CSCI946: Big Data Analytics
Assignment - 2

Venkata Sandeep Kumar. Karamsetty, Sudent No: 6228975,

Mail-Id : vskk033@uowmail.edu.au

1 Task 1 – Text Classification

1.1 Task 1 - Q1. Describe this 20 Newsgroups data set.

The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups[1]. This 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian). Here is a list of the 20 newsgroups, partitioned (more or less) according to subject matter as shown in figure 1:

As suggested by assignment download the "20news-bydate-matlab.tgz", which has 6 files namely: train.data train.label, train.map, test.data, test.label, test.map. The .data files are formatted as "docIdx wordIdx count". The .label files are simply a list of label id's. The .map files map from

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Figure 1: partitioning according to subject matter

label id's to label names. The classification of 20 news group dataset is a supervised classification problem in which we have 20 categories which has a piece of news belong to one category.

1.1.1 Preview of train.data and test.data

Command used to read and display in python:

$$trainingData = pd.read_table('train.data', sep = "", names = ['docId','wordId','count']) \quad (1)$$

print(trainingData)
Output shown in figure 2 - 2a.

$$testingData = pd.read_table('test.data', sep = "", names = ['docId','wordId','count']) \quad (2)$$

print(testingData)
Output shown in figure 2 - 2b.
Output Shown in images :

trainingData = pd.read_table

	docId	wordId	count
0	1	1	4
1	1	2	2
2	1	3	10
3	1	4	4
4	1	5	2
5	1	6	1
6	1	7	1
7	1	8	1
8	1	9	3
9	1	10	9
10	1	11	2

testingData = pd.read_table

	docId	wordId	count
0	1	3	1
1	1	10	1
2	1	12	8
3	1	17	1
4	1	23	8
5	1	27	1
6	1	29	6
7	1	30	7
8	1	33	10
9	1	42	12
10	1	48	2

Figure 2: 2a - trainingData; 2b - testingData

1.1.2 Preview of train.map and test.map

Command used to read and display in python:

$$trainingmap = pd.read_table('train.map', sep = "", names = ['labelName','labelId']) \quad (3)$$

print(trainingmap)
Output shown in figure 3 - 3a.

$$testingmap = pd.read_table('test.map', sep = "", names = ['labelName','labelId']) \quad (4)$$

print(testingmap)

Output shown in figure 3 - 3b.

Output Shown in images below :

```
  trainingmap = pd.read_table('train.map',
                   labelName  labelId
0                alt.atheism        1
1              comp.graphics        2
2    comp.os.ms-windows.misc        3
3   comp.sys.ibm.pc.hardware        4
4      comp.sys.mac.hardware        5
5              comp.windows.x        6
6                misc.forsale        7
7                   rec.autos        8
8             rec.motorcycles        9
9          rec.sport.baseball       10
10           rec.sport.hockey       11
11                  sci.crypt       12
12            sci.electronics       13
13                    sci.med       14
14                  sci.space       15
15     soc.religion.christian       16
16         talk.politics.guns       17
17      talk.politics.mideast       18
18         talk.politics.misc       19
19         talk.religion.misc       20
```

```
  testingmap = pd.read_table('test.map'
                   labelName  labelId
0                alt.atheism        1
1              comp.graphics        2
2    comp.os.ms-windows.misc        3
3   comp.sys.ibm.pc.hardware        4
4      comp.sys.mac.hardware        5
5              comp.windows.x        6
6                misc.forsale        7
7                   rec.autos        8
8             rec.motorcycles        9
9          rec.sport.baseball       10
10           rec.sport.hockey       11
11                  sci.crypt       12
12            sci.electronics       13
13                    sci.med       14
14                  sci.space       15
15     soc.religion.christian       16
16         talk.politics.guns       17
17      talk.politics.mideast       18
18         talk.politics.misc       19
19         talk.religion.misc       20
         docId  wordId  count
0            1       1      4
1            1       2      2
2            1       3     10
3            1       4      4
4            1       5      2
5            1       6      1
6            1       7      1
```

Figure 3: 3a - trainingmap; 3b - testingmap

1.1.3 Preview of train.label test.label

Command used to read and display in python:

$$traininglabel = pd.read_table('train.label', sep = "", names = ['labelId']) \quad (5)$$

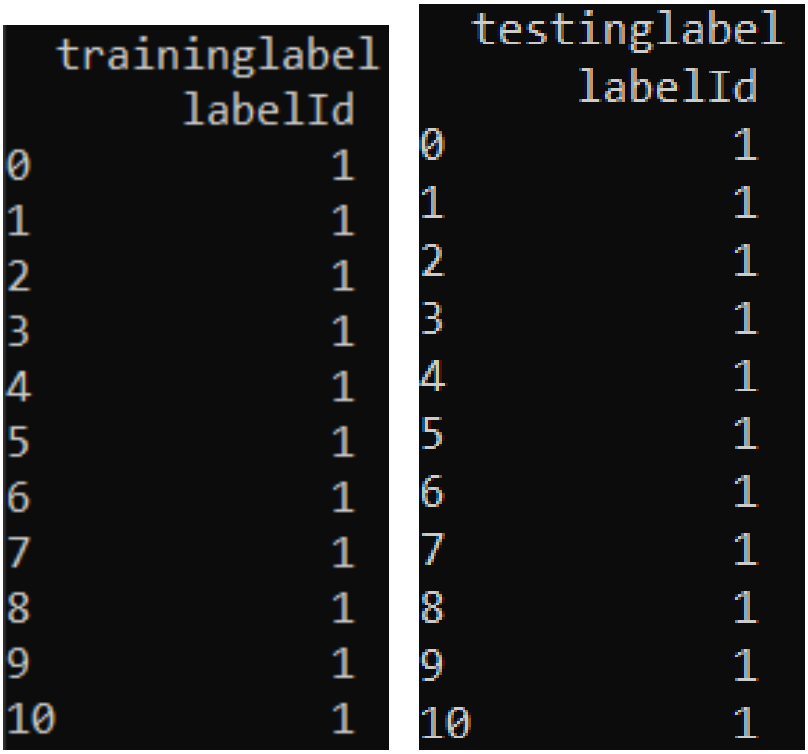
print(traininglabel)

Output shown in figure 4 - 4a.

$$testinglabel = pd.read_table('test.label', sep = "", names = ['labelId']) \quad (6)$$

print(testinglabel) Output shown in figure 4 - 4b.

Output Shown in images below :



	trainlabel
	labelId
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

	testinglabel
	labelId
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

Figure 4: 4a - traininglabel; 4b - testinglabel

1.2 Task 1 - Q2. Describe how each document is represented in your implementation.

At first the training dataset and test dataset files are read from the train.data and test.data. Then we will read the labels of train and test which will be store in variable. At last the trainmap and testmap files which is combination of a key-value pair for label id and label names.
Now focusing on mapping each of the observations from the training label with label ID on the basis of doc Id from the training data. Hence we will perform this using merge function on the training map with training label on label Id. Code used is shown below:

1.2.1 Code used for Merging the training and testing sets:

trainlabel['docId'] = list(range(1,11270))
$train_label = traininglabel.merge(trainingmap, on = "labelId")$
$train_final = train_label.merge(trainingData, on = "docId")$

$train_final = train_final[["docId","wordId","count","labelName"]]$
$print("Mergedtrainingdatashownbelow:")$
$print(train_final)$
Ouput in figure 5 - 5a.

testinglabel['docId'] = list(range(1,7506))
$test_label = testinglabel.merge(testingmap, on = "labelId")$
$test_final = test_label.merge(testingData, on = "docId")$
$test_final = test_final[["docId","wordId","count","labelName"]]$
$print("Mergedtestingdatashownbelow:")$
$print(test_final)$
Ouput in figure 5 - 5b.

Merged training data shown below:

	docId	wordId	count	labelName
0	1	1	4	alt.atheism
1	1	2	2	alt.atheism
2	1	3	10	alt.atheism
3	1	4	4	alt.atheism
4	1	5	2	alt.atheism
5	1	6	1	alt.atheism
6	1	7	1	alt.atheism
7	1	8	1	alt.atheism
8	1	9	3	alt.atheism
9	1	10	9	alt.atheism
10	1	11	2	alt.atheism

Merged testing data shown below:

	docId	wordId	count	labelName
0	1	3	1	alt.atheism
1	1	10	1	alt.atheism
2	1	12	8	alt.atheism
3	1	17	1	alt.atheism
4	1	23	8	alt.atheism
5	1	27	1	alt.atheism
6	1	29	6	alt.atheism
7	1	30	7	alt.atheism
8	1	33	10	alt.atheism
9	1	42	12	alt.atheism
10	1	48	2	alt.atheism

Figure 5: 5a - Merged training dataset; 5b - Merged testing dataset

1.2.2 Analyzing a Word using term frequency(tf-idf):

Measurement of how frequent a word occurs in a document is known as term frequency(tf). This static tf-idf is intended to measure how important a word is to a document in a collection(or corpus) of documents, for example to one novel in a collection of novels or to one website in a collection of websites. By observation we can know that given data set is already in the form of data frame, we just need to pass it to coo_matrix to separate the result. Hence we can pass document Id, World Id, count to the data frame and create a coo_matrix from the same data-frame. Once it is done, we will create training data matrix and test data matrix by using code shown below.
Code:

$$mat = sps.coo\_matrix((trainingData["count"].values, (trainingData["docId"].values - 1, trainingData["w \tag{7}$$

training_data_matrix = mat.tocsc()
training_data_matrix.shape

test_mat = sps.coo_matrix((testingData["count"].values,
(testingData["docId"].values-1, testingData["wordId"].values-1)))
testing_data_matrix = test_mat.tocsc()
testing_data_matrix = testing_data_matrix[:,:training$_{d}ata_{m}atrix.shape$[1]]

1.3 Task 1 - Q3. Describe Naïve Bayes classifier and how you use it to classify the 20 Newsgroups data set.

1.3.1 Bayes Theorom

The conditional probability of event C occuring, given the event A has already occured, is denoted as P(C—A), which can be found using the formula in Equation 9.

$$P(C|A) = \frac{P(A \cap C)}{P(A)} \tag{8}$$

Let us consider C is a class label c = c1, c2, ... cn
A is the observed attribute A = a1, a2, ... am

$$P(C|A) = \frac{P(A|C) \cdot P(C)}{P(A)} \tag{9}$$

With two simplification, Bayes theorem can be extended to become a naive Bayes classifier which are shared in section 1.3.2.

