Fake Reviews Detection for Imbalanced Dataset

Kuan-Lin Liu
Center of Data Science
New York University
New York, United States
kll482@nyu.edu

Hao-Ning Wu
Courant Institute of Mathematical Sciences
New York University
New York, United States
hnw244@nyu.edu

***Abstract*—Online reviews provide customers and business owners a good evaluation metric for the target products or services. Therefore, knowing the authenticity of reviews is essential, because they may affect how we make a decision. In this work, we tackled the fake review detection problem by designing new features, re-sampling training data and ensembling different machine learning models. Our best performance XGBoost model achieved 0.404 AP and 0.862 AUC score on test data.**

***Index Terms*—Fake Reviews, Imbalanced Classification, Ensemble, Feature Engineering, Natural Language Processing**

I. Introduction

In the old days, gossip and news were spread by word of mouth. To form an opinion, people mostly relied on the information heard from their close friends or relatives. Nowadays, people can easily read others' opinions on the Internet. Thus, the crowd consensus has become a strong source of influence toward ones' opinions. For example, when we go out for dinner, the first thing we do is probably to look up on Yelp for a good restaurant, or even better, the one with starry reviews.

Meanwhile, ill-intentioned people can easily make up fake reviews and ratings on these systems in order to gain profits or to undermine others' success. Unfortunately, researchers have shown that humans' abilities to distinguish the fake from the real are not any better than random guess [1]. Although we can formulate such a spam detection problem as a binary classification problem in machine learning, there will be two problems. First, it's hard to get the ground truth of data. Second, the dataset is usually highly imbalanced.

For the first problem, [1] used a pseudo dataset with fake reviews written by Amazon Mechanical Turk (AMT). However, [2] showed that these pseudo fake reviews share little resemblance with the actual ones. Fortunately, Yelp has released a review dataset generated by their internal filter system. It might be more reliable, since Yelp has access to the online footprints of users in addition to public information. For the second problem, most of the previous works used a balanced dataset. Whereas [3] found that random oversampling (ROS) works well on the hotel booking datasets. In this work, we will focus on fake review detection on Yelp's NYC restaurant dataset and explore more class balancing strategies.

The most common models used in the literature are Logistic Regression (LR), Naive Bayes (NB), and Support Vector Machine (SVM) on linguistic and/or behavioral features. We developed our approaches based on previous state-of-the-arts. Our main contributions are summarized as follows:

- Designing new behavioral features and improving the AP and AUC scores.
- Combining several sampling methods to solve the imbalanced dataset problem.
- Adopting ensemble methods such as Extreme Gradient Boosting (XGBoost) that previous works lacked of.

II. Architecture

We divide this end-to-end machine learning pipeline into 6 stages: data exploration, feature engineering, preprocessing, imbalanced classes, modeling, and evaluation.

A. Data Exploration

In the beginning, we explore the data from `train.csv` and `dev.csv` together by visualization and descriptive statistics. Starting from the basic features shown in Table I, we find that the target variable, label, is highly imbalanced. The fake reviews only occupy around 10.27% of the whole dataset, which can make the model unstable. Besides, it is interesting that positive reviews with rating 4 and 5 account for almost 77%. From Figure 1, we observe that spammers are inclined to give more extreme ratings (1, 4, 5). In Figure 2, the distribution of review length for non-fake reviews is more right-skewed. Our hypothesis is that people who left real reviews were more likely to spend time on writing a longer article.

TABLE I
Dataset Schema

Column	Description	Type
ex_id	unique index of the record	Integer
user_id	index of the user	Integer
prod_id	index of the product	Integer
rating	rating from 1 to 5	Float
label	fake (1) or not fake (0)	Integer
date	date of the review	Date
review	the review content	String

B. Behavioral Feature Engineering

In [2], the authors have shown that behavioral features yields better results than using linguistic features. Generally speaking, behavioral features can be further divided into reviewer-centric and review-centric features. We list the features we used as follows, with our newly-added features marked:

