Fake Review Detection on Yelp Dataset

Ferhat Turker Celepcikay & Jing Ning

Introduction

Online product reviews have been essential when people make decisions on which option to choose. The popularity of online reviews cultivated in a significant rise in fake review writing. Fake reviews are biased and misleading. In order to provide users with more reliable review information, we aim to build a classification system to detect fake reviews. Most of the fake review sentences can appear to be normal even for humans, building fake review detection systems can be a challenging task compared to some other common NLP systems.

Literature Review

Fake review or spam detection can be a challenging task as it is quite complex. According to Ahmed et al. [1], fake reviews can broadly be categorized into three groups. Their first group is the fake reviews whose main purpose is to present false information about the product either to enhance its reputation or to damage it. The second group targets the brand but does not express an experience with a certain product. The last group is non-reviews and advertisements that contain text only indirectly related to the product. Based on their research, group 3 and 2 are easier for algorithms to detect, whereas the first group is more problematic. In our work, we are only considering group 1. They propose a pre-process with n-gram feature representation and implement SVM, KNN, logistic regression, and decision tree. Like their research, we are also using n-grams. From their results, SVM in combination with Unigram was able to achieve the best result accuracy of 92%. They have not explored deep learning models, whereas we are planning to utilize DL to maximize accuracy scores. Hence, two approaches are complementary.

There are also some other non -traditional approaches. Rayana et al. [2] use a method called SpEagle where it extracts clues from all metadata and fits them into a unified framework. The networks are defined with pairs of joint probability of labels with Markov Random Field model. They used a semi-supervised model to classify results. It’s hard to compare results with supervised and semi-supervised learning, they were able to achieve AP above 90% for a good quality dataset. On the other hand, our approach is supervised, hence we think two approaches are orthogonal.

We also found out that there is a very similar work for a CS229 project in 2017 by Wang et al. [3]. Like our work, they also extracted hand-engineered features including reviewer-centric features. On top of this, we also extracted product-centric features. Also, they fit a Latent Dirichlet Allocation (LDA) topic model to the grams to obtain new features, something we have not yet explored. For modeling, besides the classical machine learning models, they only used a simple 3 layer fully connected neural network. We aim to leverage more advanced architectures to surpass their accuracy. Therefore, we think two approaches are complementary.

Dataset and Features

The dataset we use is the ‘New York City Restaurants review’ from the dataset provided by the author of ‘Collective Opinion Spam Detection: Bridging Review Networks and Metadata” [2]. The data is organized by the date of the review and is associated with a Yelp user ID and a Yelp product ID. Each review is also associated with a discrete rating value ranging between 1 and 5. The dataset also has labels generated by Yelp’s filtering algorithm. It spans reviews from 2004 till 2015. Out of total of 359052 reviews, only 36885 are fake, rest labeled real. Therefore, the original dataset is too big and imbalanced. So, we decided to use data solely from the years 2011-2013. Also, we preprocessed the original data to obtain a balanced dataset. The training dataset is generated by using reviews from the year 2011. The training dataset consists of all the 5562 fake reviews as well as an equal number of real reviews that are sampled randomly. For the validation dataset, we randomly sampled 1000 fake and 1000 real reviews from the year 2012. The testing dataset consists of 5000 fake and 5000 real reviews randomly sampled from the year 2013. Hence, we have a perfectly balanced dataset.

We also extract features by utilizing pre-trained word embeddings, and n-gram models.

At the same time, a range of new features was added to the feature vector as an experiment.

> Reviewer- Centric Features: no. of reviews given by user, the average rating given by user, average no. of words in reviews by user, max reviews given by user in a day.
>
> Product-Centric Features: no of reviews for the restaurant, the average rating for restaurant, average no. of words for a restaurant review, max reviews received by the restaurant in a day.
>
> Review-Centric Features: Character count, word count, word density, punctuation count, upper case word count, top 1000 unigrams and bigrams that occur more frequently in fake reviews than real reviews, top 1000 unigrams and bigrams that occur more frequently in real reviews than fake reviews.

The derived features provide more information about the reviews and reviewers. Our goal is to classify reviews with and without metadata and assess the significance of adding hand-engineered features.

