Anomaly Detection for Flagging Fake Product Reviews

Maha Ashour
Dept.of Electrical & Computer Engineering
Boston University
Boston, MA 02215
mash@bu.edu

Rui Liu
Division of Systems Engineering
Boston University
Boston, MA 02215
rliu@bu.edu

Yenai Ma
Dept. of Electrical & Computer Engineering
Boston University
Boston, MA 02215
yenai@bu.edu

Abstract

Online reviews often provide useful information about a product that would otherwise be hard to attain. They are one of the primary factors in a customer's decision making and consequently businesses have a lot to gain by ensuring they receive fair reviews. Thus, it is important to flag fake reviews. We apply a number of supervised and unsupervised methods to the problem of review spam detection. Of the supervised methods, we find that One-Class SVM is the most effective as measured by AUC. Unsupervised global k-NN does not perform any better than random guessing.

1. Introduction

The study of anomaly (or outlier) detection has benefited greatly from the explosion of big data in recent years, with applications in many different areas. One such application is in the field of 'opinion spam' detection. With the rapid growth of online shopping and the concurrent decline in retail business, customers and businesses alike rely increasingly on online reviews as a primary signal of product quality and customer satisfaction. It is not unusual for a customer to read several product reviews on Amazon or to look up the star rating of a restaurant on Yelp before first trying it. However, these crowd-sourced opinions may not always be reliable or genuine. The surge in online reviews has been accompanied by a corresponding increase in fake reviews, often termed opinion spam, that is difficult to monitor and filter. For instance, some retailers will submit fake positive reviews of their own products as a means of self-promotion, or conversely create fake negative reviews of competing businesses to deter customers. Customers as well might intentionally or unintentionally write biased or deceptive comments. This element of deception presents one of the biggest challenges in anomaly detection, since it is very difficult to discern whether a good or bad review is written by a satisfied or dissatisfied customer, or rather an individual or entity masquerading as one. In general, there are three types of spam reviews: untruthful reviews (otherwise known as fake reviews), reviews on brand only (promoting brand instead of product) and non-reviews (for example, advertisements) [7]. For simplicity, our research treats all three types as the same.

2. Literature Review

Jindal et al. are the first to study opinion spam in product reviews [7]. The study categories reviews from online retailer Amazon.com into the three aforementioned types and applies a variety of methods to detect review spam based on its type. They employ a logistic regression model to detect fake reviews using manually labelled training data. Lim et al. [10] take a different approach by seeking to identify the users generating spam reviews, also called review spammers. Rather than classifying reviews as spam or non-spam, they focus on identifying whether the author is a spammer by looking for clusters of anomalous activity tied to the author. They assign users scores based on suspicious activity and then compare their results against those of human evaluators.

There are two main approaches to filtering: supervised and unsupervised learning. For supervised learning, Naïve Bayes [5, 9] and Support Vector Machines (SVM) [12] have been proposed for the review spam detection problem. In [12], classification experiments are based on SVM^{Light} [8] using 5-fold cross validation and a combination of linguistic and behavioral features. Linguistic features have been shown to perform extremely well (attaining around 90% accuracy) in detecting crowd-sourced fake reviews generated using Amazon Mechanical Turk (AMT). However, for Yelp data, linguistic features have not been as effective, and Mukherjee et al. see improved detection using behavioral features [12]. Li et al. use a mix of supervised and semi-supervised techniques on partially labeled data [9]. Their finding is that, among the supervised techniques, Naïve Bayes yields the best results. Given the class imbalance inherent in anomaly detection problems, One-Class SVM is

Domain	fake	non-fake	%fake	total # reviews	#reviewers	#products
Hotel	778	5076	13.3%	5854	5026	72
Restaurant	8141	53400	13.2%	61541	33964	129

Table 1. Dataset statistics

proposed as an alternative approach [16].

Besides these supervised approaches, unsupervised modeling techniques to detect review spam in the Bayesian setting are used by Mukherjee et al. [11]. Other unsupervised approaches to anomaly detection include global k-nearest-neighbors and clustering techniques [14],[1],[4].

