Identifying Fake Restaurant Reviews

https://github.com/arendakessian/spring2020-ml-project

Aren Dakessian (agd387), Kelsey Markey (kcm312), Guido Petri (gp1655)

Abstract

We identify fake reviews in the Yelp NYC dataset with the help of machine learning text vectorization techniques and hand-crafted features, achieving a new best mean average precision score of 0.2579 and a new best area under the receiver operating curve of 0.7705 on the test set. We examine what influence the hand-crafted features have and what extends the state-of-the-art in our project.

Introduction

With the advent of customer reviews on the internet, shops can communicate trustworthiness to their future customers by showing how satisfied previous customers are. However, due to the anonymity of the Internet, it's also very easy to create fake reviews to inflate one's own customer rating, or deflate an opponent's. Sites that host customer reviews, such as Yelp, have an interest in keeping their reviews real: if people don't trust the reviews, they won't visit the website.

We used the Yelp NYC dataset, which is a severely imbalanced dataset (approximately 10% positive class) containing reviews that have been automatically categorized as real or fake by Yelp with a proprietary algorithm [1]. These categories are taken as the ground truth. After vectorizing the textual data, we trained machine learning algorithms on this vectorized data and measured performance through mean average precision (mAP) and area under the receiver operating curve (ROC-AUC).

Problem Definition

This is a binary classification task that takes as input a textual review and associated features and outputs a probability score for how likely a given review is to be fake. In our experiments we have chosen to use only some of the features associated with a review. Specifically, we did not use the `product_id` feature due to its ambiguous meaning and dataset size issues. Additionally, we did not use the `timestamp` feature directly, since we do not believe the date and time a review were written are correlated with its likelihood of being fake. Since fake reviews are generally either very positive, where fake reviewers attempt to boost their own restaurants' ratings, or very negative, to lower their competitors' ratings, we incorporated the `rating` feature into our models [Figure 1]. Instead of relying on `user_id` as a feature identifying each user, we engineered a new feature `reviews_to_date`, which counts the number of reviews written by a given user.

For this problem we focused on six different algorithms: Linear SVM, Perceptron, Naïve Bayes (NB), Random Forests (RF), Logistic Regression (LR) and Gradient Boosting Machines (GB). We

attempt both linear and non-linear models to capture both the possibility that the data is linearly separable as well as not.

The *Linear SVM* model is a classification algorithm that finds a separating hyperplane which minimizes the hinge loss. It has a hyperparameter `C` which is inversely proportional to the strength of regularization. The *Perceptron* algorithm is a binary classifier that acts iteratively to correct training classification mistakes by updating the weight vector based on the misclassified data points. *Naïve Bayes* is a Bayesian classifier which makes the strong assumption that features are conditionally independent given the label. We used three types of NB methods: Bernoulli, multinomial, and complement. Empirical comparisons suggest that the multinomial model tends to outperform the Bernoulli model if the vocabulary size is large [2], which suggests it would perform better in our problem. Complement NB is an adaptation of the standard multinomial NB algorithm that is particularly suited for imbalanced datasets, which also suggests it would improve performance on our problem [3]. *Random Forest* is a parallel ensemble method that combines variance reduction through bagging with Decision Trees, using only a subset of the total feature space. *Logistic Regression* is a generalized linear model that uses a logistic response function to model a Bernoulli dependent variable using probabilities. It has a regularization parameter similar to that of Linear SVM. *Gradient Boosting Machines* are a sequential ensemble method that iteratively builds upon weak learners (normally decision trees) by adding a scaled version of "pseudo-residual" predictions to the original prediction of the weak learner for a fixed number of iterations. These pseudo residuals represent a small step in the negative gradient direction, making it a generalizable version of gradient descent.

Experimental Evaluation

Since the dataset contains the textual feature `review`, we had to pre-process the text in order to include its features in our models. We hypothesized that fake reviews were more likely to have a higher number of exclamation points and all capital words, so we engineered two features from the raw review: a count of all capital words and a count of exclamation points in a given review. We removed stop words and stemmed remaining words using the Porter algorithm [4, 5]. We then vectorized the remaining text using an `ngram_range` of (1, 3), ignoring stop words, punctuation, and letter cases. We experimented using a count vectorizer, a TF-IDF vectorizer, and a binary vectorizer.