To alleviate overfitting to the training dataset, we created a larger dataset which is a super-set of the original training dataset. In addition to the 2011 data, we have added 2010 data: all of the 3306 fake reviews and 3306 randomly sampled real data. Similarly, we created an extended validation dataset, consisting of all of the 2012 fake reviews (6153 reviews) and 6153 randomly sampled real reviews.

We observed that fake reviewers tend to give more 1s and 5s compared to real reviewers as demonstrated in Figure 1a. Interestingly, as is evident in Figure 1b, fake reviewers tend to write more on Tuesday-Friday, while most real reviews are written on Sundays and Mondays. Another interesting note we derived is that most real reviewers write max one review a day, whereas fake reviewers may generate as high as 9 reviews a day. This is from Figure 1c, which is a CDF for max no of reviews in a day per user. Finally, fake reviews are shorter in length.

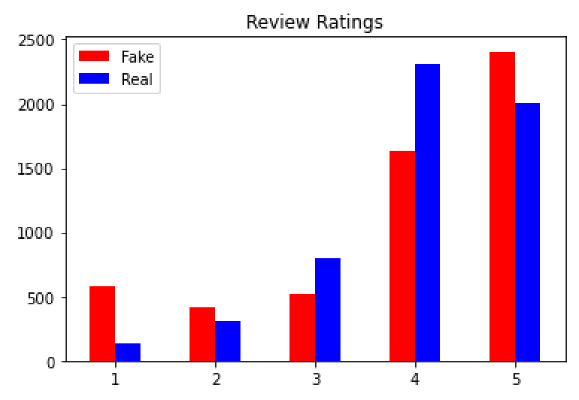

Figure 1a. Fake vs Real Review ratings

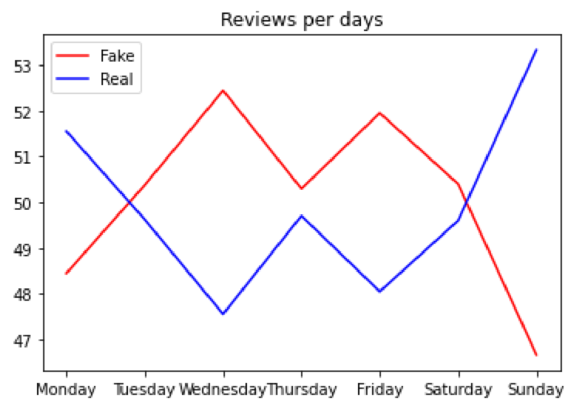

Figure 1b. Fake vs Real Review per days

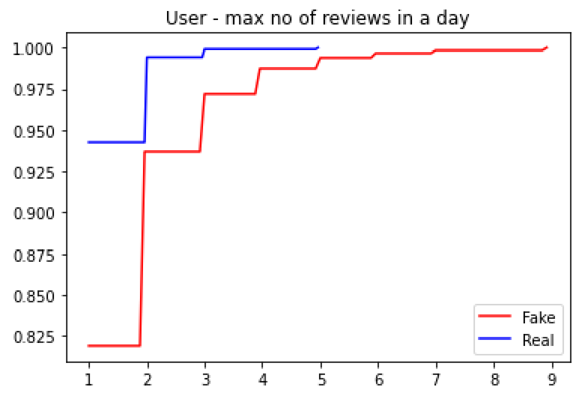

Figure 1c. Max reviews in a day by a user

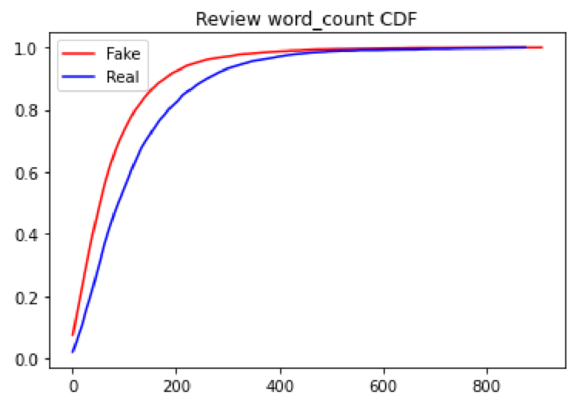

Figure 1d. Word counts

Methods

We have implemented several different machine learning models. For all models, we are using binary cross-entropy as our loss function.

