Fake Review Detection

DS-GA 1003 Machine Learning Project

Group:

Aajan Quail	aqd215	
Abha Sahay	as13492	
Christine Shen	zs1534	
Meenakshi Jhalani	mgj265	
Viraj Thakkar	vt943	(Member responsible for submission)

Contents

1 Introduction

Reading online reviews has become commonplace in our technology-driven world. Whether it be for restaurants, movies, books, TV shows, or one of the billions of products on e-commerce platforms, people often refer to the opinions of others in order to inform their own. As a result, it is important for companies like Yelp, Amazon, and AirBnB, whose business models strongly depend on accurate reviews, to be able to detect which reviews are real and which ones are not. Such a procedure will greatly benefit those businesses, as flagging fake reviews will make their platforms more reliable overall; in turn, this will boost consumer confidence in the platforms and increase their revenues. In this report, we explore a supervised-learning-based approach to efficiently and accurately detect from a large set of restaurant reviews that are fake. We explore several models, including logistic regression, Naive Bayes, Random Forest, Light GBM and Deep Neural Networks.

2 Related Work

Fake review detection is a problem that has been addressed many times in the past. [1] addresses fake reviews for consumer electronics and reached an 82% F-Score using an AdaBoost classifier. [2] experimented with logistic regression, linear discriminant analysis, Naive Bayes, support vector machines and neural networks; [3] employed Support Vector Machine (SVM), Naive Bayes (MNB), and Multilayer Perceptron, successfully detecting fake reviews with more than 86% accuracy.

3 Problem Definition

This report implements supervised machine learning methods as a means of binary classifying fake restaurant reviews. Our dataset contains Yelp data of fake and geniune reviews of restaurants. Labels denote the review's class i.e. fake or genuine. As inputs for our models, we include the rating of the restaurant and the review itself as our features. Our goal is to predict the labels correctly on the test data set based on performance of metric AUC on the validation dataset.

4 Experimental Evaluation

4.1 Data

We are given data that is already split into train and validation sets. For both datasets, we have columns ex_id, user_id, prod_id, date, rating, label and review. For our models, we kept label, rating and review and dropped the other columns.

Data set	Fake reviews (label=1)	Genuine reviews (label=0)	Total
Train	25819 (10.29%)	225055	250874
Validation	3648 (10.16%)	32270	35918

4.2 Data Preprocessing

We decided to create featurized vectors using Term Frequency-Inverse Document Frequency (TF-IDF). This method works by increasing values for words proportionally to the number of times that word appears in a document, but is offset by the number of documents that contain the word. We decided to simplify the problem by limiting the number of words used to train the model. After doing tuning for max_df and max_df [4], we found that the using 923 features in total decreased model fit time and did not significantly decrease AUC performance. In addition to the 923 text features, we also used the rating as a feature. As a result, each of our models was trained on 924 features.

4.3 SMOTE

In addition to modifying our algorithms to be better fitted to handle imbalanced data, we make use of Synthetic Minority Over-sampling Technique (SMOTE), a sampling technique, to artificially create more balanced data. SMOTE will only be applied to training data to optimize our model. Since we have already chosen algorithms appropriate for imbalanced data, we compared

the outcomes of the model using both treated and untreated training data and choose the best configuration[5].

4.4 Algorithms

4.4.1 Logistic Regression

We start with the simplest of model which serves as our baseline model. We implement Logistic Regression with sample weights as the data is imbalanced. This basically amounts to increasing the weight of the minority class by $n/(2 * n_{label=1})$ and the majority class to $n/(2 * n_{label=0})$ in the log-loss calculation, where n represents counts.

4.4.2 Compare Logistic Regression result with and without SMOTE

To test the effectiveness of SMOTE, we compared results of a logistic regression model using both original data as well as data that has been treated with SMOTE. We only applied SMOTE to the train data set. As shown in the table above, while the train AUC score of treated data is better than untreated data, its validation AUC score is worse than untreated data. The model is likely overfitting in the case of SMOTE. As a result of this experiment, we decided against applying SMOTE as a sampling technique to train data for any other models.

Data Treatment	Dataframe shape	Train AUC score	Validation AUC score
SMOTE	(170951, 924)	0.773	0.704
Weighted imbalance classes	(286792, 924)	0.730	0.717

4.4.3 Naive Bayes

The Naive Bayes classifier attempts to predict the class of an example given its features (words) using Bayes rule and assumes conditional independence between features given a particular class. Though simplifying the problem in such a way is convenient, such independence is almost never true - there usually is some degree of correlation between words. An AUC of 0.617 was achieved.

4.4.4 Random Forests

The Random Forest model is a bagged version of the decision tree [6]. Some hyperparameters that we tuned for included "n_estimators", which represents the number of trees in the forest and "max_depth", which represents the maximum depth of the tree. Since our dataset is imbalanced and the RF classifier tends to be biased towards the majority class (i.e. the review is genuine), we placed a higher penalty for misclassification of minority class (i.e. the review is fake) by setting the parameter class_weight='balanced.' After tweaking hyperparameters, we got AUC=0.697 for the validation set.

