Keepin' It Real:
How to Identify Fake and Genuine Reviews

DS-GA 1003 Machine Learning

The Realest:
Anu-Ujin Gerelt-Od ago265
Andrei Kapustin ak7671
Sumedha Rai sr5387
Stephen Roy sr5388

Abstract

Reviews have quickly become an important decision making tool for consumers considering both online and offline spending. Thus, for a new industry around buying and selling Fake Reviews has risen to meet the demand of unscrupulous businesses. The goal of this project is to generate a model that successfully classifies Fake and Genuine reviews.

Keywords: imbalanced data; NLP, BERT, word2vec, count vectorizer, TF-IDF; ML, neural network, SVM, naive bayes, logistic regression

Center for Data Science
New York University
May 19, 2020

1 Introduction

With the rising popularity of service-review platforms like Yelp comes the pressure to achieve and maintain a good reputation. Restaurant owners, in an effort to game the system, use additional services or added incentives for customers to leave a good review and boost their ratings on the website. However, this leads to deception and creates doubt in the integrity of the platform. Thus, it is important to tackle this problem and create an algorithm that is capable of detecting such solicited reviews. In this project, we attempt to create our own version of Yelp's classification algorithm for distinguishing 'genuine' and 'fake' reviews. Although Yelp is ambiguous on its detection methods, it is believed that their algorithm uses "multiple factors such as looking at multiple reviews that come from the same computer or if the review shows apparent bias in the text" [1].

A detailed overview of the dataset can be found in the Data section, in addition to pre-processing methods used to prepare it for analysis. In the Problem Definition and Algorithm section, we describe the various feature engineering techniques used to transform the text portion of the reviews and the different machine learning algorithms used for training. The Experimental Evaluation section reports the methods used to evaluate the models as well as the results from the optimal models after hyperparameter tuning.

2 Data Processing

2.1 Exploratory Data Analysis

The dataset used for this project was provided by Yelp with the review data consisting of 286,889 labeled records broken into training and validation datasets, six features per record. Review features included record ID, user ID, product ID, rating, date, and review text. While we expect some of these features to have no impact on the model (i.e., the record ID), most of the features are likely to contain relevant information. During exploratory data analysis, we identified several trends in distribution which can be seen in Figure 1. Unsurprisingly, users with numerically lower user IDs had more record entries. We saw an equally long tail for ratings in the opposite direction with the vast majority of ratings occurring at the 4+ end of the 1-5 spectrum. By contrast, product IDs exhibited no obvious pattern. When we separated the data by class, the two had similar distributions in rating as shown in Figure 2, which contradicted our initial guess about the bias towards upper and lower ends of the spectrum for fake reviews.

Our training dataset was an imbalanced dataset with a 1:10 ratio between the minority and majority classes. Class imbalance is generally a greater problem when less data is available. This is because the model has insufficient instances of the minority class to learn from and hence, it is not able to construct good representations for minority entries. To tackle the problem of a significant class imbalance, we used the techniques of oversampling and undersampling. Oversampling generates new (similar or identical) examples for the minority class from the existing data. Undersampling works by removing instances from the majority class until it is able to strike a balance between the number of instances for the minority class and the majority class. Oversampling would thus increase the size of the dataset and undersampling will reduce it depending upon the ratio between the majority and minority class.

2.2 Data Cleaning

As some of the feature transformation models, particularly word2vec, are more sensitive to non-English vocabulary than others like BERT, during the data cleaning process we removed those unrecognized words. Most of the removed words fall into one of the four categories:

1. Numbers (10, 2014)
2. Incorrectly spelled words (waitess, noddles)
3. Non-English words (heißen, 山口组)
4. Not actual words (Supercalifragilisticespialidocious, DOOOOOOOOPPPPPEEE)

To capture pertinent information contained within the features, we created additional categorical variables by transforming the existing features. The review text was converted to numeric values for modeling and

three engineered features were developed: 'rating indicator' to emphasize 1 and 5 star reviews, 'previous fake review' to indicate users who have previously written fake reviews, and 'reviews today' to indicate how many reviews a user has already written that day. These features help address areas of concern that we consider potential indicators of fake reviews.

3 Problem Definition and Algorithm

3.1 Task

The goal of this project is to output a predicted class for each review. Class, in this case, is defined as $c = 0$ if the review is genuine, and 1 if it is fake. The input is a set of Yelp reviews with metadata and the columns that we added. We hypothesize that our best model will outperform a majority class predictor. Given the high degree of class imbalance, this is a non-trivial baseline for our model.