1.3.2 Naive Bayes Classifier:

It's one of the probabilistic classification method based on Bayes theorem. It's one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm which has high accuracy and speed on large datasets.This theorem gives the relationship between the probabilities of two events and their conditional probabilities. A navie Bayes classifier assumes that the presence or absence of a particular feature of a class is unrelated to the presence or absence of other features. Some of the application areas of this classifier is fraud detection in auto insurance.

1.) Conditional Independence assumption:
Each attribute is conditionally independent of every other attribute given a class label Ci.

$$P\left(a_1, a_{2'}, \ldots, a_m|c_i\right) = P\left(a_1|c_i\right) P\left(a_2|c_i\right) \cdots P\left(a_m|c_i\right) = \prod_{j=1}^{m} P\left(a_j|c_i\right) \tag{10}$$

which simplifies the computation of P(A/Ci) in equation 11.

2.) Ignoring denominator P(A)

Removing the denominator has no impact on the probability scores. Hence the classifier modified to:

$$P\left(c_i|A\right) \propto P\left(c_i\right) \cdot \prod_{j=1}^{m} P\left(a_j|c_i\right) \quad i = 1, 2, \ldots n \tag{11}$$

$$P\left(C_i|A\right) \propto \log P\left(C_i\right) + \sum_{j=1}^{m} \log P\left(a_j|c_i\right) \quad i = 1, 2, \ldots n \tag{12}$$

In Bayesian handling of rare events must be done by the help of a technique call smooting which is explained in below section.

1.4 Smoothing

If one of the attribute value does not appear with one of the class labels within the traning set, the corresponding P(aj—ci) will equal zero which results entire equation to zero causing overfitting. Hence smoothing techniques can be employed to adjust the probabilities of P(aj—ci) and to ensure a nonzero value of P(ci—A). It pretends to see every outcome once more than it actually appears.

$$P^r(x) = \frac{\text{count}(x) + 1}{\Sigma_x[\text{count}(x) + 1]} \tag{13}$$

we can write the result as in below equation.

$$P^{*\prime}(x) = \frac{\text{count}(x) + \varepsilon}{\Sigma_x[\text{count}(x) + \varepsilon]} \tag{14}$$

1.5 How you use it to classify the 20 Newsgroups data set?

As suggested used **Naïve Bayes classifier** is used for classification of the 20 news group dataset. By observations here we will use the sparse word count feature from the 20 newsgroup corpus to show how we might classify those short documents into categories which makes easy for test data to predict from train data.

In order to use the test and train data downloaded for machine learning, convert the content of each string into a vector of numbers. For each we will use the TF-IDF vectorizer and create a pipeline that attaches it to a multinomial naive bayes classifier. with this pipeline, we can apply the model to the training data, and predict labels for the test data.

I will be classifying the Naive Bayes on the basis of Label Name with respect to document ID, Word ID and count and the result of the merged trained data. **MultinomialNB** from **sklearn.naive_bayes** by having parameters of alpha = .01 and with fit prior.

Code used in python:

```
naive Bayes Classifier
MultinomialNB(alpha=0.01, class_prior=None, fit_prior=True)
[ 1  1  1 ...  1 20  2]
Accuracy = 79.73695415439279
```

Figure 6: Accuracy for applied Navie Bayes Classifier

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Figure 7: Types of errors in confusion matrix

naiveBayesclassifier = MultinomialNB(alpha=.01, class$_prior = None, fit_prior = True$)
$naiveBayesclassifier.fit(training_data_matrix, traininglabel["labelId"])$

The result or ouput for the Naive Bayes algorithm would be the A-priori probabilities and conditional probabilities for each of the 20 document in the training dataset.

1.6 Task 1 - Q4. Report the classification accuracy and plot the confusion matrix.

1.6.1 Accuracy

Accuracy is measured based on the number of correct predictions from all the prediction made.
Code Used:
$print("Accuracy = 0".format(metrics.f1_score(testinglabel["labelId"], prediction, average =' macro') * 100))$

Output shown in figure 6

1.6.2 Confusion Matrix

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.
Options from types of errors in confusion matrix are shared in figure 7:
True Postive (TP): These are the cases that are predicted yes.
True Negative (TN): The cases that we have predicted no.
False Postive(FP): We predicted yes, but they are actually not correct. Also called as Type 1 error.
True Negative (TN): We predicted No, but they are actually correct. Also called as Type 11 error.
In our case the result for confusion matrix is a 20*20 combination of documents as we have total 20 document in the data set. In our case the confusion matrix is calcuated using $confusion_matrixfunctioncoming$
Code used in python
$cm = confusion\_matrix(testinglabel["labelId"], prediction)$
$plot\_confusion\_matrix(cm, classes = trainingmap["labelName"], title =' Confusionmatrix')$
Output shown in figure 8

1.7 Task 1 - Q5. Attach your code at the end of the report.

Code attached at last.

1.8 References

1.) https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn
2.) https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

2 Task 2 – Topic Modeling

2.1 Task 2 - Q1. Describe the Reuters-21578 corpus.

Reuters-21578 is a publically available version of the well-known Reuters-21578 "ApteMod" corpus for text categorization. ApteMod is a collection of 10,788 documents from the Reuters financial newswire service, partitioned into a training set with 7769 documents and a test set with 3019 documents. The total size of the corpus is about 43 MB. It is also available for download from http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html ,which includes a more extensive history of the data revisions.

The distribution of categories in the ApteMod corpus is highly skewed, with 36.7% of the documents in the most common category, and only 0.0185% (2 documents) in each of the five least

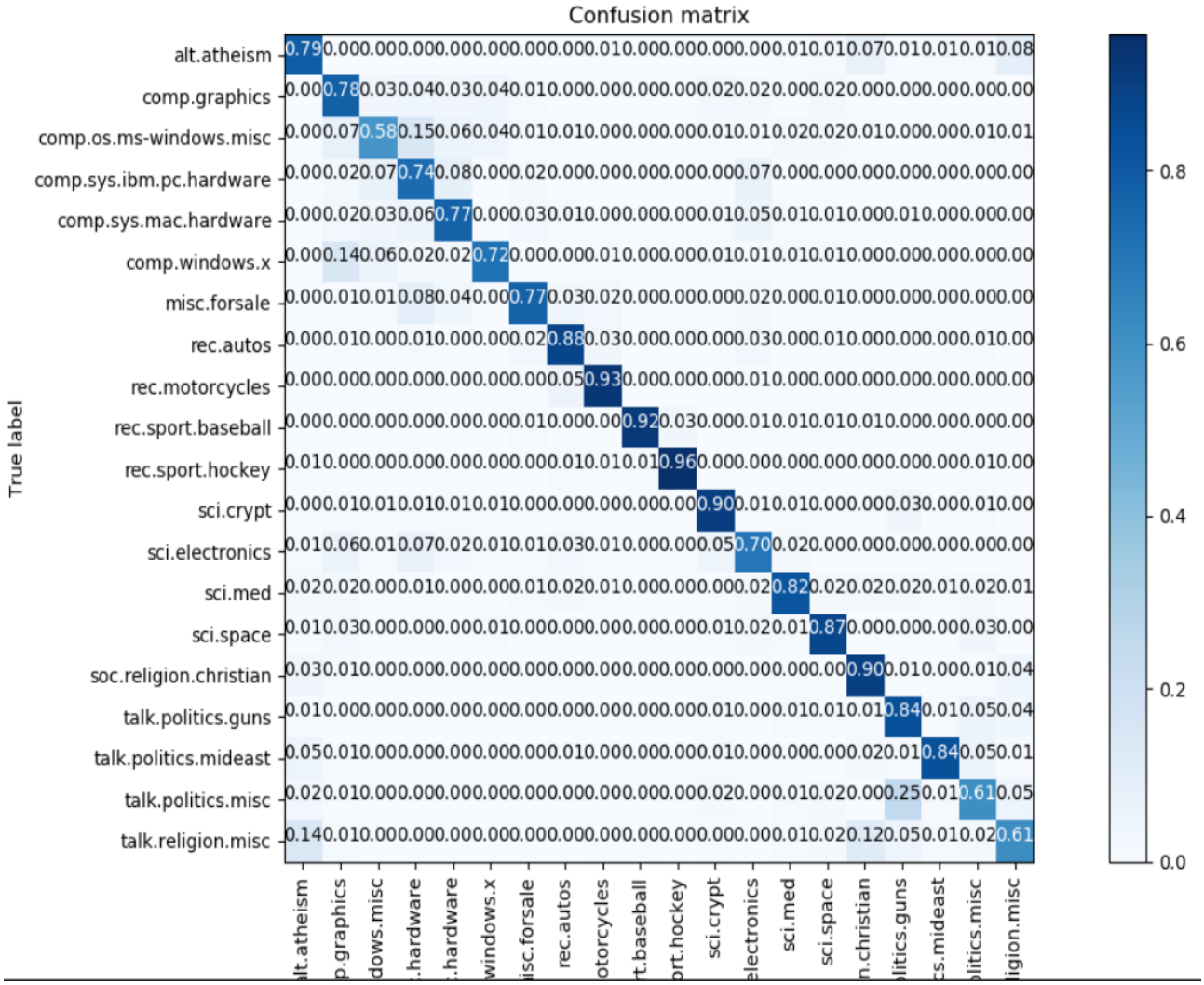


Figure 8: Confusion Matrix

Category	Training Set (YES)	Training Set (NO)	Test Set (YES)	Test Set (NO)
Acq	1615	7988	719	2580
Corn	175	9428	56	3243
Crude	383	9220	189	3110
Earn	2817	6786	1087	2212
Grain	422	9181	149	3150
Interest	343	9260	131	3168
Money-fx	518	9085	179	3120
Ship	187	9416	89	3210
Trade	356	9247	117	3182
Wheat	206	9397	71	3228

Figure 9: Categories of Reuters-21578

```
> #Summarizing the data of Reuters
> summary(Reuters21578)
      Length Class             Mode
1     2      PlainTextDocument list
2     2      PlainTextDocument list
3     2      PlainTextDocument list
4     2      PlainTextDocument list
5     2      PlainTextDocument list
6     2      PlainTextDocument list
7     2      PlainTextDocument list
8     2      PlainTextDocument list
9     2      PlainTextDocument list
10    2      PlainTextDocument list
11    2      PlainTextDocument list
12    2      PlainTextDocument list
13    2      PlainTextDocument list
14    2      PlainTextDocument list
15    2      PlainTextDocument list
16    2      PlainTextDocument list
17    2      PlainTextDocument list
18    2      PlainTextDocument list
```

Figure 10: Summary of Reuters-21578

common categories. In fact, the original data source is even more skewed—in creating the corpus, any categories that did not contain at least one document in the training set and one document in the test set were removed from the corpus by its original creator. In the ApteMod corpus, each document belongs to one or more categories. There are 90 categories in the corpus. The average number of categories per document is 1.235, and the average number of documents per category is about 148, or 1.37% of the corpus. The 10 Categories of the reuters data set is shown in figure - 9.

