Modelling the probability of a Yelp review being voted useful

Raymond Wong

1.Introduction

A quick check shows that about 52% of reviews were not voted useful. With a 'usefulness' predictor, reviewers will be motivated to produce reviews which are more helpful to readers. This project attempts to model a predictor for the probability that a new review will be voted useful(eg. will receive at least 1 useful vote) and identify the important features.

2.Method

2.1 Getting data

After looking through data file's schema, only review, user and business files contain relevant data. After data was initially loaded from the JSON file, it was saved as RDS file for faster subsequent loading. Nest data frame were flatten and nested lists were converted to new character fields.

There are 2 broad categories of data as listed below

Non textual: data about the reviews without refering to what were being written. Eg.

From business file - list of categories,stars From review file - votes.funny,votes.useful,votes.cool, stars, date, From user file - yelping_since, list of friends, fans, average_stars, list of years with elite status

Textual: data derived from review file's text field. These include character, word, sentence, paragraph counts and more advanced data from text mining and Part of Speech (POS) tagging.

2.2 Features Engineering

Some data are aggregated to better facilitate analysis. Some examples are:

isUseful is a binary data created by assigning reviews with 0 useful vote to 0 and all others to 1. This is useful for grouping data and required as outcome for machine learning.

reviewExperience is an integer field obtain by getting the months between the reviewer's 'yelping_since' date and the date of review. It represents how experience the reviewer is at the point of review.

starAgreement is a factor field obtain by taking the difference in star rating from previous review. Eg. if previous review had 2 stars and 4 stars was given for this review, the starAgreement is 2. 5 is assigned to first review. This is to test if agreement or disagreement in ratings will prompt useful votes.

matchScore is the number of matches a review text has against a list of terms which are more likely to appear in useful review based on text mining.

2.3 Data cleaning

NAs are replaced by 0 as 0 occurrence is the correct interpretation. eg. a review may not have all POS tags. NAs in missing tags were assigned to 0.

Business categories were reclassified to 20 major levels as original permutation of 8047 levels are too noisy.

There were cases where reviewers' yelping_since date are later then some of their reviews. This resulted in negative review experience. For such cases, the reviewers' first review date is taken as reference to compute reviewExperience.

2.4 Data Inference

2.4.1 Statistical Inference

To have a sense of how well the data features are able to separate the useful and non-useful reviews, t-tests was ran against several numeric features, take fans field as example:

```
t.result <- t.test(df.data[df.data$isUseful,c("fans")], df.data[!df.data$isUseful,c("fans")])

t.result$p.value
```

```
## [1] 0
```

Like the fans field above, many fields have a significant p-value. However, grouped fans by the isUseful field using table, it can be observed that the separation is not as significant.

```
head(table(df.data$fans,df.data$isUseful))
```

```
##
##      FALSE   TRUE
##   0 469602 237719
##   1 119148  91436
##   2  53968  51047
##   3  31802  36394
##   4  21769  27421
##   5  17598  22689
```

```
((469602-237719)/(469602+237719))*100 #largest separation
```

