# DS-GA 1003 Final Project : Fake Review Detection

Lizhong Wang (lw2350), Xiaocheng Li (xl3119), Weilong Chen (wc1660), Diwen Lu (dl3209)
Uploader: Diwen Lu (dl3209)

## I. INTRODUCTION

In this project, we aim to address a binary classification problem to determine whether a user's review on a restaurant is fake or genuine. The labels (fake or genuine) were generated by Yelp's filtering algorithm that identifies suspicious or fake reviews, which separates into recommended and filtered lists. Though Yelp's anti-fraud filtering is not perfect, it is shown to produce good results [1]. Therefore here we treat Yelp's recommended and filtered reviews as the ground-truth genuine and fake reviews. Differentiating the fake and genuine has a significant impact on the success of Yelp's business, as Yelp builds a platform through which user can engage and share their experience in business entities, and their ratings and reviews can be taken as references for future customers. Due to the nature of business competition, the platform risks abounding with large amount of malicious verbal attack and purposefully misleading reviews. Filtering out these ungrounded reviews is conductive to a healthy online browsing experience.

Traditional approaches in identifying fake reviews range from traditional rule-based algorithms to basic machine learning models including Naive Bayes and Logistic Regression. Due to the increasing capacity in computation and data storage, neural network methods are becoming more and more popular in Natural Language Processing (NLP) classification problems. We take in account all these algorithms into the task, as they all fit in a standard NLP classification task format. We find out that tradidional NLP methods perform at least as good as fancy neural methods, and adding information such as user information and rating information does help leverage model performance.

## II. APPROACH

For all the approaches we adopted below, and as an important preprocessing step in natural language processing, we performed tokenization, stemming, and stop-word removing on all the reviews. Tokenization helps to featurize the sequential text data into operable input vectors for machine learning models. Stemming preserves the primitive form of the word and greatly reduced the number of unique words in the dataset, and thus dimension of our vocabulary. Removing stop-words drops the words that are common in the text data but don't add much value in terms of positive/negative prediction, which, if not removed, would be weighted heavily in the count-based approach. In the end we were left with 114,051 unique tokens from training dataset.

### A. Naive Bayes

The simplest model we came up with is NB, which assumes that reviews are i.i.d, so that one review doesn't affect the genuinity of another, and given the genuinity, the appearance of each token is independent. These two assumption greatly reduces number of parameters needed for NB model and though strong they often work well in practice. NB also has the advantage that by setting a prior distribution inferred from full dataset, we can leave class imbalance as it is. For an efficient computation, we only preserve the tokens with $\geq 20$ occurrences in the training dataset. In both Bernoulli and Multinomial NB we use Laplace smoothing to address data sparsity.

1) **Bernoulli (Baseline)**: We take Bernoulli NB to be our baseline model as it simply assigns 1 to the token present in the document and 0 otherwise.

2) **Multinomial**: The Multinomial NB differs from Bernoulli NB in that it uses tokens count or term frequency–inverse document frequency (tfidf) to be the feature value. For counts feature value, the word that appears more often in a document has a larger influence on the classification. For tfidf, it increases proportionally to the word's occurrence times in the corpus but is offsetted by the number of documents in which it appears, and thus, compared to count-based values, it decreases weights of the words that are more common to many documents. Both methods adopt the concept of importance measured by certain count-based values and can better capture the characteristics of a review, as human intuition works in a sense that better matches to Multinomial NB rather than Bernoulli NB.

### B. Logistics Regression

Another common parametric approach in the field of binary classification is logistic regression. Here we adopted LR in various settings.

1) **Basic**: The best feature encoding experimented from NB is tfidf [2], so our LR settings are all based on tfidf features. In basic LR we trained the model on the full imbalanced training data.

2) **Re-sampling**: Basic LR suffers from the problem of imbalanced class shown in Table II, where the proportion of fake review is around 11.4%, as the model does not have enough training fake review data points to effectively identify fake reviews on new data. Therefore, to solve this problem, we experiment with re-sampling strategies like under-sampling and oversampling which will be discussed in detail in section III.