Summary for Reuters shared in figure 10:

2.2 Task 2 - Q2. Describe how each document is represented in your implementation.

From summary we can observe that SGML files are downloaded which can be displayed by using below commands:

```
> #observation on reuters
> cat(content(Reuters21578[[7]]))
Red Lion Inns Limited Partnership
said it filed a registration statement with the Securities and
Exchange Commission covering a proposed offering of 4,790,000
units of limited partnership interests.
    The company said it expects the offering to be priced at 20
dlrs per unit.
    It said proceeds from the offering, along with a 102.5 mln
dlr mortgage loan, will be used to finance its planned
acquisition of 10 Red Lion hotels.
 Reuter

> cat(content(Reuters21578[[3]]))
Texas Commerce Bancshares Inc's Texas
Commerce Bank-Houston said it filed an application with the
Comptroller of the Currency in an effort to create the largest
banking network in Harris County.
    The bank said the network would link 31 banks having
13.5 billion dlrs in assets and 7.5 billion dlrs in deposits.
```

Figure 11: SGML data of files 3 & 7 of Reuters-21578

Code Used:

cat(content(Reuters21578[[7]]))

cat(content(Reuters21578[[3]]))

Output shown in figure 11:

No these SGML documents of files need to represent and weight every document with respect to the set of textual features (words that appear in the document). Hence we will apply few transformation as before approaching to LDA as shared below:

1.) Pre-processing

2.) Building Document Matrix – Bag of words model

3.) Finding frequent words

2.2.1 Pre-processing

Convertion of each document content to lower case and Various stops words, punctuation's, white spaces, numbers are removed from the SGML files is done in this step.

Code Used:

pre_processing_reuters ¡- $\mathrm{tm}_map(Reuters21578, content_transformer(tolower))$

$pre\_processing\_reuters <- tm_map(pre\_processing\_reuters, stripWhitespace)$

$pre\_processing\_reuters <- tm_map(pre\_processing\_reuters, removeWords, stopwords("english"))$

$pre\_processing\_reuters <- tm_map(pre\_processing\_reuters, removePunctuation)$

$pre\_processing\_reuters <- tm_map(pre\_processing\_reuters, removeNumbers)$

$pre\_processing\_reuters <- tm_map(pre\_processing\_reuters, stemDocument)$

Output shown in figure 13:

```
> #document term matrix - Bag Of Words(BOW)
> DTM <- DocumentTermMatrix(pre_processing_reuters)
> DTM
<<DocumentTermMatrix (documents: 21578, terms: 44753)>>
Non-/sparse entries: 1063195/964617039
Sparsity           : 100%
Maximal term length: 30
Weighting          : term frequency (tf)
```

Figure 13: Pre processing applied on Reuters-21578

2.2.2 Building Document Matrix – Bag of words model

Document-term matrix describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms. The implementation of the document matrix is done using TF-IDF. **Code Used:**
DTM ¡- DocumentTermMatrix(pre_processing_reuters)
Output shown in figure 14:

Also Some words do not carry too much meaning in the setting of the entire dataset, Therefore sometimes it is necessary to use the tf–idf(term frequency–inverse document frequency) instead of the frequencies of the term as entries, tf-idf measures the relative importance of a word to a document.
Code Used for applying TF-idf:
DTM1 < $-DocumentTermMatrix(pre\_processing\_reuters, control = list(weighting = weightTfIdf))$
Output shown in figure 14:

2.2.3 Finding frequent words

Now we need to find the frequent terms in a document-term or term-document matrix before applying LDA.
Code Used for frequent words:
x < $-findFreqTerms(DTM, 15)$
Output shown in figure 15:

Now moving to application of LDA on the preprocessed data.

```
> DTM1 <- DocumentTermMatrix(pre_processing_reuters, control = list(weighting = weightTfIdf))
Warning message:
In weighting(x) :
  empty document(s): 30 31 76 77 78 92 94 95 99 101 102 103 129 132 133 171 189 216 261 283 298 383 392 396 398 400 417 4
18 430 431 433 440 445 455 459 511 519 571 593 600 607 608 611 625 632 636 648 663 665 673 675 689 694 709 713 760 789 79
0 792 796 830 831 879 886 900 901 934 949 958 972 973 981 985 993 994 1035 1049 1050 1058 1091 1107 1109 1120 1143 1164 1
165 1166 1167 1180 1199 1204 1206 1247 1253 1256 1268 1274 1307 1324 1331 1346 1368 1411 1435 1436 1437 1454 1458 1462 14
87 1508 1514 1517 1521 1530 1538 1548 1549 1589 1592 1593 1612 1613 1615 1698 1699 1710 1713 1717 1719 1758 1780 1799 182
8 1841 1851 1855 1894 1939 1953 1961 1969 1987 1988 1989 2002 2004 2014 2032 2035 2037 2038 2041 2043 2060 2067 2069 2098
 2103 2105 2113 2122 2123 2124 2125 2136 2141 2145 2146 2157 2166 2169 2171 2174 2176 2178 2181 2183 2207 2216 2220 2244
 2263 2290 2302 2313 2327 2328 2330 2336 2341 2358 2377 2401 2402 2403 2424 2441 2442 2460 2465 2466 2497 2564 2582 2587
 2588 2595 2599 2600 2607 2621 2622  [... truncated]
> DTM1
<<DocumentTermMatrix (documents: 21578, terms: 44753)>>
Non-/sparse entries: 1063195/964617039
Sparsity           : 100%
Maximal term length: 30
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
```

Figure 14: tf-idf applied on reuters-21578

```
> x <- findFreqTerms(DTM,15)
> x
   [1] "aaa"            "aaminus"        "abandon"        "abandoned"      "abc"            "abdul"
   [7] "abe"            "abide"          "ability"        "able"           "abm"            "abolish"
  [13] "abolished"      "abolition"      "abroad"         "absence"        "absolute"       "absolutely"
  [19] "absorb"         "absorbed"       "abu"            "abuse"          "abuses"         "accelerate"
  [25] "accelerated"    "accelerating"   "acceleration"   "accept"         "acceptable"     "acceptance"
  [31] "acceptances"    "accepted"       "accepting"      "access"         "accident"       "accommodate"
  [37] "accompanied"    "accompanying"   "accord"         "accordance"     "according"      "accordingly"
  [43] "accords"        "account"        "accountants"    "accounted"      "accounting"     "accounts"
  [49] "accrued"        "accumulated"    "accurate"       "accused"        "accusing"       "achieve"
  [55] "achieved"       "achieving"      "acid"           "acknowledge"    "acknowledged"   "acquire"
  [61] "acquired"       "acquires"       "acquiring"      "acquisition"    "acquisitions"   "acre"
  [67] "acreage"        "acres"          "across"         "acrossboard"    "act"            "acted"
  [73] "acting"         "action"         "actions"        "active"         "actively"       "activities"
  [79] "activity"       "acts"           "actual"         "actually"       "adams"          "adapt"
  [85] "adb"            "adbs"           "add"            "added"          "adding"         "addition"
  [91] "additional"     "additionally"   "additions"      "address"        "addressed"      "addressing"
  [97] "adds"           "adequacy"       "adequate"       "adequately"     "adhere"         "adherence"
```

Figure 15: tf-idf applied on reuters-21578

```
> result <- LDA(DTM, k, method="Gibbs", control=list(iter = 25, verbose = 25, alpha = 0.1))
K = 20; V = 44753; M = 19042
Sampling 25 iterations!
Iteration 25 ...
Gibbs sampling completed!
> result
A LDA_Gibbs topic model with 20 topics.
```

Figure 16: LDA Result

2.3 Task 2 - Q3. Describe the whole procedure on applying LDA to this corpus to perform topic modeling.

Latent Dirichlet Allocation (LDA):
It is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics. LDA is an example of a topic model. The simplest way to think about LDA is as a probability model that connects documents with words and topics.

LDA is a generative probabilistic model, where the model observations are drawn from a probability density function. In this case documents are treated as a result of a generative process and each and every document is represented as a random mixture of latent topics.

Application of LDA:
As we have applied vocabulary pruning, we have empty rows in our DTM which is not liked by LDA. Hence we remove empty rows using the below steps: **Code Used:**
$\text{selection}_i d < -slam :: row_s ums(DTM) > 0$
$DTM < -DTM[selection_i d,]$

Now its good to apply LDA using LDA function which is present in install.packages('topicmodels')
Code Used:
$\text{result} < -LDA(DTM, k, method = "Gibbs", control = list(iter = 25, verbose = 25, alpha = 0.1))$
Output shown in figure 16:

