Twitter Users' Interests Analysis

BT2101 Project Final Report

Bao Jingxian A0115420Y
Shen Lin A0116453L
Tang Jiahui A0119415J
Yuan Zihe A0115569X
Zhang Lei A0093966L

ABSTRACT

This project utilizes text mining tools to tokenize text and train classifier. By applying classification models on the testing data, we are able to evaluate the performance of the model generated before. After that, twitter users' timeline can be retrieved to analyze their interests. Besides, more complicated applications such as recommended group and common interests can be achieved.

Table of Contents

1. Introduction

Twitter is one of the most popular microblogging service where user can send and read short character messages (called tweets). These tweets express opinions about different topics. It is often hard to understand what these trending topics are about. The goal of our work is to automatically classify incoming tweets into different categories so that users are not overwhelmed by the raw data. This is particularly useful because it is directly showing users the topics that the twitter user concerns and filter the tweets of wanted topics, especially when Twitter is accessed via handheld devices like smart phones (Sriram, B. et al., 2010). Therefore, it is important and necessary to classify these topics into general categories with high accuracy for better information retrieval (Lee, K. et al., 2011).

1.1. Problem

The world is overwhelmed by the raw data and this may cause people to face problem in accurate information retrieval. Therefore, it is necessary and important to classify all information into proper categories and hence assist people in finding the most desirable and accurate information. In particular, Twitter, the popular online social networking service that enables users to send short 140-character messages and currently has more than 284 million monthly active users and 500 million tweets sent per day (Twitter,Inc, 2014), contains too much raw data which are not sorted or categorized. The social networking service provides a platform for people to interact with each other by following, commenting and re-tweeting. Hence, it is ideal for Twitter users to know the interest of others before following them. However, it is impractical to look through all his tweets if he is an active user. Thus, we see the problem of raw data.

1.2. Application and benefits

To predict the interests of particular twitter user by classifying his tweets into distinct categories (topics). This provides twitter users a clear picture of the main categories of their tweets. Since Twitter account is public, so this application could also allow Twitter users to have a general understanding of others by viewing their area of interests and hence facilitate them in following people with similar interests. On top of that, when following someone, users can choose to follow only the tweets of selected categories; therefore only tweets that are related to users' interests will be displayed on their homepages. The system helps to categorize information and allow users to filter unwanted tweets.

2. Data

2.1. Collecting Data from Twitter

Since we need to build model to categorize the tweets of a user, we are collecting our data from Twitter using Twitter API. Firstly, we need to use our Twitter account to create an application before working with the Twitter API. Application name is needed to be distinct. After creating application, unique consumer key and consumer secret will be generated for users. Then, we have to load two libraries namely twitteR and ROAuth and complete authorization by entering the PIN given.

2.2. Training Data

We obtain our data in two different manners.

At first, we predefined 18 possible categories that can be used to describe all tweets in Twitter, which are namely arts & design, books, business, fashion, food & drink, health, holidays, entertainment, music, politics, religion, science, sports, technology, TV & movies, environment, social and travel. We assume that any tweet found in Twitter can fall into exactly one of the categories. After defining the categories, we use Twitter API to collect 5000 most recent tweets related to each class by searching hash tags. For example, we collect our data under 'technology' class by searching any tweets with hashtag '#technology'. Hence, after collecting 18 sets of 5000 data, we exam the quality of data collected by looking through most tweets obtained. However, we found that a large portion of tweets collected are retweets (repost other user's tweets). This would result some of our predictors (tokenized words) stronger than others which reduces the variety of predictors. If the predictors in the repeated tweets are irrelevant, this would reduce the quality of our training data and hence the model built on it. Moreover, we found that unnegligible number of users simply use the hashtag without tweeting about the topic. This would detrimentally affect the accuracy of our model built as the predictors are not in good qualities and cannot be used to train the model. Last but not least, the data size is too large that it takes too long for us to build the model. Hence, for practical reason, we decided to discard this set of data and obtain new data using different criteria.

After reflecting on the weakness of our previous data collection, we decide to use another method to obtain the training data. We narrow down our total categories to 9 to reduce the overlap between each other, namely politics, news, music, technology, arts, fashion, businesses, food, sports. Instead of searching the tweets based on hashtag, we obtain our training data by retrieving tweets from users who

are expert or professional in that particular category. Due to their professionalism in the area, the quality of our data collected is ensured and hence the accuracy of our model. The professional accounts we choose are shown in the table below. To further improve the quality of training data, we remove the URL (http ://*) present in the tweets as each URL would be tokenized to one word and takes up our memory space but does not contribute to our model built. Due to practical reason, we sample the training data from all data collected. Hence, our final training data consists of 9 sets of 1500 data with all their categories identified.

