Mining and Learning of Tweets from *Hillary Clinton* and *Donald Trump*

Feng Zeng (fengzeng)
fengzeng@scarletmail.rutgers.edu

Jia Song (js1582)
js1582@scarletmail.rutgers.edu

I. Abstraction

In this project, we have 5722 tweets in which 2629 from *Hillary Clinton* and 3093 from *Donald Trump*. Text mining was first done to see the different features that *Clinton's* and *Trump's* tweets have. Unsupervised learning was tried first but no good result were achieved. Clustering result showed that both of them have very similar characteristics. Then 9 supervised algorithms (including "*Random Forest*", "*SVM*", "*Ada boosting*", "*GLMNET*", "*CART*", "*KNN*", "*PCA-LDA*", "*NNET*" and "*Naive Bayes*") were packaged into "*MegaLearn*". This package include training testing method and cross validation method. It was applied to the data set we have. Satisfying result can be achieved with the highest being "*GLMNET*" of 87.06% correct classification (or 87.50% with cross validation).

II. Data

We downloaded 5722 tweets data, among them, 2629 tweets of *Hillary Clinton*, 3093 tweets of *Donald Trump*. There are 28 features included in the data set like sender name, sender ID, where it was sent, time and so on. Among all these features, only text was used and processed to be the explanatory variables. User-names were used as response variable. We first processed the raw tweet data into *corpus*, cleaned, then converted to *document term matrix*. Sparse matrices were established and saved for memory purpose. Sparse matrices were converted to data frames to be input to the package to do classification.

III. Text Mining

A. Data Processing

All the steps of data processing in this text mining section were done by the functions in "*tm*" packages. The original text data were converted to *corpus* first.Then all characters were converted to lower case. URLs were removed. Anything other than English letters and spaces were removed to finish the cleaning process of the data. The build-in *stopwords* vocabulary was used with two additional ones: "*available*" and "*via*". "*Big*" was removed from *stopwords* vocabulary. Then extra white space were removed and the words left were stemed and words with same stem were completed to the same whole word. After all these steps, the final corpuses were converted to document term matrices to do further analysis like plots and clustering.

B. Data Mining

Frequent words were found out first(Figure 1, Figure 2). In *trump's* tweets, *make America great again* is the top topic in his tweets. Also, he really likes to mention himself. However, in *Clinton's* tweets, the thing she mentioned most is "*trump*". That bar is obviously longer than others. Beside this, the emphasis of her political views is "*women*", "*families*" and "*people*". "*women*" has always been the topic of all

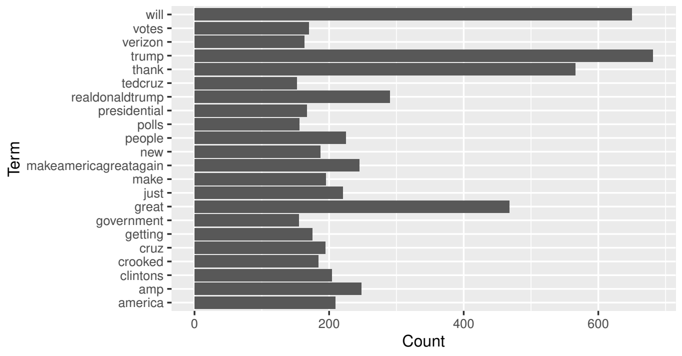


Fig. 1: Frequent words in Trump's tweets

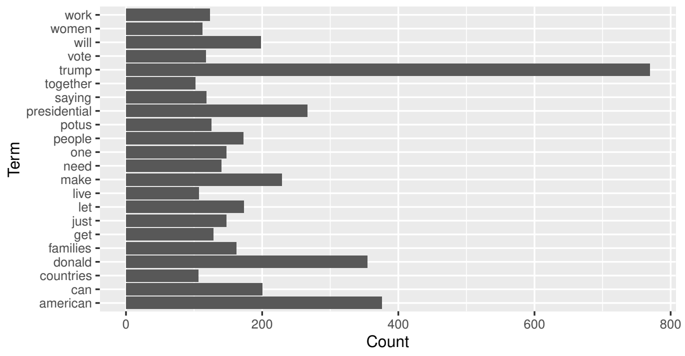
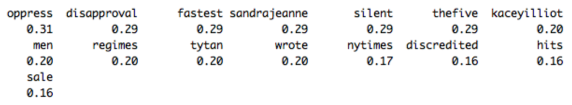