### C. Pretrained Word Embeddings

Pretrained word embeddings are provided by several platforms, such as GloVe [3] and fasttext [4]. Pretrained word embeddings are design on large-scale training of neural networks, giving words dense representations which under contexts. In experimentation, we use 6B GloVe [3] embeddings with 300 dimensions representations for our MLP classifier and LSTM classifier.

### D. Neural Network Approaches

Neural Network methods are used frequently when non-linear features and different between different tokens. In designing experiments we decide to try two types of architectures in neural approaches: A simple Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) [5]. All neural network models use sacremoses tokenizer. Each model is trained and evaluated on prince cluster requesting one Nvidia Tesla P40 GPU.

1) **MLP**: MLPs are designed to do non-linear classification in our experiments. MLPs uses activation functions which transform outputs of affine functions. By stacking up a number of affine transformations and activation functions between the model inputs and outputs you get an MLP classifier. In optimizing the model parameters we use gradient descent with several versions, i.e. Stochastic Gradient Descent (SGD), Adam, Momentum, etc., with back propagation. Hyperparameters in the model include number of layers, dimension of each layer outputs, token representations methods, and activation functions. In our experiments we only try a two-layer MLP classifier with pretrained word embeddings in GloVe [3], computing the sentence representation by taking the mean of all word embeddings, and dimension in each layer to be 128 with all Rectified Linear Unit(ReLU) activations. In optimization, we use Adam starting with 0.1 as optimizer.

2) **LSTM**: The LSTM is an architecture which belongs to the Recurrent Neural Networks(RNN) class that treats a sentence as a sequence. In computing sentence representations, LSTM outputs representations at each token position by incorporating the information before or after the token. Specifically, by concatenating the representations of the forward and backward version of LSTM, we get outputs of a Bi-directional LSTM (Bi-LSTM). In experimentation we try an LSTM classifier, in which we pass inputs through GloVe embeddings, a two-layer Bi-LSTM with 128 dimensions each direncion in each layer, a max pooling with stride 1 (to get sentence representations), and an affine transformation followed by a sigmoid function to predict scores of positive class [6]. In optimization, we use the same optimizer as we use in MLP classifier.

## III. Experiments

### A. Dataset Description

Training, validation, and testing data have been organized into 3 files, and they respectively contain 250,874, 35,918, and 72,165 reviews. A snapshot of data schema is shown in Table I. In label column, value '1' represents a fake review, and '0' represents a genuine review. From Table II, we can also see that our data suffers from imbalanced classes. In both training and validation data, the ratio of the fake to the genuine is around 1 to 10 as in Table II. To solve this problem, we tried several re-sampling strategies.

| ex_id | user_id | prod_id | rating | label | date | review |
|---|---|---|---|---|---|---|
| 0 | 923 | 0 | 3.0 | 1 | 2014-12-08 | The f... |
| 0 | 924 | 0 | 3.0 | 1 | 2013-05-16 | This li... |
| 0 | 925 | 0 | 4.0 | 1 | 2013-07-01 | order... |
| ... | ... | ... | ... | ... | ... | ... |

**TABLE I:** Data Schema

| data \ label | 1 | 0 |
|---|---|---|
| training | 25,819 | 225,055 |
| validation | 3,648 | 32,270 |

**TABLE II:** Data Imbalances

- *Oversampling V.S. Downsampling*
  We experiment with these two re-sampling strategies. By oversampling, we add more data to the fake review class(label=1). By downsampling, we sample a subset of genuine review (label=0). In both cases, the final ratio of fake reviews data to genuine reviews data is 1 to 1.
- *Random downsampling V.S. NearMiss downsampling*
  We experiment with two downsampling methods, random downsampling and NearMiss downsampling. In specific, NearMiss [7] downsampling method selects samples based on the distance between majority class and minority class, while random downsampling randomly select samples that to keep in genuine reviews data set.
- *Random Oversampling V.S. SMOTE Oversampling V.S. ADASYN Oversampling*
  We experimented with three oversampling methods, random oversampling, SMOTE [8] oversampling and ADASYN [9] oversampling. In specific, SMOTE oversampling generates samples by randomly selecting one or more of the k-nearest neighbors for each example in the minority class. ADASYN improve based on SMOTE that it focuses on the samples which are difficult to classify with a nearest-neighbors rule.