	Twitter accounts			
Fashion	@Burberry	@fashion	@womensweardaily	@Fashionista_com
	@TimesFashion	@InStyle	@voguemagazine	
	@CHANEL	@MichaelKors	@NYTFashion	@ELLEmagazine
Business	@Forbes	@HarvardBiz	@businessinsider	@WSJbusiness
	@TheEconomist	@BBCBusiness	@nytimesbusiness	@FoxBusiness
	@WSJ			
Politics	@PostReid	@postpolitics	@politicalwire	@nprpolitics
	@HuffPostPol	@foxnewspolitics	@daveweigel	@BuzzFeedPol
	@BBCPolitics	@AP_Politics		
Sports	@BBCSport	@darrenrovell	@beINSPORTSUSA	@Deadspin
	@NBCSports	@NYTSports	@universalsports	@SportsCenter
	@OnionSports	@YahooFantasy		
Technology	@BBCTech	@Gizmodo	@NASA	@newscientist
	@ScienceNews	@technology	@physorg_com	@TechCrunch
	@PopSci	@techreview		
News	@BBCWorld	@CdnPress	@BloombergNews	@FoxNews
	@news	@ABC	@ChannelNewsAsia	@LatestAusNews
	@nytimes	@STcom	@TheSunNewspaper	
Arts	@bbcarts	@nytimesarts	@MusemModernArt	@Americans4Arts
	@TheStreetArts	@DesignMuseum	@DesignMuseum	@walkerartcenter
	@mfaboston			
Food	@FoodNetwork	@Food	@foodandwine	@ItsFoodPorn
	@nytfoodfeed	@bonappetit	@YourFoodPorn	@thefoodsection
	@WholeFoods			
Music	@billboard	@TwitterMusic	@SonyMusicGlobal	@amazonmusic
	@sonymusicsg	@iTunesMusic	@beatsmusic	

Fig 2.1 Nine categories of Twitter accounts selected as training data

2.3. Testing Data

After obtaining our training data by sampling 1500 from each category, we also need to have our testing data to prove the accuracy of our model build based on training data. Thus, we need to use tweets with category identified to test the model instead of tweets without identified category. We can put the testing data into our model and compare the category returned by the model and its pre-identified model to show the accuracy of our model. Hence, we decide to use the rest of data collected from professional accounts that are not used in model training. Firstly, this ensures the data integrity as the testing results are not biased for this model. Secondly, this approach is efficient since we do not need to read through selected tweets and categorize them manually. Lastly, the accuracy of our final conclusion is ensured as we obtain testing data from professional accounts, hence the categories pre-identified are highly possibly correct.

3. Approach

The business analytics model we used to address this problem is classification, in order to predict twitter users' interests based on labeled data. The data origins and its retrieval process are described in the previous section. To be specific, we will briefly introduce how we utilized the training data and text classification tools to build the classifier.

3.1. Text Pre-processing

After doing simple random sample from each predefined interest category, we used *rbind* function to binding all the extracted data by row.

At text pre-processing stage, we apply the training data into *create_matrix* function, which is a major text tokenization function in RTextTools library. It is able to help us generate a feature vector with all the frequent words. By default, the language for analysis is English. Besides, word stemming and stopword removal are adopted in the tokenization process. Numbers, punctuation and sparse terms are also removed in our feature vector, making the final matrix more condensed and relevant. Eventually, in our model, the feature vector is a large matrix with 9000 variables and 749 predicting words.

The related R code is shown below.

```
matrix <- create_matrix(total[,1], language="english",
    removeStopwords=T, removeNumbers=T,stemWords=T, toLower=T,
    removePunctuation=T, removeSparseTerms=0.998)
mat <- as.matrix(matrix)
```

3.2. Classifier Training

After successfully building the feature vector, we compared different classifiers and selected a classifier model. Support Vector Machine (SVM) and Naive Bayes (NB) are chosen as our classifiers.

SVMs are supervised learning models that are used for classification and regression analysis. Given a set of labeled training examples, an SVM training algorithm is able to build a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Naive Bayes is a family of simple probabilistic classifiers based on applying Bayes' Theorem with strong independence assumptions between the features (Aaron, 2014). It has advantage that it only requires a small amount of training data to estimate the parameters necessary for the process of classification. As variables are assumed to be independent, only the variances of the variables for each class need to be determined, in order words, not the entire covariance matrix. For NB model, we use the same set of training data to conduct machine learning and the same set of testing data to test.

```
svm_classifier = svm(mat,as.factor(total[,2]))
nb_classifier = naiveBayes(mat,as.factor(total[,2]))
```

By loading the R library *e1071,* SVM and NB can be used to build our classifier. Approximately, it takes about half an hour to train the SVM classifier and few seconds for NB model. After the training process is finished, the two classifiers can be used to predict a user's interest. By inputting text sentences in *predictTest* function, the classifier will predict and output the interest category it falls into.
In the following section, more details will be introduced regarding how we use our testing data to evaluate the accuracy and performance of our classifier.

