Yelp Fake Review Detection

Xiang Li, Monika Dagar, Jeffrey Tumminia, Yibo Liu

1. Introduction

In recent years, online reviews have gained more significance in e-commerce. Because of their potential influence on customers, many fake reviews have been made up by unethical writers. Fake review detection has thus drawn the attention of researchers, and various approaches have shown promising results on this problem. In this project, we explored neural and non-neural approaches with textual and structural features, and we built a detection system to filter out fake reviews based on Yelp customer reviews dataset.

2. Related Work

Fake review detection problem is first introduced by Jindal and Liu (2007) from the spam filtering perspective. For non-neural methods, many traditional text features for classification are used in the following researches, such as n-gram, part-of-speech, topic models(Hernández-Castañeda et al.), psycho-linguistic features(Ott et al., 2011), while non-linguistic features, e.g. reviewer behavioral features(You et al., 2018) have also proved to be helpful. For neural methods, the features include word2vec. This problem is also related to sentiment analysis, so the methods for sentiment analysis can also be applied.

3. Problem Definition

Fake review detection is usually treated as binary classification problem. Our experiments follows the procedure for a classification task in general: (1) text pre-processing, (2) feature engineering, (3) learning algorithm.

4. Dataset

Yelp restaurant reviews dataset used in our experiments contains 274420 samples, with 250874 for training set and 23528 for development set respectively. Each sample consists of customer review, including a 0-5 rating and the text based review, and some additional information in the log, such as user ID, date, product ID.

The original dataset in highly imbalanced, where the ratio of positive samples is only 10.3%. So we either: (1) used the weight proportional to the inverse of the class frequencies, (2) or adapted over-sampling and under-sampling technique on it to create a balanced dataset;the balanced dataset includes 501748 samples in total, with equal number of positive examples and negative samples.

5. Methodology

We have applied four learning algorithms to this problem, including non-neural methods and neural methods. Non-neural methods include multinomial naive Bayes (MNB) and random forest (RF), while neural methods include convolutional neural network (CNN).

We are implementing different feature engineering for different models, mainly because of the difference between neural models and non-neural models.

5.1 Multinomial Naive Bayes

5.1.1 Feature Engineering

Features includes textual features and structural features. Textual features includes bi-gram and uni-gram generated from review texts in training set, structural features include rating and user behavior statistics, which are calculated on each user, namely (1) rating mean, (2) the total number of reviews, (3) the number of days with reviews, (4) average reviews per day, (5) maximum reviews in one day.

5.2 Random Forest

5.2.1 Feature Engineering

Features includes textual features and structural features. Textual features includes uni-gram generated from review texts in training set, passing through PCA filter. Structural features include (1) rating and its absolute deviation from 3, (2) user behavior statistics: the total number of reviews, mean of user rating and its absolute deviation from 3, maximum reviews in one day, maximum reviews in one day (3) product statistics: mean of rating and its absolute deviation from 3.

5.3 Convolutional Neural Network

5.3.1 Feature Engineering

One of our primary areas of research is into deep learning techniques to classify the reviews. Specifically, we focus on the ability to classify the reviews using *only* the text of the review. We have two reasons for this. Firstly, any latent features describing the user can be brought in on top of a classifier relying on information extracted through the text to improve it's performance later on. Secondly, we can imagine real-world scenarios and consider the context of the test set in which we have no prior information on a user before having to classify their review. Thus we find optimal utilization of the text to be the highest priority and becomes our main focus for this portion of the research. Our text pre-processing only consists of downsampling 50% of the genuine reviews out of the training data and tokenize and pad all reviews to be of the same length. The tokenization involves limiting the vocabulary and the padding introduces a max review length, introducing two parameters to our model.

5.3.2 Model

We experiment with a Convolutional Neural Network (CNN), first introduced for sentence classification in Kim (2014). The reason for choosing this model is because of it's utilization of an embedding layer. In Collobert et al. (2011) they introduce a method of transforming words into feature vectors. Their transformation learns a vector representation (or embedding) for each word in a vocabulary, using only the raw words found in real text of the language. This vector representation of a vocabulary (commonly summarized as word2vec), has a distinct advantage in being storable and reusable, allowing researchers to evaluate the capabilities of these representations across many domains and tasks, making them popular for transfer learning. Some popular ones include GoogleNews (goo) and GloVe (Pennington et al., 2014). We test both of these implementations, as well as implement a method to build, store and reuse our own custom embedding based on the training data.