### B. Methodology

Due to class imbalance nature of training data, we use area under receiver operating characteristic (AUROC) and average precision (AP) as our evaluation metrics because they are immune to the effect of imbalanced class problem.

1) **AUROC**: AUROC is the area under the ROC curve. ROC is plotted by calculating true positive rate and false positive rate under various classification thresholds. Often in a binary classification scenario, we want to reach high true positive rate and low false positive rate given a decision threshold, thus an ROC closer to top left corner of the plot is better, corresponding an AUROC closer to 1.

2) **AP**: Average precision is a summary of precision-recall curve where precision reached at each classification threshold is weighted by the increment of recall from the previous threshold. The equation is given by: $AP = \sum_n (R_n -$

$R_{n-1})P_n$, where $R_n$ and $P_n$ are the recall and precision achieved at $n^{\text{th}}$ threshold.

3) **Feature Engineering**: In our first attempts of the above approaches, we only took review text data to featurize as our model inputs. But then we realized rating could also be particular helpful. The idea is that if we assume all users leave reviews to business entities that he had experienced, then we can construct a rating matrix just like in collaborative filtering recommendation system, and the latent factors learned for users and business entities can reflect the preference of a user and the characteristics of a business. The rating from a fake user who left a fake review for a business he didn't interact with must deviate from the true rating as if he did much more than that difference of ratings for genuine users.

Furthermore, in the development of feature engineering, we additionally noticed that user_id can also be a great feature in detecting fake reviews as experience from life informs us there are bot user accounts on online platforms and they specialize in making fake reviews. Thus we can count the number of fake reviews that each fake user ever posted in training data and feed that along with all other features previously come up with into the model.

We noticed the magnitude of rating features and user features is much larger than the magnitude of TFIDF review features. This inconsistency will lead to more regularization penalty for the features with smaller magnitude. Therefore, we re-scaled the features MaxAbsScaler. The advantage of this scaling method is that it will not destroy the sparsity of the original input features.

### C. Results

Our baseline model is Bernoulli NB. Results of baseline and other more complex models are listed below.

| model\metrics | AUROC | AP |
|---|---|---|
| Bernoulli NB | .679 | .192 |
| Multinomial NB (counts) | .688 | .176 |
| Multinomail NB (TFIDF) | .691 | .193 |
| **LR (basic)** | **.725** | **.215** |
| LR (random downsample) | .720 | .209 |
| LR (NearMiss downsample) | .598 | .145 |
| LR (random oversample) | .719 | .209 |
| LR (SMOTE oversample) | .688 | .185 |
| LR (ADASYN oversample) | .688 | .183 |
| MLP (basic) | .716 | .216 |
| MLP (random oversample) | .716 | .218 |
| LSTM (basic) | .664 | .179 |
| LSTM (random oversample) | .700 | .196 |

**TABLE III:** model validation metrics

| rescale | user_id | rating_features | tfidf | AUROC | AP |
|---|---|---|---|---|---|
| | | | x | .725 | .215 |
| | | x | x | .859 | .317 |
| | x | x | x | .896 | .545 |
| **x** | **x** | **x** | **x** | **.900** | **.562** |

**TABLE IV:** LR basic + feature engineering

### D. Results Explanation

We can see among 3 naive bayes classifers, the one with tfidf as feature value reaches the best AUROC and AP. We compare different models, sampling methods, and usage of different information.

1) **Model Complexity**: As we compare our baseline model with fancier neural network methods, we find that leveraging model complexity doesn't really give rise to performances either in AUROC or AP. Furthermore, we find that when we use more complex models, like neural network models, evaluation for AUROCs and APs give even lower than linear methods like logistic regression. Therefore, when performing extensions, we only apply a random over-sampling to the two neural network architectures.