3. Problem Formulation and Solution Approaches

3.1. Problem Formulion

We explore anomaly detection in the context of detecting fake product reviews on Yelp hotel and restaurant data. Our aim is to classify a review as either genuine (normal) or fake. We use the YelpCHI dataset which has 13% fake reviews, as summarized in Section 4.1 and Table 1. We treat the labels generated by Yelp's Anti-Fraud filter as the ground truth in our analysis. We use both supervised learning methods such as One-Class Naïve Bayes and One-Class SVM, and an unsupervised learning method, namely, Global k-Nearest-Neighbors [4] to flag the fake reviews. We use common metrics to compare the performance of the three models against each other and against benchmarks in the literature.

3.2. Naïve Bayes and One-Class Naïve Bayes

Naïve Bayes Naïve Bayes is a popular and effective method for text categorization since 1960s [15]. The fundamental idea is to apply *Bayes' Rule* (Equation (1)) while assuming all features are conditionally independent for a given class (Equation (2)). Although the independence assumption does not hold in many real-world applications, Naïve Bayes classifier still works surprisingly well [9]. The parameters of each feature are learned using either term frequency for categorical features or Gaussian model for real-value features. Naïve Bayes classifier uses MPE decision rule (Equation (3)).

$$p(y|\mathbf{x},\theta) = \frac{p(\mathbf{x}|y,\theta) \times p(y)}{p(\mathbf{x})} \tag{1}$$

$$p(\mathbf{x}|y,\theta) = \prod_{i=1}^{d} p(x_i|y,\theta_i) \tag{2}$$

$$\hat{y} = argmax_y p(y|\mathbf{x},\theta) = argmax_y p(y) \times \prod_{i=1}^{d} p(x_i|y,\theta_i) \tag{3}$$

One-Class Naïve Bayes One challenge associated with fake review detection is unbalanced data: the majority of the reviews are normal while only a few are abnormal. As a result, there might be very limited (or no) data to train the abnormal class. In addition, the lack of similarity in the abnormal 'class' could also cause a problem. In the cats vs. non-cats analogy, it is not easy to extract the similarity between a human and a desk, although both of them are non-cats. Similar to One-Class SVM, One-Class Naïve Bayes uses only normal data to train the parameters for each features, based on which to assign labels. The decision is made by comparing the probability of being in a normal class to a threshold (Equation (4)), where the threshold can be either a constant (e.g., minimum probability of the normal class) or a function.

$$\hat{y} = normal, if \quad p(y = normal|\mathbf{x},\theta) > p_{threshold} \tag{4}$$

3.3. One-Class Support Vector Machine

In classic binary SVM, problems usually have two well-defined classes, such as men vs. women, dogs vs. cats and so on. However, when it comes to distinguishing cats from non-cats, it is hard to train the features of non-cats, because an infinite number of items could qualify as non-cats. On the other hand, in real life applications, abnormal samples are rare and it may cost a lot to generate abnormal samples. To handle these challenges, one proposal is One-Class SVM [16]. Given that our data set only contains a small percentage of abnormal reviews, we believe One-Class SVM is suitable for our detection problem. In this section we introduce the One-Class SVM method [16] in the setting of our problem.

We consider training data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $n \in \mathbb{N}$ is the number of samples and $x_i \in \mathbb{R}^N$, $y_i \in \{+1, -1\}\ \forall i = 1, \ldots, n$. Assume that if a sample i is a non-fake review, then $y_i = +1$; otherwise, $y_i = -1$. Let $I = \{i|y_i = +1, i = 1, \ldots, n\}$ and let $l = |I|$. Then $\{x_i|i \in I\} \in \mathcal{X}$ where $\mathcal{X}$ is the non-fake review set. For simplicity, we think of it as a compact subset of $\mathbb{R}^N$. Assume that the kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be expressed as an inner product of a feature map $\Phi : \mathcal{X} \to V$, i.e., $k(x, x^{'}) =< \Phi(x), \Phi(x^{'}) >$.

The aim of one class SVM is to develop a function $f : \mathcal{X} \to \{+1, -1\}$ such that f returns $+1$ for those non-fake reviews' feature x and -1 elsewhere. To separate the data set $\{x_i|i \in I\}$ from the origin, we need to solve the following optimization problem first

$$\min_{\omega \in V, \xi \in \mathbb{R}^l, \rho \in \mathbb{R}} \frac{1}{2}\|\omega\|^2 + \frac{1}{\nu l}\sum_{i \in I} \xi_i - \rho \tag{5}$$

subject to

$$< \omega, \Phi(x_i) > -\rho \geq -\xi_i,\ \xi_i \geq 0\ \ \forall i \in I \tag{6}$$

where $\nu \in (0, 1)$ is a parameter.