Model 1: Baseline: Naïve Bayes (NB):

We adopted a Multinomial Naïve Bayes (MNB) model as our baseline because of its simplicity and it is a very standard choice for NLP related tasks. Naïve Bayes is a generative model. The set of words in the text are encoded into a feature vector, which is called the vocabulary. This model has a very strong assumption that features are conditionally independent given the label. This method is known to perform well on text-based tasks.

We first used only the review data from users, excluding other features. We ended up getting 57% accuracy, which is low. Also, using a count vectorizer in the training with the default setting of hyperparameters resulted in high variance. To mitigate the high variance issue, we tried feature reduction (selection) and hyper-parameter tuning. We first used Principal Component Analysis (PCA) to reduce the number of features to 300. Then, we tuned hyperparameters by running a grid search. Moreover, we decided to incorporate hand-engineered features including reviewer-centric features, review-centric features and product-centric features. Finally, we overcome high variance and got 65.4% test accuracy.

Model 2: A Simple Neural Network (NN):

A simple Neural Network approach with word embedding is used as an alternative approach. For word embedding, it uses an embedding network that maps text to a 50-dimensional embedding vector. This embedding is based on feed-forward Neural-Net Language Models with pre-built OOV. Then, we have a 64 unit fully connected layer. A linear learning rate schedule is used in the training of the model. The initial learning rate is set to 0.001, then the learning rate is halved at every 5 epochs. The total number of parameters of the model is 48,193,929. The batch size of 256 is used for training on a single GPU. We also use checkpoints to keep track of the best validation model. It achieved an accuracy of 77.8% for training, whereas the test accuracy is 63.6%. The model was trained for 20 epochs and is certainly overfitting. Besides alleviating the overfitting issue, we would like to utilize hand-engineered features.

The following graph shows the architecture of the network.

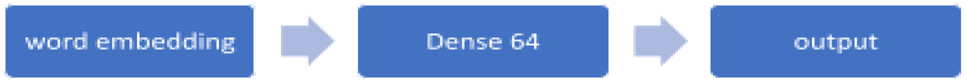

Figure 2. Simple pre-trained embedding neural network architecture graph

Model 3: LSTM Network (LSTM):

A simple LSTM network is also trained since LSTMs are one of the best tools for NLP-related tasks. Index tokens are used for this model's embedding layer. We choose a vocabulary size of 1000 and the same training strategy as Model 2. The total number of parameters of the model is 174,129. All other hyperparameters are the same as the second model. With fewer parameters, it achieved an accuracy of 71.1% for training, whereas the test is 61.1% which is around 2.5% less compared to model 2. The following graph shows the architecture of the network.

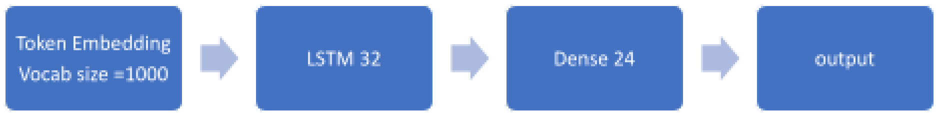

Figure 3. LSTM network with token embedding architecture graph

Model 4: LSTM Network with CNN encoder (LSTM-CNN):

From our previous analysis, the model is not generalizing well to unseen test restaurant reviews from a different year. In order to increase model robustness, firstly, we added an extra 17,737 reviews with 50% of each class distribution to our training dataset. In addition to that, we also combined the fraud-related hand-engineered features with word embedding. Log-scale normalization was performed to hand-engineered features to reduce the input scale.

For the neural network architecture, we want to add some encoding layers to enhance the feature extraction process. According to Soroush et al. [4], they used CNN as a feature extractor for sentiment analysis. Hence, 2 layers of 1D CONV with max-pooling were added to represent the feature extraction process. The features will then feud into 3 layers of LSTM network. The total number of parameters for this model is 940,385.

A similar training strategy is used for this model. It was able to achieve a high level of training accuracy of 97.6% after 20 epochs but suffered from overfitting. So we picked the best validation model. Compared to Model 3, the test accuracy increased by 3.5% to 64.6%.

The following graph shows the architecture of the network.