Additionally, we suspected that there could be a relation between `user_id` and the negative class, since a frequent user is more likely to write real reviews [Figure 2]. For this purpose, we created a feature `reviews_to_date` that contains the number of reviews written by the user up until a review's date. In order to simulate a real deployment situation, this feature was created in a rolling manner so that reviews from the same user in the training set are numbered incrementally from earliest review to most recent. Due to overlap between users in the training and validation sets, these had to be combined to properly implement this feature, then re-split. The operating assumption here is that the probability that a given review is fake is independent from the probability that previous reviews by the same user are fake. Through this assumption, there is no information leak from the validation set into the training set. We think that this treatment best mirrors the deployment environment where this feature would be assigned in real-time for an incoming review using the number of reviews already existing in the database.

To correct the dataset class imbalance, the negative class in the training set was downsampled since downsampling has been shown to perform reasonably well while saving computation on

large datasets [6]. We also experimented with upsampling in order to utilize more of the training data. These sampling techniques were only applied to the training set in order to preserve the validation set as close to a deployment situation as possible.

We also experimented with feature reduction techniques. Specifically, we used truncated SVD [7], a process similar to PCA, to reduce the number of features, as well as manually filtering ngrams that did not appear in more than a specific number of reviews. This led to a dramatic decrease in feature space. Finally, for the appropriate algorithms, maximum absolute scaling was used to lessen the effect of imbalanced feature distributions.

All experiments were done by training on our subsampled training set with the relevant feature transformations and predicting class probabilities on the validation set given to us. The metrics used were MAP and ROC-AUC, both with a range of 0-1 and where higher is better. In order to ensure our models were in fact improving upon a baseline model, we crafted a heuristic model based solely on the `reviews_to_date` column, since we found that the first review a user wrote was most likely to be fake [Figure 3]. This simple model considered a review fake if it was a user's first review and real otherwise, achieving a mAP of 0.1398 and a ROC-AUC of 0.6491.

Our best model was also evaluated on the test set that was given to us and underwent the appropriate feature transformations. This simulates first exposure to new reviews and is the best means for evaluating a model in production. Unlike existing approaches such as SpEagle [1], our model only outputs the probability that a product review is fake, and does not use user nor product metadata. Our approach also necessitates labelled data, since it views the problem as a supervised problem, unlike SpEagle which is largely an unsupervised method.

Results

We evaluated our models using ROC-AUC and mAP on the validation set, using class probability estimates to calculate these metrics. This gave us a consistent measure to compare all of our models against each other. The results are in Figure 4 and Table 1 of the appendix. It is clear from these results that the best estimators were the linear SVMs. Additionally, the TF-IDF version of our Multinomial NB and LR model were strong contenders. Using these three estimators, we conducted further experiments into upsampling and downsampling, as well as filtering certain features out or scaling features [Table 2 and 3].

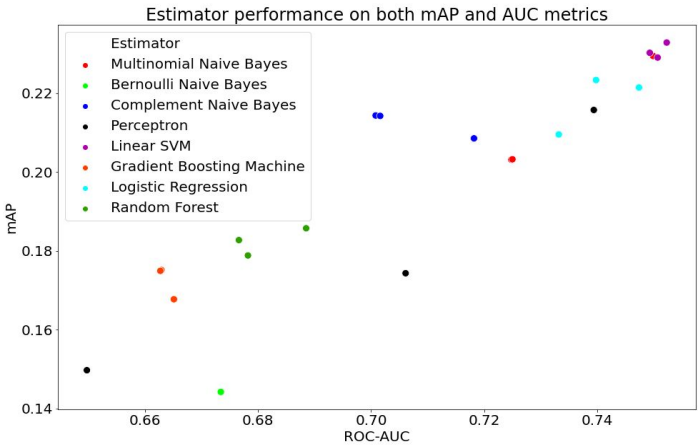

Figure 4: Model results, calculated on the validation set.

From these further investigations, we noticed that reducing the feature space for our estimators improved their performance slightly. This was present mostly in the LR estimator, but the other two estimators also had a smaller performance loss when the feature space was reduced. Upsampling instead of downsampling made performance far worse in recall and slightly better in precision, which means that overall, mAP and AUC also worsened [Table 2].

We also had a high discrepancy between the metrics on the training and validation set. This indicated that the training and validation sets were themselves very different. In order to test this, we split the training set further into a smaller training set and a second validation set. The performance of models trained on the smaller training set was measured using the second validation set. The performance was also very poor, which indicates that instead of there being a difference between our training and validation sets, we instead have a problem that is particularly difficult to solve [Table 4]. To better understand the difficulty in classification, two team members experimented with labeling by hand whether the reviews were fake or real. The results of this were far better than the baseline model, with an AUC of 0.6058 and a mAP of 0.5877. However, this was considered an upper bound for our results, since it involved human interaction.