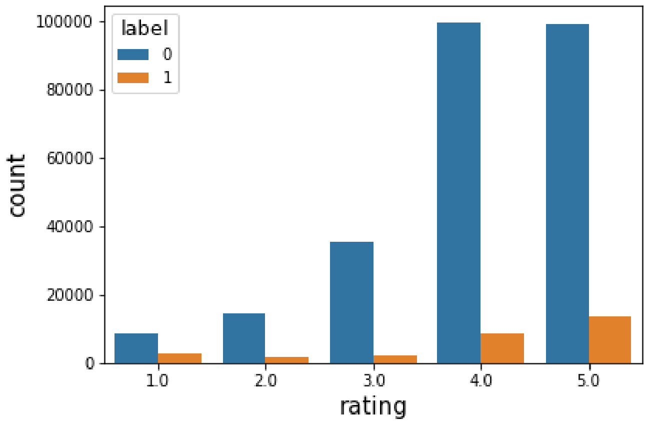

Fig. 1. Rating vs. Label

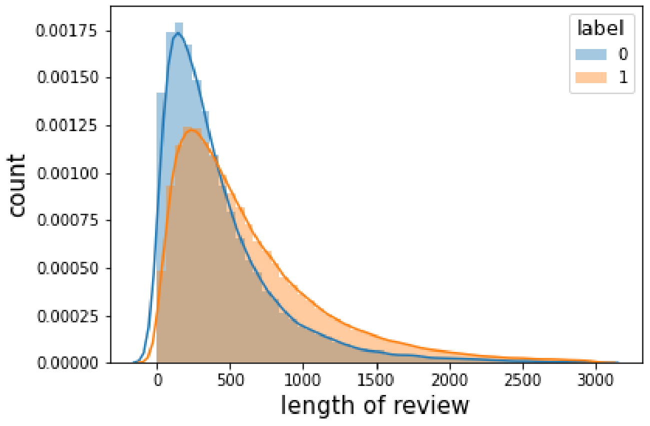

Fig. 2. Rating vs. Label

review-centric features

- Review length
- Absolute rating deviation from product's average rating [4]: $d_{ij} = |r_{ij} - \frac{1}{J}\sum_j r_{ij}|$, where r_{ij} is the rating from user i to business j.
- Earlier time review [5]: $max(0, 1 - (\frac{t_{now} - t_{first}}{T}))$. Whether the current review is written within the $T = 70$ months from the first review for this product. It's reasonable for a spammer to post a review as early as possible to increase their chances of being viewed.
- Whether a review is written on holiday or not [new]: It's more effective to write fake reviews on holidays, since people are more likely to go out on these days.
- Whether a review is written on weekend or not [new]

reviewer-centric features

- Number of reviews per user
- Max/avg reviews per user per day [2]
- Percentage of positive/negative reviews per user [2]
- Burstiness [6]: $max(0, 1 - (\frac{t_{last} - t_{first}}{T}))$. Whether all reviews from a user are written within $T = 28$ days.
- Avg rating deviation [6]: $\frac{1}{I}\sum_i d_{ij}$. How different are one's reviews from others.
- Avg/Var review length per user [new]
- Avg content similarity (n-gram) [6]: Compute the average of pairwise jaccard similarities among user's reviews. A review is represented as a 500d uni-gram vector.
- Avg content similarity (GloVe) [new]: Compute the average of pairwise cosine similarities among user's reviews. A review is represented as a 50d GloVe vector. We argue that GloVe can better embed the meaning of the text into a smooth manifold. As a result, it can capture the information where uni-gram cannot.

C. *Linguistic Feature Engineering*

Considering there may be useless words or characters in the review column, before creating new text features, we need to clean the content. At first, we found out the whole cleaning process takes a long time to finish by NLTK and spaCy in Python. Therefore, we took advantage of Spark's batch processing ability by integrating the code in Python and PySpark with Spark NLP, a Spark's library. The cleaning steps is listed in the following:

- **HTML Tags and URL**: Web-scraped data may contain HTML tags which are not useful in text data, so we use a Python library, BeautifulSoup, to remove them.
- **Accented characters**: Since we are dealing with English reviews, accented characters, e.g. café, can be confusing. In this project, we convert all accented characters into normal English characters, e.g. cafe.
- **Punctuation & Special Characters**: The Punctuation marks, such as question marks and quotes, inclusive of any special characters are also removed from the text data
- **Stopwords**: In the Spark NLP pipeline, we use the predefined stopwords collection from NLTK to extract the words which do not add much value in the article.
- **Lemmatization & Lower Case**: One word can be written in different form with different tenses. Lemmatization helps us unifying them into a single format. After that, we convert all tokens into the lower case.

It's not done yet after removing redundant characters and convert the sentences into several tokens. String-type data are not allow to be fitted into the model. We have two major strategies here. First of all, based on the 50-dimension pretrained GloVe embeddings, one token is transformed into an array with a length of 50. We take the average of all the tokens in one review as the 50 new text features. Second, we extract multiple sparse matrices by the bag-of-words (BOW) method with uni-gram and bi-gram. Because of a high variance among the length of reviews, we respectively generate the matrices with the top 200, 500, and 1000 frequent tokens.