Intuitively, most NLP publications tell that when model complexity increases, or a specific architecture is designed for the task, the model tends to give more accurate predictions. This intuition doesn't apply to our task, in which sizes of classes are extremely imbalanced, and by getting an idea of sentence meaning doesn't help much give predictions whether sentences are fake. Even human beings are sometimes fooled by fake reviews.

2) **Sampling Methods**: We try four sampling methods in experimentation, i.e. Random, NearMiss, SMOTE, and ADASYN.

By different sampling methods, we find that fancier sampling methods don't necessary improve model performance compared to random oversampling and downsampling. One explanation is that as we use word counts or tf-idf values to form our sentence vector representations, we have already assume that token-level features are enough for comparing similarity. However, in natural languages the actual sentence representations are far more complex than token-level similarities. For example, paraphrasing generates the same meaning with different uses of tokens.

3) **Additional Information**: We find that when adding additional information such as user information and rating information, model performances are better. We first try with models that deals with merely text data as is described in feature engineering section. Across different models we train, results in AUROCs and APs are similar.

| Ratings | Label | Count |
|---|---|---|
| 1.0 | 0 | 7496 |
| | 1 | 2565 |
| 2.0 | 0 | 12894 |
| | 1 | 1643 |
| 3.0 | 0 | 31047 |
| | 1 | 2214 |
| 4.0 | 0 | 87066 |
| | 1 | 7481 |
| 5.0 | 0 | 86552 |
| | 1 | 11916 |

**TABLE V:** Label Counts on Ratings-Train Data Set

Then, we trained a latent factor model using SVD matrix factorization with all the genuine reviews in the train dataset. For a given user and a given product, this latent factor model will generate a predicted rating. However, the model will not be able to calculate a predicted rating if the user has no rating history before. We use an additional feature pred_rating_is_missing to capture missing values,

and we populate the missing pred_rating with the average rating of that product.

Moreover, we counted the number of fake review a user has produced in the training dataset, and use this count as a new feature.

Finally, we rescaled all the user_features, rating_features, and the text_features.

### E. Error Analysis

As we look at validation accuracies, AUROC scores, AP scores, and especially the confusion matrix, we come to conclusion that models trained on original data are inclined to predict the majority class.

One possible reason behind this phenomenon is the imbalance of dataset, having roughly 90% fake reviews and only 10% real reviews. After training the model on resampled dataset, alghouth the model starts to generate positive predictions, the AUC scores and the AP scores are similar or even worse. This means that the ranking result is not improved by resampling, which implies that the reason for models in predicting the majority class is not class imbalance.

According to our results with additional information, we find that user ratings are features which contributes to model performances. Moreover, the count of a user's historical fake reviews also contributes to the prediction of fake reviews, as we might discover accounts which are commonly used to generate these fake reviews.

## IV. Discussion

### A. Findings Evaluation

The user review history, including counts of past fake reviews and rating history, is helpful when we already have user information in our system, as in this case we can find accounts that generate fake reviews, and will have strong evidence to predict other reviews generated by these accounts. However, if a new user with zero review history comes in, the model has to predict the review based on only its text features, which has weaker performance as indicated at TABLE IV.

### B. Future Work

For next step of this project, we want to do a full sanity check on all combinations of models with both oversampling and downsampling methods, validating whether model performances are un-correlated with model complexity. We will also try more methods in injecting user and rating information into more complexed models like BERT. Moreover, we want to train our models for more times in order to obtain a statistical significant confident conclusion for our model performances.

## V. Contribution

Lizhong Wang: Build Logistic Regression with ALS model, add rating feature.
Xiaocheng Li: Build MLP and LSTM Classifier with and without sampling methods, set up HPC pipeline for model training and evaluation.
Weilong Chen: Build Logistic Regression and Naive Bayes with oversampling method.
Diwen Lu: Naive Bayes and Logistic Regression, add user_id feature.
All: Paper Writing.

## References


[1] K. Weise, "A lie detector test for online reviewers," *Bloomberg Business Week*, 2011.

[2] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 670–680, Association for Computational Linguistics, September 2017.

[7] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of workshop on learning from imbalanced datasets*, vol. 126, 2003.

[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] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328, IEEE, 2008.