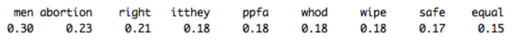
Fig. 2: Frequent words in Clinton's tweets

presidents. If we try to find the words that associates with "*women*" the most in *Trump's* and *Clinton's* tweets, the results are shown in figure 3 and figure 4. *Clinton* always showed her rigorous opinion of equal right, safety issues and abortion problems. However, when *Trump* talked about women, it seems to be the opposite. *Wordcloud* graphs were also generated to better visualize the topics in *Trump's* and *Cinton's* tweets. (Figure 5 and Figure 6)



```
 oppress  disapproval      fastest sandrajeanne       silent      thefive  kaceyilliot
    0.31         0.29         0.29         0.29         0.29         0.29         0.20
     men      regimes        tytan        wrote      nytimes  discredited         hits
    0.20         0.20         0.20         0.20         0.17         0.16         0.16
    sale
    0.16
```

Fig. 3: Words associates with "women" in Trump's Tweets



```
 men abortion    right   itthey     ppfa     whod     wipe     safe    equal
0.30     0.23     0.21     0.18     0.18     0.18     0.18     0.17     0.15
```

Fig. 4: Words associates with "women" in Clinton's Tweets

IV. Clustering

From the frequent words view, *Trump's* and *Clinton's* tweets seems to share some similar topics like talking about "*trump*". On the other hand, their political topics are quite different. So before heading to supervised classification algorithms, we are interested whether unsupervised clustering can separate them if we random mix all the tweets together.

Figure 7 shows the result of *kmeans* (k=2). They tends to fall into the same cluster. This might mean that the "*bag of word*" property is not that much different in their tweets by using the distance features only.

Figure 8 gave the result of *PAM*. It made 10 clusters, it seems that *Clinton's* and *Trump's* Tweets distribute similarly among all these 10 clusters. Which agrees with the result of *kmeans*, meaning that only using difference can be separate these two's tweets. We need other algorithms, maybe supervised to do classification.

V. Text Learning

A. Loading data

Through data processing, corpus of tweets of 549 words with suitable frequency threshold was produced as a *document term matrix*, and was saved as "HCDT.dat". The saved data file "HCDT.dat" of sparse matrix form (5722*550) was read into *R*, and converted into data frame for learning. The first 549 columns are the words counting, and the last column is the user-name of each tweet, either *realDonaldTrump*, or *HillaryClinton*.

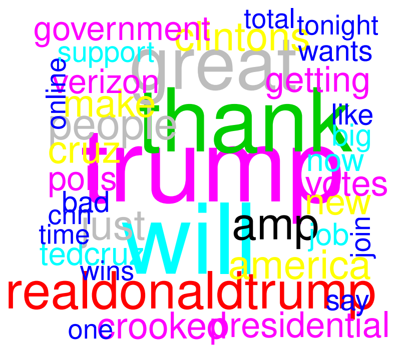


Fig. 5: Wordcloud of Trump's tweets

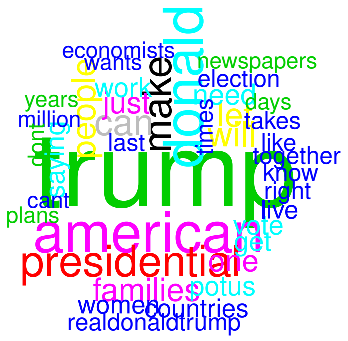
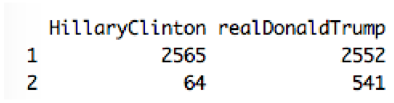


Fig. 6: Wordcloud of Clinton's tweets



```
  HillaryClinton realDonaldTrump
1           2565            2552
2             64             541
```

Fig. 7: kmeans result

B. Weight Calculation

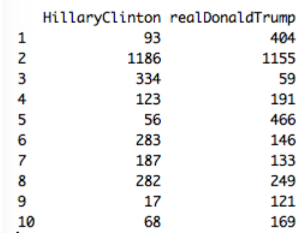
User can provide their own well-defined weight to be input parameter, while we also provide the weight function of two methods for calculating the weight, which depends on whether the variable to be an continuous variable or categorical variables. If the variable is continuous, we suggested using *anova* function to get *p* value; if the variable is categorical, we suggested using *fisher information* function to get *p* value. Once the user identified the property of the variables, and appropriate *pvalue* were calculated, and the *weight* function can calculate the weight using the library *qvalue*.