Hence LDA fit a generative topic model which accounts for both the words which occur in a

```
> terms(result, 10)
      Topic 1   Topic 2    Topic 3   Topic 4   Topic 5      Topic 6      Topic 7  Topic 8   Topic 9  Topic 10
 [1,] "mln"     "said"     "billion" "said"    "said"       "said"       "cts"    "pct"     "said"   "said"
 [2,] "dlrs"    "reuter"   "dlrs"    "dlrs"    "tax"        "market"     "mln"    "will"    "reuter" "shares"
 [3,] "net"     "will"     "mln"     "will"    "budget"     "stock"      "net"    "mln"     "coffee" "dlrs"
 [4,] "loss"    "corp"     "said"    "debt"    "house"      "trading"    "loss"   "said"    "nil"    "stock"
 [5,] "year"    "inc"      "year"    "company" "president"  "exchange"   "shr"    "issue"   "will"   "inc"
 [6,] "profit"  "company"  "pct"     "reuter"  "government" "analysts"   "dlrs"   "ltd"     "drug"   "reuter"
 [7,] "cts"     "systems"  "bank"    "corp"    "reuter"     "new"        "reuter" "reuter"  "new"    "will"
 [8,] "reuter"  "new"      "quarter" "inc"     "reagan"     "futures"    "revs"   "bond"    "food"   "mln"
 [9,] "share"   "contract" "last"    "mln"     "party"      "will"       "profit" "bank"    "week"   "company"
[10,] "company" "computer" "first"   "also"    "chairman"   "securities" "note"   "manager" "year"   "common"
      Topic 11     Topic 12      Topic 13  Topic 14   Topic 15   Topic 16   Topic 17     Topic 18 Topic 19
 [1,] "said"       "said"        "said"    "pct"      "said"     "said"     "said"       "said"   "mln"
 [2,] "will"       "tonnes"      "mln"     "said"     "banks"    "dollar"   "trade"      "oil"    "said"
 [3,] "japanese"   "wheat"       "dlrs"    "year"     "bank"     "bank"     "will"       "prices" "oil"
 [4,] "government" "reuter"      "stg"     "february" "debt"     "rates"    "japan"      "dlrs"   "pct"
 [5,] "ltd"        "agriculture" "pct"     "january"  "new"      "rate"     "officials"  "price"  "year"
 [6,] "banks"      "sugar"       "year"    "reuter"   "loans"    "pct"      "states"     "will"   "gas"
 [7,] "pct"        "export"      "reuter"  "sales"    "foreign"  "market"   "told"       "market" "tonnes"
 [8,] "companies"  "price"       "will"    "rose"     "billion"  "currency" "countries"  "new"    "last"
 [9,] "bank"       "corn"        "company" "last"     "brazil"   "west"     "united"     "reuter" "production"
[10,] "firms"      "farm"        "sales"   "march"    "interest" "economic" "government" "opec"   "reuter"
      Topic 20
 [1,] "said"
 [2,] "company"
 [3,] "reuter"
 [4,] "inc"
 [5,] "corp"
 [6,] "will"
```

Figure 17: Top Words

collection of documents as well as the links between the documents. The goal of LDA is to infer the underlying topics, topic proportions, and topic assignments for every document. Here k refers to topics which is 20, and 25 is the total number of iterations. Now let's take a look at the 10 most likely terms within the term probabilities beta of inferred topics.

Code Used:

terms(result, 10)

Output for top words in figure 17:

Assigning the topics with more descriptive names than just numbers. So, simply concatenate the five most likely terms of each topic to a string that represents a pseudo-name for each topic. **Code Used:**

top5termsPerTopic ¡- terms(result, 5)

nameOfTopics ¡- apply(top5termsPerTopic, 2, paste, collapse=" ")

Output for descriptive names in figure 18:

2.4 Task 2 - Q4. Describe the parameter setting that you use in the LDA and explain their meanings.

LDA Helpin R Environment shown in figure 19:

LDA Code Used:

```
> nameOfTopics <- apply(top5termsPerTopic, 2, paste, collapse=" ")
> nameOfTopics
                            Topic 1                             Topic 2                             Topic 3
        "mln dlrs net loss year"          "said reuter will corp inc"       "billion dlrs mln said year"
                            Topic 4                             Topic 5                             Topic 6
    "said dlrs will debt company"    "said tax budget house president" "said market stock trading exchange"
                            Topic 7                             Topic 8                             Topic 9
         "cts mln net loss shr"           "pct will mln said issue"       "said reuter coffee nil will"
                           Topic 10                            Topic 11                            Topic 12
     "said shares dlrs stock inc"  "said will japanese government ltd" "said tonnes wheat reuter agriculture"
                           Topic 13                            Topic 14                            Topic 15
          "said mln dlrs stg pct"     "pct said year february january"          "said banks bank debt new"
                           Topic 16                            Topic 17                            Topic 18
    "said dollar bank rates rate"    "said trade will japan officials"       "said oil prices dlrs price"
                           Topic 19                            Topic 20
          "mln said oil pct year"       "said company reuter inc corp"
```

Figure 18: Assigning descriptive names

Latent Dirichlet Allocation

Description

Estimate a LDA model using for example the VEM algorithm or Gibbs Sampling.

Usage

```
LDA(x, k, method = "VEM", control = NULL, model = NULL, ...)
```

Arguments

`x`	Object of class `"DocumentTermMatrix"` with term-frequency weighting or an object coercible to a `"simple_triplet_matrix"` with integer entries.
`k`	Integer; number of topics.
`method`	The method to be used for fitting; currently `method = "VEM"` or `method= "Gibbs"` are supported.
`control`	A named list of the control parameters for estimation or an object of class `"LDAcontrol"`.
`model`	Object of class `"LDA"` for initialization.
`...`	Optional arguments. For `method = "Gibbs"` an additional argument `seedwords` can be specified as a matrix or an object of class `"simple_triplet_matrix"`; the default is `NULL`.

Figure 19: LDA Help in R environment

result $< -LDA(DTM, k, method = "Gibbs", control = list(iter = 25, verbose = 25, alpha = 0.1))$

2.4.1 Document Term Matrix (STM)

By using term-frequency weighting or an object coercible to a "simple_triplet_matrix" with integer entries.

2.4.2 K - Scalar Integer

A scalar integer indicating the number of latent topics for the model. In this case K is assigned to 20 which is based on number of topics.

2.4.3 Method

The method used is "Gibbs" for LDA.

2.4.4 Control

Named list of the control parameters for estimation of an object of class "LDAcontrol".

2.4.5 Alpha(not Mandatory)

Alpha-parameter is used to control the topic distribution. For even distribution of topics within a document an Higher alpha priors used for topics result. Low alpha priors ensure that the inference process distributes the probability mass on a few topics for each document.

2.5 Task 2 - Q5. Describe the output of your code and visualize the obtained topics in appropriate ways.

As of now by applying LDA_Gibbs on topic model we obtained with 20 topics which were shown in figure 15.

2.5.1 Selecting most likely terms

Code Used:
top5termsPerTopic $< -terms(result, 5)$
$nameOfTopics < -apply(top5termsPerTopic, 2, paste, collapse = "")$
Output shown in figure 20:

```
> nameOfTopics <- apply(top5termsPerTopic, 2, paste, collapse=" ")
> nameOfTopics
                                Topic 1                                 Topic 2
    "said market trading stg futures"            "said reuter gulf oil air"
                                Topic 3                                 Topic 4
        "pct said bank billion banks"            "pct said year last years"
                                Topic 5                                 Topic 6
          "pct billion mln year said"          "said trade oil japan will"
                                Topic 7                                 Topic 8
   "said will analysts president new"         "said tax sugar will reuter"
                                Topic 9                                Topic 10
           "pct mln will said reuter"               "mln cts net loss dlrs"
                               Topic 11                                Topic 12
         "said shares dlrs stock inc" "said banks spokesman reuter foreign"
                               Topic 13                                Topic 14
      "said will reuter company mln"         "said debt billion dlrs bank"
                               Topic 15                                Topic 16
    "said shares company dlrs reuter"           "said debt reuter inc corp"
                               Topic 17                                Topic 18
      "said will company reuter corp"          "said dollar pct rates bank"
                               Topic 19                                Topic 20
         "dlrs mln said year quarter"           "said will reuter bank new"
```

Figure 20: Top five arranged

2.5.2 Applying posterior

Determining the posterior probabilities of the topics for each document and of the terms for each topic for a fitted topic model.

Code Used:

tmposterior $< -posterior(result)$

$attributes(tmposterior)$

Output:

$names

"terms" "topics"

2.5.3 observing sample data

let us take a look at the content of eight sample documents by applying lappy function over the whole prepreprocessed corpus.

Code Used:

examples $< -c(2, 150, 250, 450, 850, 1050, 1250, 1450)$

lapply(pre_processing_reuters[examples], as.character)

Output of Content in examples shown in figure 21:

2.5.4 Processing content for probabilities

Getting posterior probabilities and applying to dataframe to observe the repetition of contents in the topic model.

tpExamples $< -theta[examples,]$

```
> examples <- c(2, 150, 250, 450, 850, 1050, 1250, 1450)
> lapply(pre_processing_reuters[examples], as.character)
$`2`
[1] "standard oil co  bp north america inc said  plan  form  venture  manage  money market borrowing  investment a
ies   companies bp north america   subsidiary  british petroleum co plc bp  also owns   pct interest  standard oil
ure will  called bpstandard financial trading  will  operated  standard oil   oversight   joint management committ
ter "

$`150`
[1] "parlux fragrances inc said  completed  offering   unis   dlrs   underwriters rc stamm  co  rosenkrantz lyon
nc  unit consists  two shares  common stock  one redeemable warrant  entitles  holder  buy  additional common shar
 dlrs  feb   feb   company creates  markets fragrances  beauty products including  anne klein line  department
lty stores reuter "

$`250`
[1] " reserve bank said  cancelled  regular weekly treasury bill tender scheduled  march   said   statement  forec
 net cash withdrawal   system   settlement week cash flows   government  expected   offset cash injections  added
 said  expects  conduct open market operations   week   cash balances  fluctuate around  mln nz dlrs reuter "

$`450`
[1] "canadas industrial product price index rose  pct  january  falling  pct    two previous months statistics ca
aid  rise  led  price gains  papers pharmaceuticals  petroleum  coal products price declines  recorded  meat produ
mber  motor vehicles   year  year basis  federal agency said  index fell  pct  january  largest yearly decline  re
euter "
```

Figure 21: Content in 8 documents

$colnames(tpExamples) < -nameOfTopics$
$vizDataFrame < -melt(cbind(data.frame(tpExamples), document = factor(1 : N)), variable.name =$
$"topic", id.vars = "document")$
Output of vizDataFrame in figure 22:

2.5.5 Plotting Results

Obtained vizDataFrame is plotted by using ggplot for easy analysis of topic occurance and its value.
Code Used:
ggplot(data = vizDataFrame, aes(topic, value, fill = document), ylab = "proportion") + $geom_bar(stat =$
$"identity") + theme(axis.text.x = element_text(angle = 90, hjust = 1)) +$
$coord_flip() + facet_wrap(\ document, ncol = N)$

output of ggplot shown in figure 23:

ggplot display that most of the topics have occured in the documents 1[2], 7[1250] and 8[1450]. In document 2[150] only "min.cts.loss.net.dlrs" has highest value of 0.65 and have presence of "said.will.bank.mln.reuter", "said.company.shares.stock.offer" and "pct.said.will.ltd.mln". Similar observation can be observed from ggplot.