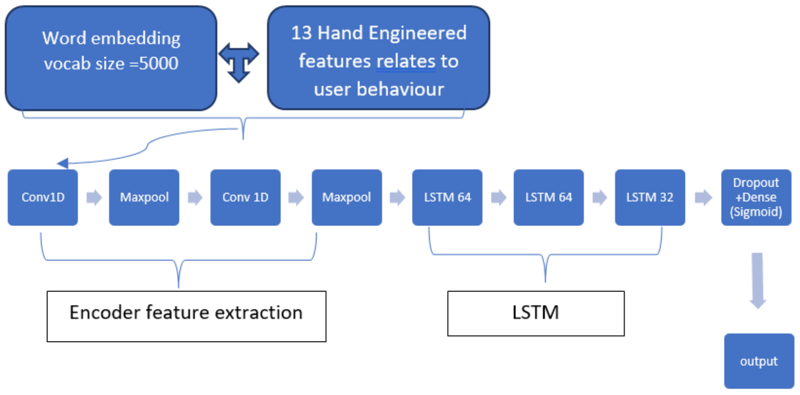
Word embedding vocab size =5000
13 Hand Engineered features relates to user behaviour
Conv1D
Maxpool
Conv 1D
Maxpool
LSTM 64
LSTM 64
LSTM 32
Dropout +Dense (Sigmoid)
Encoder feature extraction
LSTM
output

Figure 4. LSTM network with CNN encoder architecture graph

Model 5: Light Gradient Boosting Tree (GBM):

The Gradient Boosting model is known to work well with sparse feature representations. In order to compare the difference between a neural network and a tree-based model, we trained a light gradient boosting tree. It uses both sentence embeddings and hand-engineered features. A binary log loss was used with 400 leaves and a maximum tree depth of 7. To prevent overfitting, a subsample of 0.8 is used. A comparison between the results on the confusion matrix is shown in Figure 7 and it will be analyzed in-depth in the error analysis section. The gradient boosting tree was able to achieve around 3% higher test accuracy compared to LSTM-CNN.

Model 6: Transfer Learning using BERT (BERT):

BERT became very popular in the machine learning community after presenting superior results in a wide variety of NLP tasks. Its key difference with previous efforts is to look at a text sequence either from left to right or combined left-to-right and right-to-left training. By using a novel technique - Masked Language Modelling (MLM), bidirectional training in models is achieved. We utilized BERT via transfer learning to increase the accuracy of fake review prediction. To adapt BERT to our sentiment task, we added a final layer for which parameters are trainable.

Hyperparameters including ‘keep_prob’, the batch size and the number of train epochs were tuned. ‘keep_prob’ is the dropout hyperparameter which helps prevent overfitting. For the best-performing model, ‘keep_prob’ was set to 0.55. The mini-batch size is 32 and the learning rate is 2e-5.

Evaluation Metric

To evaluate the success of our problem, accuracy is our primary evaluation metric as the dataset becomes balanced after the data preprocessing. In the referenced works, accuracy is the common metric. So, we are aligned with them in our metric choice. To better understand the quality of the prediction, we also provide F1 scores and AUC-ROC curves. We also look at the confusion matrix and perform error analysis.

Result & Analysis

We have fully implemented our baseline as well as 5 other models. The main differences between the models are two-fold: the model architecture and word embedding methods. For model 2 (simple neural network) and model 3 (LSTM), word embedding is used as a feature representation of the input sentences. However, these models are not using hand-engineered features. On the other hand, all other methods, namely NB, GBM, LSTM-CNN and BERT, are utilizing hand-engineered features. Therefore we have generated with and without metadata results for these models.

In Table 1, we compared the results for all models in two metrics, namely, accuracy and F1-score. Based on the results in the table, our baseline model, Multinomial Naïve Bayes (NB), seems to be a good fit. We have already incorporated the aforementioned hand-engineered features into NB. We also performed a grid search for hyper-parameter tuning. Hence, there is not much room for NB to improve. GBM gets better results compared to NB, yet it is worse compared to BERT. As expected, BERT model is the best performing model. However, it definitely overfits the training dataset.