C. Splitting Data for Training and Testing sets

The `split_data` function can split the input data into training and testing sets by random sampling, under the specified split ratio. This function will be called inside the function *MegaLearning* to split the data before learning.

D. MegaLearning

MegaLearning function was designed to perform the simultaneous learning tasks using 9 supervised algorithems (including "*Random Forest*", "*SVM*", "*Ada boosting*", "*GLMNET*",

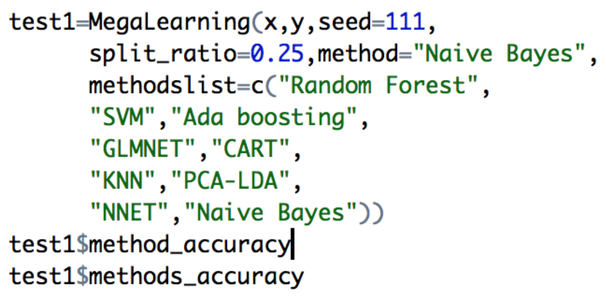


```
   HillaryClinton realDonaldTrump
1              93             404
2            1186            1155
3             334              59
4             123             191
5              56             466
6             283             146
7             187             133
8             282             249
9              17             121
10             68             169
```

Fig. 8: PAM result

”*CART*”, ”*KNN*”, ”*PCA-LDA*”, ”*NNET*” and ”*Naive Bayes*”). User need input the *split ratio* for creating training and testing sets. This function is also versatile to adopt main parameters for each algorithm. For example, for *GLMNET*, user can specify *glmnetalpha*, *glmnetfamily*, and *glmnet-standardize*. The structure of *MegaLearning* was shown in Fig.11.

For each algorithm, user can specify particular parameters, or the default parameter will be used. We applied the *MegaLearning* function to the tweets data(5722*550),using all nine algorithms, and specifying outputting the model of *Naive Bayes*. The coding was captured in Fig.9. All other parameters were under default setting, and the result was in Fig.12, which is a table-like representation. Among the results, *GLMNET* gave the best accuracy rate, while *PCA-LDA* had the worst performance, which might be the reason of the multinomial distribution is naturally suitable for *GLMNET*, while *PCA* might not be good for the sparse representation of document term matrix.



```
test1=MegaLearning(x,y,seed=111,
      split_ratio=0.25,method="Naive Bayes",
      methodslist=c("Random Forest",
      "SVM","Ada boosting",
      "GLMNET","CART",
      "KNN","PCA-LDA",
      "NNET","Naive Bayes"))
test1$method_accuracy
test1$methods_accuracy
```

Fig. 9: MegaLearning Test

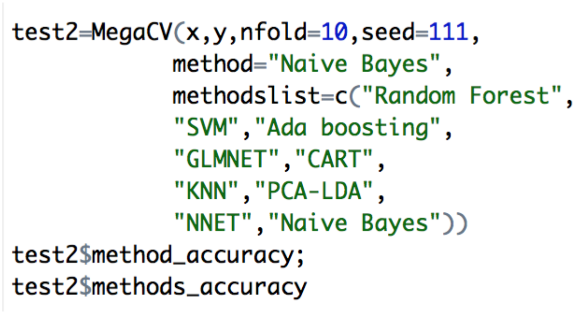
E. *MegaCrossValidation*

Similar to *MegaLearning*, the *MegaCV* function can also perform the learning tasks with different algorithms simultaneously, while with cross validation procedure. User can specify the parameter of *nfold*, and the default value for *nfold* is 3. The function can return *n=nfold* fit models for the specified method. User can further using these models for prediction of new data, and the majority vote can be regarded as prediction result. The structure of *MegaCV* was presented in Fig.13.

The learning of tweets data was also performed with *MegaCV*, the coding was captured by Fig.10. And the result was presented in Fig.14. The result from *MegaCV* is consistent with the result of *MegaLearning*.