2.6 Task 2 - Q6. Attach your code at the end of the report.

Code attached at last.

```
> vizDataFrame <- melt(cbind(data.frame(tpExamples), document = factor(1:N)), variable.name = "topic", id.vars =
 "document")
> vizDataFrame
    document                              topic        value
1          1          billion.pct.dlrs.mln.said 0.0022222222
2          2          billion.pct.dlrs.mln.said 0.0062500000
3          3          billion.pct.dlrs.mln.said 0.0007936508
4          4          billion.pct.dlrs.mln.said 0.0229166667
5          5          billion.pct.dlrs.mln.said 0.0003636364
6          6          billion.pct.dlrs.mln.said 0.0005617978
7          7          billion.pct.dlrs.mln.said 0.0055555556
8          8          billion.pct.dlrs.mln.said 0.0083333333
9          1       said.company.reuter.inc.will 0.0022222222
10         2       said.company.reuter.inc.will 0.0062500000
11         3       said.company.reuter.inc.will 0.0087301587
12         4       said.company.reuter.inc.will 0.0020833333
13         5       said.company.reuter.inc.will 0.0003636364
14         6       said.company.reuter.inc.will 0.0623595506
15         7       said.company.reuter.inc.will 0.0055555556
16         8       said.company.reuter.inc.will 0.0083333333
17         1          said.will.bank.mln.reuter 0.4688888889
18         2          said.will.bank.mln.reuter 0.0687500000
19         3          said.will.bank.mln.reuter 0.0007936508
20         4          said.will.bank.mln.reuter 0.0020833333
21         5          said.will.bank.mln.reuter 0.0040000000
```

Figure 22: vizDataframes Data

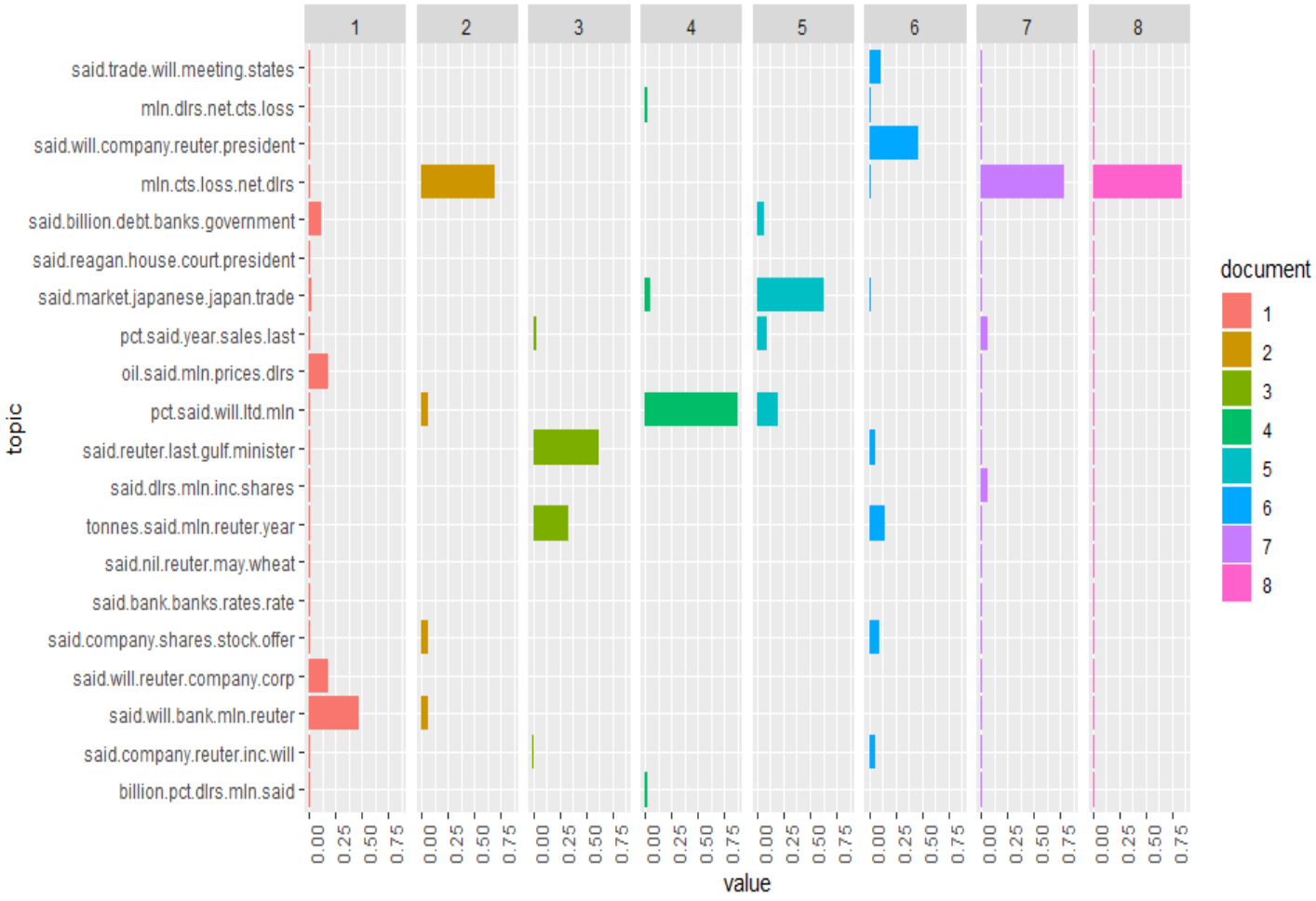


Figure 23: ggplot of vizDataFrame

2.7 References:

1.) https://rstudio-pubs-static.s3.amazonaws.com/132792$_8$64e3813b0ec47cb95c7e1e2e2ad83e7.$html$
2.) $https://miguelmalvarez.com/2015/03/20/classifying-reuters-21578-collection-with-python-representing-the-data/$
3.) $https://www.rdocumentation.org/packages/lda/versions/1.4.2/topics/rtm.collapsed.gibbs.sampler$
4.) $https://tm4ss.github.io/docs/Tutorial_6_Topic_Models.html6_topic_proportions_over_time.$

3 Task 3 – Sentiment Analysis

Sometimes known as opinion mining, sentiment analysis is the process of contextually mining text to identify and categorize the subjective opinions expressed by the writers. Normally it is used to determine whether the writer's attitude towards a particular topic or product, etc. is positive, negative, or neutral. It is also often use by businesses to help them understand the social sentiment of their brand, product or services while monitoring online conversations. In the context of a twitter sentiment analysis, at its simplest, sentiment analysis quantifies the mood of a tweet or comment by counting the number of positive and negative words. By subtracting the negative from the positive, the sentiment score is generated.

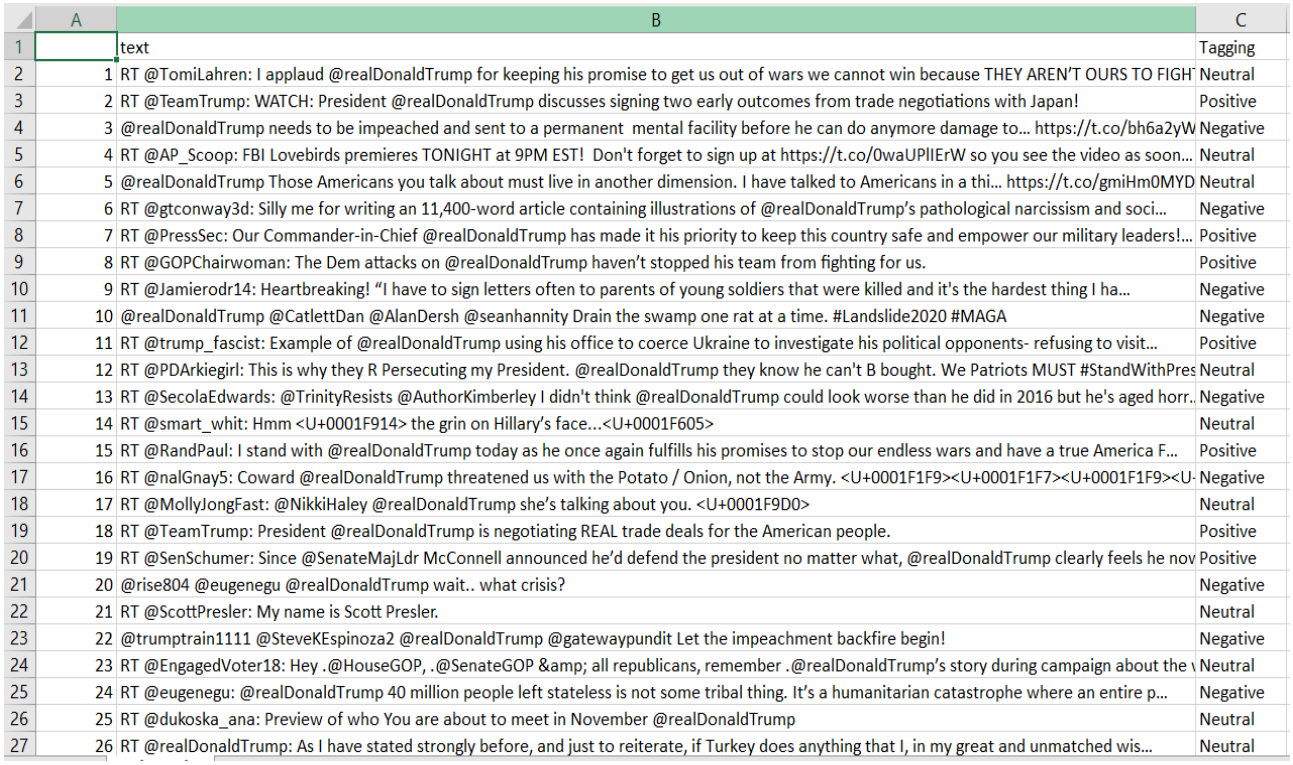
3.1 Task 3 – Q1. Describe the procedure of collecting the tweets and manually tagging them.

3.1.1 Collecting Tweets from twitter

1.) Create a twitter account if you do not already have one.
2.) Go to https://dev.twitter.com/apps and log in with your twitter credentials.
3.) Click "Create New App" for assignment.
4.) Fill out the form and agree to the terms. Click on Create your Twitter application. Put your Callback URL as http://localhost:1410.
5.) On the next page, click the "Keys and Access Tokens" tab along the top, then scroll all the way down until you see the section "Your Access Token"
6.) Click the button "Create My Access Token" we can get access token to get handshake between twitter and R.
7.) Finally we actually get to use the API to get data. Using the following code below, we can get twitter data.