D. *Preprocessing*

Features with distinct scales can affect the model-training efficiency and make the result unstable, so before fitting the selected columns into the models, we choose to do standardization. By doing so, we can re-scale all columns with a zero mean and one standard deviation. Furthermore, many of the features have a right-skewed distributions, e.g., number of reviews per day, review length, etc. We perform log normalization on these features, and it always leads to better results.

E. Imbalanced Data

With a imbalanced dataset, our model can cheat by predicting the most common class during training and achieve high accuracy score, yet with low AP and AUC scores. The common approaches for this problem are class re-weighting (RW) and data re-sampling. The former assign higher weights for the classes with fewer samples. It's simple to implement and efficient. So, we use it as our default setting. On the other hand, Data re-sampling try to make each class have the same number of samples. With the help of Python's imbalanced-learn package [7], we tried the following approaches:

- **Random under-sampling (RUS)**: Only a subset of samples from the majority class are used. Some important information in the out-of-sample data may be lost.
- **Random over-sampling (ROS)**: Each sample in the minority class might be sampled for multiple times. However, replicating too much information may result in over-fitting.
- **SMOTE (Synthetic Minority Oversampling Technique)** [8]: Basically, the technique generates new samples by interpolating of its nearest neighbors.
- **ROS + RUS**: This method can benefit from the advantages of both approaches.
- **SMOTE + RUS**: For the model with only the linguistic features, we use this method to create a dataset with 50,000 in both of the target classes.
- **Ensemble of RUS**: Perform RUS several times, each time with a different subset of samples from the majority class. Make the final prediction by voting.

F. Modeling

We compare several well-known machine learning algorithms supported by scikit-learn package [9], such as LR, SVM, NB. They have been proven effective for other opinion spam detection. LR is common in binary classification problems. It uses a linear function to fit the natural log odds, which is then transformed into the probability of occurrence for the positive class with a logistic function. We set our baseline model as a regularized LR since it can be quickly estimated. But, it cannot work well with a non-linearly separable problem. Although SVM is able to deal with the linearly inseparable problem by kernel tricks, it took more time than linear SVM. Furthermore, it requires training time proportional to the number of samples. So, most of the time we use it along with RUS. XGBoost [10] is an ensemble method which builds several weak classifiers, decision tree. In theory, XGBoost tends to reduce both the bias and variance. Boosting methods are also known for its ability to solve imbalanced data problem. From our experiment, we found that XGBoost not only performs better than other models, it also converges very fast. The only drawback is that it has the most hyper-parameters to tune. For all of our models, we use 5-fold cross validation and grid search to find the optimal hyper-parameters.

G. Evaluation Metrics

Instead of using accuracy which is not informative enough in imbalanced data as the metric, two other evaluation methods, average precision (AP) and Area Under the Receiver Operating Characteristic Curve (AUC) score [11], are implemented in this work. ROC curve is a probability curve for comparing sensitivity and specificity. AP score summarizes the precision-recall curve at n thresholds:

$$AP = \sum_{i=1}^{n}(R_n - R_{n-1})P_n \tag{1}$$

where R_n and P_n are the recall rate and precision rate at the nth threshold on the precision-recall curve.

III. Experiments

We train with only linguistic (Table. II) or behavioral features (Table. III and Table. IV) independently because training them together will deteriorate the results.

A. Dataset

The dataset[1] used in this work contains reviews for restaurants located in New York City. The schema is shown in Table I. There are 7 columns and 250,874 rows for `train.csv`, 35918 rows for `dev.csv`, and 72,165 rows for `test_no_label.csv`.

B. Result

From Table II, we know there is room for improvement if we only use the linguistic features in both the Logistic Regression or XGBoost. Among all methods, the BOW uni-gram model generally outperform the pre-trained 50-dimension GloVe embedding. Also, AUC score and AP increase slightly with more frequent tokens in the BOW.

TABLE II
Training Result with Linguistic Features

Models	Feature Set	AUC Score	AP
Logistic Regression	BOW 1-gram + top 200 tokens	0.687	0.189
Logistic Regression	BOW 1-gram + top 500 tokens	0.698	0.199
Logistic Regression	BOW 1-gram + top 1000 tokens	0.706	0.207
Logistic Regression	GloVe	0.600	0.162
XGBoost	BOW 1-gram + top 200 tokens	0.682	0.184
XGBoost	BOW 1-gram + top 500 tokens	0.691	0.191
XGBoost	BOW 1-gram + top 1000 tokens	0.696	0.197
XGBoost	GloVe	0.549	0.141

