Predict User Ratings Based on Review Texts

MAI Vinh Tuong
EURECOM
Email: mai@eurecom.fr

CAI Kehe (*)
EURECOM
Email: cai@eurecom.fr

Pietro Michiardi
EURECOM
Email: michiard@eurecom.fr

***Abstract*—Yelp has introduced a deep dataset for research-minded academics from their wealth of data. The dataset includes businesses, users, reviews and tips etc. information. How can we predict the user ratings based on its text alone? It is a mapping problem from textural information to a discrete number (Normally 1 to 5). In this paper, we use cutting-edge methodologies (SVM and Random Forest) that are believed to be the most accurate tools as our classifiers. We mainly focus on feature selection part and appropriate parameter space exploration, which are very important since we only have textual information. We study in detail error metrics and confidence to make sure our prediction is generic and our method has good generalization properties.**

I. Introduction

Yelp has introduced a public dataset for academic use which includes businesses, users, reviews and tips etc. information generated by users of their well known web spoliation for travelers. The so called "Challenge Dataset" includes data from Phoenix, Las Vegas, Madison, Waterloo and Edinburgh, with more than 1M reviews, consisting of roughly 1 GB worth of data. Recently, large attention has been paid to information extraction from web data, opinion mining and review mining. How can we predict the user rating of a business based on the review textual elements alone? The main purpose of this paper is to build statistical models out of text data, which are then used for prediction tasks: specifically, we consider a classification problem of predicting the number of stars associated to a business review. Many modern use cases (Amazon, Facebook, Google, Twitter etc) rely on statistical models to offer better user experience: in the specific case of Yelp, being able to predict ratings opens up new applications, for example, Yelp could automatically prompt users to choose an automatic star rating for their review. The main difference between opinion mining and our user rating prediction problem is that we do not use sentiment analysis to assign positive or negative labels to the reviews: our statistical models are used to predict user ratings, that can vary from 1 to 5 stars.

Current existing methods [1-4] are not sufficient for user rating prediction due to the fact that they consider the sentimental analysis task as binary classification, while in our case it is multi-classification problem. Besides, most of the these works does not analyze the selected features. They tune the parameters to achieve a high accuracy and we only know that this feature combination can achieve this accuracy while we do not know exactly which feature contributes the most. In our case, we use some methodologies (Support Vector Machine, Naive Bayes and Tree-Based methods) that are believed to be the most accurate tools as our classifiers. We mainly focus on feature selection and appropriate parameter space exploration, which play a key role in prediction since we only use textual information for training or models. We study in detail error metrics and confidence to make sure our prediction is generic and our method has good generalization properties.

The main contributions of this paper include a detailed feature selection study (including BOW-Bag of Words), application of sound resampling methods, analysis of predicting accuracy and confidence, and result visualization.

II. Related Work

In 2009, Alec Go etc. introduced a novel method to automatically classify the sentiment of Twitter messages. They deployed Naive Bayes, Maximum Entropy, and SVM algorithms to achieve accuracy above 80% with emoticon data [1]. The authors of the paper [6] handled the prediction task as both a regression and a classification modeling problem. The experiment results suggested that classification techniques achieve better performance than ranking modeling when handling evaluative text. In [2], Alexander Pak. indicated that Microblogging websites are full of data for opinion mining and sentimental analysis. They also showed how to collect a corpus automatically and determine positive, negative and neutral documents by using a classifier. Efthymios Kouloumpis et. al. [3] evaluated the usefulness of existing lexical resources and features which capture information about the informal and creative language in 2011. Authors of paper [4] described a system for real-time analysis of public sentiment toward presidential candidates in the 2012 U.S. election as expressed on Twitter. They demonstrated that while traditional content analysis takes days or weeks to complete, this real-time system analyzes sentiment in the entire Twitter traffic about the election, delivering results instantly and continuously. More recently, In 2013, Julian McAuley etc. introduced a model by fusing latent review topics and latent rating dimensions for better product recommendation. It is a novel way to combine different sources for data mining. The work in [7] employed text analysis provided in SAS Text Miner to predict the overall and feature-based ratings for the online application reviews. These findings maybe helpful in promoting the sales of applications by better satisfying customer demands.

