W4201 Advanced Data Analysis

Classification and Analysis of Keywords of Yelp Reviews

Yanyu Zheng / YZ2690
Binglei Shao / BS2918
Bin Zhou / BZ2267
Zhe Li / ZL2421

Abstract

With the increasing popularity of Yelp and the huge amount of data it collects, the analysis of users' reviews has become an essential technique to understand customer behaviors. In this project, we focused on the classification and analysis of keywords of yelp reviews. The Yelp Challenge Dataset, which contains 2.2M reviews made by 552K users, is used in the project. In the first phase of the project, we built an adaboost model to predict the star rating of a review from the corresponding text. The model achieved 91.91% accuracy rate on a test set with 40K+ reviews and generated keywords in the process. In the next phase of the project, we analyzed the keywords in the sense of words frequency, part of speech and positive/negative effects. Through showing the power of the keywords in classifying reviews, this research shows that from 13.18% of the words we can detect 90%+ sentiment in the reviews.

1. Introduction

Problem of Interest

Each yelp review is consisted of a text comment and a star rating. This project focuses on predicting the star rating from the corresponding text. And after successfully building the model, we analyzed the keywords influencing the predicting power of the model.

Dataset

Source: https://www.yelp.com/dataset_challenge
Data Contains fields userID, reviews, friends, stars, etc. This project focused on reviews and stars for modeling and prediction.

After filtering out determiners, prepositions, interjections, etc, We converted reviews into word frequency matrix with each column being a word and each row being a review. For example, a review “Went to chipotle today. Food was good, service was good, too” will be converted to an entry below (counts for words that do not appear in this review will be recorded as 0):

	“went”	“chipotle”	“today”	“food”	“was”	“good”	“service”
count	1	1	1	1	2	2	1

Now we get a huge sparse matrix with thousands columns (Entries for most of the words are 0 since a sentence covers way less words than the entire dictionary). We represent each word as a variable, For example, the word “went” is x_1, “chipotle” is x_2, and corresponding star rating as y.

Organization of the Report

This report consists of model setup for svm, logistic regression and adaboost; model comparison of the above mentioned models; bias variance trade off in adaboost; error analysis and model improvement in adaboost; analysis of keywords in the sense of negative/positive, frequencies and part of speech allocation; conclusion. Also, please check the appendix for source codes and contributions of team members.

2. Model Setup

SVM

Since the dataset used has high dimensions and finding support vectors is quite time efficient in large dataset, the first model came into our mind was SVM. However, the result was quite disappointing. Given the fact that the original proportion of five star reviews is approximately 75%, an accuracy rate of 76.81% was unacceptable. As it turned out, svm on high dimensions has its own limitations.

Logistic regression

The second model we used was logistic regression, which we learned in class this semester. Basically, logistic regression is a generalized linear model that gives results between 0 and 1. And for binary classification problem, if the output is larger than say .5, we assign it to five star, vice verse. The result was disappointing again, we got 53% accuracy rate on test dataset, just a little bit better than random guess. And the reason behind it was probably overfitting on training dataset.

Adaboost

As presented above, SVM and logistic regression did not give a desirable outcome. So going forward, we tried adaboost for variable selection and classification. Adaboost uses decision stumps as weak learners, applies forward selection on them and assigns non-zero weights only to words that can improve the predictive power of the model. For a test set of size over 40,000, adaboost reached a 91.91% accuracy rate. 139 words are selected as influential on prediction. We further explored some interesting features of these words and detailed analysis is given in Analysis of Keywords section.

3. Model checking

Comparison of SVM logistic and adaboost

The major problem of SVM is that all of the 20,000+ variables are given the same weights in defining the distances between reviews, which brings the curse of high dimensions. And the large number of variables results in extreme over-fitting problem for logistic regression. Therefore, dimension reduction is needed. And here we applied adaboost to achieve that. Since adaboost selects only those features known to improve the predictive power of the model, it is more likely to give a reasonable result than unsupervised methods like PCA do.

Adaboost is a recursive method, so computation time is of essence here. However, the training process turned out to be faster than expected. The main reason was that during forward selection, most of the variables are kept out of the model and therefore saved time.

Bias variance trade off in adaboost

During the process of training the adaboost model, we selected maximum number of variables of 200 and learning rate of 1. The adaboost algorithm used in this project implements forward selection in training. So even if we set the threshold at 200, the algorithm only selected 139 variables in the final model. Thus making tuning this parameter unnecessary. Furthermore, there is actually a bias variance trade off between the learning rate and the number of variables selected. Since the larger the learning rate is, the more restrictions we are putting on each variable, thus resulting in more variables included in the model to achieve the same accuracy rate. And the smaller the learning rate is, each variable is more flexible, so less variables are needed in the model. In conclusion, the trade off between learning rate and the number of variables made it unnecessary to tune these two parameters. So we went on with the default parameters as mentioned.

