Jeff Levesque

Professor Gates

IST 565

Homework #8

**Introduction**

As industry rapidly digitizes customer reviews, the need to distinguish genuine statements become important. How do customers, and industry accept that reviews were legitimate? The level of importance for these questions vary between organizations and may even incur financial consequences. Therefore, analyzing corresponding data, is an important practice.

As often found, many different techniques can be implemented to attain an optimal solution. For example, when choosing between naïve bayes, or svm a question could surface regarding which algorithm has a shorter runtime. However, choosing the right approach is often an art, and no solution is absolute.

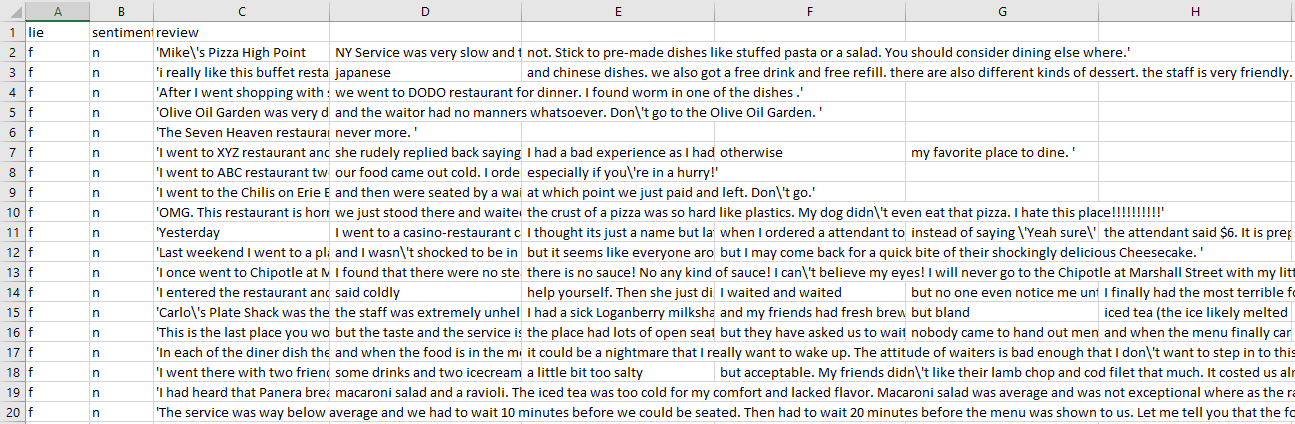
In this study, both naïve bayes, and svm’s will be implemented to determine if an accurate model can be generated for lie detection, and sentiment analysis for restaurant reviews. Techniques for dealing with sparse data will be implemented.

**Analysis**

Data Preparation:

A single dataset was represented twice with two different file formats. The first was an arff file format, native to the weka application, and the second a csv file. This provided an option to use either weka, or an alternative csv reader. Since custom R code was written, the csv file was used.

When inspecting the csv dataset, it was immediately noticed there was 92 records, while the number of columns being non-standard. More generally, the number of columns varied in a given row, between three, to any 3+n columns. This was because the review could split into any number of columns:



Therefore, the 3+n columns needed to be collapsed into a single review column. Therefore, the column name needed to be read separately from successive rows. However, initial attempts to load the provided dataset, produced incorrect number of rows. More generally, the corresponding dataframe yielded only 45 entries:

> filepath = 'data/deception\_data\_converted\_final.csv'

> df.colnames = read.table(filepath, nrow=1, stringsAsFactors=FALSE, sep=',')

> df.full = read.table(filepath, skip=1, header=FALSE, sep='\n', fill=TRUE)

> count.fields(filepath, sep='\n', blank.lines.skip=FALSE)

[1] 1 NA NA NA 1 1 NA 1 NA 1 1 NA NA NA NA NA 1 NA NA NA NA 1 1 1 1 NA NA 1 1 NA NA 1 1 1 1 1

[37] 1 1 1 1 1 1 1 1 1 NA NA NA NA NA NA NA NA 1 NA NA NA 1 1 1 1 1 1 1 1 1 NA 1 NA NA NA NA

[73] NA 1 1 1 NA NA NA NA NA NA NA NA NA NA 1 1 1 NA NA 1 1

To account for missing attributes, additional attributes for read.table was needed:

## import dataset

filepath = 'data/deception\_data\_converted\_final.csv'

df.colnames = read.table(filepath, nrow=1, stringsAsFactors=FALSE, sep=',')

df.full = read.table(

filepath,

skip=1,

header=FALSE,

sep='\n',

quote = '',

comment.char = ''

)

Then, the corresponding single column dataframe was exploded into exactly the lie, sentiment, and review columns:

out = stri\_split\_fixed(str = df.full[, c(1)], pattern = ',', n = 3)

df.split = as.data.frame(do.call(rbind, out))

colnames(df.split) = df.colnames

df.split$review = as.character(df.split$review)

Once the dataframe loaded, the feature set was reduced, to help decrease the sparsity of train, and test data. Specifically, a feature set of 1462 was reduced to 118. This was done by removing any column with a sum less than 0.3:

df.merged = df.merged[, colSums(df.merged) > 0.3]

After the dataset preprocessing completed, the text2vec package was used to tokenize a bag of words vocabulary, into a document term matrix. Then, the term frequency-inverse document frequency was applied to the document term matrix:

it\_train = itoken(

df.split$review,

preprocessor = tolower,

tokenizer = word\_tokenizer

)

vocab = create\_vocabulary(it\_train)

vectorizer = vocab\_vectorizer(vocab)

model\_tfidf = TfIdf$new()

dtm\_tfidf = model\_tfidf$fit\_transform(create\_dtm(it\_train, vectorizer))

