Prediction of User Appreciation of Yelp's Businesses Based on Text Reviews Hidden Features

Umberto Di Fabrizio
University of Illinois at Chicago
udifab2@uic.edu

Vittorio Selo
University of Illinois at Chicago
vselo2@uic.edu

Abstract

This work aims to predict the vote (i.e.stars) that an user gives to a business on Yelp. Each user is modeled by a set of his most important words, which are extracted from his reviews and his *tastes* are used to predict how much he will like a business. The contribution given by this work are: 1) A new framework is build to profile the users' tastes based only on the text in his reviews 2)The user profiling algorithm is used to predict the likelihood that the user likes a business. The method reaches 52% of accuracy revealing the effectiveness of the approach and encourages further investigation on this area, in particular combining this approach with classical high level features.

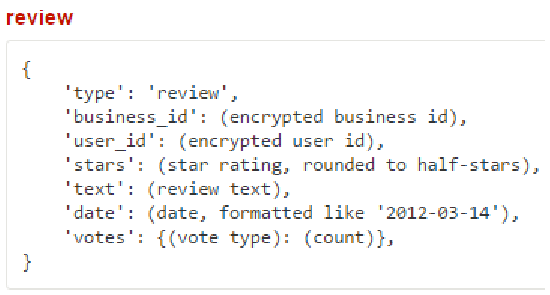
1 Introduction

Nowadays with the growth of websites that offer user-generated content (e.g. Yelp, IMDb, TripAdvisor) there is an increasing need, for companies, to extract the tastes of the users in order to make unique the user experience.
For this reason, recently, there have been a lot of researches in this area aimed to increase the effectiveness of the contents provided to the user (e.g. Netflix competition(**?**), Amazon, Yelp(**?**)). The traditional approach is based on a system that tries to predict the rating that a user would give to an item, this type of engines are called recommendation systems and have gained popularity in the recent years.
Generally to achieve this outcome the system is trained using high-level features, for instance in a movie platform those features would be: the rating of the film, the genre, the actors and so on.
Those characteristics are easily obtained from the database and are extensively used.
Our aim is to extract *hidden-features* of an user in order to understand their personal tastes and extract this information only by the text reviews. We call *hidden-features* those personal tastes of an user which can even be unconscious for the user himself. For example suppose a user writes reviews of mexican restaurants, why is one restaurant better than another although the food is good in both? Usually, people pay attention to details such as lights, atmosphere, the type of customers and so on, and sometimes without even being aware of their pet peeves.
The issue is <u>how</u> to find out, in an automatic way, those tastes that rule people feelings about a place (or an item) and <u>which</u> source to use.
The idea to address the user profiling by exploiting the text reviews has been used also by (**?**), anyway the usual approach is to use LDA to extract topics and then use a similarity measure to evaluate the distance between

items. Our approach tries to dig deeper in the user profiling, with the purpose to find those unique characteristics of the user and to suggest a business that he will surely like. To follow this idea we decided not to use LDA but to extract frequent words in a user reviews and use them as indicators for user tastes.
To extract the *hidden-features* we will mine the reviews of an user in order to collect the most common topics (furniture, lights, etc.) and understand what he really observes to judge a business (e.g. restaurant, pub, hotel). The hypothesis is that if an user always talks about certain topics then he cares a lot about that theme and this can guide the recommendation system.
Once the user hidden features are extracted we use them to train an ensemble of classifiers (SVM, Naive Bayes, Linear Regression) that learn how to predict the starts vote by those features.
We develop two methods: the first one is composed by one classifier that predicts one of the 9 possible classes (i.e. {1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5}) the second method is composed by three classifier which independently predict three subsets of the classes (i.e. {1, 1.5 ,2},{2.5 ,3 ,3.5},{4, 4.5, 5}) and then trains a Naive Bays classifier on the predictions of those three preceding classifiers.
The approach developed has the strength that it does not make any assumption on the type of reviews, user or platform, nor it make use of any specific domain-knowledge, this makes the framework highly flexible and ready to be used across different platform with no changes.