VI. R Package: MegaLearn

MegaLearning, *MegaCV*, as also as the *weight* function was compressed into the R Package *MegaLearn*. User can install this package and its related dependency, then user can directly call function *MegaLearning* and *MegaCV*to perform learning tasks.The detailed user manual of this package will be supplied in a separated PDF file.



```
test2=MegaCV(x,y,nfold=10,seed=111,
             method="Naive Bayes",
             methodslist=c("Random Forest",
             "SVM","Ada boosting",
             "GLMNET","CART",
             "KNN","PCA-LDA",
             "NNET","Naive Bayes"))
test2$method_accuracy;
test2$methods_accuracy
```

Fig. 10: MegaLearning Test

VII. Conclusions

In this project, we utilized both unsupervised method(*TextMining*) and the supervised method(*TextLearning*) to analyze the tweets of *Hillary Clinton* and *Donald Trump*. The tweets of them show obvious different patterns of using different words, and the wrapped-together functions *MegaLearning* and *MegaCV* successfully performed the simultaneous learning tasks, among the learning results, *Random Forest*, *SVM*, *Ada boosting*, *GLMNET*, *NNET* achieved the accuracy rates above 80%.

VIII. Appendix

TABLE I: Method/function and their dependency in the MegaLearn Package

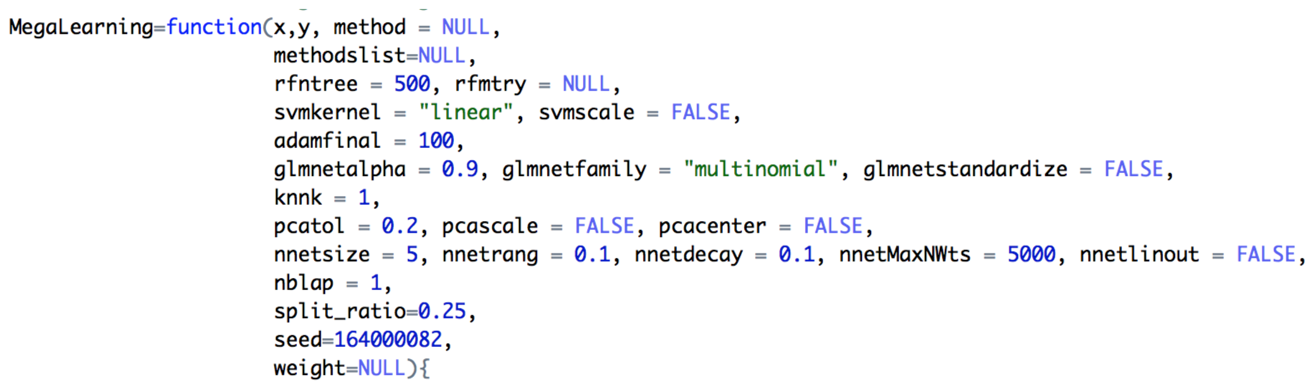
Method/Procedure	Library Dependency			
Data Processing	tm			
Text Mining	wordcloud	graph	Rgraphviz	fpc
Load data	e1071	SparseM	methods	Matrix
Weight function	qvalue			
randomForest	randomForest			
SVM	e1071			
Ada boosting	adabag	rpart	mlbench	caret
CART	rpart			
GLMNET	glmnet	Matrix	foreach	slam
KNN	class			
PCA-LDA	MASS			
NNET	nnet	caret		
Naive Bayes	e1071			

References

[1] http://www.rdatamining.com/examples/text-mining

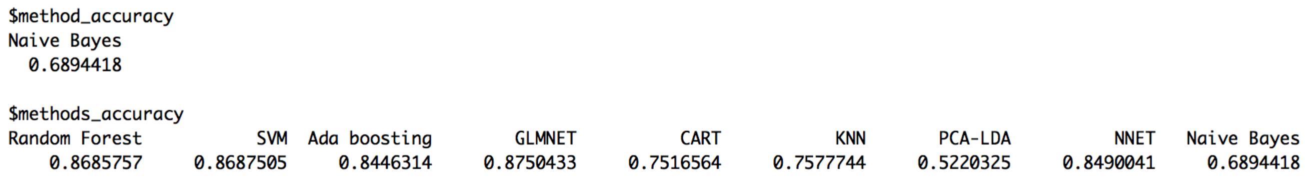
[2] http://varianceexplained.org/r/trump-tweets/