Finally, we tested an ensemble model composed of all three of our best estimators (SVM, Multinomial NB, and LR, all with TF-IDF) on our validation set. This had the best performance when the class probabilities from each model were averaged out instead of being vote-based. Finally, we tested this ensemble model on the test set and had a mAP of 0.2579 and an ROC-AUC of 0.7705.

Discussion

We suspect that the feature space for the reviews was too large. This can be seen in the much smaller decrease in performance by limiting the feature space through L1 regularization or through only selecting non-rare features for the models [Table 3]. As compared to scaling, using SVD, or only using the handcrafted features, we found that the raw feature space composed of ngrams from the textual reviews is far more important for our models to learn an appropriate set of weights for predicting whether a review is fake. However, the handcrafted features also exerted a large influence in our models, as evidenced in our further experiments [Table 3] where removing handcrafted features worsened the overall performance.

Additionally, the ensemble model that we crafted performed far better than its individual components. This indicates that the different model types capture different, complementing elements of the feature space.

Conclusions

Textual data is a particularly difficult problem for machine learning approaches due to its large feature space and semantic meaning. These issues have also presented a computational challenge throughout our experiment. Despite this difficulty, we have found that we are able to achieve a new state-of-the-art in the Yelp NYC dataset by both ROC-AUC and mAP, using only a combination of extensive feature engineering and simple algorithms that are not computationally intensive.

Further research could be done on this problem by using different vectorization techniques such as word2vec [8] or transformers such as BERT [9]. This approach involves considerably more computational work, but it promises to provide even better results than ours. This being said, our model is far faster and can thus be used in real time on a website such as Yelp.

Bibliography

[1] Rayana, S., & Akoglu, L. (2016). Collective opinion spam detection using active inference. In *Proceedings of the 2016 SIAM International Conference on Data Mining* (pp. 630-638). Society for Industrial and Applied Mathematics. https://epubs.siam.org/doi/pdf/10.1137/1.9781611974348.71.

[2] McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization* (Vol. 752, No. 1, pp. 41-48). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.9324&rep=rep1&type=pdf.

[3] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/modules/naive_bayes.html.

[4] Bilbro, R., Ojeda, T., & Bengfort, B. (2019). Chapter 4: Text Vectorization and Transformation Pipelines. In Applied text analysis with Python. O'reilly. https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/ch04.html.

[5] Jabeen, H. (2018). Stemming and Lemmatization in Python. https://www.datacamp.com/community/tutorials/stemming-lemmatization-python.

[6] Liu, Y., Shriberg, E., Stolcke, A., & Harper, M. (2004). Using machine learning to cope with imbalanced classes in natural speech: Evidence from sentence boundary and disfluency detection. In *Eighth International Conference on Spoken Language Processing.* https://www.isca-speech.org/archive/archive_papers/interspeech_2004/i04_1525.pdf.

[7] Hansen, P. C. (1987). The truncated SVD as a method for regularization. *BIT Numerical Mathematics*, 27(4), 534-553. http://doi.org/10.1007/BF01937276.

[8] Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *International Conference on Learning Representations*. https://arxiv.org/abs/1301.3781.

[9] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT 2019*, 4171-4186. https://arxiv.org/abs/1810.04805.

[10] Wang, Z., Gu, S., & Xu, X. (2018). GSLDA: LDA-based group spamming detection in product reviews. *Applied Intelligence*, 48(9), 3094-3107. https://link.springer.com/article/10.1007/s10489-018-1142-1.

[11] Fontanarava, J., Pasi, G., & Viviani, M. (2017). Feature analysis for fake review detection through supervised classification. *In 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 658-666). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8259828.

Appendix

Figures

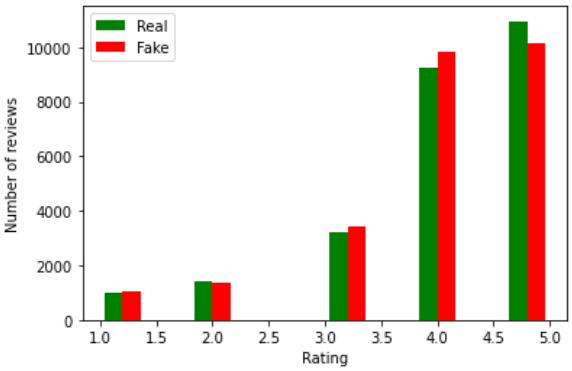

Figure 1: Number of real and fake reviews by rating.