If (ω^*, ρ^*) solves this optimization problem, then the decision function $f(x) = sign(< \omega^*, \Phi(x) > -\rho^*)$ will be

positive for most non-fake reviews. The dual problem can be written as

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i,j \in I} \alpha_i \alpha_j k(x_i, x_j) \tag{7}$$

subject to

$$0 \leq \alpha_i \leq \frac{1}{\nu l}, \ \sum_{i \in I} \alpha_i = 1 \ \ \forall i \in I \tag{8}$$

By the dual problem, the decision function can be expressed as $f(x) = sign(\sum_{i \in I} \alpha_i k(x_i, x) - \rho^*)$.

3.4. Global k Nearest Neighbor

Nearest neighbors methods are quite useful in anomaly detection. Here we explore a variant called global k-nearest-neighbors (k-NN). Unsupervised global k-NN works by assigning an anomaly score to each review in an unlabeled data set. Reviews are then ranked by their anomaly scores, and a threshold is used to determine which reviews qualify as anomalous. As a result, global anomalies are more easily detected than local anomalies, since they are less likely to have normal points in their neighborhood and consequently will receive a higher anomaly score.

Anomaly scores are generated based on the distance of a given review to its neighbors, specifically its k^{th} neighbor. We take two approaches, k^{th}-NN [14] and k-NN [1]. The first approach, k^{th}-NN, considers only the distance of a point to it's k^{th}-nearest-neighbor whereas the latter, k-NN, takes the average distance of a point to all its k-nearest-neighbors. For clarity we follow the same naming convention used by Goldstein et al. [4].

We use $D^k(p)$ to denote the distance of point p from its k^{th} nearest neighbour. For a given threshold T, p is an anomaly if its anomaly score is in the top T of all points. Therefore, the algorithm relies only on two parameters, the neighborhood size k and the threshold T to determine which points are anomalies.

4. Implementation

4.1. Datasets, Data Cleaning and Pre-processing

There is a lot of business and product review data available through the Outlier Detection DataSets [13] at Stony Brook University. We use YelpCHI datasets for our analysis, which contains two domains, one is for hotel reviews and the other is for restaurant reviews. The dataset statistics are summarized in Table 1, where there are 5854 reviews from 5026 reviewers on 72 hotels and 61541 reviews from 33964 reviewers on 129 restaurants. The YelpCHI dataset gives information on date, review ID, reviewer ID, product ID, fake or not, user feedback (i.e., how many people think this review is cool, useful, or funny), star rating, and text review [12]. An example review is shown in Figure 1. All reviews are labeled as either fake or genuine. The truth labels provided in the datasets are imperfect they are generated by Yelps anti-fraud filter and as such may be misclassifying some reviews. However, this is not an uncommon problem in anomaly detection [3] and for our exploration, we will use the provided label as the ground truth.

One characteristic of review spam detection is that we have to deal with textual data. Thus, inevitably, we have to clean and pre-process the text or unstructured text data before we start to implement any anomaly detection technique. We take a few steps to perform data cleaning and pre-processing: (i) replace review ID, reviewer ID and product ID with numbers starting from 0, (ii) count text review length and use it as a new feature, (iii) break down the text reviews into lists of words (unigrams), (iv) remove punctuation, (v) convert all words to lower case, (vi) remove 'stopwords' (we use standard stopwords downloaded from NLTK library for English [17]), (vii) build up the vocabulary for all reviews, (viii) count the appearance of each remaining word in each review, and (iv) finally, vectorize the '*bag-of-words*' to a sparse matrix $(I, J) = V$, where I is the review index, J is the word index in the vocabulary, and V is the number of appearances of Word J in Review I.

