Happy Hour Detection - Using tips from Yelp users to figure out the best deals at a time
CMPS 242 Fall 2016

Devashish Purandare DPURANDA@UCSC.EDU
Karthik Mohan Kumar KMOHANKU@UCSC.EDU
Priyanshu Sharma PSHARMA8@UCSC.EDU
Namrata Simha NPSIMHA@UCSC.EDU

Abstract

Stepping into a new place and knowing which is the most beneficial business setup to visit is a highly sought luxury. This project aims to do just that - direct the user towards hot deals that could be happening in their area. This could range from all genres of the industry, such as Happy hours pubs, free pizza slices restaurants, 40% off on clothing boutiques and many more. The recommender system kicks off by collecting user locations, fetching tips within 30 miles of their location radius, Naive Bayes classifying relevant tips from irrelevant ones and concludes by plotting the place, deemed most worthwhile by the classifier, on google maps.

1. Code

https://github.com/devashishp/Happy-Hour-Detection

2. INTRODUCTION

Recommender Systems are widely popular today. They take the level of convenience to greater and more accurate heights. This project implements a basic recommender system that focuses on extracting information from the yelp data set and learns from it to provide a best effort recommendation. The user is first asked to enter their location. The corresponding latitude and longitude information is extracted from google map APIs and passed onto the data extraction module. This module calculates a 30 mile radius around the coordinates obtained and extracts Business IDs falling under that region of interest from the Yelp Business Data Set. All the tips associated with these Business IDs are subsequently extracted. These tips then head to the Naive Bayes classifier module that classifies these tips as relevant or irrelevant. The features are also extracted in this module through a training data set. Initially, we create a list of 20 odd discriminative words such as free, $1, Discount and so on. Using this list we classify all the tips for a particular city as relevant or irrelevant and increase our bag of words by learning from the relevant classified tips.

When a new tip is encountered, the probability of each word being useful is calculated and cumulatively, we classify the tip as a whole. The tips classified as relevant are then matched with their Business IDs. The Business ID which has the highest number of positive and relevant tips is assumed to be more popular and consistent. This will be our recommendation to the user and will be plotted on a google map.

The content on this report are divided as follows: The Feature Engineering section talks about our feature extraction process and approach in great detail. Algorithm/Implementation section covers details about our implementation of the Naive Bayes algorithm, our approach and how we have extended it to fit our objective. Evaluation and Results section composes details of our results obtained and observations made along the way. We conclude with references to amazing sources which helped us concoct this project.

3. PROBLEM STATEMENT

To find the best location for the user to go to at a particular time or a place based on the tips given in the yelp dataset.

4. FEATURE ENGINEERING

We performed several steps to preprocess the data:

4.1. SQL database :

In the first step, we extracted all the data that we wanted - tips and business information and put it into a SQL

This paper is submitted as a part of CMPS 242 class work.

database. This offered us several advantages :

1. Very fast access to data, without the need to load it repeatedly in memory.

2. Ability to call different sections of data

3. Data anamolies and outliers are automatically eliminated or converted

4. Ability to run the program in an effiecient on commodity hardware – laptop.

5. The recommendation can be done with a simple SQL query like : `SELECT business_id FROM data WHERE (latitude BETWEEN '+Xmax+' AND '+Xmin+' ) AND (longitude BETWEEN '+Ymax+' AND '+Ymin+') "` Results to which are instantenous.

4.2. Pandas

We used `pandas` to deal with most of the data because `pandas` is natural for a table based dataset and can speed up a lot of functions

1. `pandas` offers direct methods to import and export `csv` or `json` files.

2. `pandas` allowed us to use lambda functions to map an entire column to another, easily calculating frequency and probability.

3. `pandas` also allowed us to vectorize and optimize code using functions such as intersection instead of loops and slicing instead of iteration. The whole code can be executed over the entire tips dataset (649,000 tips) in less than a minute.

4.3. Good Words and Bad Words

We initially filter the data into good set and bad set. Because we do not have any baseline to go by, we handpicked a few words and made them our prior with the expectation that our algorithm would pick other words.

All the words in a 'good' sentence were classified as good with a certain probability, with the same logic for bad words. This helps us calculate both probabilities and whichever is greater is selected as the case by the algorithm.

We do not limit the number of words that our algorithm goes through, and the number of features is the number of unique words. We did restrict our algorithm to US cities though, to reduce language issues.

5. ALGORITHM

The recommender system we implemented first takes in the users location and retrieves the coordinates from Google's database. These are then sent to the range finder API of ours to ascertain a diameter of 30 miles. The range thus obtained is then used to filter through our extracted Business data database file to get business IDs of all Enterprises in that circle. Using these IDs, we obtain all of their corresponding tips. Next we proceed to process these tips for valuable information.

Our first approach was to use LDA (Topic modelling) algorithm. But it did not align well with our use case. We then changed our problem statement to a classification problem. We are using Naive Bayes classifier to classify the tips as relevant tips and irrelevant tips.

We formed a small prior list of words we felt would most discriminatively signify positive and relevant tips. We then used this prior to classify all the tips we obtained from the city of Phoenix in Arizona. This forms our feature vector of useful words. Similarly, we maintain a feature vector of words not useful to us. These vectors exclude the frequently used stop words. We have managed to accumulate about 59000 words in this way. We hoped to gain a high accuracy by using high dimensional data set. When a new tip is encountered, we calculate its probability of being useful. This is done through Naive Bayes estimation:

6. IMPLEMENTATION

$$P(Useful|Word) = \frac{P(Word|Useful) * P(Word)}{P(Useful)}$$

$P(Word|Useful)$ is obtained through the counter value for that particular word in the useful vocabulary divided by the total number of unique words in the useful vocabulary.