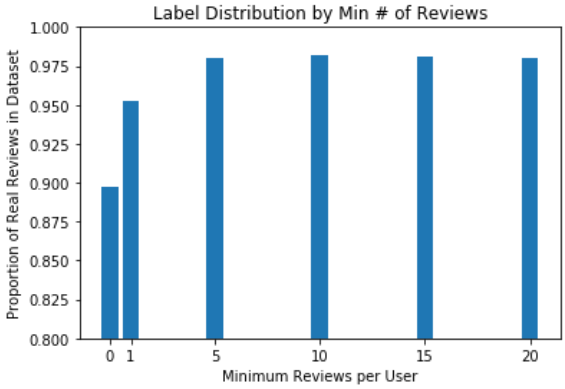

Figure 2: Distribution of real reviews in training dataset by minimum number of reviews per user.

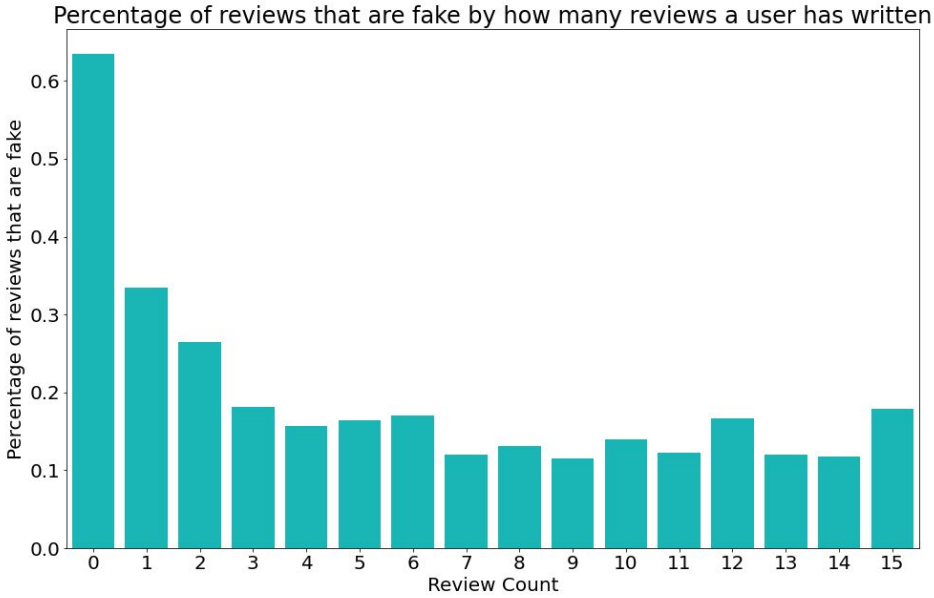

Figure 3: The percentage of reviews that are fake versus how many reviews a user has previously written. This was calculated on the downsampled set.