In addition to the pre-processing steps described above, we create word bi-grams (also called 2-grams) which results in a ten-fold increase in the vocabulary size. Due to the increased computational and memory complexity, we re-run our detection algorithms using the word bi-grams only for the hotel data. While the purpose of the bi-grams is to capture more of the context of the original text, we do not observe any performance improvement, rather a slight degradation, at the cost of $10\times$ the simulation time. So in Section 5, we only report results associated with uni-gram features.

After data cleaning and pre-processing, we have generated three csv files for each domain. The first file is the vocabulary of all reviews, including a list of words and a list of word indices. The second file is the *bag-of-words*, which uses three columns (review index, word index in the vocabulary, and word count, respectively) to represent a sparse matrix listing the word count of each word in each review. The last file contains the cleaned logistic information of the reviews, including date, review ID, reviewer ID, product ID, fake or not, cool, useful, funny, and star rating which are originally provided in the dataset, and an additional feature of text length.

4.2. Naïve Bayes

We use 5-fold cross-validation to separate our data into training and test sets. All the results we collect are the average of these 5 runs to reduce the noise resulting from training and test data separation. We use *bag-of-words* model for text review to get a $n \times N$ sparse matrix, where n is the number of reviews, N is the vocabulary size, and the matrix values are word counts. We train and get the term frequen-

date	reviewID	reviewerID	productID	fake or not	cool	useful	funny	star rating
2/5/2010	i4HIAcNTjabdpG1K4F5Q2g	Sb3DJGdZ4Rq__CqxPbae-g	tQfLGoolUMu2J0igcWcoZg	N	9	4	11	3
text review	The only place inside the Loop that you can stay for $55/night. Also, the only place you can have a picnic dinner and get a little frisky on the 17th floor roof and then wake up in your room the next morning to an army of ants going in on your picnic							

Figure 1. Example review

cies for the normal class and the abnormal class, and then compute the probability for a review being normal (or abnormal) using Equation (2). All the computation is in the log scale to resolve the underflow issue due to large vocabulary size.

Text Review and Metadata To jointly use text reviews and metadata, we treat text review as one feature whose probability is computed by *bag-of-words* model described above. We use the metadata each as a feature and train them separately. For categorical metadata (such as reviewerID, productID, star rating), the parameters to learn for each class are the term frequencies β (i.e., the occurrence of each categorical value in a class over the occurrence of all categorical values in the class). For real-value metadata (such as text length and user feedbacks of cool, useful and funny), we use Gaussian model to get the mean and variance of the feature. It should be noted that we observe zero-mean-zero-variance distribution for user-feedback features in the abnormal class, which are essentially all zeros. This introduces 'multiplying by zero' issue. To handle that, we eliminate the feature while observing the user feedbacks are zero.

One-Class Naïve Bayes We use normal data only for One-Class Naïve Bayes. For the abnormal class, instead of using the actual data, we set all term frequencies (β) to $1/N$, assuming the word occurrence are uniformly distributed for text feature [18]. For metadata features, we apply the same trick, assuming the empirical data X_{emp} follow uniform distribution.

4.3. One-Class SVM

We implement One-Class SVM on the Yelp Hotel dataset. We first separate the data into training and test sets. We use word frequency, and data on whether a review is cool, useful, or funny as features. LIBSVM tools [2] are used to implement One-Class SVM methods.

We apply 5-fold cross-validation to the training data. We perform a grid search for both an RBF kernel and a linear kernel. For the linear kernel, ν is the only parameter, while the RBF kernel has two parameters ν and γ to be determined. In order to compare the two kernels and several parameters, we use AUC (which is the area under the receiver operating characteristic (ROC) curve) as a performance metric. We first compare the AUC of the two kernels. For the RBF kernel, we choose γ_ν for each ν such that (ν, γ_ν) has the highest AUC. Then we compare the AUC generated by (ν, γ_ν) to the AUC generated by the linear kernel with the same ν. Figure 2 shows that the RBF kernel performs better than the linear kernel for our dataset. Furthermore, we use the best pair of parameters from cross-validation for the RBF kernel, training the entire data and then testing the model. The result of the grid search for the RBF kernel is shown in Figure 3.

