Fake News Classification

Andy Weng **Mengning Geng**

The University of Texas at Austin
{andyweng, mengninggeng}@utexas.edu

Abstract

In this project we investigated the topic of fact-checking, determining the trustworthiness of a given article or statement. We divided this project into 2 further subsections of fact-checking. In the first section, we tried to classify an example into 4 categories of fake news to examine the linguistic characteristic of fake news to find patterns in fake news categories. Rather than classifying news into categories, in the second part, we tried to predict where the news lay on the scale of the trustworthiness scale from 0-5. Although both are types of fact-checking, we see that they are in fact different tasks through our results.

1 Introduction

Fake news detection assesses the truthfulness of claims in articles. Fake news has a significant impact on everybody in society, especially in a time where news about Coronavirus are impacting our daily lives. The original way of doing fake news detection, which involved having multiple experts check a claim's validity based on previously written facts, is too slow. It would be faster and more convenient if we have this process automated through computers.

Recently, multiple datasets have been proposed to aid this task. For example, LIAR (Wang, 2017), FEVER (Thorne et al., 2018) and POLITIFACT (Rashkin et al., 2017) contain short claims annotated by journalists and trained annotators. FAKENEWSNET (Shu et al., 2018) and UNRELIABLE NEWS DATA (Rashkin et al., 2017) contain long articles. In this work, we pick one dataset from claims and one from articles.

The news types we use in the first task of classifying news can be illustrated in **Figure 1**. We can classify the news types along two dimensions: information quality and the intention of the author. Keep in mind that realistically there is not a clear-cut border between each category of fake news. In addition, not all have to be on the extremes of each side. For instance, propaganda often combines truth and false statements to try to mislead other people. But for the purpose of this project, we need a clear-cut boundary between each category because we are predicting one category and not components.

Satire	**Hoax**
Trusted News	**Propaganda**

No desire to Lie Desire to Lie

Fake Trustworthy

Figure 1: Categories fake news articles based on the intent of the author and the trustworthiness of the information they provide.

For the second task instead of predicting categories, we are predicting the trustworthiness of the news on a scale. The scale used is shown in **Figure 2.**

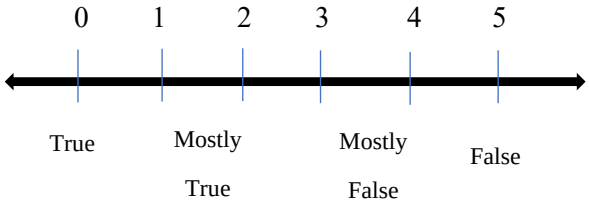

Figure 2: Rates fake news based on a scale from 0 -5.

2 Implementation

2.1 Data Preprocessing

As mentioned previously, we are investigating 2 types of fact checking. Therefore, we also have 2 datasets for this project. The first task uses *Unreliable News Data* and the second task uses *Politifact Data* (Raskin et al., 2017). The first dataset, *Unreliable News Data,* is organized as the first column being the label and the second column is the article see. This dataset has a total of 4 labels (see **Figure 1**). The description provided by the authors of the dataset include:

- ***Satire (label 1)***: mimics real news but still cues the reader that it is not meant to be taken seriously
- ***Hoax (label 2)***: convinces readers of the validity of a paranoia-fueled story
- ***Propaganda (label 3)***: misleads readers so that they believe a particular political/social agenda.
- ***Reliable News (label 4)***: Not fake news.

The second dataset, *Politifact Data,* is given the order of speaker, statement, and truth-rating [0-5] (see **Figure 2**). Since this dataset is different from the first one, we had to come up with two different ways to preprocess the data, to obtain the string and label for each example.

Since some examples were very large, up to 20000 words, we decided to restrict the train dataset by imposing a maximum length. The max length parameter was determined based on the model we are training. For the BERT model, there is a maximum length of 512 wordpiece because that is the restriction the huggingface (Wolf et al., 2019) BERT model has. For the LSTM model, we looked at the entire dataset and saw that majority of the dataset was less than 2048 length.

After refining the training set, we needed to tokenize the examples. To tokenize this dataset, we used the Bert tokenizer provided in the huggingface library, which has 30522 words in the vocabulary. Although this tokenizer was built for BERT, we also used this tokenizer as the indexer for the LSTM model.

The last preprocessing step we did was batching the inputs into the corresponding batch size. Implementing batching allowed us to speed up training for both models.

2.2 Creating and Training BERT

BERT, Bidirectional Encoder Representations from Transformers, model is a bidirectional transformer. This model combines left-to-right and right-to-left training. Unlike the default LSTM model, which only looks at previous states, the BERT model looks at both the previous and next states. The BERT model was pre-trained using only an unlabeled, plain text corpus (namely the entirety of the English Wikipedia, and the Brown Corpus). (Devlin, 2018).

To create the BERT model, we used the huggingface library's pre-trained BERT model. In order to use this model, we needed to get tokenize each example using the BERT tokenizer, so the input is acceptable. In addition, we needed to get the input_ids and attentions_masks of each example, which is needed to pass the example into the pre-trained BERT model.

The final BERT model we trained for 10 epochs, with a batch size of 4, dropout of .1, a leaning rate of 4e-6, Adam optimizer, and cross-entropy loss. When playing around with the hyperparameters, we realized that the learning rate has a big impact on accuracy. **Table 1** shows the different learning rates and the accuracies associated with each learning rate. From the table, we can see that a higher learning rate will lead to higher accuracy. This might be because the dataset is on average smaller, so a lower learning rate will be more precise. Similarly, when tweaking the epochs, we did see a small increase in accuracy.