[1]https://worksheets.codalab.org/worksheets/0x33171fbfe67049fd9b0d61962c1d05ff

TABLE III
HANDLING IMBALANCED DATA

	RW	ROS	SMOTE	RUS	RUS+ROS	Ensemble
LR						
AP	0.268	0.268	0.268	0.269	0.268	0.268
AUC	0.797	0.797	0.796	0.797	0.797	0.797
XGB						
AP	**0.424**	**0.424**	0.333	0.365	0.415	0.370
AUC	**0.867**	**0.867**	0.829	0.851	0.864	0.854

Table. III shows that each balancing strategy have almost no effects on the performance of LR models. For XGBoost, which itself is already an ensemble method. It's not helpful to apply any other re-sampling method.

TABLE IV
FEATURE ABLATION STUDY

	Full	-Cosine	-Weekend	-Holiday	Review Len		None
					Var	Avg	
AP	0.424	0.418	0.419	0.421	**0.408**	**0.408**	0.374
AUC	0.867	0.865	0.864	0.865	**0.861**	**0.861**	0.845

We studied the importance of our new behavioral features by removing them one at the time. Table. IV summarizes the results, where the Full models use all the features, the None model uses only features from previous work. The statistics of review length per user give us the most performance gain. Whereas the holiday feature only provide slight improvement. This result suggests that we did discover some useful features. The ROC curve of our best model is shown below.

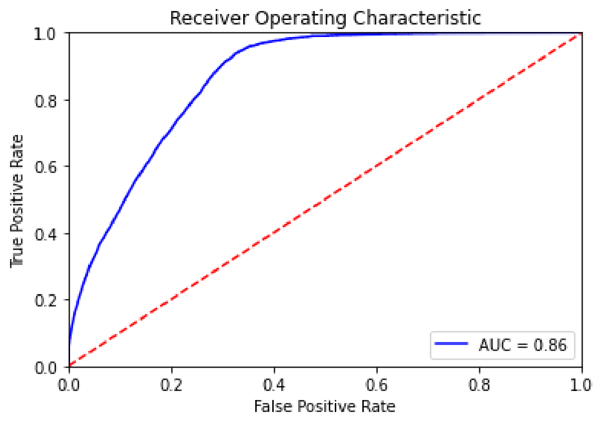

Fig. 3. ROC Curve of Our Best Model

IV. CONCLUSION

In this work, we learned that the most critical factor affecting the final scores is feature engineering. The same as previous findings, behavior features outperform linguistic feature. However, combining two types of feature together didn't improve the result. Hyper-parameter tuning plays an important role in XGBoost, whereas it has less impact on other models. Moreover, it cannot increase the higher bound of the final scores. Lastly, different data balancing strategies perform roughly the same. This might be why previous works didn't focus on this part. All in all, obtaining greater insights from data and then designing features for it seems to be the best recipe to develop a good machine learning model (except for neural network probably).

During our research, we have brainstormed about some potential directions for opinion spam detection. [12] modeled the user-product-review relationships as a graph and solved for MLE. Inspired by them, we think it's promising to use a two-stage approach: First, predict who are spammers. Then, predict which are fake reviews. It's intuitive to utilize these kind of probability-based methods because spammers do not always fake. Their faking pattern might be controlled by variables such as time, business type, etc. Our another idea is that, we can incorporate an auxiliary model which predicts the rating of reviews. A good spammers should write reviews that match their ratings. So we should be able to predict their ratings more easily compared to regular users. We did some preliminary prototyping of our ideas but didn't get any significant results.

REFERENCES

[1] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 2011, pp. 309–319.

[2] A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance, "What yelp fake review filter might be doing?" in *Seventh international AAAI conference on weblogs and social media*, 2013.

[3] A. S. Abu Hammad, "An approach for detecting spam in arabic opinion reviews," *An Approach for Detecting Spam in Arabic Opinion Reviews*, 2013.

[4] F. H. Li, M. Huang, Y. Yang, and X. Zhu, "Learning to identify review spam," in *Twenty-second international joint conference on artificial intelligence*, 2011.

[5] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh, "Spotting opinion spammers using behavioral footprints," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 632–640.

[6] G. Fei, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh, "Exploiting burstiness in reviews for review spammer detection," in *Seventh international AAAI conference on weblogs and social media*, 2013.

[7] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[11] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[12] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, 2015, pp. 985–994.

[13] M. Crawford, T. M. Khoshgoftaar, J. D. Prusa, A. N. Richter, and H. Al Najada, "Survey of review spam detection using machine learning techniques," *Journal of Big Data*, vol. 2, no. 1, p. 23, 2015.