III. Methodology

The main workflow of our method is illustrated in Fig. 1. The review file provided by Yelp is in JSON format, and we can extract review text easily from it. After the data preparation stage, we extract several main features from the texts, these features are in different dimensions. Before we go forward to the part in which statistical models are combined, we shall discuss feature selection and normalization in order to keep features in certain data range. Then, we explain why Naive

* The first author and the second author have the same contribution to this paper

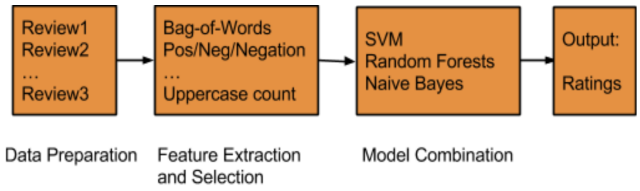

Fig. 1: The main flow chart of our process

Bayes, SVM and Random Forests have been chosen as our classifiers, and how we build an aggregate model to combine the outputs of different classifiers, in order to boost their individual performance.

The length of review texts in Yelp Dataset varies a lot, some of them have more than 200 words while some of them have only 30 words. In this case, its not easy to select some features which will represent every review text without bias. In our experiment, we extract 25 features from the raw text, in order to preserve as much information as possible. The features are shown in Table I.

TABLE I: Features used in our experiment

1	Strong Positive	14	Negative (WordNet)
2	Weak Positive	15	Character Count
3	Strong Neutral	16	Uppercase Count
4	Weak Neutral	17	Lowercase Counts
5	Strong Negative	18	Punctuation Count
6	Weak Negative	19	Alphabetic Count
7	Positive (WordStat)	20	Numeric Count
8	Negative (WordStat)	21	1-Star (BoW)
9	Negation	22	2-Star (BoW)
10	Length	23	3-Star (BoW)
11	Day	24	4-Star (BoW)
12	Vote	25	5-Star (BoW)
13	Positive (WordNet)		

Our features are extracted mainly based on some mainstream dictionaries. 4 dictionaries are totally used in our work. By using the first dictionary [8], we extract keywords from the original text. The keywords are among types: strong or weak, and its prior polarity, includes positive, negative, neutral. So totally six features are generated (features 1, 2, 3, 4, 5, 6 in table I).

The second dictionary is the WordStat Sentiment dictionary [9]. In this dictionary we can extract a list of positive words and a list of negative words. It contains more than 5500 words considered as positive and more than 9500 considered as negative words. These are the features 7,8 in table I.

The third dictionary is the SentiWordNet dictionary [10] . In our experiment, we use version 3.0.0. In this dictionary, one word can have positive meaning, negative meaning or even both. The degree of sentiment (how much positive or negative) is indicated by two real nonnegative numbers, PosScore and NegScore $(0 \leq PosScore + NegScore \leq 1)$, which correspond to positive level and negative level of a specified word. For other words, the values PosScore and NegScore are the positive and negative score assigned by SentiWordNet to its synset.

Due to the fact that the three predefined dictionaries above are in different formats, we need carry out preprocessing to extract the desired information and store it in a suitable format for later use.