Table 1. Model accuracies and F1-scores

	Accuracy			F1-Score		
	Training	Validation	Test	Training	Validation	Test
NB	0.708	0.657	0.654	0.716	0.682	0.681
NN	0.778	0.641	0.636	0.787	0.655	0.654
LSTM	0.711	0.631	0.611	0.731	0.664	0.653
LSTM-CNN	0.731	0.655	0.646	0.760	0.704	0.703
GBM	0.789	0.695	0.674	0.792	0.717	0.709
BERT	0.999	0.813	0.757	0.999	0.816	0.766

On the other hand, in order to compare the influence of different neural network architectures, the LSTM-CNN was created with much more layers compared to the simple LSTM and NN. With large parameters, LSTM-CNN was able to achieve a test accuracy improvement of 2-3% and also an F1 score improvement of 5%. We also trained LSTM-CNN with additional hand-engineered features to improve performance. In order to reduce overfitting, “recurent_dropout” and the dropout layer with probability = 0.5 are used. However, due to the large parameters in the model and relatively small training dataset, the model still overfits easily after a few epochs. Some strategies include data augmentation, collecting more data, adding regularization, reducing network complexity, better learning rate schedule, and also early stopping can be used for fine-tuning better results.

Our best-performing model, BERT, is overfitting to the training dataset. To mitigate the overfitting, we decided to use more data. As demonstrated in Table 2, we utilized extended training as well as extended validation datasets. The first row in Table 2 denotes the original result that is obtained by utilizing BERT and it is the same as the corresponding row in Table 1. In the second row, we have used the extended training dataset, which is a super-set of the original training dataset. This yields accuracy increases in both validation and test results. In the third row, we used the extended validation dataset in conjunction with the extended training dataset. This also improved the accuracies in both validation and test results. Hence, by adding more data to the training dataset as well as to the validation dataset, we improved the performance of our model.

Table 2. BERT – Using more data

	Accuracy		
	Training	**Validation**	**Test**
Original Dataset	0.999	0.813	0.757
Extended Training	0.999	0.826	0.766
Extended Validation	0.999	0.842	0.779

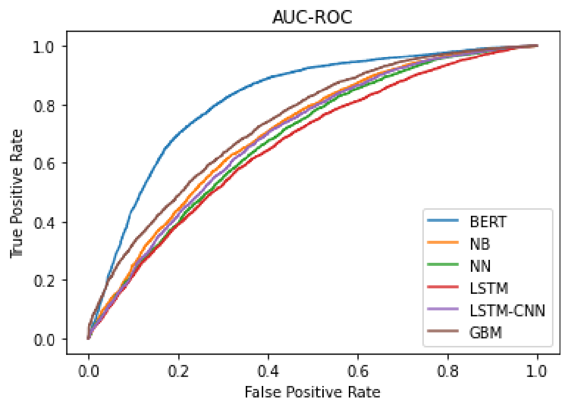

Figure 5. AUC-ROC curves for all models

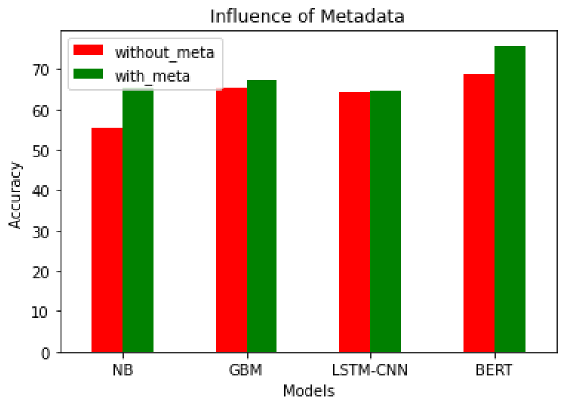

Figure 6. Influence of Metadata

In addition, the receiver operating curve for the test dataset, also noted ROC, is the plot of TPR versus FPR by varying the threshold. From the AUC-ROC, we can see that BERT has the largest area under the curve, followed by GBM and NB. This explains the performance of the models in terms of accuracy as well.

In order to compare the influence of using Metadata in hand-engineered features, we have generated with and without metadata results comparison. Figure 6 shows that all the models have about 60-65% accuracy when just the reviews (without Metadata) were used. But once the metadata was utilized using the aforementioned hand-engineered features, the accuracy got boosted in almost all the models. BERT got an accuracy improvement of about 10%. This explains that Yelp's spam filter works not just based on the reviews, but also based on the metadata. So, with the metadata, the model was able to predict the reviews better.