2 DATA

We have used the Yelp dataset(**?**) which is right now freely available because of the Yelp challenge that is currently ongoing.
The dataset is composed by 1.6M reviews and 500K tips by 366K users for 61K businesses and spans across several cities(Edinburgh, Karlsruhe , Montreal, Waterloo, Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison).
Because of the dimensions of the dataset and the computational power we have access to, we decided to limit our investigation to businesses in the area of Edinburgh(23780 reviews, 3150 users,4576 businesses). Anyway this work can be easily generalized to the all dataset or virtually to any dataset with the same info.
The format of the dataset is shown in fig 1.


review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

Figure 1: Yelp's Dataset review format.

3 PREPROCESSING

The reviews of Edinburgh businesses are tokenized using the tweeter tokenizer (keeps smilies ':)'), then we use the nltk package for python trained on the Penn Tree Bank dataset for the POS-tagging.
For our first approach we scan each user's reviews, we remove stop-words and detect the nouns that he uses finally for each noun we collect three features. Let M be the set of all users, R the set of all reviews. Given user m $\in$ M let r_m be the set of all reviews of user m then define X_m as the set of all nouns of the user m. Let n_{xmz} be the number of times that the word x belonging to X_m appear in review z $\in r_m$. Let s_{mz} be the rank that user m gave

to review z.
Now we can define:

- **Frequency**: frequency of that noun compared to the other nouns used
  i.e. how much the user talks about X?

$$f(x,m) = \frac{\sum_{i \in r_m} n_{xmi}}{\sum_{i \in r_m} \sum_{k \in X_m} n_{kmi}}$$

- **Regularity**: how constantly is that noun used in the reviews
  i.e. does the user talks about X in most of the reviews or not?

$$r(x,m) = \frac{\sum_{i \in r_m} n_{xmi} \frac{1}{\sum_{k \in X_m} n_{kmi}}}{|r_m|}$$

- **Relevance**: how influent is the noun to predict the rank?
  i.e. is X important to the final business score?

$$i(x,m) = \frac{\sum_{i \in r_m} n_{xmi} \frac{s_{mi}}{\sum_{k \in X_m} n_{kmi}}}{|r_m|}$$

As shown in the formulas above all the scores are normalized, in this way we avoid any biasing problem for the classifier.
Once we have those 3 features for each noun of a user we select the top 30 based on frequency and we intersect them with the top 30 based on regularity, this is because we want features that are frequent AND regular, finally we compute the union between those features and the top 30 based on relevance, this is because the influence of a feature to the rank is fundamental by itself. We run the same algorithm to collect noun and their features for the businesses, scanning all the reviews of a business. In the end for each user we have his *best nouns* with their relative 3 hidden-features and for each business we have its nouns with their 3 hidden-features. For each best noun of a user we look in the business best nouns to check if there is a a match, in this case the business values for that word are collected otherwise a default $[0, 0, 0]$ is assigned. In the end we have a vector:
[User best nouns][Business word in common with user]
for each user-business couple. In the end the length of the features vector is $2 * |\text{best user noun}|$, so the inputs to the classifier are between $(30 * 3) * 2 = 180$ and $(90 * 3) * 2 = 540$.

For our second approach we identified three subset based on the users' review stars [1-2], [3], [4,5]. For each subset we used two features: frequency and regularity and we ranked the features accordingly to the score:

$$S = log(frequency) + log(regularity)$$

This measure takes into account both features and avoid underflow. We extracted the 100 most important nouns (or less if there were not enough) for each of those subsets.
As in the first approach we applied the same algorithm to the businesses and then we extracted the business nouns in common with the user. The input to the classifier was again a vector of features combining user and business.

4 EXPERIMENTS SETUP

For our first method we selected all the user with more than 20,25,50,100 reviews and compared results. The random baseline is theoretically 1/9 = 11.1% but in our dataset there are not half stars reviews so the classes to predict are only 5 (i.e.{1,2,3,4,5}) which raises the random baseline to 20%. The majority class baseline (we always predict the majority class) is 44.9% which is really high. Fig 2 reveals how skewed is the curve of rankings on Yelp, in fact the number of 4 and 5 stars by them selfs are 69.2% of the all dataset. We divided the test between