We call the last dictionary we use the Bag of Words (BOW) dictionary. Unlike the traditional BOW model, in which one review is represented by a n-dimension vector (n is the number of distinct words in the whole dataset), every reviews vector in our bag of words just has 5 dimension, no matter how long the review is. Even though sometimes the word in this dictionary has no positive or negative meaning, the main advantage of this dictionary is that it adapts tightly to our dataset. The detail process is as follows. First we build 5 datasets which are: dataset containing one-star reviews, dataset containing two-star reviews, and so on. From these 5 datasets, we build 5 bags that represent 5 categories (1-star, 2-star, , 5-star.). The number of samples in each dataset is generated randomly. After we get these datasets, we generate n-grams to build our own dictionary. A n-gram is a contiguous sequence of n words from a given sequence of text. Empirical experiments show that larger n-gram provides better accuracy while require larger computational capacity. In our case, unigram, bigram and trigram are used in the experiment. For unigram, we select every single word from all reviews belonging to each dataset, and then put it into the bag that contains all the words in the same category. For example, we select all the words in 1-star dataset and then put it into the bag of 1-star, if one word appears more than once in the bag, instead of storing all of them, we just store it once. Along with the words, we store their frequency of appearance. We continue this process with bigram and trigram until we build the complete dictionary which contains 5 bags. One item (unigram, bigram, trigram) may appear in various bags. Finally we have a dictionary that associates every n-gram with its frequency of appearance in that bag (that category). In order to build feature histograms based on this dictionary, for every review we also generate all n-grams and their frequency, ending up with a vector in 5 dimensions ([a, b, c, d, e]), where a is the number of times this n-gram appears in the bag of 1-star, b is the number of times this n-gram appears in the bag of 2-star, and so on. To represent every review, we sum all vectors which represent all the n-gram belonging to that review. Finally, we obtain a 5-dimension vector for every review that represents the distribution that characterizes the review in terms of to 1-star, 2-star, ... reviews.

The other proposed features come from our assumptions. For example, we assume that the review with many votes is a good one and the longer too. That is the reason why we choose reviews votes and reviews length as features. The reviews day can also affect the user, if they make review on weekend and sunny day, the reviews rating may be better than in weekday and rainy day, even with the same person. We also have assumptions for other features. To verify our assumptions, we need to do experiment and compute some value to make sure these features are relevant to the ratings.

Based on the features extracted from these dictionaries and other features that we show in the table above, we build a histogram, that is a vector representation for each review text, and use it as the input for classifiers.

We can extract many features from review texts, but how can we know which feature is better than others? Which feature combination can achieve higher performance? Its maybe not a good idea to try features one by one and see which one can give the highest precision. We should complete this process in a more efficient way. Even though in this way we can find the best feature, but what is the underlying proof to support this result? Before we select our individual statistical models and delve into the details of their combination, we calculate the Mutual Information (MI) and Pearson Correlation Coefficient (PCC) between every feature and the user ratings which is the response we want to predict. The MI of two discrete random variables X and Y is defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log \frac{p(x,y)}{p(x)p(y)}$$

Where p(x,y) is the joint probability distribution function of X and Y, and p(x) and p(y) are the marginal probability distribution function of X and Y respectively. It measures the mutual dependence of variables.

PCC is a measure of the strength and direction of the linear relationship between two variables that is defined as the covariance of the variables divided by the product of their standard deviations.

I-Score: feature_importances_score - an attribute on the fitted model (tree-based model). This is an array with shape (n_features,) whose values are positive and sum to 1.0. The higher the value, the more important is the contribution of the matching feature to the prediction function.

In order to know which feature is more important than others in Yelp Dataset, we randomly select 1000 review text from the original dataset which includes more than 1 million review texts, and compute the MI, I-Score and PCC measures for each feature. The random process is carried out like this: we generate 1000 pseudo random numbers between 1 and the total number of review text in Yelp dataset by using sample function in random module. Table II shows the results.

TABLE II: MI, PCC and IScore between every feature and user ratings

NO.	MI	PCC	IScore	NO.	MI	PCC	IScore
1	0.0928	0.0904	0.0460	14	1.4622	-0.2398	0.0517
2	0.0943	-0.0570	0.0303	15	1.1787	-0.1460	0.0323
3	0.0504	-0.1083	0.0214	16	0.1798	-0.1264	0.0380
4	0.0648	-0.1370	0.0255	17	1.1032	-0.1430	0.0343
5	0.0935	-0.2794	0.0352	18	0.2191	-0.1341	0.0355
6	0.0728	-0.2406	0.0276	19	1.1266	-0.1434	0.0228
7	0.1359	-0.0568	0.0352	20	0.0602	-0.1406	0.0818
8	0.1457	-0.2698	0.0373	21	0.9918	-0.0133	0.0543
9	0.0757	-0.2576	0.0319	22	0.7934	0.1204	0.0622
10	0.6385	-0.1531	0.0324	23	0.6780	0.1198	0.0874
11	0.0027	-0.0017	0.0116	24	0.9904	-0.1389	0.0632
12	0.0449	-0.0266	0.0232	25	0.8340	0.1487	0.0228
13	1.4597	-0.0405	0.0461				

Based on the values in Table II, features [1,2,3,4,5,6], [7,8], [13,14] and [21,22,23,24,25] are combined individually, each combination come from the same dictionary.