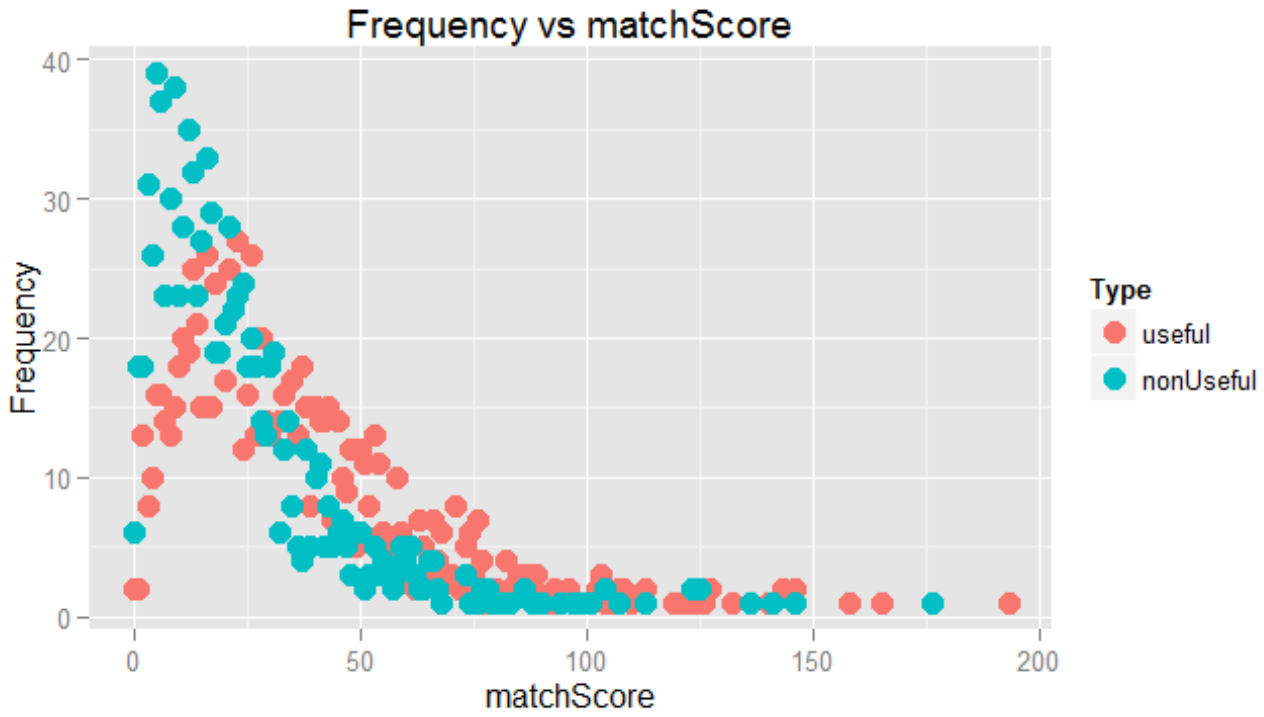
```
## [1] 32.78328
```

As shown above, the largest separation is only about 33%. This is a hint that the predictor's accuracy is at best in the 60%-70% range.

2.4.2 Data Visualization

Plotting the matchScore field of 1000 sample useful and nonUseful reviews(each) against its frequency, we get the following plot which imply a weak separation between the groups.

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```



As the above inference results all points to weak separation, we can only expect the prediction model to have moderate accuracy.

2.5 Machine Learning

We will use 3 most popular machine learning techniques: Random Forest, Gradient Boost and Naive Bayes to train the model. As we are prediction probability, we will be performing regression and the outcome, isUseful, will be an integer field.

3.Results

3.1 Model selection

A quick test with small dataset (11771 training and 3923 testing rows) show that all 3 techniques had similar accuracy but Gradient Boost has the shortest run time. Thus, Gradient Boost model was selected for further tuning.

Initial test also shows that POS tag data had a high error rate:

```
model_gb_POS <-readRDS("model_gb_POS.rds")
df.gb_POS_test<-readRDS("df.gb_POS_test.rds")

gb_tresult <- predict(model_gb_POS, df.gb_POS_test[,-c(46)])
```

```
## Loading required package: gbm

## Warning: package 'gbm' was built under R version 3.1.3

## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1.1
## Loading required package: plyr

## Warning: package 'plyr' was built under R version 3.1.3

RMSE(gb_tresult, df.gb_POS_test[,46])

## [1] 0.4751766
```

As POS tags are time consuming to create and they does not improve the model accuracy, we will drop the POS tags from subsequent tuning.

3.2 Model tuning

The model was tuned by increasing the traing and testing data to 23540 and 7846 rows respectively and adding a new feature, review_count(to represent users' experience in writing reviews). Unfortunately, the accuracy did not improve significantly and the error rate is still at 39.8%. The error rate and features of the final model was as follows:

```
gb_present <- readRDS("model_rf_20kmatchedReview.rds")
df.test <- readRDS("df.test20kFinal.rds")

gb_tresult <- predict(gb_present, df.test[,-c(12)])

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.1.3

## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.

RMSE(gb_tresult, df.test[,12])

## [1] 0.4025458

names(df.test)

##  [1] "review.Stars"      "votes.funny"       "votes.cool"
##  [4] "starAgree"         "review_count"      "fans"
##  [7] "user_avg_stars"    "frenCount"         "isElite"
## [10] "strcat"            "words.Count"       "isUseful"
## [13] "para.Count"        "revExp"            "sentence.Count"
## [16] "avg.word.Sentence" "avg.Char.Word"     "biz.Avg.Stars"
## [19] "diff.Biz.Stars"    "matchedScore"
```

4.Discussion

The final prediction accuracy of 59.7% was slightly lower than our earlier expectation of a moderate accuracy due to weak separation between the groups. This suggest the separation between groups is smaller than expected and we did a good set features to adequately answer the question.

This exercise illustrated that once the sample data used for training has reached a representative level, increasing rows of training data does not improve accuracy of the model. In this case, the accuracy of model did not improve after doubling the training data from 1% to 2% (eg. 11771 to 23540 rows).

This exercise also showed that adding weak predictor only improve accuracy marginally.

In this project, we tried modelling a generalized predictor for reviews of all business. It is pausible that different key terms will surface reviews for different businesses. Thus, modelling a predictor for a specified business may achieve a higher accuracy.

Finally the top 5 important features in this model are:

votes.cool 375.06

words.Count 185.98

votes.funny 153.36

matchedScore 149.30

frenCount 76.01

Thus, qualitatively, best way to get useful votes is to write something 'cool'. This is significantly more important that the other features listed.

Your review should have sufficient content as word.Count is the 2nd most important feature. Based on the summary result below, your review should at least have 126 (mean value) word.

```
summary(df.data$words.Count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    49.0    93.0   126.7   166.0  1047.0
```

Humor also plays an important part at 3rd.

Correct choice of words will improve communication with reader as indicated by matchScore field being the 4th important feature.

Finally, further away at 5th, having more friends will probably increase number of interested reader and lead to higher chance of useful vote.