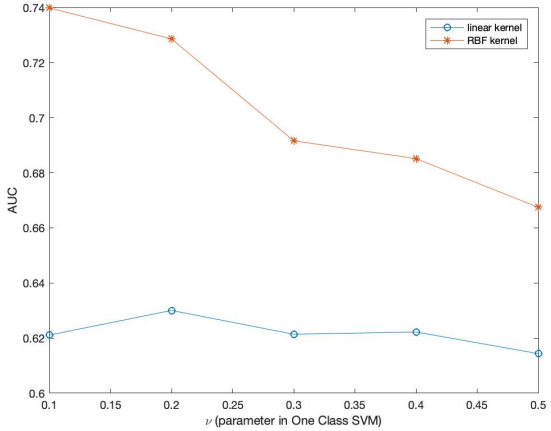

Figure 2. AUC of linear and RBF kernel.

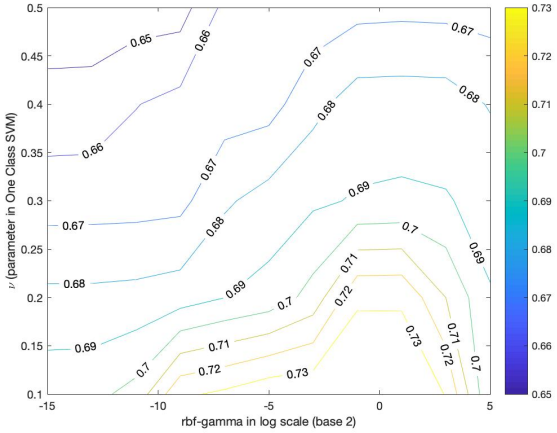

Figure 3. Grid search for RBF kernel.

4.4. Global k-NN

To implement global k-NN we considered a few common distance metrics, such as euclidean distance and cosine distance to calculate the distance between every pair of reviews. The two metrics are defined as follows:

The euclidean distance between two vectors p and $q \in$

$\mathbb{R}^d$ is defined as:

$$d_{p,q} = \sqrt{\sum_{i \in D} (q_i - p_i)^2} \tag{9}$$

And the cosine distance between p and q is given by:

$$d_{p,q} = 1 - \cos(\theta) \tag{10}$$

where the cosine similarity is defined as:

$$\cos(\theta) = \frac{p.q}{\|p\|\|q\|} \tag{11}$$

Another metric used to compare textual data is the Jaccard distance which is a measure of the similarity of two documents and is defined as the size of the intersection of two sets over the size of their union. While the Jaccard distance can be used to detect duplicates - a kind of fake/spam review - it is not appropriate in this context since our data set has been already cleaned to remove duplicate reviews.

The reviews are represented by vectors of words counts and the distances between the vectors are calculated as previously described. For a given neighborhood size, k, we repeated the ranking algorithm for different values of T to produce the ROC. The whole process is then repeated for a range of k values. We picked k = 10 as a starting point based on [4] who suggest a value of k between 10 and 50. We incremented k in steps of 10 up to 100.

5. Experimental Results

5.1. Performance metrics

One challenge associated with fake review detection is working with unbalanced data. As a result, common performance metrics such as the accuracy rate (CCR) are not appropriate. Blindly predicting all the reviews as normal would yield quite a high accuracy (87% in our case). For anomaly detection tasks, performance is better measured by the share of anomalies that are successfully detected. However, solely looking at the detection rate (recall) is not informative either, since a high detection rate can be achieved by blindly flagging every review as abnormal. Similarly, neither precision nor specificity are sufficient metrics on their own. The F1score, which balances both recall and precision, and the AUC, which balances recall and specificity, give a better sense of the performance of an anomaly detection algorithm and are more commonly used for unbalanced datasets [6]. In this context, the AUC can be interpreted as the probability that an anomaly detection algorithm will assign a higher score to a randomly picked abnormal instance than to a randomly picked normal instance [4]. For our analysis, we use AUC as the main performance metric.