Error Analysis and Model Improvement in adaboost

After taking a close look at the length and the business type of misclassified reviews. We found high correlation between the misclassification rate and the business type. Moving forward, there is possible improvement for the model if different models are built for different type of businesses.

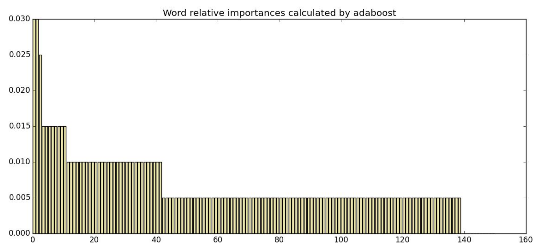
4. Analysis of Keywords

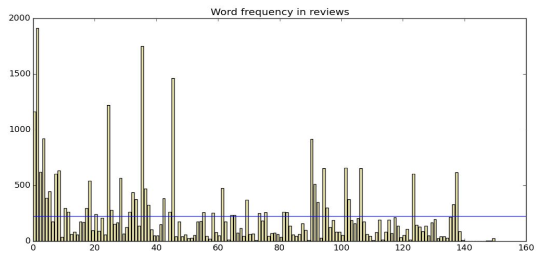
With the success of using adaboost on this classification problem, we were able to get the relative importances of the words as a byproduct in the process. The cover picture of the report, a word cloud in which the size of keywords are in direct proportions to their relative importance, gives a good overview. From the word cloud, it is easy to see that most words has strong positive or negative tendencies.

Negative or Positive

In this part, we listed part of the words that has positive or negative effect on the model. That is, if a certain review has this word in it, it will be more likely to be classified as five star or one star.
Positive: great love years needed selection best little amazing good just staff food business family want waitress etc.
Negative: rude terrible slow horrible left ordered told worst manager money etc.
Most of the words in positive and negative classes are commonly recognized positive and negative words. However, there are also mutual words in these two classes, such as family, waitress, manager and money. And we believe this is due to the sentencing habits of common users and the things certain words can remind us of. For example, family is often linked to happy family get together, while money and manager is more like a complaining customer asking the manager for refund.

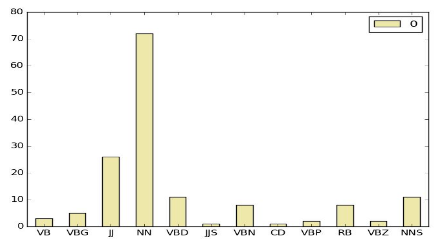
Frequencies

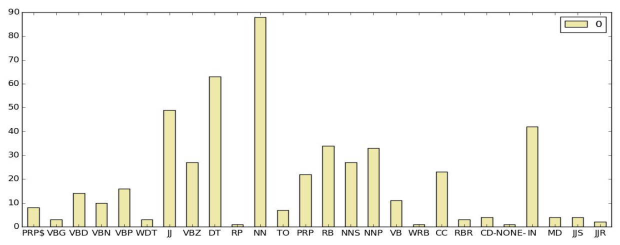




Moving forward, we tried to detect if there is correlation between the relative importance and the frequencies of the words. Here the left plot shows the relative importance of keywords from high to low while the right plot shows the corresponding word frequencies in the original reviews with a horizontal line showing the mean frequency of all words. There is no seeable correlation between these two criteria, so no further analysis was performed.

Part of Speech





Apart from frequencies, the roles keywords played in a sentence was also a topic of interest. Here the left plot is the part of speech plot of keywords and the right plot is the part of speech plot of the original review. As shown in the left plot, adjective or numeral (JJ) and noun(NN) are the two dominant components in keywords. Correspondingly, these two components also played important parts in the original reviews. Also, because we deleted words like determiners, prepositions, interjections, these components went missing in keywords.

5. Conclusion

In the classification step, we constructed an Adaboost model which performs well with over 90% accuracy rate and has an advantage over SVM and logistic regression. The selection of keywords suggested that 13.18% of words can represent more than 90% sentiments. Furthermore, by digging into the selected words, it is indicated that the sentiment tendency of keywords is consistent with common sense, the weights of the words has no obvious correlation with frequency, and adjectives and nouns are most related to sentiments.