**Results**

The naviebayes package was used to fit models for the lie detection, and sentiment classification cases. The generated resubstitution error indicates that the lie detection case was not accurate, while the sentiment classification was moderately accurate:

===========================================================

resubstitution error (lie detection)

===========================================================

[1] "class error: 0.516129032258065"

fit.nb.lie.class f t

f 8 7

t 9 7

===========================================================

resubstitution error (sentiment classification)

===========================================================

[1] "class error: 0.32258064516129"

fit.nb.sentiment.class n p

n 12 3

p 7 9

The gmum.r package for multiclass svm was initially attempted. However, the package was removed from the cran-r repository. Additionally, performing a git clone, then attempting to load the package within R produced errors. Specifically, the package could not be found and loaded into memory. Therefore, the e1071 package was used for svm classification.

Since the e1071 package implements svm model as a binary classifier, multiclass classification could not directly be achieved. Rather, a single class, could be compared against all other class (aggregated as a single class). The required number of one vs one comparison would be proportional to the number of classes in the system. Since the train dataset was reduced to 118 features, the process would require significant amount of work to train 118 different classifiers.

Instead of generating many different classifiers, just one classifier was computed for each case. Thus, the svm classifier was forced to attempt a binary classifier on multiclass situation. These limitations clearly identify the shortcomings of R, compared to other languages, such as pythons SVC classifiers.

The confusion matrix, and resubstitution error, was computed after the svm models were fitted. The results indicate poor performance for the lie detection, while producing better results for sentiment classification:

===========================================================

resubstitution error rate (lie detection)

===========================================================

[1] "class error: 0.548387096774194"

svm.lie.pred f t

f 4 4

t 13 10

===========================================================

resubstitution error rate (sentiment classification)

===========================================================

[1] "class error: 0.32258064516129"

svm.sentiment.pred n p

n 11 2

p 8 10

Finally, the FSelector package was implemented to determine the top20 classifiers, using the gain.ratio, and the chi2 functions:

===========================================================

gain ratio (sentiment):

===========================================================

[1] "sentiment (top 20): best" "sentiment (top 20): minutes" "sentiment (top 20): we"

[4] "sentiment (top 20): terrible" "sentiment (top 20): at" "sentiment (top 20): amazing"

[7] "sentiment (top 20): is" "sentiment (top 20): love" "sentiment (top 20): noodle"

[10] "sentiment (top 20): japanese" "sentiment (top 20): environment" "sentiment (top 20): some"

[13] "sentiment (top 20): your" "sentiment (top 20): need" "sentiment (top 20): bar"

[16] "sentiment (top 20): will" "sentiment (top 20): don" "sentiment (top 20): if"

[19] "sentiment (top 20): always" "sentiment (top 20): favorite"

===========================================================

gain ratio (lie detection):

===========================================================

[1] "lie (top 20): service" "lie (top 20): i" "lie (top 20): love"

[4] "lie (top 20): noodle" "lie (top 20): japanese" "lie (top 20): environment"

[7] "lie (top 20): some" "lie (top 20): your" "lie (top 20): need"

[10] "lie (top 20): bar" "lie (top 20): will" "lie (top 20): don"

[13] "lie (top 20): if" "lie (top 20): always" "lie (top 20): favorite"

[16] "lie (top 20): around" "lie (top 20): quality" "lie (top 20): much"

[19] "lie (top 20): friendly" "lie (top 20): definitely"

The top20 words were not identical between the gain.ratio, and the chi2. Some commonalities exist between the two methods, including words such “your”, “love”, “will”, “always”; while several other words differed, including “favorite”, “friendly”, “environment”. However, the earlier resubstitution errors could likely improve had either the gain.ratio, or the chi2 be used for feature selection, rather than only implementing earlier feature set reduction.

Lastly, svm model fitting slightly outperformed naïve bayes, while naïve bayes significantly outperformed model prediction:

===========================================================

performance (minutes)

===========================================================

[1] "fitting naive bayes (lie detection): 0.206180095672607"

[1] "fitting naive bayes (sentiment): 0.211071014404297"

[1] "fitting svm (lie detection): 0.161234140396118"

[1] "fitting svm (sentiment): 0.191520929336548"

[1] "predicting naive bayes, class (lie detection): 0.0508151054382324"

[1] "predicting naive bayes, prob (lie detection): 0.0596039295196533"

[1] "predicting naive bayes, class (sentiment): 0.0605859756469727"

[1] "predicting naive bayes, prob (sentiment): 0.0684049129486084"

[1] "predicting svm, (lie detection): 0.130943059921265"

[1] "predicting svm, (sentiment): 0.11139702796936"

**Conclusions**

While many different classification approaches exist, understanding the limitations of these methods are important. More generally, computation with natural language processing often involve large sparse data, which can quickly complicate model fitting, and prediction. Therefore, understanding which features of the data is redundant, and can be removed, is an important task.

Without knowing which features to remove, computing models can either be computationally expensive, or result in poor accuracy. Additionally, understanding the limitations of the tools being used, is another important dimension during analysis. In the case of implementing model fitting, it is crucial to understand whether a model is capable of a multi-classification. For example, performing a binary classifier on a multiclass case, would yield poor results.

In this study, techniques identified potential future improvements. These include removing redundant features prior to model fitting, as well as knowing the limitations of the tools being implemented. Therefore, this study proved to be a valuable exploratory analysis.