Combination	train CCR	test CCR	precision	recall	specificity	f1score	AUC
text review only	0.9544	0.8107	0.2736	0.2558	0.8958	0.2641	0.6556
text + reviewerID	0.9544	0.8107	0.2736	0.2558	0.8958	0.2641	0.6556
text + productID (P)	0.9561	0.8150	0.2873	0.2648	0.8993	0.2753	0.6579
text + cool	0.9544	0.8107	0.2736	0.2558	0.8958	0.2641	0.6556
text + useful	0.9544	0.8107	0.2736	0.2558	0.8958	0.2641	0.6556
text + funny	0.9544	0.8107	0.2736	0.2558	0.8958	0.2641	0.6556
text + star$_r ating(S)$	0.9564	0.8159	0.2876	0.2609	0.9009	0.2732	0.6587
text + length (L)	0.9529	0.8095	0.2750	0.2648	0.8930	0.2695	0.6554
text + P + S	0.9582	0.8170	0.2898	0.2597	0.9025	0.2733	0.6611
text + P + S + L	0.9561	0.8141	0.2840	0.2622	0.8987	0.2722	0.6608
text + all metadata	0.9561	0.8141	0.2840	0.2622	0.8987	0.2722	0.6608
metadata only	0.1529	0.1510	0.1342	0.9884	0.0227	0.2363	0.6860

Table 2. Performance of Naïve Bayes using combinations of text review and metadata.

5.2. Naïve Bayes

Table 2 shows performance of Naïve Bayes classifier using various combinations of text review and metadata using YelpCHI-Hotel data. Although we use AUC as our main metric, we still show other metrics for comparison purposes. There is no performance improvement when using text review and one of reviewerID, cool, useful and funny. As we mentioned in Section 4.2, cool, useful and funny features are deterministic zeros for the abnormal class. For Hotel data, we have 5854 reviews and 5026 reviewers, so effectively only a few reviewers contribute to more than one reviews. For the metadata only case, it have a highest AUC. However, telling from the other metrics (15% test CCR, 99% recall (detection rate), and 2% specificity (98% false alarm rate), the classifier is mistakenly predicting almost all reviews as 'fake', which is as expected. Overall, combining text review and metadata does improve the performance (AUC), but slightly. This makes sense since most information are contained in the text portion.

Comparing Naïve Bayes with One-Class Naïve Bayes (see Figure 5), surprisingly, Naïve Bayes performs better. One reason might be the One-Class Naïve Bayes could still be improved by using more advanced method to set the threshold function. Another reason might be Naïve Bayes classifier is stable enough with our dataset that 13% anomalies are enough to train the abnormal class.

Thus, we also explore how Naïve Bayes and One-Class Naïve Bayes work with various percentage of fake reviews. The reviews to be removed from the dataset for a certain class are randomly chosen. Figure 4(a) shows AUC values when we decrease number of fake reviews while keep the number of normal reviews fixed to 5076 (aka. reducing the anomaly percentage from 13.3% to 0.2%). Naïve Bayes consistently outperforms One-Class Naïve Bayes, especially when the fake review count is larger than 150 or so. For fake review count smaller than 100, both of them have a sudden drop in AUC. As shown in Figure 4(b), while fixing the fake review count to 778 and decreasing the number of normal reviews (aka. anomaly percentage increases from 13.3% to 85%), Naïve Bayes' performance degrades, while One-Class Naïve Bayes performs more stable and outbeats Naïve Bayes when the normal review count drops below 4000. As a result, the performance of Naïve Bayes highly

test CCR	precision	recall	specificity	f1score	AUC
0.7822	0.023	0.0154	0.8997	0.0185	0.7424

Table 3. Performance of One-Class SVM.

	Threshold	CCR	precision	recall	specificity	f1score	AUC
k-th NN	20%	0.7079	0.1037	0.1568	0.7924	0.1249	0.4207
	15%	0.7474	0.1031	0.117	0.844	0.1096	0.4207
	10%	0.7899	0.1156	0.0874	0.8976	0.0996	0.4207
k NN	20%	0.7084	0.1033	0.1555	0.7931	0.1242	0.4207
	15%	0.7482	0.1036	0.117	0.845	0.1099	0.4207
	10%	0.7904	0.1162	0.0874	0.8981	0.0998	0.4207

Table 4. Performance of Global k-NN using word uni-grams only.

depends on the size of the normal class, with respect to which One-Class Naïve Bayes is more robust.