3.3. Model Testing

After building the model, we need to use testing data to test the accuracy of our model. We have 9 sets of testing data, each having [, 1] being the text of tweets and [, 2] being the type of the corresponding category. Due to the large size of our testing data, we further reduce each testing dataset down to 100 except for music testing dataset, which is reduced to 24 due to the restriction in its maximum size. After sampling out the testing data, we have a variable *'count'* to record the correctness of our testing data. Then we use the *predictTest* function to predict the category of each incoming testing data. We put the text of the testing data [, 1] into the function, and compare the result given by the function with category [, 2] of the text. If the two match, then the correctness increases by one. Hence, when finishing all testing cases, we would be able to calculate out the accuracy percentage by dividing the total number of testing cases.

The testing result indicates that the average percentage of tweets that can be successfully classified using SVM model is 59.0%, while the percentage for NB model is only 12.5%, showing that SVM model is

much more accurate than the NB model for our dataset. Therefore, we choose SVM model for our project.

Though the accuracy of the SVM model is approximately 50%~60%, the accuracy for some specific category, such as *news*, is not as high as what we expected. We would discuss about this problem in the later limitation section. More details on the testing result can be found on appendix 9.

4. System

4.1. Input and Output of the System

The input of our system is twitter usernames. With the username, our system can generate interest table, Nightingale's Statistical Diagram, and word cloud diagram. If the input is only one username, the output would be interests allocation of this twitter user through those tables and diagrams. If the input is two usernames, the output would be the common interests between two twitter users as well as a number of tweets after filtering the unwanted categories of tweets.

4.2. User Timeline Retrieval

As our system is about knowing interests of the users, we must acquire the tweets on the user timeline first. The process of "handshake" is the same as how we retrieved tweets of training data and testing data. After loading "twitter_auth.Rdata" and finishing "registerTwitterOAuth(Cred)", we use "userTimeLine()" function in Library "twitteR" to get certain number of tweets of the specific twitter user. The code for getting tweets of specific user is shown here:

```
user <- userTimeline(user_name,n=100)
user <-twListToDF(user)
tweets <- as.data.frame(user$text)
tweets[,1] <- as.character(tweets[,1])
for (i in 1:length(tweets[,1])) {
    tweets[i,1] <- gsub('http\\S+\\s*', '', tweets[i,1])
    tweets[i,1] <- gsub('[\r\n\t]', '', tweets[i,1])
}
```

4.3. User Testing Result

After collecting the user timeline, we then calculate the percentage of tweets of the user in 10 pre-defined categories and put the statistics into the result table. A new function "get_result_table" is defined to return the classes of the retrieved tweets from timeline, number of times of appearance of each class and percentage of each class. We chose one account from each of the nine pre-defined categories and call "get_result_table" function for each account to display their interests. At the same time, a pie-chart will also be shown to illustrate the different component of tweets. One example of "David_Cameron" is shown in Fig 4.1 below. The result tables of other eight accounts can be accessed in Appendix 1 to Appendix 8.

	classes	times	percentage
7	news	37	0.42528736
3	business	13	0.14942529
8	politics	12	0.13793103
2	arts	8	0.09195402
9	sport	5	0.05747126
10	technology	5	0.05747126
4	food	3	0.03448276
6	music	2	0.02298851
1	others	1	0.01149425
5	fashion	1	0.01149425

Fig 4.1 One example of "get_result_table" on "David_Cameron"

4.4. Visualization of results

4.4.1. Florence Nightingale's Statistical Diagram

We use Baidu ECharts to create a special pie-chart, namely Florence Nightingale's Statistical Diagrams, to better display the weightage of each class. The pie-chart is simple and easy to understand. It helps audiences to conduct an immediate analysis and comprehend information quickly (Carol, 2014). Nightingale's Statistical Diagram further evaluate the advantage of pie-chart. The example of "David_Cameron" is shown in Fig 4.2 below.