6. Experiments

6.1 Experiment Details

6.1.1 Multinomial Naive Bayes

In preprocessing part, y labels are mapped to $\{-1, 1\}$. In feature extraction part, structural features are regularized to $(0, 1)$ with max-min scaler. To avoid the large feature space for bi-gram and uni-gram features, we are excluding features with less than 30 counts. Textual features are regularized with L_2 norm.

learning rate is 0.01, the class prior probabilities are learned from data (instead of Bernoulli, which is usually used for binary classification tasks).

6.1.2 Random Forest

The number of estimators is 500 and the minimum samples required for one leaf is 50.

6.1.3 Convolutional Neural Network

Kim's techniques have been implemented by many, and we utilize one available implementation on github (Cmasch, 2019). His implementation includes the GloVe word embedding, and we add the GoogleNews and custom word embedding to the model as options. From here we would typically take the suggestions of some available literature for hyperparameters and start our own tuning from there. Zhang and Wallace (2015) offer a deep sensitivity analysis of CNNs for text classification and insight into achieving optimal performance. Specifically, they recommended 4 layers of around 100 feature maps and a filter size of about 7 per layer. Unfortunately, we ran into serious computational limitations, where these suggested parameters could take anywhere from days to weeks to train on our hardware (considering we are not utilizing a GPU).

As such, we move on to finding a suitable parameter range to train. We used 3 layers with 10 feature maps and a filter size of 3,4, and 5 respectively. These parameters offered forgiving training time with the down-sampled data, allowing us to proceed with classification. We found that increasing vocabulary size past 500 and max review length past 300 yielded little increase in performance for significant computational cost, so we hold those values

constant. We evaluate above parameters for all of our included word embeddings, with the GoogleNews embedding CNN featured in our cross model comparison. After 1 run of 10 epochs with a batch size of 100 examples, we evaluate on the validation set. A table with the performance will be found in the results section.

6.2 Experiment Results

6.2.1 Overall Results

Table 1 shows the performances of different models. Random Forest outperforms the other three models in terms of AUC (73.60%) and AP (21.70%), while other classifiers also achieve acceptable performances.

Table 1: Optimal Results for Different Models

Model	precision	recall	f1 score	ROC	AP
MNB	86%	66%	73%	65%	15%
RF	88.28%	52.87%	61.37%	73.60%	21.70%
CNN	85.51%	45.23%	29.42%	64%	12.4%

6.2.2 Embedding for Neural Method

Below is a breakdown of the performance of each embedding on our holdout validation dataset. All models are ran with the parameters discussed in section 5.

Table 2: Metrics for CNN on Validation Data

Embedding	Loss	Accuracy	Recall	AUC
Custom	.5	83.44%	17.60%	63%
GloVe	.43	84.60%	16.83%	63%
Google	.42	85.51%	17.98%	64%

We saw that the despite the specialized task, more robust word embeddings seemed to perform the best. Overall, performance was weak which is somewhat unsurprising given the lack of parameter exploration. A clear next step in this research is to extend the training to a GPU to allow us to take full advantage of the power of deep learning.

7. Conclusion and Future Work

We have conducted a series of classification experiments with non-neural and neural classifiers. Our experiments show that for non-neural models, the Rf performs the best. For neural models, Google embedding is the best, where word embedding can be seen as text features. In all three models, Random Forest achieves the best performance in our experiments, which reaches an 73.60% ROC and 21.70% AP on classification task.

References

Google code archive - long-term storage for google code project hosting. URL https://code.google.com/archive/p/word2vec/.

Cmasch. cmasch/cnn-text-classification, Dec 2019. URL https://github.com/cmasch/cnn-text-classification.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

Ángel Hernández-Castañeda, Hiram Calvo, Alexander Gelbukh, and Jorge J. García Flores. Cross-domain deception detection using support vector networks. *Soft Computing*, pages 585–595.

Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 1189–1190, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi: 10.1145/1242572.1242759. URL https://doi.org/10.1145/1242572.1242759.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey 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*, pages 309–319, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-1032.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Zhenni You, Tieyun Qian, and Bing Liu. An attribute enhanced domain adaptive model for cold-start spam review detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1884–1895, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1160.

Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.