For the purpose of finding the most suitable model for this problem, each member ran a different machine learning algorithm with a different review text feature engineering technique: Logistic Regression with Count Vectorizer, Naïve Bayes with TF-IDF, SVM with Word2vec, and Neural Network with BERT. While each member started with the same search space, the transformers and hyperparameter tuning approaches varied. After tuning our models separately, we measured performance for each transformer/model pairing in a bake-off selecting the best performing pairing for each model to use on the test dataset. The next sub-section describes each of the models used with the initial transformation method applied to the data before running the model.

3.2 Algorithm

3.2.1 Logistic Regression

Logistic Regression is the go-to method for binary classification in the field of machine learning. It is a strong algorithm for a baseline model. It partitions the classes by drawing a linear decision boundary in the feature space. The loss function used to train logistics regression is the Cross Entropy Loss which tries to find the parameters that maximize the log probability of the correct class for all instances in the training set. The cross entropy loss is given by: $L(w) = \sum_i -y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$, where $p_i = \frac{1}{1+e^{-w^T x_i}}$ is the probability that our model assigns to instance x_i as belonging to class 1. Here, we can also include a regularization term to our loss function to make the model learn better, thereby improving the generalizability of the model. The hyperparameters are the regularization strength and the height of regularization.

Count Vectorizer (CV) is the most basic feature extraction strategy to extract features from text. It converts a collection of text documents to a sparse matrix of token counts. Given a vocabulary V, the vectorizer constructs $|V|$ number of features, and the value of its feature for Document D_j is simply the count of token V_i in document D_j. The design choices while constructing a count vectorizer are size of vocabulary, if n-grams should be used, and tokenization decisions such as standardizing capitalization/tense or using stop words.

3.2.2 Naïve Bayes

The Naïve Bayes classifier is a simple classification algorithm based on the Bayes rule of probability for a document d and class c: $P(c|d) = P(d|c)P(c)P(d)$. Applying this rule to the feature space with $x_1, ..., x_n$ features, the complete formula becomes: $p(c_k|x_1, x_2, ..., x_n) \propto p(c_k, x_1, x_2, ..., x_n) = p(c_k)p(x_i|c_k)$. The classifier reports the probability of each document belonging to a class, fake and genuine in our case, and returns a prediction. In conjunction with this model, we used the Term Frequency - Inverse Document Frequency feature engineering method, also known as TF-IDF. This vectorizer first counts the number of times a word or term appears in a document, then counts its occurrence in the whole set and returns its inverse as a fraction depicting the informativeness of each term. The higher the value, the more important and useful that term is for classification.

The main assumption the model makes is that each feature is conditionally independent given its class. However, this presents problems when new terms enter the dataset and result in zero total probability, also known as Zero Frequency. To deal with this, we use the Laplace smoothing technique to add a constant parameter, alpha, to the prior and conditional probabilities, making sure that we avoid cases where $p(x_i|c_k) = 0$. After evaluating the baseline model, we ran a cross-validation grid search with $\alpha \in (0.01, 1)$, but the model with the best hyperparameters yielded the same results as the baseline model. Additionally, we utilized the Random Over and Under Sampler methods to balance the dataset to a 1:1 ratio. However, the Naïve Bayes model was not sensitive to these changes and the resulting accuracy scores were slightly lower than the baseline model, thus we kept the original dataset.

3.2.3 Support Vector Machine

Support Vector Machine (SVM) is a model that tries to separate the data into two classes with a hyperplane. To do so, it minimizes the regularized hinge loss: $L(w) = \frac{1}{2}\|w\|_2 + C\sum_{i=1}^{n} \max(0, 1 - y_i w^T x_i)$. Using kernels allows SVM to use nonlinear features to separate the data. The main hyperparameters of SVM are the regularization parameter C and kernel type. To achieve the best performance the RBF and polynomial kernels were tested, the regularization parameter was searched for in range [0.001, 10].

Word2vec is a model that assigns each word a vector in such a way that if two words have similar meaning, their vectors are close to each other. To vectorize the reviews, the word2vec model pretrained on Google News Dataset was used. The model assigns a 300 dimensional vector to each word of the text. To get the single vector for the review, the vectors assigned to the words of the review were averaged. As mentioned in the Data Processing section, the model can work only with words that were present during training. So all the words that had no vectors assigned were removed from the reviews. After that some reviews became empty strings and were not used for training the classifier.

3.2.4 Neural Network

Neural Networks have a rich, albeit short, history with respect to supervised classification problems. While various neural network implementations exist, the scikit-learn Multi-layer Perceptron classifier was chosen to minimize dependencies, simplify integration, and promote like comparisons with the other models being evaluated. Similarly to the SVM model, neural networks generally perform better when the input data is normalized. A scikit-learn MinMaxScaler((-1,1)) [4] was chosen over the StandardScaler for regularization since none of the dataset features resembled a normal distribution [5]. The hyperparameters explored through cross-validation include hidden layer size, activation method, solvers for weight optimization, L_2 penalties, learning rates, and solver iterations.