4.4.5 Light GBM

Light GBM is a gradient boosting framework that uses a tree based learning algorithm[7]. Since we have a relatively large dataset, light GBM seems to be a good choice to handle large size of data and take lower memory to run. Num_leaves and max_depth were tuned to control the complexity of the model and to reduce overfitting. We set bagging fraction and feature fraction so that we could select only a subset for samples and features to decrease training time. Furthermore,we

set 'is_unbalance' to True to deal with class imbalance. We used grid search to determine the optimal set of hyperparameters for the best results. This resulted in AUC=0.708 on validation set for the configuration

4.4.6 Deep Learning Networks

As a build-up on Logistic Regression, we implemented Deep Neural Networks as our final model which resulted in the best performance on AUC[8]. This was a sequential dense Neural Network with the following architecture was used: 1 Input Layer: 924 nodes, 3 Hidden Layers: 924 nodes each and 1 Output Layer: 1 node(Sigmoid probability). ReLU and sigmoid activation functions were used to fit the highly non-linear dataset [9], [10]. Droupout rate was set to 0.25 and used with l2-regularization to avoid overfitting of data [11]. The loss function used was Mean Square Error(MSE). We obtained the best AUC score for this model as shown in Fig.1.

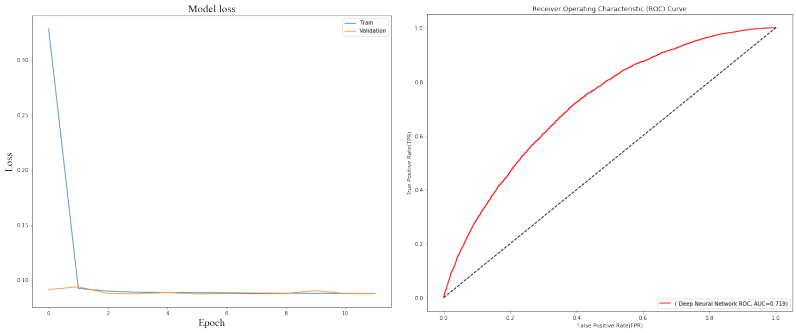

Figure 1: Left: The learning curve for Training and Validation set. Right: ROC curve for DNN which resulted in the best AUC≈0.72

5 Results and Conclusions

We obtain the following results from the models trained above. The performances are better with complex models like LigthGBM and Deep Neural Networks as compared to simpler models, which is expected. However, there is not much significant difference in the AUC. This may probably be due to the fact that the 924-dimensional features essential captures most of the information from the dataset.

Model	AUC score
Logistic Regression	0.717
Naive Bayes	0.671
Random Forest	0.697
Light GBM	0.712
Deep Neural Network	0.719

6 Outlook and Shortcomings

We used relatively simple techniques to process text data and did not employ models specifically designed for natural language processing. Methods such as word embeddings and semantic orientation are not explored in this paper. Numerous neural network models, such as BERT, are proven to be extremely effective at handling natural language based data. We did not explore these methods and models that are specific to text based data, resulting in less than ideal outcomes.

In one of our experiments, however, input text is featurized with GloVe embeddings and we built a simple BiLSTM classifier. Unfortunately, this experiment failed to run successfully locally due to memory issues. In the future, we hope to experiment on high computer power clusters to ensure the smooth and efficient execution of neural network models.

Contributions

Aajan Quail(aqd215):	Data preprocessing, Logistic Regression and Naive Bayes
Abha Sahay(as13492):	Light GBM model, hyperparameter tuning and evaluation
Christine Shen(zs1534):	SMOTE Sampling techniques for imbalance class, GloVe Embedding and BiLSTM classifier
Meenakshi Jhalani(mgj265):	Random forests and hyper-parameter tuning and evaluations for models.
Viraj Thakkar(vt943):	Data preprocessing, Logistic-Regression(weighted) and Deep Neural Networks

Github: https://github.com/virajthakkar/Fake-review-detection

References

[1] Barbado, Rodrigo, Oscar Araque, and Carlos A. Iglesias. "A framework for fake review detection in online consumer electronics retailers." Information Processing & Management 56.4 (2019): 1234-1244.

[2] Z. Wang, Y. Zhang, T. Qian. "Fake Review Detection on Yelp" (2017).

[3] Hossain, Md Forhad, "Fake Review Detection using Data Mining" (2019). MSU Graduate Theses. 3423.

[4] https://stackoverflow.com/questions/27697766/\penalty-\@Munderstanding-min-df-\penalty-\@Mand-max-df-in-scikit-countvectorizer

[5] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[6] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[7] https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-\penalty-\@Mhow-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

[8] https://keras.io/

[9] https://www.machinecurve.com/index.php/2019/09/04/relu-sigmoid-and\penalty-\@M-tanh-todays-most-used-activation-functions/#rectified-linear-unit-relu

[10] https://towardsdatascience.com/exploring-activation-functions-for-neural-networks-73498da59b02

[11] https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/