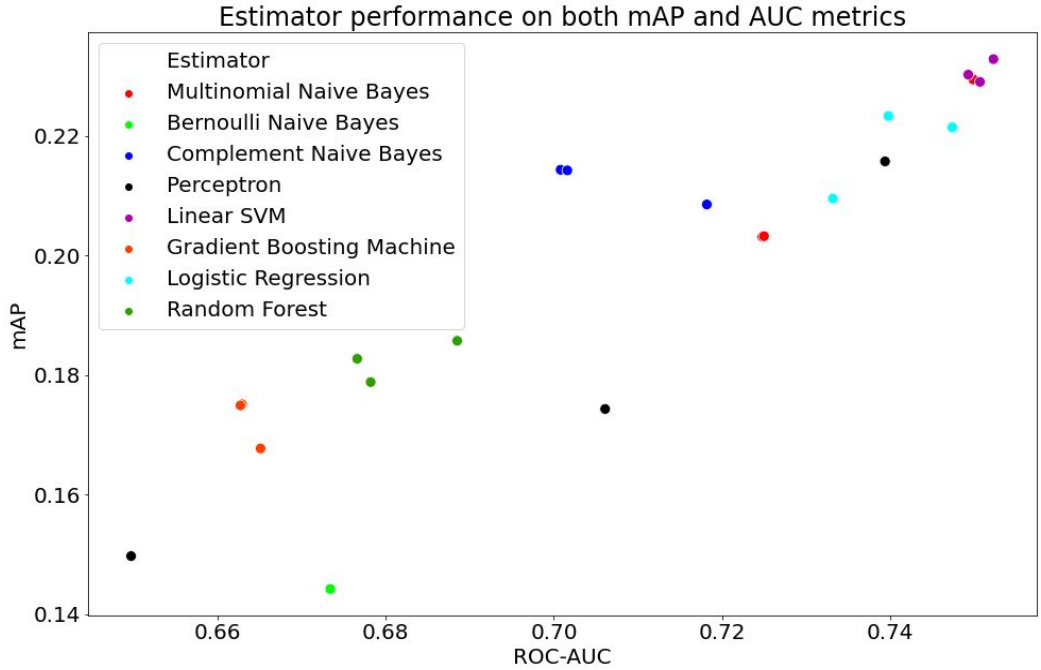

Figure 4: All model results, as calculated on the validation set.

Tables

Table 1: Results of all tuned models on validation set.

Model	Vectorizer	Dev AUC	Dev AP
Multinomial NB	count	0.7248	0.2031
Multinomial NB	tfidf	**0.7499**	0.2294
Multinomial NB	binary	0.725	0.2032
Bernoulli NB	binary	0.6734	0.1608
Complement NB	count	0.7008	0.2143
Complement NB	tfidf	0.7182	0.2085
Complement NB	binary	0.7016	0.2142
Linear Perceptron	count	0.7061	0.1743
Linear Perceptron	tfidf	0.7394	0.2157
Linear Perceptron	binary	0.6497	0.1497
Linear SVM	count	0.7507	0.229
Linear SVM	tfidf	0.7523	**0.2328**
Linear SVM	binary	0.7493	0.2302
GradientBoosting	count	0.6629	0.1751
GradientBoosting	tfidf	0.6651	0.1677
GradientBoosting	binary	0.6627	0.1749
Logistic Regression	count	0.7474	0.2214
Logistic Regression	tfidf	0.7398	0.2233
Logistic Regression	binary	0.7332	0.2095
Random Forest	count	0.6782	0.1788
Random Forest	tfidf	0.6885	0.1857
Random Forest	binary	0.6766	0.1827

Table 2: Multinomial Naive Bayes with TF-IDF vectorizer with various sampling methods.

	Dev AUC	**Dev AP**	**Dev Precision**	**Dev Recall**
Downsampling	**0.7499**	**0.2294**	0.192	**0.7297**
Upsampling	0.7199	0.212	**0.2963**	0.0973

Table 3: Additional experiments into model performance, all completed with TF-IDF vectorizer. Logistic Regression (LR) measured with ngram range (1, 2), Linear SVM (SVM) and Multinomial Naïve Bayes (MNB) with ngram range (1, 3).

Experiment	**LR AUC**	**SVM AUC**	**MNB AUC**	**LR AP**	**SVM AP**	**MNB AP**
No modifications	0.7497	**0.7515**	**0.7499**	0.2293	**0.2318**	**0.2294**
With MaxAbsScaler	0.7221	0.7335	0.6536	0.2115	0.2194	0.1595
Only engineered features	0.6991	0.6942	0.6581	0.1830	0.1812	0.1437
Only non-engineered features	0.6933	0.7125	0.7070	0.1917	0.2054	0.2059
Only engineered features, with scaling	0.6983	0.6904	0.6794	0.1827	0.1766	0.1610
TruncatedSVD, 100 components	0.7429	0.7402	0.7097	0.2222	0.2219	0.1875
TruncatedSVD, 10 components	0.7174	0.7148	0.6879	0.2028	0.2016	0.1571
With scaling and SVD, 100 components	0.7227	0.6917	0.6703	0.2092	0.1933	0.1808
L1 penalty	0.7497	0.7205	N/A	0.2293	0.2061	N/A
Limiting to only ngrams that show up in more than x emails, where x is the mean	**0.7548**	0.7512	0.7158	**0.2315**	0.2316	0.1888
Limiting to only ngrams that show up in more than 30 emails	0.7547	0.7503	0.6932	**0.2315**	0.2306	0.1649

Table 4: Multinomial Naive Bayes model performance on the given datasets versus the training set split into a smaller training set and a second validation set, each downsampled with TF-IDF.

	Dev AUC	**Dev AP**	**Dev Recall**
Using given train and dev datasets	0.7499	0.2294	**0.7297**
Training set split into 80% train & 20% dev	**0.7506**	**0.2299**	0.6405