[3] http://www.rtexttools.com/



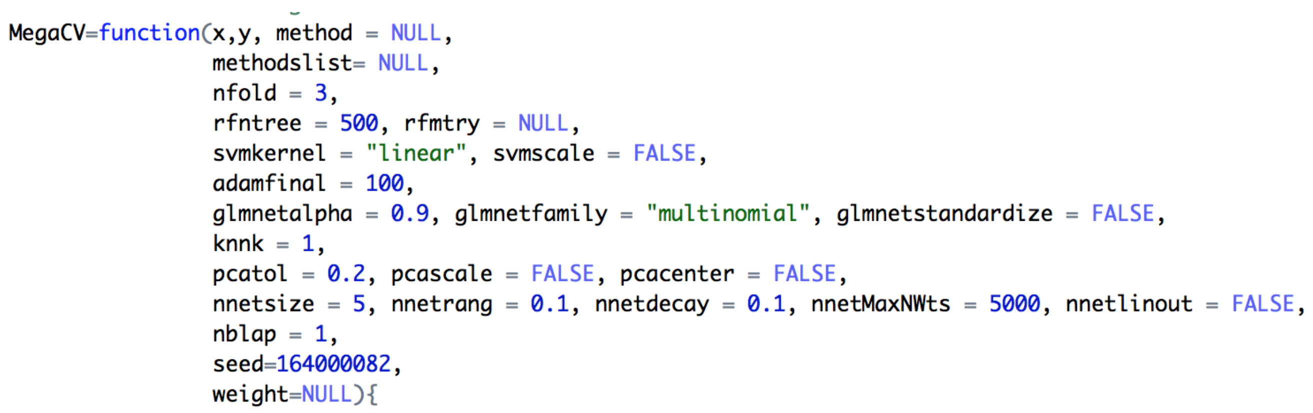
```
MegaLearning=function(x,y, method = NULL,
                      methodslist=NULL,
                      rfntree = 500, rfmtry = NULL,
                      svmkernel = "linear", svmscale = FALSE,
                      adamfinal = 100,
                      glmnetalpha = 0.9, glmnetfamily = "multinomial", glmnetstandardize = FALSE,
                      knnk = 1,
                      pcatol = 0.2, pcascale = FALSE, pcacenter = FALSE,
                      nnetsize = 5, nnetrang = 0.1, nnetdecay = 0.1, nnetMaxNWts = 5000, nnetlinout = FALSE,
                      nblap = 1,
                      split_ratio=0.25,
                      seed=164000082,
                      weight=NULL){
```

Fig. 11: MegaLearning Function Structure



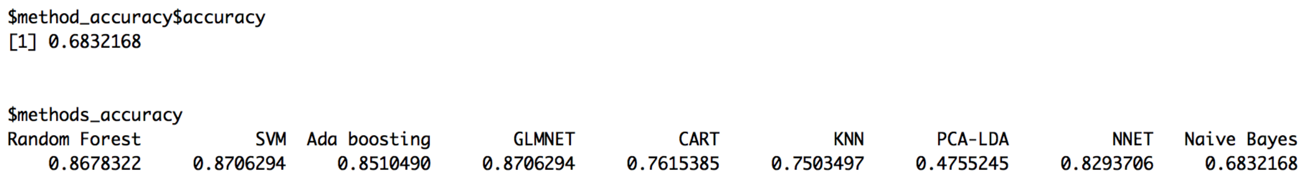
```
$method_accuracy
Naive Bayes
  0.6894418

$methods_accuracy
Random Forest           SVM  Ada boosting        GLMNET          CART           KNN       PCA-LDA          NNET   Naive Bayes
    0.8685757     0.8687505     0.8446314     0.8750433     0.7516564     0.7577744     0.5220325     0.8490041     0.6894418
```

Fig. 12: MegaLearning Result



```
MegaCV=function(x,y, method = NULL,
                methodslist= NULL,
                nfold = 3,
                rfntree = 500, rfmtry = NULL,
                svmkernel = "linear", svmscale = FALSE,
                adamfinal = 100,
                glmnetalpha = 0.9, glmnetfamily = "multinomial", glmnetstandardize = FALSE,
                knnk = 1,
                pcatol = 0.2, pcascale = FALSE, pcacenter = FALSE,
                nnetsize = 5, nnetrang = 0.1, nnetdecay = 0.1, nnetMaxNWts = 5000, nnetlinout = FALSE,
                nblap = 1,
                seed=164000082,
                weight=NULL){
```

Fig. 13: MegaCV Function Structure



```
$method_accuracy$accuracy
[1] 0.6832168


$methods_accuracy
Random Forest           SVM  Ada boosting        GLMNET          CART           KNN       PCA-LDA          NNET   Naive Bayes
    0.8678322     0.8706294     0.8510490     0.8706294     0.7615385     0.7503497     0.4755245     0.8293706     0.6832168
```

Fig. 14: MegaCV Result