Table 1: Changes in learning rate versus accuracy.

For instance, for epochs of 10 the accuracy is around 26% and 28% for epochs of 20. This increase is most likely because our model converges slowly; the loss decreases at a slower rate. The final BERT model we choose to use 10 epochs because of the tradeoffs between the accuracy and time and potential of overfitting the model.

2.3 Creating and Training LSTM

We created the LSTM model (Hochreiter and Schmidhuber, 1997) using the PyTorch LSTM. The first thing we needed to do was embed our input. We used the BERT tokenizer mentioned earlier as the indexer to convert each word with its associated unique id. We did not use a pre-trained word embedding, instead, we created out our embedding layer, with dimensions we are able to modify.

The final LSTM model we trained for 10 epochs, hidden size of 128, input size of 128, batch size of 16, dropout = .1, 2 layers, a learning rate of .01, and using the Adam optimizer.

When trying to optimize the LSTM model, we found out that parameters that had the biggest impact on the performance of the model were dropout and number of layers. The model with no dropout and one layer had a performance of around 57% for the first task of classifying fakes news into 4 categories. After we added a dropout and an extra layer, the performance jumped to around 77% for classifying task.

3 Result/Analysis

3.1 Baseline Comparison

The only baseline comparison we found was for task 2 using LSTM. The accuracy that the baseline model had was 21%, which was lower than our model's accuracy (Raskin et al., 2017).

3.2 Task 1 vs Task 2

As mentioned above task 1 is classifying fake news into one of the 4 categories. The categories include satire, hoax, propaganda, and trusted news. On the other hand, task 2 predicts where the news lay on the scale of the trustworthiness scale from 0-5. Both tasks are considered to be part of fact checking; however, by looking at our results we can see that they are indeed very different.

	LSTM	BERT
Task 1	77.8%	89.05%
Task 2	21.9%	27.5%

Table 2: Performance of task 1 and task 2 using the BERT or LSTM model.

Table 2 shows the final performance for task 1 and task 2, using the LSTM model and the BERT model. From the table, we can see that there is a big difference in performance between the two tasks. Task 1, the classifying task has around a 50% better performance than tasks 2.

There are a few things that can cause this to happen. One reason could be because the dataset we used for the first task and the second task is very different. An example in the first dataset is on average longer than the second. For instance, the longest example in the first dataset is around 20000 words. This means that there is more context to look at for each timestep. That could impact the performance of the model.

Another reason, which I think is the most feasible reason, is because of the nature of each task is different. The labels for task one are different categories of fake news. They have a more defined difference between them. For instance, you can tell that an article is a satire by looking at its characteristics. However, for task 2

Learning Rate	Accuracy
1e-5	21.91
2e-5	21.6
2e-3	17.6
2e-6	26.26
4e-6	27.52

the boundaries between each label are very blurry. How do we tell the difference between true and mostly true? It is much harder to determine the difference between labels for task 2 so the accuracy for task 2 is a lot lower than task one.

Similarly, one could say that the number of classes for task 1 and task 2 is different, which can lead to a performance difference. Task one has 4 labels, while task 2 has 6. This means that task 4 has a higher percentage chance of predicting the correct label even if we randomly guess labels.

3.3 BERT vs LSTM

Based on the results in Table 2 we can see that the BERT model outperforms the LSTM model for both task 1 and task 2. After seeing such a trend, we investigated the difference by looking at what each model predicts vs the expected label.

For the LSTM model, we saw that majority of the predictions that LSTM got right were the labels that were two. Similarly, the label that the model precited the most is also two. For the BERT model, we saw that majority of the correct predictions were labels 1 and 4. The BERT model gave a larger range of predictions. Unlike the LSTM model, which predicted mostly 2's and 0's, the BERT model's predictions were more distributed.

By looking at the difference, I believe that the BERT model does a better job predicting the extremes compared to the LSTM model. Label 2 would be considered to the middle of the scale, so it seems like the LSTM model does not do a good job determining the degree truthfulness. The LSTM model just predicts 2 for many examples because it does not know the degree of how true a statement is. The BERT model, on the other hand, is more confident in its predictions. Rather than predicting the middle like the LSTM model, the model labels that the model predicts are closer to the extremes. This means that the BERT model does a better job predicting the truthfulness for a given example.

4 Conclusion

Fake news classification is a very important topic because it effects a large population of people. Similarly, it is a very hard task because of the most articles are not all completely true or false, so it is harder to determine. This can be shown through our low accuracy from task 2. We have shown in our paper that classifying fake news into categories is not as hard as putting them on a number line scale. Similarly, we found out that the BERT model performs better than the LSTM for this task. Although the BERT model outperforms the LSTM model, the accuracy is still not good. There might be better models out there more suited for this task 2.

References

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780

Oshikawa, R., Qian, J., & Wang, W. Y. (2018). A survey on natural language processing for fake news detection. arXiv preprint arXiv:1811.00770.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Desmaison, A. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (pp. 8024-8035).

Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017, September). Truth of varying shades: Analyzing language in fake news and political fact-checking. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 2931-2937).

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.

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.

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

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. ArXiv, abs/1910.03771