Looking at the results, we can interpret and verify our assumptions above. For example, the feature #10 - review length has very high MI (0.6385), compare to feature #9 - negation (0.0757), but its not exactly #10 is more important than #9, because the variance of review_length is much higher than negation_count, hence its MI bigger is reasonable. In this case, the negation count also plays a very important role. The I-Score between them supports our explanation (0.0324 vs. 0.0319). So in order to conclude that which feature is more important than other, we need to consider many factors, and verify by doing the experiment.

Table III shows the classification accuracy by using SVM and Random Forests (RF) classifiers on several feature combinations. R2 score means the coefficient of determination in statistics. It can indicate how well data fit a statistical model. (The Wrong Rate in this case indicates how many predicted values are different from the original values, $|predicted\ value - original\ value| > 1$)

From table III we can find that normally Bag-of-Words features can give better performance in both SVM and RF cases.

In this part, the 1000 samples are sequentially processed in a 64-bit Ubuntu Laptop, which has 2Gb RAM and Intel Core i5 CPU (1.2 GHz).

IV. Experiments and Results Analysis

After we calculate the MI and PCC values for the selected features, we can visualize our analysis by using ROC (Receiver Operating Characteristic) curve and COS distribution curve. COS means the cosine similarity distance. ROC curve is a graphical figure that demonstrates the performance of a binary classifier system. This curve is generated by plotting the true positive rate against the false positive rate at various threshold settings. COS is computed as the cosine similarity distance between every two samples in each class. So if there are 200 samples in each category, there should be $200 * \frac{199}{2}$ = 19900 points in the COS figure. In order to show these points in a two dimensional figure, we set these distances as the Y axis, and the X axis is random values generated between [-0.5, 0.5]. The ROC curve and COS distribution of some combinations are shown in Fig. 2 and Fig. 3.

Looking at the COS distribution, we can know which combination is better than others. The more converging COS distribution to 1.0, the higher precision the combination yields.

In our experiment, we first tried to use regression models to predict user ratings, but in this case the output of user ratings are continuous, how to round the float value to int value is tricky. For example, the predicted user rating maybe is 3.4 star by using regression model, how can we decide whether it is 3 star or 4 star, or even 2 star, 5 star? If we label it directly to 3 or 4 star, however, in the training set maybe it belongs to 1 star indeed. This is a multi-class classification problem, it is different from regression problem, and much more complicated than binary classification. It needs an effective scheme to deal with this issue. So we investigate classification models instead of regression models to fulfill this prediction task.

After we finish feature extraction and feature selection process, we implement various classification models in scikit-learn

TABLE III: The classification accuracy by using SVM and Random Forests (RF) classifiers on several feature combinations