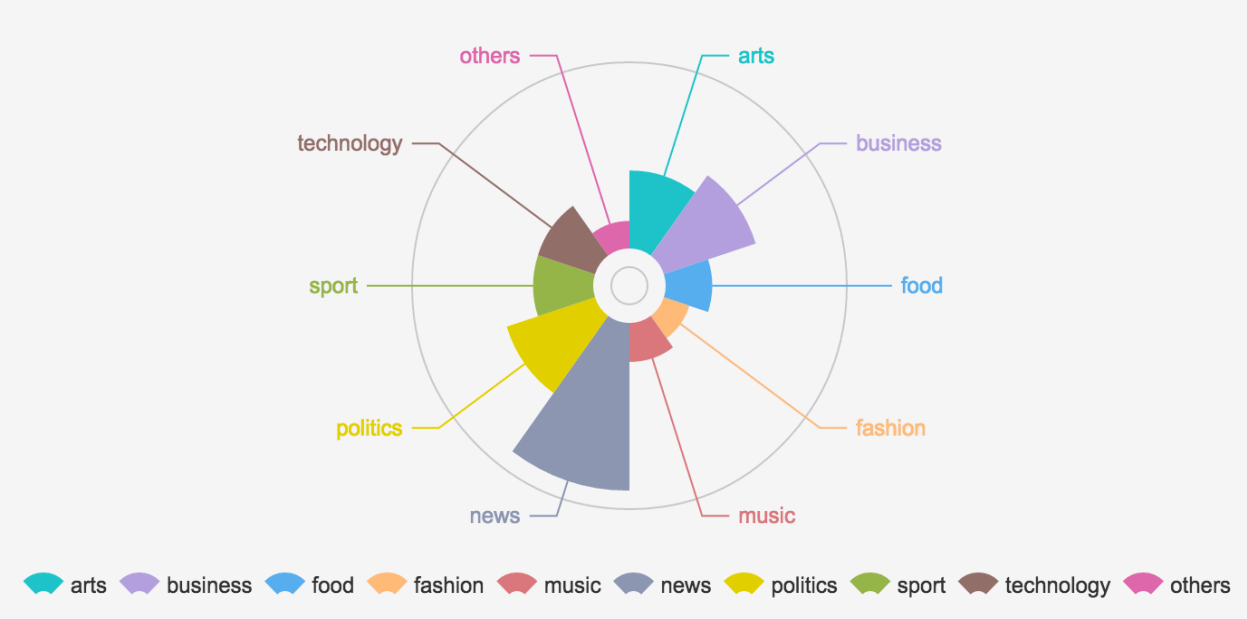

Fig 4.2 One example of Nightingale's Statistical Diagrams on “David_Cameron”

4.4.2. Word Cloud

Besides the pie-chart, we also use word cloud to establish more friendly visualization. While calling “get_result_table”, a word cloud diagram will be displayed at the same time. Word cloud emphasizes the most frequently used words in the document, allowing audiences to focus on the different weightage of words (Frances&Andrew, 2011). We use Library “wordcloud” and define a new function “print_word_cloud” to show the word cloud. One example of “David_Cameron” is shown in Fig 4.3 below. The word cloud of other eight accounts can be accessed in Appendix 1 to Appendix 8 as well. The R code for building the word cloud is shown below.

Fig 4.3 One example of Word Cloud Diagram on “David_Cameron”

5. Application

5.1. View user's interested areas on his/her twitter profile page

A pie chart of a specific user's will be displayed on the user's profile page. As we can observe in the picture below, most tweets sent by David Cameron, the prime minister of UK, are related with political issues according to our classification.

