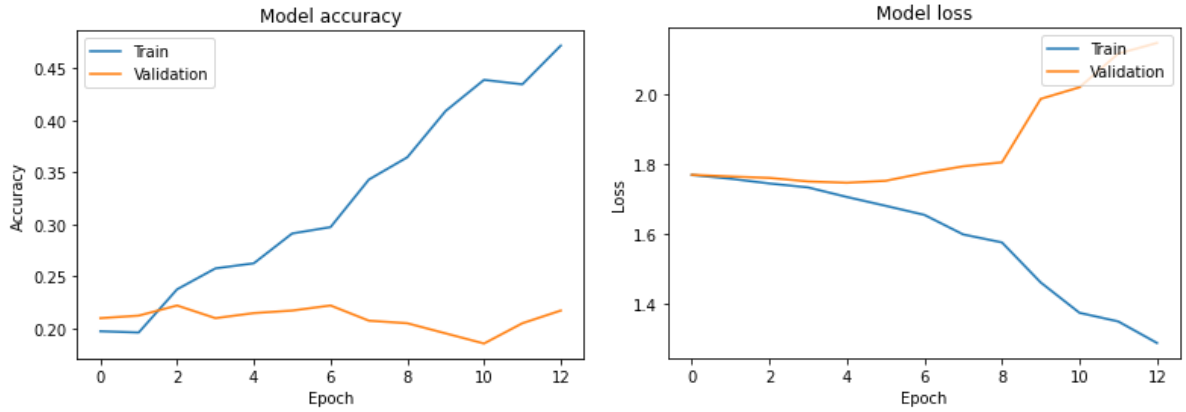

Figure 4. Model Loss and Accuracy of Neural Network Training

Conclusion

Using the LIAR dataset, the LR model and the CNN model, this short paper tried to replicate the results generated by William Yang Wang in his article. We found a similar accuracy for the prediction by using the LR model, however, we did not achieve the exact same accuracy like what the paper shows by using the specific CNN model we used. We noticed that the dataset could be the cause of the relatively low accuracy we got with the CNN model, since the prediction could not be well tuned by using a small dataset in our model. Therefore, we will feed the models with a large size dataset and also optimize our CNN model to adapt for more application scenarios in the future.

References

Wang, W. Y. (2017). ” liar, liar pants on fire”: A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Vlachos, A. and Riedel, S. (2014). Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22.

Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Conroy, N. J., Rubin, V. L., and Chen, Y. (2015). Au- tomatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Khurana, U. and Intelligentie, B. O. K. (2017). The linguistic features of fake news headlines and statements.

Shu, K., Mahudeswaran, D., Wang, S., Lee, D., and 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*.

Bhattacharjee, S. D., Talukder, A., and Balantrapu, B. V. (2017). Active learning based news veracity detection with feature weighting and deep-shallow fusion. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 556–565. IEEE.

Hassan, N., Arslan, F., Li, C., and Tremayne, M. (2017). Toward automated fact-checking: Detecting check- worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1803– 1812. ACM.

Oshikawa, R., Qian, J., Wang, W.Y.(2020). A Survey on Natural Language Processing for Fake News Detection. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 6086–6093. ELRA.

Karimi, H., Roy, P., Saba-Sadiya, S., and Tang, J. (2018). Multi-source multi-class fake news detection. In *Pro- ceedings of the 27th International Conference on Computational Linguistics*, pages 1546–1557.

Long, Y., Lu, Q., Xiang, R., Li, M., and Huang, C.-R. (2017). Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth Interna- tional Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 252–256.

Kirilin, A. and Strube, M. (2018). Exploiting a speaker’s credibility to detect fake news. In *Proceedings of Data Science, Journalism & Media workshop at KDD (DSJM’18)*.

Pham, T. T. (2018). A study on deep learning for fake news detection.

Huilgol, Purva. “BoW Model and TF-IDF For Creating Feature From Text.” *Analytics Vidhya*, 25 June 2020, www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/.

Pennington, J. (n.d.). GloVe Library. Retrieved November 17, 2020, from https://nlp.stanford.edu/projects/glove/

Brownlee, J. (2019, August 07). What Are Word Embeddings for Text? Retrieved November 17, 2020, from https://machinelearningmastery.com/what-are-word-embeddings/

Saha, S. (2018, December 17). A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way. Retrieved November 17, 2020, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Brownlee, J. (2019, August 06). A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. Retrieved November 17, 2020, from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

Summary of Team Roles

We did all model work, video filming, and report compilation together as a team.

Report sections:
Jasmine - Abstract, Part of Background, CNN Results, Video Design and Slides
Ashish - Approach, Methodology & Result for Binary and Multi-class Logistic Model.
Leon - Approach and Methodology for Word embedding, Neural Network.
Charlie - Introduction, Literature review, Conclusion, part of video contents