BERT is the latest state-of-the-art language model developed by Google in 2018. It has achieved groundbreaking results on various natural language processing tasks by simultaneously conditioning on the elements to both the left and right sides of the target [6]. Unlike previously unidirectional language models, BERT is able to obtain more comprehensive information and richer context while learning, thus generally outperforms similar LSTM models. While no retraining was done to focus BERT on the review corpus; BERT serves as a useful tool to transform our unconstrained text review feature into 768 component numerical features for the neural network to learn from. It should be noted that BERT was the slowest transformer to convert text to numeric features by several orders of magnitude; for application with streaming data or real-time analysis, significant compute resources are likely required to obtain timely results.

The major concerns regarding this model are scaling, reproducibility, and interpretability. Since the model relies on scaled inputs (based on just the training data), the MinMaxScaler must be stored along with the model to scale any new input data. Additionally, due to the complexities inherent in neural networks such as drop-out on hidden layers, it is very difficult to reproduce and interpret the model.

4 Experimental Evaluation

4.1 Methodology

The data was given to us as separate training, validation and test sets, no additional splitting was done. To address the problem of an imbalanced dataset, we used the RandomUnderSampler and the RandomOverSampler from the imbalanced-learn toolbox on our training dataset. To evaluate the models we used two metrics: area under ROC curve (AUC) and average precision (AP). These metrics were chosen because they will be used for the final evaluation of the submitted model. AUC is equal to the probability that a model will rank a random positive example higher than a random negative. It is computed as an area under the plot of true positive rate (ratio of true positives to all positives) versus false positive rate (ratio of false positives to all negatives) at different classification thresholds. AP is equal to $\sum_n (R_n - R_{n-1})P_n$, where R_n and P_n are recall and precision at the n-th threshold.

4.2 Results

For evaluation, we first compared the results from the models with the feature transformation pairings as described in the Problem Definition and Algorithm section. As part of the bake-off, we ran the four models with different transformation vectorizers and compared the results to the initial pairings and selected the model-transformer pairings that yielded the highest scores, which are plotted on Figure 3. The first plot shows the ROC curves with the AUC and AP scores. The right hand side of the figure illustrates the confusion matrices generated using the validation set.

4.3 Discussion

As our best model, we selected Logistic Regression transformed with TF-IDF as it had the highest AUC and AP scores. For TF-IDF, we built a vocabulary using the top 1,000 features ordered by term frequency. Next, using a cross-validation grid search with $C \in (10^{-1}, 10^3)$, and L_1 and L_2 as penalty functions for logistic regression, we got our best model configuration as ($C = 0.1$, penalty $= L_2$). Using this configuration, we achieved an AUC of 0.812 and an AP of 0.487 on the validation set. We also explored random sampling by training our data on two different datasets - an oversampled version (with replacement) and an undersampled version. With our class imbalance of 1:10, transforming the data using random sampling did not lead to an improvement in performance on our best model.

The TF-IDF vectorizer performed better in our experiments with Logistic Regression than CV because the latter simply tells the model if a word is present or not whereas TF-IDF also quantifies the importance of a token by counting not only the token frequency in one document but also the presence of tokens in all the documents. This leads to a better selection of tokens as features for the model because a lot of redundant words do not make it to the vocabulary. The LR model was further constrained by only selecting the top 1,000 results from the TF-IDF transformer for training. Finally, we found that our engineered feature 'previous fake review' was three times more important (_coef = 4.41) than the next best feature (_coef = -1.45). This type of feature helps convey temporal trends for models lacking some form of inherent memory.

5 Conclusion

We found that the context of a review was an important factor for classifying reviews. In our best model 48 of the top 50 features came from numerical transformations of the review. However, past performance from users was by far the largest indicator of a fake review with the number of reviews submitted so far on a given day ranking 39th. It seems that more advanced NLP techniques such as word2vec and BERT struggle in this regard - possibly because most fake reviews are still being written by people and don't have clear language indicators separating them from legitimate reviews.