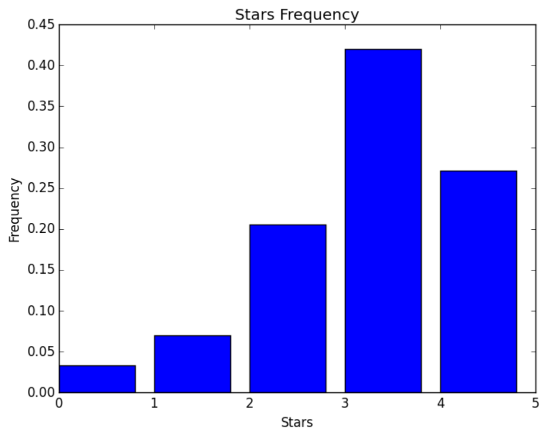


Figure 2: Distribution of stars of Yelp Edinburgh dataset.

train(80%), validation set (10%) and test (10%). For each user we trained an ensemble of classifiers.
The validation set has been used in a variation of the algorithm to choose the best classifier for each user among Naive Bayes, Random forest, SVM and Linear Regression.

In the second approach we selected users with more than 50 reviews and trained three classifier as follows (on the right the majority class baseline):

1. 1, 2, Other - 91.7%
2. 3, Other - 76.8%
3. 4, 5, Other - 46.3%

This is motivated by the fact that we need to discriminate very well between 4 and 5,and to avoid to bias the classifier towards the majority class.
Again it appears that the first classifier has an high accuracy, anyway it mostly predict the 'Other' class because of the unbalanced dataset.

Table 1: Accuracy based on number of reviews per user

MaxReviews	20	25	50	75
Accuracy	0.445	0.446	0.455	0.445

Table 2: Accuracy per class

Class	[1-2]	[3]	[4-5]
Baseline	91.7%	76.8%	46.3%
Our method	91.7%	79.5%	58.7%

5 RESULTS

The results for the first approach shows that 50 reviews are enough to profile an user, as presented in Table 1, this reduces the number of user selected to 72.
For the second method we achieved a significant improvement. Fig. 2 shows the comparison of each class with respect to its baseline; the approach cannot improve the accuracy of the class [1-2] but it outperform the baseline for the classes [3] and [4-5], especially the accuracy improves more than 12% for class [4-5].
Combining the three prediction with a Naive Bayes model we achieve an overall accuracy of 52% outperforming by more than 6% both the baseline and the first method.

5.1 Error Analysis

When dealing with recommendation system it is important to understand how much does the system fails when it does not predict the right class.
In the experiments we found out that the baseline majority class has an average error of -0.15, which means that it usually overestimate its prediction, moreover this is by itself a very low error.
Our first model has an average error of -0.10

which means that it still overestimates the stars but with less bias.
Finally our second model has an average error of 0.008 which represent the fact that the model is not consistently wrong on any direction and the error is unpredictable which is highly advisable for predictors, infact it means that there is not additional information that can be extracted from the data.

6 CONCLUSION

The work has presented two methods to profile the users' tastes from the text reviews. The comparison with the baseline shows that the second one increases the accuracy up to 52%.
This demonstrates that it is possible to extract user related information only from the text. The approach is very flexible and can be adopted to any web platform because it does not depend on any domain-specific knowledge.
The limitation of our approach is mainly in the fact that the dataset is very biased towards higher scores. For this reason we would like to extend our approach and try to consider a more balanced dataset both by applying stratified sampling or by adopting a different dataset which may be very useful to compare the performances.
We would also like to extend the noun extraction by extending each noun with its synonyms and/or hypernym so that the match between user nouns and business nouns can be increased.
Finally,the framework that we developed opens the way to several extensions, for instance the sentiment linked to a certain noun can be extracted so that the feeling towards the topic can be added to the features.

References

[Yelp] Yelp_Dataset www.yelp.com/dataset_challenge

[Netflix] Netflix_Challenge www.netflixprize.com/index

[sim] Aletras et al. Measuring the Similarity between Automatically Generated Topics,2014,Aletras at al.

[altroYelp] Hybrid Recommender System for Prediction of the Yelp Users Preferences,Vladimir Nikulin, 2014

[rev] Content Profiling from Text Reviews, Christophe Dupuy et al., 2014