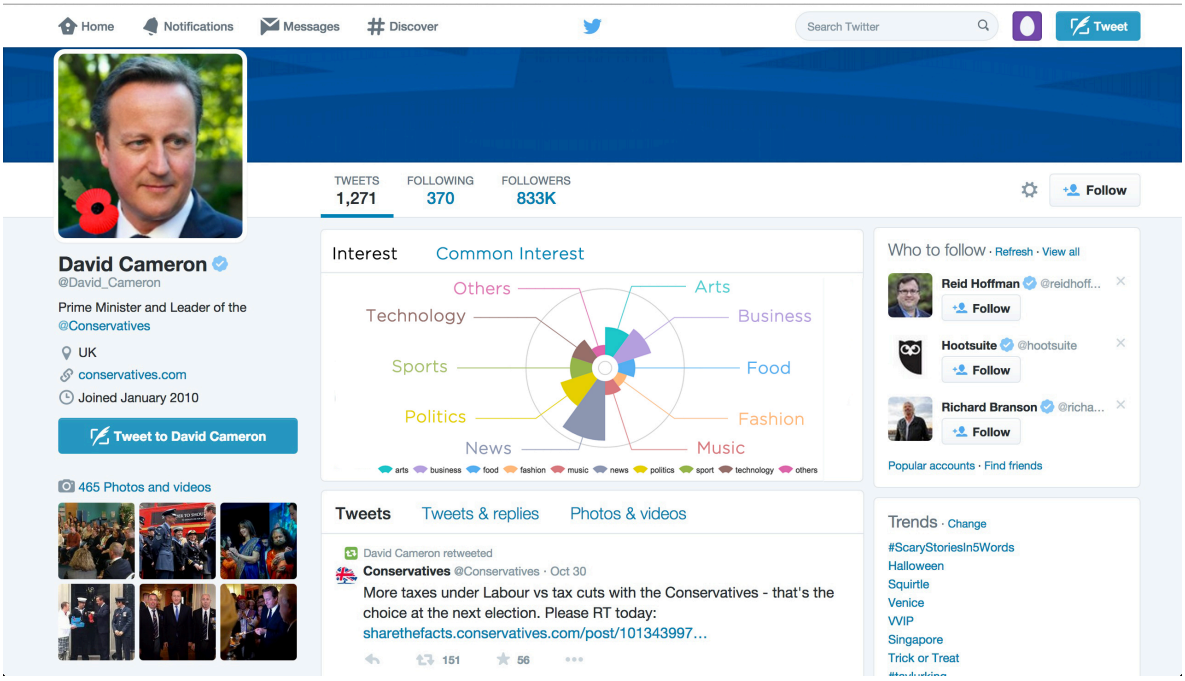

Fig 5.1 New Profile Page of Twitter

The pie chart serves as a brief summary of the user's tweets sent recently, giving other people a direct impression on the topics that the user likes. By knowing this information, other people can recognize their common interests between themselves and the user so as to better decide whether they want to follow the user. For example, people who are interested in politics, business and news have higher tendency to follow David Cameron.

Moreover, after people click on the topic name on the pie chart, only tweets that belong to that specific category will be displayed on the user's timeline. This function is extremely helpful if another twitter user is only interested in that certain topic. The person can quickly browse through the tweets related to their area of interests without being spammed by other unrelated information.

5.2. Display only the tweets under certain categories on user's home page

Currently, if user A decides to follow user B, all tweets sent by the user B will be displayed on the front page of the user A which may includes many tweets that are out of A's interests. Our classification method allows twitter user to follow only the tweets under certain categories sent by another user, which means tweets from other categories will be filtered and hidden from the user.

After clicking the "Follow" button, the user will be able to see a page similar to the picture below.

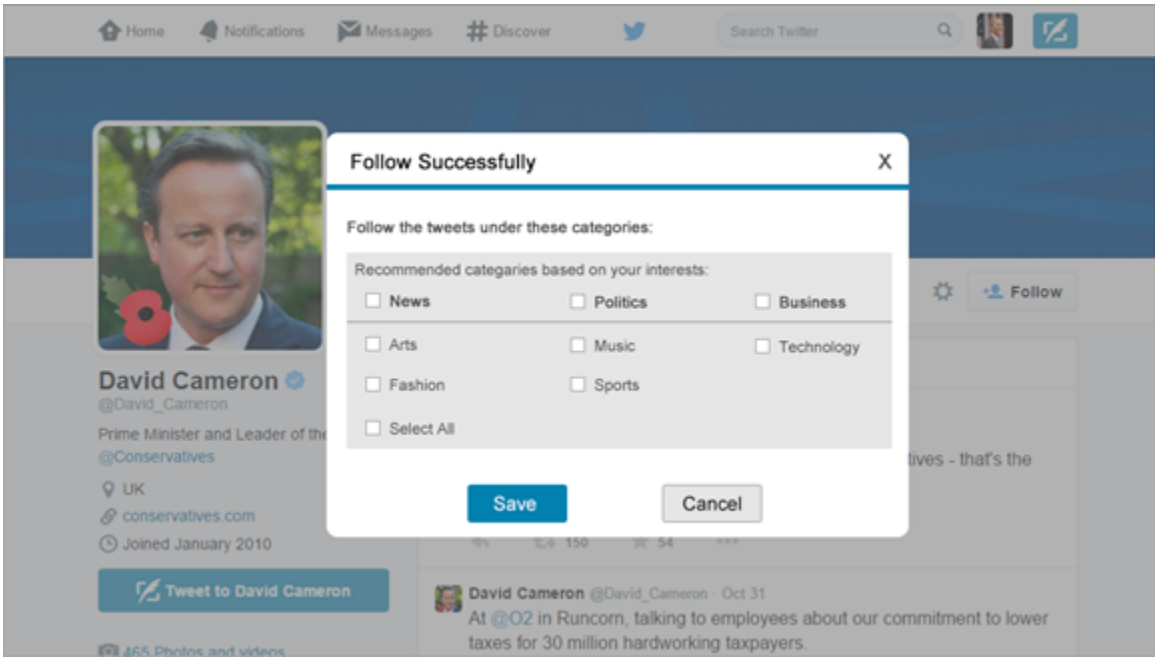

Fig 5.2 The pop-up window when following a new twitter user

Based on the interested topics of both the follower and the user will be followed, we can obtain a list of top few common interested areas of the two. We created a function called *InterestedArea* and *make_friend* to implement this function.

```
interestedArea <- function(user_name) {
    result_table <- get_result_table(user_name)
    list <- result_table[c(1,2,3,4,5),]
    list <- list[,1]
    as.data.frame(list)
}
```

```
make_friend <- function(userA, userB) {
    areaA <- interestedArea(userA)
    areaB <- interestedArea(userB)
    common <- merge(areaA,areaB,all=F)
    if (length(common)==0) {
        cat("No common interested area!")
    } else {
        cat("The common interested areas between ", userA,
            " and ", userB, ":\n")
    }
    common
}
```