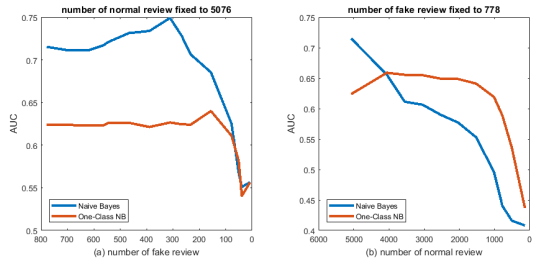

Figure 4. Performance while varying anomaly percentage by removing (a) fake reviews (b) normal reviews.

5.3. One-Class SVM

Table 3 shows the performance of One-Class SVM. The confusion matrix indicates that One-Class SVM can identify almost all fake reviews, however, 40% of non-fake reviews are flagged as fake reviews. So while the detection rate is good, the false alarm rate is quite high. We also try binary SVM algorithm on our dataset. It does not work well because of unbalanced data. The confusion matrix shows that binary SVM tends to label all the reviews as non-fake reviews.

5.4. Global k-NN

Global k-NN performed poorly on our data. Table 4 gives the results for both the k^{th}-NN and k-NN implementations using a neighborhood size k = 10 and using euclidean distance to generate anomaly scores. The AUC is a little below 0.5 which implies that we could beat the k-NN algorithm's results by randomly labeling reviews as anomalous or not. The results for larger values of k are very similar and therefore not reported. Using cosine distance instead of euclidean distance did not significantly impact results either.

One explanation for the failure of global k-NN at the detection task, is that the distance between two reviews represented as word count vectors is not adequately measuring the similarity or dissimilarity of reviews. When examining the distance of a review to its neighbors, we found that there was not a lot of variation in the distance of a review to its closest and furthest neighbor. As a result, we re-calculate the distance using word bi-grams; however the results do not improve. The results are presented in Table 6 in the Appendix.

Our hypothesis is supported by Murkherjee et al. [12] who find that the word distributions of fake and non-fake reviews in the Yelp data are very similar which they suggest is a reason why word uni-grams and bi-grams are less effective in anomaly detection of textual data.

We also try calculating anomaly scores using the metadata only but the AUC is close to 0.5 as before.

5.5. Overall Comparison

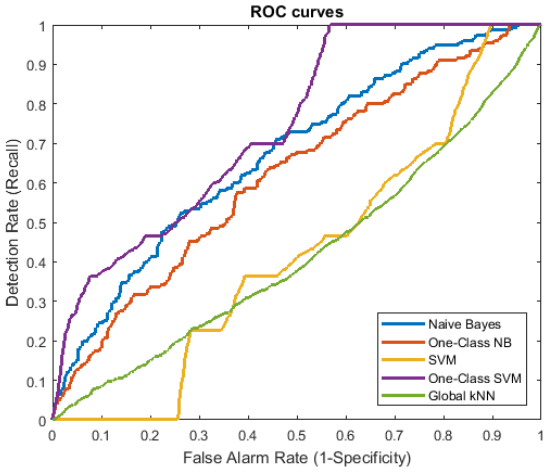

Figure 5. ROC curves of all the methods that are explored.

Figure 5 shows the ROC curves of all the methods we have explored. The curves have been discussed in each indivisual subsections. As a conclusion, for YelpCHI Hotel and Restaurant datasets, One-Class SVM performs best. To be noted, for YelpCHI Restaurant dataset, we observe similar trends and performance as the Hotel dataset for all the methods. So we don't explicitly show the results for Restaurant data.

6. Conclusion

To tackle the review spam problem, we study a number of supervised and unsupervised machine learning methods. For supervised methods, we explore Naïve Bayes, One-Class Naïve Bayes, and One-Class SVM; for unsupervised method, we explore Global k-NN. Using YelpCHI Hotel and Restaurant data, One-Class SVM performs best, achieving the highest AUC.

To improve the learning models, we can try more data pre-processing such as further cleaning the textual data with lemmatization and stemming. This could lead to improved performance of the supervised models, and at the same time produce more meaningful distance measures between reviews for Global k-NN. We can also try using word and sentence embeddings to capture some of the information that is lost in our current pre-processing of the original review.

7. Description of Individual Effort

Each of us took the lead on one specific anomaly detection technique, as listed below, and the corresponding tasks related to the topic, including literature review, learning and implementing the model, analyzing data, preparing slides, writing up the related parts of the report (theory, implementation and result sections). We worked together on general tasks, including topic selection, dataset search, general literature review, as well as problem approach, strategy, presentation, and report writing (the remaining sections).