Code used for R and twitter authentication:

consumerAPIKey $<-$"$4XsqFw3PwDZWRNUtyF5WSuFlz$"
$consumerAPISecretKey <-$"$CuOT4iPxlyoMuk5cconiUpF0eX8e7LQ1ltxs27XbmYWCNJ8b6O$"



	A	B	C
1		text	Tagging
2	1	RT @TomiLahren: I applaud @realDonaldTrump for keeping his promise to get us out of wars we cannot win because THEY AREN'T OURS TO FIGH	Neutral
3	2	RT @TeamTrump: WATCH: President @realDonaldTrump discusses signing two early outcomes from trade negotiations with Japan!	Positive
4	3	@realDonaldTrump needs to be impeached and sent to a permanent mental facility before he can do anymore damage to... https://t.co/bh6a2yW	Negative
5	4	RT @AP_Scoop: FBI Lovebirds premieres TONIGHT at 9PM EST! Don't forget to sign up at https://t.co/0waUPlIErW so you see the video as soon...	Neutral
6	5	@realDonaldTrump Those Americans you talk about must live in another dimension. I have talked to Americans in a thi... https://t.co/gmiHm0MYD	Neutral
7	6	RT @gtconway3d: Silly me for writing an 11,400-word article containing illustrations of @realDonaldTrump's pathological narcissism and soci...	Negative
8	7	RT @PressSec: Our Commander-in-Chief @realDonaldTrump has made it his priority to keep this country safe and empower our military leaders!...	Positive
9	8	RT @GOPChairwoman: The Dem attacks on @realDonaldTrump haven't stopped his team from fighting for us.	Positive
10	9	RT @Jamierodr14: Heartbreaking! "I have to sign letters often to parents of young soldiers that were killed and it's the hardest thing I ha...	Negative
11	10	@realDonaldTrump @CatlettDan @AlanDersh @seanhannity Drain the swamp one rat at a time. #Landslide2020 #MAGA	Negative
12	11	RT @trump_fascist: Example of @realDonaldTrump using his office to coerce Ukraine to investigate his political opponents- refusing to visit...	Positive
13	12	RT @PDArkiegirl: This is why they R Persecuting my President. @realDonaldTrump they know he can't B bought. We Patriots MUST #StandWithPres	Neutral
14	13	RT @SecolaEdwards: @TrinityResists @AuthorKimberley I didn't think @realDonaldTrump could look worse than he did in 2016 but he's aged horr..	Negative
15	14	RT @smart_whit: Hmm <U+0001F914> the grin on Hillary's face...<U+0001F605>	Neutral
16	15	RT @RandPaul: I stand with @realDonaldTrump today as he once again fulfills his promises to stop our endless wars and have a true America F...	Positive
17	16	RT @nalGnay5: Coward @realDonaldTrump threatened us with the Potato / Onion, not the Army. <U+0001F1F9><U+0001F1F7><U+0001F1F9><U+	Negative
18	17	RT @MollyJongFast: @NikkiHaley @realDonaldTrump she's talking about you. <U+0001F9D0>	Neutral
19	18	RT @TeamTrump: President @realDonaldTrump is negotiating REAL trade deals for the American people.	Positive
20	19	RT @SenSchumer: Since @SenateMajLdr McConnell announced he'd defend the president no matter what, @realDonaldTrump clearly feels he nov	Positive
21	20	@rise804 @eugenegu @realDonaldTrump wait.. what crisis?	Negative
22	21	RT @ScottPresler: My name is Scott Presler.	Neutral
23	22	@trumptrain1111 @SteveKEspinoza2 @realDonaldTrump @gatewaypundit Let the impeachment backfire begin!	Negative
24	23	RT @EngagedVoter18: Hey .@HouseGOP, .@SenateGOP & all republicans, remember .@realDonaldTrump's story during campaign about the	Neutral
25	24	RT @eugenegu: @realDonaldTrump 40 million people left stateless is not some tribal thing. It's a humanitarian catastrophe where an entire p...	Negative
26	25	RT @dukoska_ana: Preview of who You are about to meet in November @realDonaldTrump	Neutral
27	26	RT @realDonaldTrump: As I have stated strongly before, and just to reiterate, if Turkey does anything that I, in my great and unmatched wis...	Neutral

Figure 24: Manual tagging on excel data

$accessTokenKey <- "1179219778579226630 - hFXeJSppsWWY7JllGrtjOmkjvMUQap"$
$accessTokenSecretKey <- "utFFVPO8J711V6JL1JCJx0HNFgjqyCKQdniCLqBWspwmz"$
$reqURL <- "https://api.twitter.com/oauth/request_token"$
$accessURL <- "https://api.twitter.com/oauth/access_token"$
$authURL <- "https://api.twitter.com/oauth/authorize"$
$setup\_twitter\_oauth(consumerAPIKey, consumerAPISecretKey, accessTokenKey, accessTokenSecretKe$

3.1.2 Manual Tagging

From above section we have successfull hand shake between R and twitter, now we will be focusing on collecting the data from twitter to a excel sheet for Manual tagging.

Code used for extracting Excel data from twitter:

$Tweets <- searchTwitter("@realDonaldTrump", n = 200, lang = "en")$
$Tweets <- twListToDF(Tweets)$
$write.csv(Tweets, file = "twitterList.csv")$

From above code we can observe that tweets of donald trump are collected in tweets object and then they are converted to dataframe using twListToDF() function. Then the data frames are extracted to a excel and tagging manual tagging have been done on the excel data as positive, Negitive and Neutral for the tweets made by trump as shown below.

```
> TrumpResults<- read.csv("twitterList.csv")
> # count the tagging done in file
> countTagging <- table(TrumpResults$Tagging)
> countTagging

Negative  Neutral Positive
      75       50       75
> #Corpus to know details in file
> corpusVector <- Corpus(VectorSource(TrumpResults$text))
> corpusVector
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 200
```

Figure 25: Tagging types and count

Each and every tweet of trump which are collected are tagged to one of the options like positive, negative and neutral as shown in figure 24.

3.2 Task 3 – Q2. Describe the statistics of the obtained data set.

Before applying statistic read data from the .csv five with code shown below
TrumpResults $< -read.csv("twitterList.csv")$
Now data can be explore to get details of manual tagging done by using below code.
count the tagging done in file
countTagging $< -table(TrumpResults\$Tagging)$
$Corpustoknowdetailsinfilelikenumberofdocuments$
$corpusVector < -Corpus(VectorSource(TrumpResults\$text))$

Output for above code shown in figure 25:

Hence there are total 200 observations with Negative - 75, Neutral - 50 and Positive - 75.

3.3 Task 3 – Q3. Describe how you represent each tweet for classification.

By proper observation of figure 24 which consists of excel data where tweets consists of noises other than tweets like @,#,— etc., which makes difficult to apply any classifier on the tweets. So below two steps are done.

3.3.1 pre-processing data

The process involved is as follows:
1.) Firstly, we will transform the dataset and remove all the extra URL that are the part of the text in the tweets, remove numbers, stop words.
2.) Then we will remove all the extra spaces from the text dataset.
3.) Then we will remove all the extra symbols like @, , %, $, ' $<, >$ *etc.*
Code Used:

```
> inspect(Tweets_Pre_Processed[1:10])
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 10

 [1] rt tomilahren  applaud realdonaldtrump  keeping  promise  get us   wars   win   aren't   fight f…
 [2] rt teamtrump watch president realdonaldtrump discusses signing two early outcomes  trade negotiations  japan
 [3] realdonaldtrump needs   impeached  sent   permanent mental facility   can  anymore damage …
 [4] rt apscoop fbi lovebirds premieres tonight  pm est dont forget  sign     see  video  soon…
 [5] realdonaldtrump  americans  talk  must live  another dimension   talked  americans   thi…
 [6] rt gtconwayd silly   writing  word article containing illustrations  realdonaldtrump's pathological narcissis
 [7] rt presssec  commanderinchief realdonaldtrump  made   priority  keep  country safe  empower  military leaders
 [8] rt gopchairwoman  dem attacks  realdonaldtrump haven't stopped  team  fighting  us google just agreed  create
 [9] rt jamierodr heartbreaking "   sign letters often  parents  young soldiers   killed    hardest thing  ha…
[10] realdonaldtrump catlettdan alandersh seanhannity drain  swamp one rat   time landslide maga
```

Figure 26: Pre processed data

$pre - processingthetweets$
$Tweets_Pre_Processed$¡- $\mathrm{tm}_map(corpusVector, content_transformer(tolower))$
$url <- content_transformer(function(x)gsub("(f|ht)tp(s?) : //$
$S + ", "", x, perl = T))$
$Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, url)$
$Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, removePunctuation)$
$Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, stripWhitespace)$
$Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, removeWords, stopwords("english"))$
$Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, removeNumbers)$
$Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, stemDocument)$
$inspect(Tweets_Pre_Processed[1 : 10])$
Output shown in figure - 26

3.3.2 Building Document Term Matrix:

Document-term matrix is a matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms. The implementation of the document matrix is done using TF-IDF.
Code Used:dtm $<- DocumentTermMatrix(pre\_process\_tweets)$

3.3.3 Splitting testing and training data

Now we can split the data into training and testing data by using the below code.
partitioning of training and testing data is done by below code:
training_Data $<- TrumpResults$Tagging[1:160] trainig set
$test_Data <- TrumpResults$Tagging[161:200] test set

Some words do not carry too much meaning in the setting of the entire corpus Therefore sometimes it is necessary to use the tf–idf(term frequency–inverse document frequency) instead of the frequencies of the term as entries, tf-idf measures the relative importance of a word to a document.

Now here we will divide the training and test data dataset from the TF-IDF which is as shown below in code.