The function *interestedArea* returns a list of top 5 interested topics of the user. And the function "make_friend" returns the interception part of the two user's interested topics lists. These common interested topics will be shown under the "Recommended categories based on your interests" since they are the most possible set of tweets user would like to follow. Moreover, users are always free to follow as many topics as they like and they can also edit their choices anytime.

```
filter <- function(topics,user_name) {
    tweets <- get_tweets(user_name);
    for (i in 1:length(tweets[,1])) {
        current <- tweets[i,1]
        result <- predictTest(current, mat, svm_classifier)
        for (j in 1:length(topics)) {
            if (result == topics[j]) {
                cat(current,"\n")
                break
}}}}
```

We create a function called "filter" to filter out the unwanted tweets. "topics" is list of topics the user wants to follow. We check through every tweets and classify them accordingly. Only the tweets that are classified to the topics in "topics" will be displayed on the follower's timeline.

6. Discussion and Conclusion

6.1. Evaluation on Performance

As aforementioned summary in model testing, the accuracy of the SVM classifier is approximately 50%~60%. However, it is not for all the categories. For some specific category, such as *news*, it is not as high as what we expected.

Furthermore, when applying the model to some certain users, it could be shown that the classifier is relatively accurate. We choose some specific users who focused on particular areas, whose timeline are supplied into the SVM classifier and subsequently draw pie charts on their interest. It could be seen that the major area in the pie charts mostly corresponds to the focused area of that particular user. This point can also prove the accuracy of our classifier. The details of the process are attached in the appendix 1-8. In a nutshell, the performance of our classifier is quite acceptable.

6.2. Limitations

Though the testing result is relatively accurate, the classification model still has various limitations that we could possibly work on to make further improvements.

Firstly, the data amount is considered as a major limitation in our project. In order to make time controllable when training our classifier, we only sampled 1500 testing data from each category. As a result, the predicting words in our feature vector consists only 749 words, which to some extent is not comprehensive enough. Furthermore, when we are analysing a single users' interest, we could only predict his or her recent interest, as the function *userTimeline* has maximum number 3200 restriction. It is unlikely for us to predict the users' interest based on the whole dataset of his or her past tweets.

Secondly, some hidden overlapping might occur when we take a look at some specific tweets. For example, one can mention arts and music in a single tweet, as they are highly related. Sports, politics and business could also be included in the news category, as there is always news about these aspects. As our classifier can only assign one final predict result to a text sentence, these possible overlapping might lead to inaccurate result.

Thirdly, some tweets with same content can be possible collected various times in our data retrieval process, which may affect our results. After sampling, it is still likely that there are repeating tweets in final training dataset. This flaw might reduces the information we get from data retrieval process, and to

some extent lowering the accuracy of our classifier as the feature vector would be smaller than expected due to this problem.

Fourthly, we make an assumption in our data collecting. We assume that every tweet under the chosen professional accounts is related to the topic. However, this assumption may not hold as some unrelated tweets may also be posted by the twitter owner. This limitation would cause our predicting words in the feature vector not accurate enough to be used to train the model.

6.3. Improvements

6.3.1. More data should be collected

Since the limitation of our devices and time, the data collected is not sufficient to training a very accurate classifier. To improve the accuracy of our classifier, we should use a computer with better CPU to collect more data from more users, which may increase the amount of predicting words.

6.3.2. More models should be build to improve the accuracy

Due to the time constraint, we only compare the result accuracies between SVM model and Naive Bayes model in our project. Other classification algorithms, such as Fisher's linear discriminant, Logistic regression, Kernel estimation, and Decision trees, should also be used to test the accuracies. The model with the highest accuracy should be chosen to obtain the best classification result.

6.3.3. Modify the model

Due to the nature of most tweets of which can be categorized into more than two categories, we can modify the model such that a tweet can belong to multiple categories. We expect the modified model can predict each tweet multiple times and return the categories that are most suitable for it.

6.4. Conclusion

For this project, we spot the problem of raw data and proposed a solution to it by building a model from our training data obtained through Twitter and practically using it to suggest the possible interests of the Twitter user. This model allows people to have a clearer understanding on other Twitter users and ease their information search journey in Twitter, which further enhance the Twitter user experience. Though

the model we built based on SVM have reasonable and acceptable accuracy percentage, it also has a few limitations that dent its efficiency. Hence, the improvements on our data collection process, model selection and model modification are favoured to increase the efficiency of our solution. In conclusion, our model can practically solve our problem and be further used to build few applications to enhance Twitter user experience.