References

[1] R. Penafloridam, "Everything You Need to Know About Fake Yelp Reviews" *Review Trackers*
https://www.reviewtrackers.com/blog/fake-yelp-reviews/

[2] K. Weise, "A Lie Detector Test for Online Reviewers" *Bloomberg*
https://www.bloomberg.com/news/articles/2011-09-29/a-lie-detector-test-for-online-reviewers

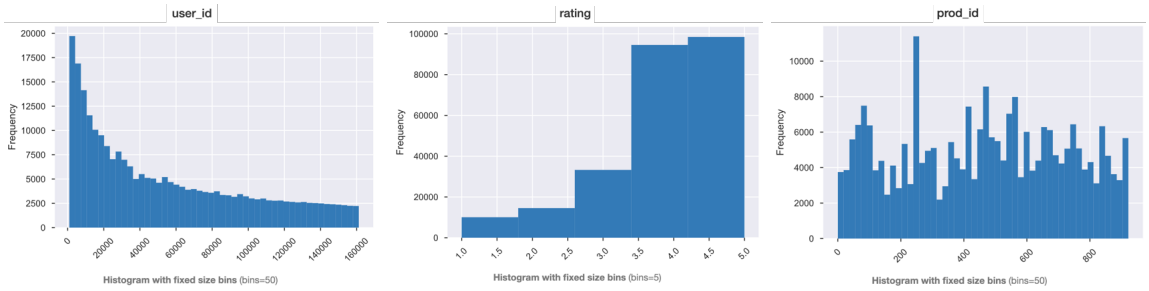
[3] S. Yang, "An Introduction to Naïve Bayes Classifier" *Towards Data Science*
https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf

[4] W. Sarle, "Should I Normalize/Standardize/Rescale the Data" *Usenet newsgroup*
http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-16.html

[5] Scikit Learn Preprocessing Package
https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler

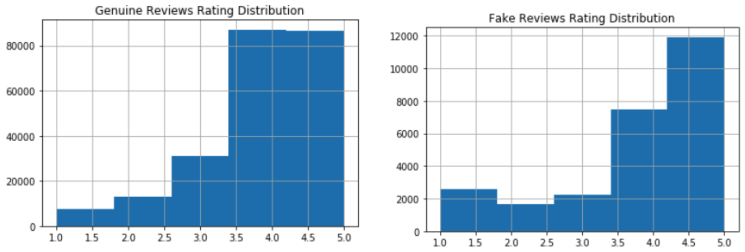
[6] J. Devlin, M. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" *Google AI Language*
https://arxiv.org/pdf/1810.04805.pdf

Appendix



(a) (b) (c)

Figure 1: Distribution of (a) User IDs, (b) Ratings, and (c) Product IDs



(a) (b)

Figure 2: Distribution of ratings separated by class

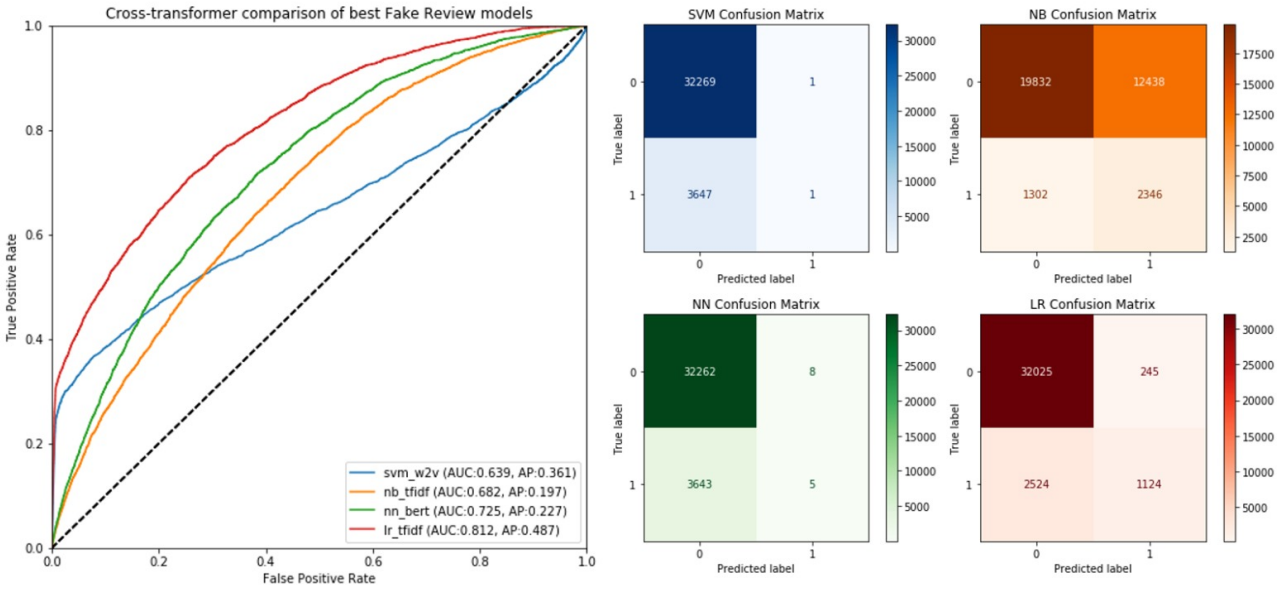


Figure 3: Results from the best model-transformer pairings