DTM_TRAIN_TFIDF $< -DocumentTermMatrix(Tweets\_Pre\_Processed, control = list(weighting = weightTfIdf))$
$DTM\_TRAINING\_SET < -DTM\_TRAIN\_TFIDF[1:160,]$
$DTM\_TEST\_SET < -DTM\_TRAIN\_TFIDF[161:200,]$

3.4 Task 3 – Q4. For each classifier, describe its working principle, classification procedure, and parameter setting.

3.4.1 Naive Bayes classifier

As Naive Bayes classifier is explained in section 1.3.2 for Question 1, As concept remains same will be skipping and talking about Classification Procedure and Parameter Setting for Naïve Bayes Classifier.
1.) Once the dataset is partitioned into training (80%) and test (20%).
2.) Now we will call the naïve Bayes function for performing classification with Laplace smoothing = 1 on the training data set.
3.) Now we will predict the result on the test data.
4.) After this we will build the confusion matrix for the prediction result and actual result. We will call the confusion Matrix function to compute the result.
Code used:
NAIVE_BAYES_RESULT $< -naiveBayes(as.matrix(DTM_TRAININGSET), training_Data, laplace = 1)$
$PREDICTED\_RESULT < -predict(NAIVE\_BAYES\_RESULT, as.matrix(DTM\_TEST\_SET))$
Output shown in figure - 27

3.4.2 Support Vector Machine - SVM:

SVM is a supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). 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 the side of the gap on

```
             mostly
training_Data [,1] [,2]
     Negative    0    0
     Neutral     0    0
     Positive    0    0

             thousands
training_Data [,1] [,2]
     Negative    0    0
     Neutral     0    0
     Positive    0    0

> # predict with testdata
> PREDICTED_RESULT <- predict (NAIVE_BAYES_RESULT,as.matrix(DTM_TEST_SET))
> PREDICTED_RESULT
 [1] Neutral Neutral Neutral Neutral Neutral Neutral Neutral Neutral Neutral
[10] Neutral Neutral Neutral Neutral Neutral Neutral Neutral Neutral Neutral
[19] Neutral Neutral Neutral Neutral Neutral Neutral Neutral Neutral Neutral
[28] Neutral Neutral Neutral Neutral Neutral Neutral Neutral Neutral Neutral
[37] Neutral Neutral Neutral Neutral
Levels: Negative Neutral Positive
```

Figure 27: Navie predicted result

which they fall. If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them.

Classification Procedure and Parameter Setting for Support Vector Machine:

Once the dataset is partitioned into training (80%) and test (20%). Now we will call the SVM function for performing classification with setting parameters like Gamma(parameter needed for all kernels accept linear default = 1), cost(cost of constraints violation, default=1), and Kernel function(Selected as "radical") to convert non-linearity to linearity. Now we will predict the result on the test data.After this we will build the confusion matrix for the prediction result and actual result. We will call the confusion Matrix function to compute the result.

Code Used:

SVM_TRAIN $< -svm(as.matrix(DTM\_TRAINING\_SET), training_Data, type =' C-classification', ker$
"$radial$", $cost = 10, gamma = 0.5$)

SVM_TRAIN

$SVM\_PREDICT < -predict(SVM\_TRAIN, as.matrix(DTM\_TEST\_SET))$

$SVM\_PREDICT$

Output shown in figure - 28

Hence from figure we can know that 147 vectors are created by Support Vector Machine.

```
> SVM_TRAIN

Call:
svm.default(x = as.matrix(DTM_TRAINING_SET), y = training_Data, type = "C-classification", kernel = "radial",
    gamma = 0.5, cost = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10

Number of Support Vectors:  147

> #Prediction Result
> SVM_PREDICT <- predict(SVM_TRAIN, as.matrix(DTM_TEST_SET))
> SVM_PREDICT
     161      162      163      164      165      166      167      168      169      170      171      172      17
Negative  Neutral Positive Positive Positive  Neutral Positive Negative Negative Negative  Neutral Positive Negati
     176      177      178      179      180      181      182      183      184      185      186      187      18
Negative Negative Positive Negative Positive Negative Negative Negative Negative Negative Negative Positive Positi
     191      192      193      194      195      196      197      198      199      200
Negative Positive Negative Negative Positive Negative  Neutral Positive Negative Negative
Levels: Negative Neutral Positive
```

Figure 28: Support Vector Machine Result

```
> confMatrixNB <- confusionMatrix(xtab)
> confMatrixNB
Confusion Matrix and Statistics

          Predictions
Actual     Negative Neutral Positive
  Negative        0      15        0
  Neutral         0       9        0
  Positive        0      16        0
```

Figure 29: Confusion Matrix for Naive Bayes

3.5 Task 3 – Q5. For each classifier, report the classification accuracy and plot the confusion matrix.

3.5.1 Classification accuracy and confusion matrix for Naive Bayes

Code Used for Confusion matrix and accuracy for Naive Bayes:

xtab $< -table("Actual" = test_Data, "Predictions" = PREDICTED\_RESULT)$

$confMatrixNB < -confusionMatrix(xtab)$

$confMatrixNB$

Output of confusion matrix for Naive Bayes classification is shown in figure - 29

Output of Classification accuracy for Naive Bayes classification is shown in figure - 30

```
Overall Statistics

               Accuracy : 0.225
                 95% CI : (0.1084, 0.3845)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Negative Class: Neutral Class: Positive
Sensitivity                       NA          0.225              NA
Specificity                    0.625             NA             0.6
Pos Pred Value                    NA             NA              NA
Neg Pred Value                    NA             NA              NA
Prevalence                     0.000          1.000             0.0
Detection Rate                 0.000          0.225             0.0
Detection Prevalence           0.375          0.225             0.4
Balanced Accuracy                 NA             NA              NA
```

Figure 30: Classification accuracy for Naive Bayes

```
> confMatrixSVM <- confusionMatrix(xSVM)
> confMatrixSVM
Confusion Matrix and Statistics

          Predictions
Actual     Negative Neutral Positive
  Negative       11       0        4
  Neutral         8       1        0
  Positive        3       4        9
```

Figure 31: Confusion Matrix for Support Vector Machine

3.5.2 Classification accuracy and confusion matrix for Support Vector Machine

Code Used for Confusion matrix and accuracy for Support Vector Machine:

xSVM $< -table("Actual" = test\_Data, "Predictions" = SVM\_PREDICT)$

$xSVM$

$confMatrixSVM < -confusionMatrix(xSVM)$

$confMatrixSVM$

Output of confusion Matrix for Support Vector Machine is shown in figure - 31

Output of Classification accuracy for Support Vector Machine is shown in figure - 32

Hence from the figures we can observe that: Accuracy for Naive Bayes classification is : 22.5% Accuracy for Support vector machine is : 52.5%

```
Overall Statistics

               Accuracy : 0.525
                 95% CI : (0.3613, 0.6849)
    No Information Rate : 0.55
    P-Value [Acc > NIR] : 0.684414

                  Kappa : 0.2527

 Mcnemar's Test P-Value : 0.006909

Statistics by Class:

                     Class: Negative Class: Neutral Class: Positive
Sensitivity                   0.5000         0.2000          0.6923
Specificity                   0.7778         0.7714          0.7407
Pos Pred Value                0.7333         0.1111          0.5625
Neg Pred Value                0.5600         0.8710          0.8333
Prevalence                    0.5500         0.1250          0.3250
Detection Rate                0.2750         0.0250          0.2250
Detection Prevalence          0.3750         0.2250          0.4000
Balanced Accuracy             0.6389         0.4857          0.7165
```

Figure 32: Classification accuracy for Support Vector Machine

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	**TP**	**FN**
Class 2 Actual	**FP**	**TN**

Figure 33: class prediction

3.6 Task 3 – Q6. (CSCI946 only) When you run more than one classifiers, report which classifier performs better than the others and describe the methods you use to reach this conclusion.

Classifier performance can be easily predicted by checking and comparing the percision, recall and f measure of navie bayes and support vector machines.

All TP(True Positive), FN(False Negitive), FP(False Positive) and TN(True Negitive) belongs to figure 33.

Precision = TP/TP+FP

Recall = TP/TP+FN

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Accuracy = TP+TN/TP+FP+FN+TN

All the above formulas are collected in R and ouputs are shown below.

3.6.1 Naive Bayes classification - Precision, Recall, Fmeasure, and Accuracy is output shown in figure - 34

```
> #Performance for Naive Bayes - Precison, Recall, Fmeasure
> n1 = sum(xtab) # number of instances
> nc1 = nrow(xtab) # number of classes
> diag1 = diag(xtab) # number of correctly classified instances per class
> rowsums1 = apply(xtab, 1, sum) # number of instances per class
> colsums1 = apply(xtab, 2, sum) # number of predictions per class
> p1 = rowsums1 / n1 # distribution of instances over the actual classes
> q1 = colsums1 / n1 # distribution of instances over the predicted classes
> accuracy1 = sum(diag1) / n1
> precisionNB = diag1 / colsums1
> recallNB = diag1 / rowsums1
> f1NB = 2 * precisionNB * recallNB / (precisionNB + recallNB)
> data.frame(precisionNB, recallNB, f1NB, accuracy1)
          precisionNB recallNB      f1NB accuracy1
Negative          NaN        0       NaN     0.225
Neutral         0.225        1 0.3673469     0.225
Positive          NaN        0       NaN     0.225
```

Figure 34: class prediction

```
> n2 = sum(xSVM) # number of instances
> nc2 = nrow(xSVM) # number of classes
> diag2 = diag(xSVM) # number of correctly classified instances per class
> rowsums2 = apply(xSVM, 1, sum) # number of instances per class
> colsums2 = apply(xSVM, 2, sum) # number of predictions per class
> p2 = rowsums2 / n2 # distribution of instances over the actual classes
> q2 = colsums2 / n2 # distribution of instances over the predicted classes
> accuracy2 = sum(diag2) / n2
> precisionSVM = diag2 / colsums2
> recallSVM = diag2 / rowsums2
> f1SVM = 2 * precisionSVM * recallSVM / (precisionSVM + precisionSVM)
> data.frame(precisionSVM, recallSVM, f1SVM, accuracy2)
          precisionSVM recallSVM     f1SVM accuracy2
Negative     0.5000000 0.7333333 0.7333333     0.525
Neutral      0.2000000 0.1111111 0.1111111     0.525
Positive     0.6923077 0.5625000 0.5625000     0.525
```

Figure 35: class prediction

3.6.2 Support Vector Machine - Precision, Recall, Fmeasure, and Accuracy is output shown in figure - 35

By observing figure 34 and 35 we can easily say that Support vector Machine has hight accuracy of 52.5% which is far greater than 22.5% of Naive Bayes classification. Hence we can conclude that Support vector machine is better than the Naive Bayes Classifier.

3.7 Task 3 – Q7. Attach your code at the end of the report.

Code attached at last

3.8 References

1.) https://medium.com/@GalarnykMichael/accessing-data-from-twitter-api-using-r-part1-b387a1c7d3e

3.) https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

4.) https://eight2late.wordpress.com/2017/02/07/a-gentle-introduction-to-support-vector-machines-using-r

5.) https://opensource.com/article/17/6/collecting-and-mapping-twitter-data-using-r

4 Code

4.1 Task1 - Code

Task 1 has three parts which are displayed in figures 36, 37 and 38.