7. References

Aaron (2014). Independent variables in Naïve Bayes, Retrieved from CrossValidated, http://stats.stackexchange.com/questions/117062/independent-variables-in-na%C3%AFve-bayes

Finch, C. (2014). Advantages & Disadvantages of a Pie Chart. Retrieved November 2, 2014, from http://www.ehow.com/list_6715678_advantages-disadvantages-pie-chart.html

Frances Miley, Andrew Read (2011). Journal of the Scholarship of Teaching and Learning, Vol. 11, No. 2, April 2011, pp. 91-110.

Lee, K., Palsetia, D., Narayanan, R., Ali Patwary, M., Agrawal, A., & Choudhary, A. (2011). Twitter Trending Topic Classification. Retrieved November 2, 2014, from http://users.eecs.northwestern.edu/~drp925/pdf/LeePal11.pdf

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., & Demirbas, M. (2010). Short Text Classification in Twitter to Improve Information Filtering. Retrieved November 2, 2014, from http://www.cs.bilkent.edu.tr/~hakan/publication/TweetClassification.pdf

Twitter. Inc., 2014. Retrieved November 2, 2014, from https://about.twitter.com/company

8. Appendix

Appendix 1 The testing result of *@LouisVuitton*:

```
> get_result_table("LouisVuitton")
      classes times percentage
5     fashion    51       0.51
2        arts    11       0.11
4        food    10       0.10
9       sport     9       0.09
6       music     8       0.08
10 technology     5       0.05
1       others     3       0.03
3    business     2       0.02
8    politics     1       0.01
7        news     0       0.00
```

Appendix 2 The testing result of *@NBA*:

```
> get_result_table("NBA")
      classes times percentage
9       sport    27 0.52941176
4        food     6 0.11764706
6       music     5 0.09803922
5     fashion     4 0.07843137
1      others     2 0.03921569
3    business     2 0.03921569
7        news     2 0.03921569
10 technology     2 0.03921569
2        arts     1 0.01960784
8    politics     0 0.00000000
```

Appendix 3 The testing result of *@cnnbrk*:

```
> get_result_table("cnnbrk")
      classes times percentage
7        news    33     0.4125
8    politics    28     0.3500
9       sport     7     0.0875
3    business     5     0.0625
10 technology     4     0.0500
1      others     1     0.0125
2        arts     1     0.0125
4        food     1     0.0125
5     fashion     0     0.0000
6       music     0     0.0000
```

Appendix 4 The testing result of *@ReutersBiz*:

```
> get_result_table("ReutersBiz")
      classes times percentage
3    business    55 0.62500000
7        news     9 0.10227273
4        food     7 0.07954545
10 technology     6 0.06818182
8    politics     5 0.05681818
6       music     2 0.02272727
9       sport     2 0.02272727
1      others     1 0.01136364
5     fashion     1 0.01136364
2        arts     0 0.00000000
```

Appendix 5 The testing result of *@ForbesTech*:

```
> get_result_table("ForbesTech")
      classes times percentage
10 technology    37 0.37373737
3    business    26 0.26262626
4        food     9 0.09090909
7        news     7 0.07070707
9       sport     6 0.06060606
1      others     5 0.05050505
8    politics     3 0.03030303
2        arts     2 0.02020202
5     fashion     2 0.02020202
6       music     2 0.02020202
```

Appendix 6 The testing result of *@HuffPostArts*:

```
> get_result_table("HuffPostArts")
      classes times percentage
2        arts    25     0.3125
10 technology    18     0.2250
4        food    11     0.1375
3    business     5     0.0625
5     fashion     5     0.0625
6       music     5     0.0625
9       sport     4     0.0500
1      others     3     0.0375
8    politics     3     0.0375
7        news     1     0.0125
```

Appendix 7 The testing result of *@allsongs*:

```
> get_result_table("allsongs")
      classes times percentage
6       music    37 0.41111111
4        food    14 0.15555556
9       sport     8 0.08888889
2        arts     7 0.07777778
1      others     6 0.06666667
5     fashion     6 0.06666667
7        news     4 0.04444444
10 technology     4 0.04444444
3    business     3 0.03333333
8    politics     1 0.01111111
```

Appendix 8 The testing result of *@latimesfood*:

```
> get_result_table("latimesfood")
      classes times percentage
4        food    31 0.39743590
9       sport     9 0.11538462
10 technology     8 0.10256410
5     fashion     6 0.07692308
8    politics     6 0.07692308
6       music     5 0.06410256
3    business     4 0.05128205
1      others     3 0.03846154
2        arts     3 0.03846154
7        news     3 0.03846154
```