Appendix

Contribution

Yanyu Zheng(YZ2690)
code(analysis of keywords/error analysis)
PPT
Presentation
report(Introduction/Abstract/Model Checking/Analysis of keywords)

Binglei Shao / BS2918
code(data cleaning/word cloud/error analysis)
report(Adaboost/Model Checking/Conclusion)

Bin Zhou / BZ2267
code(svm/logistic/word cloud)
report(SVM/Logistic)

Zhe Li / ZL2421
code(data cleaning/adaboost/word cloud)
report(Data/Code)

Code

```
#### Modules used omitted
#### Read in data, vectorize, split train and test
df = pd.read_csv('../data/review.csv')
dfUser = pd.read_csv('../data/user.csv')
vectorizer = CountVectorizer(stop_words='english')
# matrix is a n* 2000+ matrix, with each words represented by integer
matrix = vectorizer.fit_transform(df.text)
X = matrix
y = df.stars.as_matrix()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

#### Adaboost for 1 star and 5 star, predict for test set
bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),
n_estimators=200,learning_rate = 0.5)
bdt.fit(X_train, y_train)
z = bdt.predict(X_test)

#### Calculate error rate
print 'adaboost15'
print 'error: {}'.format(sum(z!=y_test))
print 'total test: {}'.format(len(y_test))

#### Build up dataset for plotting
vocab = vectorizer.vocabulary_ # vocab of all appeared words in corpus
importances = bdt.feature_importances_
index = range(len(importances))
count = np.array(matrix.sum(axis=0))[0]
length = np.array(X_test.sum(axis=1))
length = [item for sublist in length for item in sublist]
vocabNew = dict((value, key) for (key, value) in vocab.iteritems())
wordImptDict = {}
imptWordDict = {}
frequent = {}
#for value in index:
for ind,word in vocabNew.iteritems():
	wordImptDict[word] = importances[ind]
	frequent[word] = count[ind]
nMostImpt = 150
words = heapq.nlargest(nMostImpt, wordImptDict, key = wordImptDict.get)
words = [word.encode() for word in words] # array of important words
impt = [wordImptDict[word] for word in words  ] # array of importances
cot = [frequent[word] for word in words] # array of counts of words

#### bar plot of importances against words and words frequencies
%matplotlib inline
plt.figure()
plt.title("Word relative importances calculated by adaboost")
plt.bar(range(len(impt)), impt, color = "palegoldenrod")
plt.figure()
plt.title("Word frequency in reviews")
plt.bar(range(len(cot)), cot,color = "palegoldenrod")
plt.plot([0, 160], [np.mean(cot), np.mean(cot)])

#### plot part of speech allocation for keywords and original reviews
%matplotlib inline
tagged = nltk.pos_tag(words)
tags = [t for (w,t) in tagged]
letter_counts = Counter(tags)
dfWords = pd.DataFrame.from_dict(letter_counts, orient='index')
dfWords.plot(kind="bar",rot=0,color = "goldenrod")
tokenizer = RegexpTokenizer(r'\w+')
tex = [tokenizer.tokenize(sentence) for sentence in df.text[range(500)]]
tex = [item for sublist in tex for item in sublist]
%matplotlib inline
tagged = nltk.pos_tag(tex[0:500])
tags = [t for (w,t) in tagged]
letter_counts = Counter(tags)
dfWords = pd.DataFrame.from_dict(letter_counts, orient='index')
dfWords.plot(kind="bar",rot=0,color = "palegoldenrod")

#### Error analysis
%matplotlib inline
length = pd.DataFrame(length,columns = ['length of review'])
length['Categories'] = pd.Series(z==y_test)
pd.options.display.mpl_style = 'default'
length.boxplot(column = 'length of review', by='Categories')
```

```
#### analysis of misclassified samples
count = 0; misc = []
for i in range(len(z)):
    if z[i] != ynew[i]:
        count += 1
        misc.append(i)
rev3=rev2.ix[misc,:]
X_misc = vectorizer1.fit_transform(rev3.text)
y_misc = rev3.stars.as_matrix()
bdt_misc = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200)
bdt_misc.fit(X_misc, y_misc)
b=bdt_misc.feature_importances_.copy()
kwords=[]
for i in range(len(b)):
    if b[i] != 0:
        kwords.append(i)
newdict={}
for key,value in vocab.iteritems():
    if (value in kwords)==True:
        newdict[key]=value
ind=[]
for value in newdict.values():
    if value in kwords:
        ind.append(value)
B=b[ind]
df={'word':pd.Series(newdict.keys()),'weight':pd.Series(B)}
df=pd.DataFrame(df)
print df

#### Word cloud
wordcloud = WordCloud(stopwords=STOPWORDS).generate(text)
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
text =''
for review in badReviews:
            text = text+' '+review
wordcloud = WordCloud(stopwords=STOPWORDS).generate(text)
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```