#	Feature Combination	SVM			RF		
		Accuracy	Wrong Rate	R2 Score	Accuracy	Wrong Rate	R2 Score
1	[1,2,3,4,5,6,9]	0.480	0.215	0.0003	0.350	0.305	0.2829
2	[1,2,3,4,5,6,9, 10,11,12]	0.430	0.205	-0.0393	0.405	0.225	-0.0393
3	[1,2,3,4,5,6,9, 15,16,17,18,19, 20]	0.470	0.225	-0.1186	0.435	0.230	0.0790
5	[7,8,9]	0.440	0.205	-0.0535	0.435	0.230	-0.0110
5	[7,8,9,10,11,12]	0.485	0.220	-0.2092	0.370	0.240	-0.3508
6	[7,8,9,15,16,17, 18,19,20]	0.400	0.210	-0.2262	0.400	0.215	-0.2574
7	[1,2,3,4,5,6,9, 10,11,12,15,16, 17,18,19,,20]	0.480	0.205	-0.1186	0.415	0.215	-0.2290
8	[7,8,9,10,11,12, 15,16,17,18,19, 20]	0.450	0.210	-0.1044	0.430	0.195	-0.0223
9	[9,13,14]	0.505	0.205	-0.0846	0.355	0.295	-0.3367
10	[9,10,11,12,13, 14]	0.425	0.205	-0.1356	0.345	0.245	-0.2574
11	[9,13,14,15,16,17,18,19,20]	0.480	0.200	-0.0563	0.405	0.230	-0.1413
12	[9,10,11,12,13, 14,15,16,17,18, 19,20]	0.430	0.195	-0.0563	0.385	0.225	-0.2205
13	[21,22,23,24, 25]	0.535	0.165	0.1504	0.490	0.185	0.0315
14	[10,11,12,21, 22,23,24,25]	0.460	0.190	-0.0790	0.445	0.175	-0.0450
15	[15,16,17,18, 19,20,21,22,23, 24,25]	0.495	0.180	0.0457	0.515	0.165	0.1533

open source library (http://scikit-learn.org/stable/) to carry out the prediction. We have a lot of reasons to choose scikit-learn: its robust, fast, easy to use, well-documented and permissive licenced.

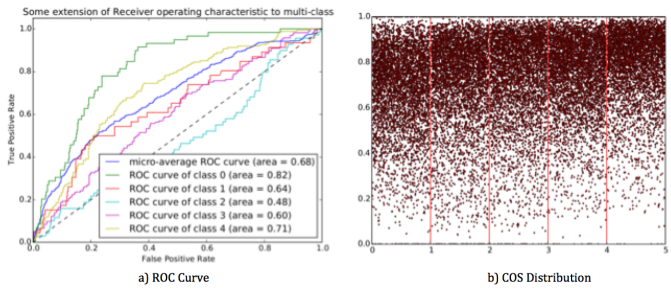

Fig. 2: Features [0,1,2,3,4,8]

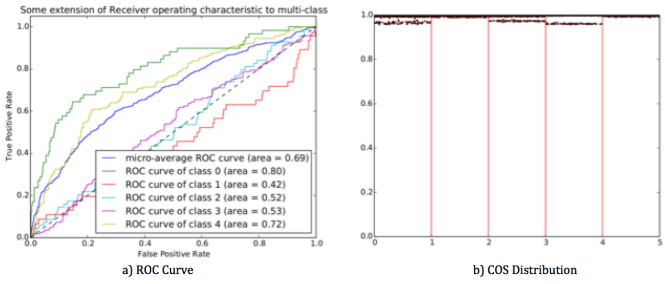

Fig. 3: Features [15,16,17,18,19,20,21,22,23,24,25]

The first model we have implemented is Naive Bayes. Because its very fast and easy to implement. After that we try SVM to compare the results. We see that Naive Bayes gives low accuracy with small amount of data, and it assumes independence of features while SVM gives higher accuracy. The main problem with SVM is that we can not interpret the result because SVM is a black box model. So we move to investigate tree-based method, which is simple and easier to