Error Analysis:

Figure 7 shows confusion matrices for both BERT and GBM. Confusion matrix provides per group performance of a model. In this figure, it is evident that both BERT vs GBM is more successful in predicting fake reviews while struggling in predicting real reviews. GBM has as high as 2214 false positives, whereas BERT has 1411. The models may not be able to achieve good accuracy, however, recall is relatively high around 79% compared to precision. The evaluation matrix selection could be totally different based on the real-world scenario. For example, for credit card fraud transactions where banks are willing to trade-off some false positives to get more true positive fraud transactions. In this situation, recall can be chosen as the primary evaluation matrix as it is a risk-averse choice.

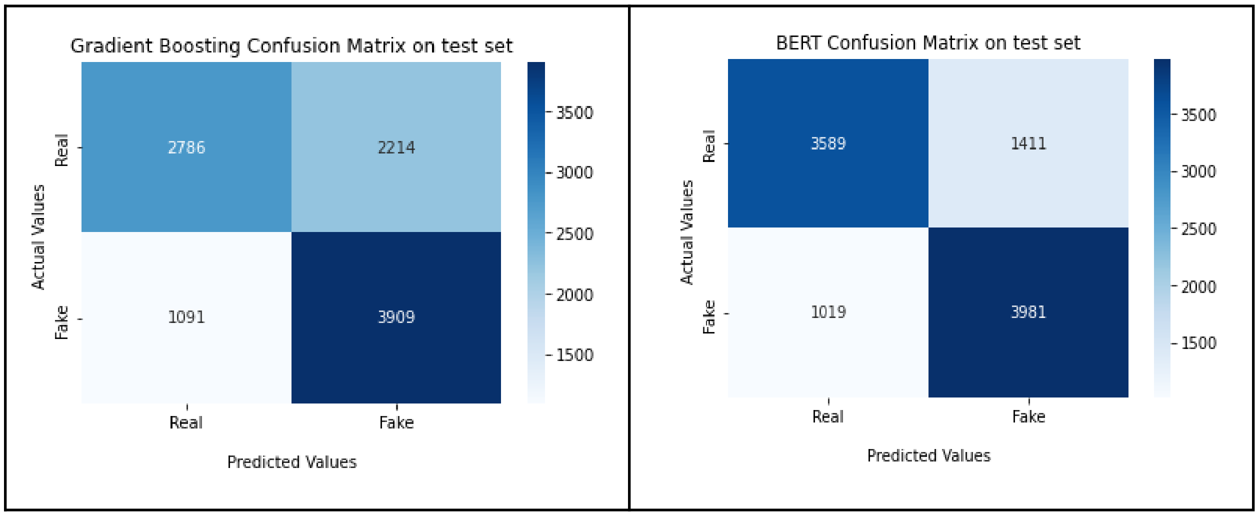

Figure 7. BERT vs GBM confusion matrix

The training callbacks for LSTM-CNN are also analyzed. As it can be seen from Figure 8, the model overfits to the training dataset with high accuracy at 97.6% under 20 epochs. However, as the loss of validation starts to go up at around epoch 5, the training is starting to overfit. We use callbacks to save the best validation model, but an early stopping strategy could also be used to prevent further training.

Relatively small training size result in overfitting. To improve on that we can collect more data and reduce model complexity. Secondly, there's a data mismatch problem between training, validation and test set since they are all from different years restaurant review. Lastly, as discussed from the confusion matrix section, the per-group accuracy such as recall could be a better choice for the primary evaluation metric based on the scenario.

As we also experienced human labeling fake and real sentences, the performance varied between 30% - 70% accuracy. So the models can outperform human labeling just based on limited information. The problem was expected to be a challenging one.