- Maha: k-NN Global
- Yenai: Data Clean and Pre-processing, Naïve Bayes
- Rui: One-class SVM Models

References

[1] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–27. Springer, 2002.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *In Proceedings of the International Conference on Machine Learning*, pages 255–262. Morgan Kaufmann, 2000.

[4] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLOS ONE*, 11(4):1–31, 04 2016.

[5] A. Hammad and A. El-Halees. An approach for detecting spam in arabic opinion reviews. *The International Arab Journal of Information Technology*, 12, 2013.

[6] M. Jiang, P. Cui, and C. Faloutsos. Suspicious behavior detection: Current trends and future directions. *IEEE Intelligent Systems*, 31(1):31–39, 2016.

[7] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the 2008 international conference on web search and data mining*, pages 219–230. ACM, 2008.

[8] T. Joachims. Making large-scale svm learning practical. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund, 1998.

[9] F. Li, M. Huang, Y. Yang, and X. Zhu. Learning to identify review spam. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2488–2493. AAAI Press, 2011.

[10] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948. ACM, 2010.

[11] A. Mukherjee and V. Venkataraman. Opinion spam detection: An unsupervised approach using generative models. *Techincal Report, UH*, 2014.

[12] A. Mukherjee, V. Venkataraman, B. Liu, and N. S. Glance. What yelp fake review filter might be doing? In *ICWSM*, pages 409–418, 2013.

[13] Outlier detection datasets. `http://odds.cs.stonybrook.edu/`. Accessed: 2018-11-5.

[14] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *ACM Sigmod Record*, volume 29, pages 427–438. ACM, 2000.

[15] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[16] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[17] B. Steven, E. Loper, and E. Klein. Natural language processing with python. *OReilly Media Inc*, 2009.

[18] K. Wang and S. J. Stolfo. One-class training for masquerade detection. In *Workshop on Data Mining for Computer Security, Melbourne, Florida*, pages 10–19, 2003.

8. Appendix

Combination	train CCR	test CCR	precision	recall	specificity	fscore	AUC
text review only	0.8222	0.6791	0.2198	0.5590	0.6974	0.3155	0.6848
text + reviewerID	0.8221	0.6791	0.2197	0.5590	0.6974	0.3155	0.6848
text + productID (P)	0.8235	0.6817	0.2217	0.5602	0.7002	0.3177	0.6863
text + cool	0.8222	0.6791	0.2198	0.5590	0.6974	0.3155	0.6848
text + useful	0.8222	0.6791	0.2198	0.5590	0.6974	0.3155	0.6848
text + funny	0.8222	0.6791	0.2198	0.5590	0.6974	0.3155	0.6848
text + star$_{r}ating(S)$	0.8242	0.6824	0.2222	0.5600	0.7011	0.3181	0.6871
text + length (L)	0.8163	0.6729	0.2170	0.5645	0.6894	0.3135	0.6851
text + P + S	0.8257	0.6849	0.2239	0.5605	0.7038	0.3200	0.6887
text + P + S + L	0.8195	0.6782	0.2212	0.5682	0.6949	0.3184	0.6889
text + all metadata	0.8195	0.6781	0.2212	0.5682	0.6949	0.3184	0.6890
metadata only	0.1779	0.1776	0.1363	0.9778	0.0556	0.2393	0.6703

Table 5. Performance of Naïve Bayes using combinations of text review and metadata for YelpCHI-Restaurant data.

	Threshold	CCR	precision	recall	specificity	f1score	AUC
k-th NN	20%	0.7053	0.0997	0.1517	0.7902	0.1203	0.4191
	15%	0.7485	0.1057	0.1195	0.845	0.1122	0.4191
	10%	0.789	0.1134	0.0861	0.8968	0.0979	0.4191
k NN	20%	0.7074	0.1008	0.1517	0.7926	0.1211	0.4191
	15%	0.7489	0.1059	0.1195	0.8454	0.1123	0.4191
	10%	0.7901	0.1145	0.0861	0.898	0.0983	0.4191

Table 6. Performance of Global k-NN using word bi-grams.