Appendix 9 The testing process based on remaining dataset

It could be shown that in the **business** category, the correct prediction is 56 out of 100, which is 56% accuracy.

```
> correct <- 0
> for (i in 1:length(business_remain[,1])){
+   text <- business_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(business_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of business: ", correct)
Number of correct classification of business:  56
```

However, using Naive Baynes classifier, the accuracy is only 0%.

```
> for (i in 1:length(business_remain[,1])){
+   text <- business_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(business_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of business: ", correct)
Number of correct classification of business:  0
```

It could be shown that in the **fashion** category, the correct prediction is 63 out of 100, which is 63% accuracy.

```
> correct <- 0
> for (i in 1:length(fashion_remain[,1])){
+   text <- fashion_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(fashion_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of fashion: ", correct)
Number of correct classification of fashion:  63
```

However, using Naive Baynes classifier, the accuracy is only 1%.

```
> for (i in 1:length(fashion_remain[,1])){
+   text <- fashion_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(fashion_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of fashion: ", correct)
Number of correct classification of fashion:  1
```

It could be shown that in the **politics** category, the correct prediction is 60 out of 100, which is 60% accuracy.

```
> correct <- 0
> for (i in 1:length(politics_remain[,1])){
+   text <- politics_remain[i,1]
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(politics_remain[i,2])){
+     correct = correct+1 }
+ }
> cat("Number of correct classification of politics: ", correct)
Number of correct classification of politics:  60
```

However, using Naive Baynes classifier, the accuracy is only 2%.

```
> for (i in 1:length(politics_remain[,1])){
+   text <- politics_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(politics_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of politics: ", correct)
Number of correct classification of politics:  2
```

It could be shown that in the **music** category, the correct prediction is 16 out of 23, which is 69.6% accuracy.

```
> correct <- 0
> for (i in 1:length(music_remain[,1])){
+   text <- music_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(music_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of music: ", correct)
Number of correct classification of music:  16
```

However, using Naive Baynes classifier, the accuracy is only 4.3%.

```
> correct <- 0
> for (i in 1:length(music_remain[,1])){
+   text <- music_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(music_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of music: ", correct)
Number of correct classification of music:  1
```

It could be shown that in the **art** category, the correct prediction is 47 out of 100, which is 47% accuracy.

```
> correct <- 0
> for (i in 1:length(art_remain[,1])){
+   text <- art_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(art_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of arts: ", correct)
Number of correct classification of arts:  47
```

However, using Naive Baynes classifier, the accuracy is only 9%.

```
> for (i in 1:length(art_remain[,1])){
+   text <- art_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(art_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of arts: ", correct)
Number of correct classification of arts:  9
```

It could be shown that in the **food** category, the correct prediction is 60 out of 100, which is 60% accuracy.

```
> correct <- 0
> for (i in 1:length(food_remain[,1])){
+   text <- food_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(food_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of food: ", correct)
Number of correct classification of food:  60
```

Using Naive Baynes classifier, the accuracy is 81%. As a matter of fact, the most result given by naive baynes is food.

```
> for (i in 1:length(food_remain[,1])){
+   text <- food_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(food_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of food: ", correct)
Number of correct classification of food:  81
```

It could be shown that in the **news** category, the correct prediction is 59 out of 100, which is 59% accuracy.

```
> correct <- 0
> for (i in 1:length(news_remain[,1])){
+   text <- news_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(news_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of news: ", correct)
Number of correct classification of news:  59
```

However, using Naive Baynes classifier, the accuracy is only 0%.

```
> correct <- 0
> for (i in 1:length(news_remain[,1])){
+   text <- news_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(news_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of news: ", correct)
Number of correct classification of news:  0
```

It could be shown that in the **sports** category, the correct prediction is 65 out of 100, which is 65% accuracy.

```
> correct <- 0
> for (i in 1:length(sports_remain[,1])){
+   text <- sports_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(sports_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of sports: ", correct)
Number of correct classification of sports:  65
```

However, using Naive Baynes classifier, the accuracy is only 14%.

```
> for (i in 1:length(sports_remain[,1])){
+   text <- sports_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(sports_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of sports: ", correct)
Number of correct classification of sports:  14
```

It could be shown that in the **technology** category, the correct prediction is 51 out of 100, which is 51% accuracy.

```
> correct <- 0
> for (i in 1:length(technology_remain[,1])){
+   text <- technology_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, svm_classifier)
+   if (result == as.character(technology_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of technology: ", correct)
Number of correct classification of technology:  51
```

However, using Naive Baynes classifier, the accuracy is only 1%.

```
> for (i in 1:length(technology_remain[,1])){
+   text <- technology_remain[i,1]
+   if (text != "") {
+   result <- predictTest(text, mat, nb_classifier)
+   if (result == as.character(technology_remain[i,2])){
+     correct = correct+1 }}
+ }
> cat("Number of correct classification of technology: ", correct)
Number of correct classification of technology:  1
```