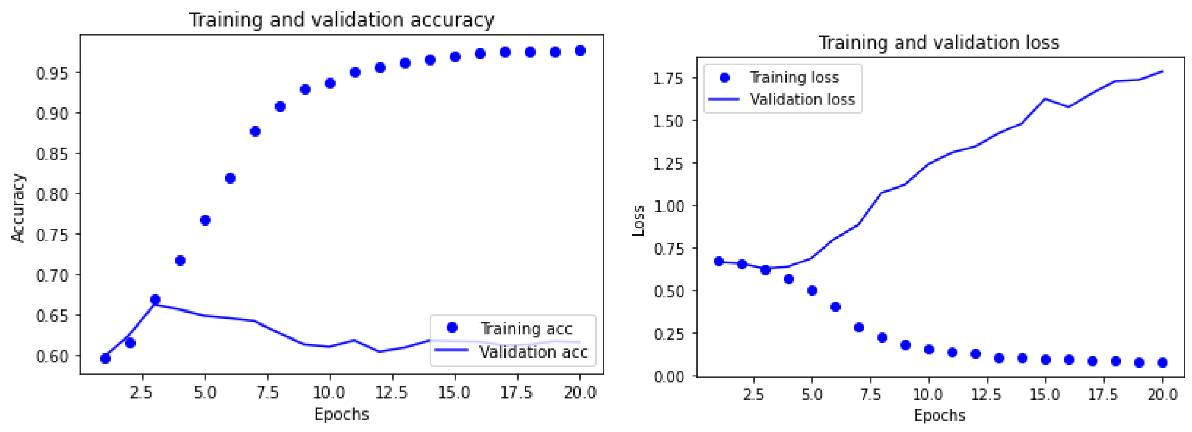

Figure 8. Accuracy and loss info from model 4 LSTM with CNN encoder

Conclusion

To conclude, from our various model building and data manipulation experiences, adding hand-engineered behavior features has proven to be useful for detecting fake reviews. The model's performance also benefited from more complex architecture such as BERT or CNN-LSTM for a given fixed dataset. Different model architecture, such as GBM could be an efficient algorithm to quickly train for a classification problem. It works well with large sparse data such as our word embedding representation. However, the robustness of any model over a time horizon is still a challenging task as the online environment is changing dramatically every day. For a fast-paced online environment, there's a need for real-time spam detection models.

Future Work

The problem of detecting fake reviews is considered to be more challenging compared with some of the traditional NLP sentiment analysis tasks. We believe collecting more genuine user behavior data could benefit the algorithm to extract important features. Some other methods such as knowledge graphs are commonly used to identify the connection between new users and existing known fraud groups. According to Qing et al. [5] , they use a knowledge graph to detect fraud in credit cards by using the borrower's phone network. As fake restaurant reviews can be deceptive to humans, this will also introduce confusion to the machine learning models.

We believe with the ability to get much more user behavior data, real-time model updates, it is possible to build a robust algorithm to detect fake reviews. Some future work is as follows:

- Hyperparameter tuning of the LSTM-CNN model has to be done to achieve the best possible accuracy.
- Hyperparameters of BERT have to be fine-tuned in order to alleviate overfitting and achieve better accuracy.
- Confusion matrices suggest maybe using more real reviews would be beneficial. We are going to explore different proportionalities of real and fake data.
- The influence of adding hand-engineered features is obvious. We will explore more hand-engineered features.
- Ensembling of models generally yields an improved result.

Video Link: https://youtu.be/TmiAaI77m24

Code: Please see the zipped file

Dataset:
https://drive.google.com/file/d/1DDjfE64jqARg5Svmjf9X87uZmlJHj5r7/view?usp=sharing

Shared Google Drive:

https://drive.google.com/drive/folders/1LoFtjhSClNgBVO-3QgCiv3hjD0WONy0Q?usp=sharing

References:

[1] Hadeer Ahmed, Issa Traore, and Sherif Saad."Detecting opinion spams and fake news using text classification." Security and Privacy 1.1 (2018): e9.

[2] Shebuti Rayana, and Leman Akoglu. "Collective opinion spam detection: Bridging review networks and metadata." Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining. ACM, 2015.

[3] Zehui Wang , Yuzhu Zhang , Tianpei Qian , http://cs229.stanford.edu/proj2017/final-reports/5229663.pdf

[4] Soroush Vosoughi, Prashanth Vijayaraghavan and Deb Roy. "Tweet2Vec: Learning Tweet Embeddings Using Character-level CNN-LSTM Encoder-Decoder" . Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016

[5] Qing Zhan , Hang Yin. "A loan application fraud detection method based on knowledge graph and neural network" .Proceedings of the 2nd International Conference on Innovation in Artificial Intelligence . ACM, 2018