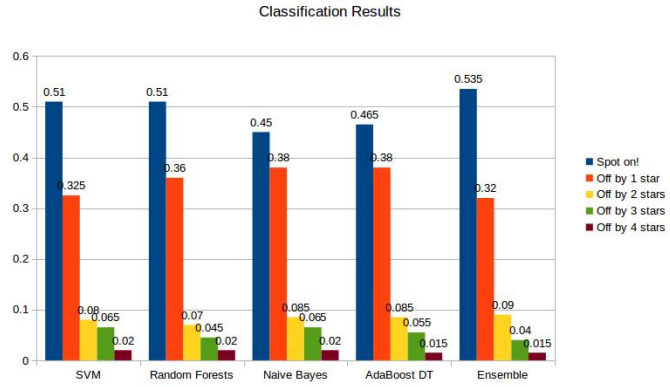

Fig. 4: Classification Results

interpret. We can also evaluate the features and visualize the tree so that non-expert user can understand. Decision Tree is a good candidate. With limited number of features and classes, Decision Tree performs very well and the results are easy to explain. We can evaluate the importance of every feature based on its position in the tree, or by the features important attribute in the fitted model (the I-Score above). The main disadvantage is that its easy to overfit. That is the reason why ensemble methods like bagging and boosting come in. They are general-purpose procedures for reducing the variance of a statistical learning method. Bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model. In other words, averaging a set of observations reduces variance. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting. Random Forests overcome this problem by forcing each split to consider only a subset of the predictors, provide an improvement over bagged trees

TABLE IV: Experiment Results

Classifier	Absolutely correct (Spot on)	Off by 1 star	Off by 2 stars	Off by 3 stars	Off by 4 stars	R2 Score	Running Time
SVM (kernel=rbf)	51%	32.5%	8%	6.5%	2%	0.122111463525	1s
Random Forests (n_ests=510)	51%	36%	7%	4.5%	2%	0.226891708201	<3s
Naive Bayes	45%	38%	8.5%	6.5%	2%	0.0796329859538	<1s
AdaBoost Decision Tree	46.5%	38%	8.5%	5.5%	1.5%	0.175917535116	3s
Ensemble (RF & SVM)	53.5%	32%	9%	4%	1.5%	0.275033982782	3s

by way of a small tweak that decorrelates the trees. Boosting works in a similar way with bagging, except that the trees are grown sequentially: each tree is built using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree fits on a modified version of the original data set. A practical implementation of boosting named Adaboost (Adaptive Boosting) is use in our experiment.

After trying with these single models, we want to improve accuracy by combining them together. There are a lot of strategies proposed. The most suitable one which we get by experiment is combining the best models (SVM and random forest) together as inputs for Logistic Regression model. By doing this, with the same dataset, the accuracy was increased in the acceptable time. The details are shown in table IV and Fig. 4.

V. Conclusion and Future Work

In order to provide a business overview, one solution is giving the business a star (such as from 1 to 5). However, this rating can be subjective and biased toward users personality. In this paper, we provide the framework to predict the user ratings by using the review text alone. We mainly contribute to the feature selection and model combination parts. We provide some statistical measures to evaluate the selected features and explain why these features affect the ratings. Our model can deal with multiclass classification problem, it can be adaptable with more classes. Besides, our model is generic; it includes both black box and white box models, SVM and tree-based methods; it can be extended to solve other sentiment analysis problem, or other classification problem easily, with high accuracy rate and interpretability. However, to understand more clearly the meaning of textual information, negation should be considered. Even though on this paper we have already used number of negation words in every review as a feature, negation is still complicated and it needs more time to investigate. Future work mainly concentrates on building more powerful features and negation words, exploiting new model combination schemes rather than the existing stacking, bagging and boosting strategies. As the dataset is huge and not feasible to be processed in a single machine, parallel and scalable process should be utilized.

References

[1] Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision

[2] Alexander Pak, Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining

[3] Efthymios Kouloumpis, Theresa Wilson, Johanna Moore. Twitter Sentiment Analysis:The Good the Bad and the OMG!

[4] Hao Wang, Dogan Can, Abe Kazemzadeh, Franois Bar, Shrikanth Narayanan. A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle.

[5] Julian McAuley, Jure Leskovec. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text.

[6] Narendra Gupta, Giuseppe Di Fabbrizio, Patrick Haffner. Capturing the stars: predicting ratings for service and product reviews.

[7] Tianxi Dong, Jonghyun Kim, Zhangxi Lin. Predicting Application Review Rating with SAS Text Miner.

[8] Theresa Wilson, Janyce Wiebe and Paul Hoffmann . Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. Proceedings of HLT/EMNLP 2005, Vancouver, Canada.

[9] http://www.provalisresearch.com/wordstat/Sentiment-Analysis.html

[10] http://sentiwordnet.isti.cnr.it/downloadFile.php