P(Word) is obtained by the counter value of the word in either the useful or unuseful vocabulary divided by the total unique words in both the vocabularies.

P(Useful) is computed by dividing the total number of tips classified as useful by the total tips used as training data.

We do this computation on a logarithmic scale to avoid overflow. The probability of each word in the tip being useful is added and the tip as a whole is classified as useful if greater than the tip of the word being classified as not-useful. This process is repeated on all the tips that we filtered out based on the users location. We then rematch the useful tips with their respective Business IDs. We make a simple recommendation by assuming that the most popular business units will most likely be awarded with the highest number of positive tips by users. So, our highest endorsement goes with the Business Unit that has the most number of tips classified as useful by our classifier. We then extract its Latitude and Longitude coordinates and depict it on google maps.

7. EVALUATION

The service being a “Best-Effort” service, we consider the recall of suggestions more important than the precision. To come up with an specific criteria for accuracy is very hard in our case because the way we have selected the training data, it leaves for a large number of false-positives and missing entries in the training data. The algorithm performing well will still be deficient because of this.

We addressed this problem in two ways :

7.1. Recall

Recall is an important measure, because the success of any recommender system is actually measured by the number of relevant recommendations it gives. Because the ratio of happy hour tips to general tips is very small in the data, we boosted the probability a bit to make it a bit overzealous in recommending places, because in a best case service : some result is better than no result.

7.2. Confidence

But Recall does not tell us anything about the accuracy of the system. To have some measure, we came up with a measure of confidence : the smaller the difference between the probabilities of good and bad, the lesser the confidence the model has in its prediction. The most discriminatory words such as ‘deal’ meanwhile inspire a very high confidence.

8. RESULTS

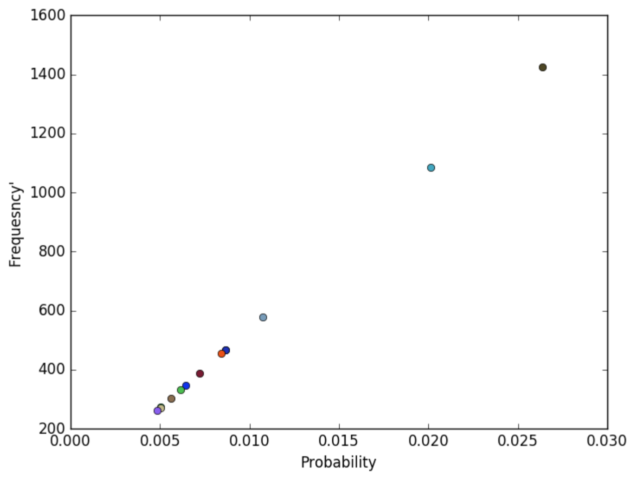
8.1. Discriminatory Words

These are the most discriminatory words for the set. These words appearing in the data almost invariably cause it to be marked as a useful tip.

Word	Frequency	Probability
free	1424	0.026371
day	1086	0.020111
great	580	0.010741
get	468	0.008667
sunday	467	0.008648
special	455	0.008426
happy	389	0.007204
friday	347	0.006426
$5	332	0.006148
good	304	0.005630
&	273	0.005056
hour	271	0.005019
saturday	262	0.004852

8.2. Sorting Tips

We select the closest businesses to a given location and get all the tips available for the businesses nearby. We can then sort tips into useful or not useful for our project. These are then compiled into a set of useful tips for each of the nearby businesses.



9. OBSERVATIONS

While our system shows some really good results, more is needed for a working system. Currently the system does return a lot of false positives. And while we argue about the best effort nature of the system, the system itself can be made much more accuracte by adopting a few measures.

The training data was not sampled as well as it could have, and this resulted in a few of false positives. The training data could be much better if each tip was manually evaluated and the number of endorsements of that tip were taken into account.

Another issue that we are facing is a tip which says something like ”crowded on Sundays” can mislead our system into thinking that it is saying something special about Sundays. This causes the system to mark it as a good deal even though it isn’t.

A simple way to fix this would be to run some sort of sentiment analysis on the tips, only looking for positive or non negative tips about a place rather than all tips. This, we believe would improve the recommendation engine.

Finally the amount of useful tips is very low which affects the probability of any tip being classified so. This could be boosted by handpicking the training data and as we did with our system, make the system a little overzealous in classifying something as useful.

10. FUTURE WORK

The work we presented is a basic recommendation system. We can further extend this by classifying tips and reviews based on sentiment analysis. We can incorporate CRFs to bring out the word dependency and improve our accuracy. We can extend its classification to strong, medium and weak tip/reviews by using multinomial logistic regression. Interface wise, we can map out the shortest path to our recommendation. We can also have multiple recommendations.

Acknowledgments

We would like to thank the TAs Params Raman and Andrei Ignat for helping us throughout the project and setting us on the right path when our project was too ambitious. We would like to thank Dr. Vishwanathan, his teaching and schedule ensured that we worked on this project very effectively. Finally we would like to thank the countless contributers on StackOverflow and other forums from where we got help whenever something went wrong.

References

1. https://github.com/googlemaps/google-maps-services-python
2. http://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification
3. https://docs.python.org/3/tutorial/index.html
4. http://scikit-learn.org/stable/modules/naive_bayes.html
5. https://www.cs.cmu.edu/ tom/mlbook/NBayesLogReg.pdf
6. https://documents.software.dell.com/statistics/textbook/naive-bayes-classifier
7. http://www.cs.ubbcluj.ro/ gabis/DocDiplome/SistemeDeRecomandare/Recommender_systems_handbook.pdf
8. Pattern Recognition and Machine Learning - Book by Christopher Bishop