4.2 Task2 - Code

Task 2 has two parts which are displayed in figures 39 and 40.

4.3 Task3 - Code

Task 3 has three parts which are displayed in figures 41, 42 and 43.

All the code is displayed in images as used latex for documentation. Based on time concern images were shared of the code. Code is submitted in moodle.

```
"""
Student Name: Venkata Sandeep Kumar Karamsetty
Student ID: 6228975
Student Mail ID: vskk033@uowmail.edu.au
"""
import os
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix
from sklearn import metrics
import matplotlib.pyplot as plt
from scipy import interp
from sklearn.metrics import roc_curve, auc
from itertools import cycle
import itertools
import scipy.sparse as sps

#Confusion Matrix function

def plot_confusion_matrix(cm, classes,normalize=True,title='Confusion matrix',cmap=plt.cm.Blues):
    """
        This function prints and plots the confusion matrix.
        Normalization can be applied by setting `normalize=True`.
        """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    plt.figure(figsize = (20,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),horizontalalignment="center",color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True label')
```

Figure 36: Task1 - part1

```
    plt.xlabel('Predicted label')
    plt.grid('false')
    plt.show()

os.chdir("C:/Users/venka/Desktop/semester - 3/Big data Analytics/Assignment 2/PythonApplication1/PythonApplication1")

#Read the data
trainingData = pd.read_table('train.data',sep=" ",names=['docId', 'wordId', 'count'])
print(trainingData)
testingData = pd.read_table('test.data',sep=" ",names=['docId', 'wordId', 'count'])
print(testingData)
traininglabel = pd.read_table('train.label',sep=" ",names=['labelId'])
print(traininglabel)
testinglabel = pd.read_table('test.label',sep=" ",names=['labelId'])
print(testinglabel)
trainingmap = pd.read_table('train.map',sep=" ",names=['labelName','labelId'])
print(trainingmap)
testingmap = pd.read_table('test.map',sep=" ",names=['labelName','labelId'])
print(testingmap)

#Merging the training and testing sets
traininglabel['docId'] = list(range(1,11270))
train_label = traininglabel.merge(trainingmap, on="labelId")
train_final = train_label.merge(trainingData, on="docId")
train_final = train_final[["docId","wordId","count","labelName"]]
print("Merged training data shown below:")
print(train_final)

testinglabel['docId'] = list(range(1,7506))
test_label= testinglabel.merge(testingmap, on="labelId")
test_final = test_label.merge(testingData, on="docId")
test_final = test_final[["docId","wordId","count","labelName"]]
print("Merged testing data shown below:")
print(test_final)


#Document matrix for testing and training
mat = sps.coo_matrix((trainingData["count"].values, (trainingData["docId"].values-1, trainingData["wordId"].values-1)))
training_data_matrix = mat.tocsc()
training_data_matrix.shape
print(training_data_matrix)

test_mat  = sps.coo_matrix((testingData["count"].values, (testingData["docId"].values-1, testingData["wordId"].values-1)))
testing_data_matrix = test_mat.tocsc()
```

Figure 37: Task1 - part2

```
testing_data_matrix = testing_data_matrix[:,:training_data_matrix.shape[1]]
print(testing_data_matrix)

#Naive Bayes classifier
naiveBayesclassifier = MultinomialNB(alpha=.01, class_prior=None, fit_prior=True)
naiveBayesclassifier.fit(training_data_matrix, traininglabel["labelId"])
print("naive Bayes Classifier")
print(naiveBayesclassifier)

# Predict the test results
prediction = naiveBayesclassifier.predict(testing_data_matrix)
print(prediction)
#Printing accuracy
print("Accuracy = {0}".format(metrics.f1_score(testinglabel["labelId"], prediction, average='macro')* 100))

#Confusion Matrix
cm = confusion_matrix(testinglabel["labelId"], prediction)
plot_confusion_matrix(cm, classes=trainingmap["labelName"],title='Confusion matrix')
```

Figure 38: Task1 - part3

```r
#Task 2 -
#Name: Venkata Sandeep Kumar Karamsetty
#Student ID: 6228975
#Mail ID: vskk033@uowmail.edu.au

#install.packages('XML')
#install.packages("tm.corpus.Reuters21578", repos = "http://datacube.wu.ac.at")
#install.packages("wordcloud")
#install.packages('topicmodels')
require(topicmodels)
library(tm.corpus.Reuters21578)
library(wordcloud)
library(lda)
library(ggplot2)
library(reshape2)

data(Reuters21578)

#Summarizing the data of Reuters
summary(Reuters21578)

#observation on reuters
cat(content(Reuters21578[[7]]))
cat(content(Reuters21578[[3]]))

#Pre-Processing the Reuters-21578
pre_processing_reuters <- tm_map(Reuters21578,  content_transformer(tolower))
pre_processing_reuters <- tm_map(pre_processing_reuters, stripWhitespace)
pre_processing_reuters <- tm_map(pre_processing_reuters, removeWords, stopwords("english"))
pre_processing_reuters <- tm_map(pre_processing_reuters, removePunctuation)
pre_processing_reuters <- tm_map(pre_processing_reuters, removeNumbers)
pre_processing_reuters <-tm_map(pre_processing_reuters,stemDocument)
pre_processing_reuters

#document term matrix - Bag Of Words(BOW)
DTM <- DocumentTermMatrix(pre_processing_reuters)
DTM

#document term matrix TF-IDF
DTM1 <- DocumentTermMatrix(pre_processing_reuters, control = list(weighting = weightTfIdf))
DTM1

#reducing the dimensions of the DTM
#By removing the less frequent terms such that the sparsity is less than 0.95
REVISED_DTM = removeSparseTerms(DTM1, 0.99)
REVISED_DTM
#frequency - Simple word cloud
x <- findFreqTerms(DTM,15)
x


#LDA
# due to vocabulary pruning, we have empty rows in our dtm
#LDA does not like this. So we remove those docs from the dtm and the metadata
selection_id <- slam::row_sums(DTM) > 0
```

Figure 39: Task2 - part1

```
DTM <- DTM[selection_id, ]

#number of topics
k <- 20

# compute the LDA model, inference via 25 iterations of Gibbs sampling
result <- LDA(DTM, k, method="Gibbs", control=list(iter = 25, verbose = 25, alpha = 0.1))
result

#Let's take a look at the 10 most likely terms within the term probabilities beta of the inferred topics.
terms(result, 10)

#five most likely terms of each topic to a string
top5termsPerTopic <- terms(result, 5)
nameOfTopics <- apply(top5termsPerTopic, 2, paste, collapse=" ")
nameOfTopics
# have a look a some of the results (posterior distributions)
tmposterior <- posterior(result)

# format of the resulting object
attributes(tmposterior)

#Let us first take a look at the contents of eight sample documents:
examples <- c(2, 150, 250, 450, 850, 1050, 1250, 1450)
lapply(pre_processing_reuters[examples], as.character)

theta <- tmposterior$topics
theta
N <- length(examples)
N

# get topic proportions form example documents
tpExamples <- theta[examples,]
colnames(tpExamples) <- nameOfTopics
vizDataFrame <- melt(cbind(data.frame(tpExamples), document = factor(1:N)), variable.name = "topic", id.vars = "document")
vizDataFrame


#ggplot
ggplot(data = vizDataFrame, aes(topic, value, fill = document), ylab = "proportion") +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  coord_flip() +
  facet_wrap(~ document, ncol = N)

```

Figure 40: Task2 - part2

```r
# Task - 3
#Name: Venkata Sandeep Kumar Karamsetty
#Student ID: 6228975
#Mail ID: vskk033@uowmail.edu.au

#import libraries
#install.packages("twitteR")
#install.packages(c('ROAuth','RCurl'))
#install.packages("dplyr")
#install.packages('purrr')
#install.packages('caret')
#install.packages('RTextTools')
#install.packages('maxent')
#install.packages('sos')
#install.packages(c('ROAuth','RCurl'))
#install.packages('RTextTools')
library(sos)
library(twitteR)
library('ROAuth')
library('RCurl')
library('dplyr')
library(purrr)
library(plyr)
library(stringr)
library(tm)
library(e1071)
library(caret)
library(RTextTools)
library(maxent)

library(twitteR)
library(RCurl)
#twitter authentication
consumerAPIKey <- "4XsqFw3PwDZWRNUtyF5WSuFlz"
consumerAPISecretKey <- "CuOT4iPxlyoMuk5cconiUpF0eX8e7LQ1ltxs27XbmYWCNJ8b6O"
accessTokenKey <- "1179219778579226630-hFXeJSppsWWY7JllGrtjOmkjvMUQap"
accessTokenSecretKey <- "utFFVPO8J711V6JL1JCJxOHNFgjqyCKQdniCLqBWspwmz"
reqURL <- "https://api.twitter.com/oauth/request_token"
accessURL <- "https://api.twitter.com/oauth/access_token"
authURL <- "https://api.twitter.com/oauth/authorize"
setup_twitter_oauth(consumerAPIKey,consumerAPISecretKey,accessTokenKey,accessTokenSecretKey)
TrumpTweets<- searchTwitter("@realDonaldTrump",n=200,lang="en")
TrumpTweets
#convert to data frame
TrumpTweets <- twListToDF(TrumpTweets)
#write to csv file and save the tweets - change file location
write.csv(Tweets,file="twitterList.csv")
TrumpResults<- read.csv("twitterList.csv")

# count the tagging done in file
countTagging <- table(TrumpResults$Tagging)
countTagging

#Corpus to know details in file
corpusVector <- Corpus(VectorSource(TrumpResults$text))
corpusVector
```

Figure 41: Task3 - part1

```
57
58  #pre-processing the tweets
59  Tweets_Pre_Processed <- tm_map(corpusVector,  content_transformer(tolower))
60  url <- content_transformer(function(x) gsub("(f|ht)tp(s?)://\\S+", "", x, perl=T))
61  Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, url)
62  Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, removePunctuation)
63  Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, stripWhitespace)
64  Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, removeWords, stopwords("english"))
65  Tweets_Pre_Processed <- tm_map(Tweets_Pre_Processed, removeNumbers)
66  Tweets_Pre_Processed <-tm_map(Tweets_Pre_Processed,stemDocument)
67  inspect(Tweets_Pre_Processed[1:10])
68  #Extracting Document term matrix from pre_processed_data
69  DTM <- DocumentTermMatrix(Tweets_Pre_Processed)
70  DTM
71  #partitioning of training and testing data
72  training_Data<- TrumpResults$Tagging[1:160] #trainig set
73  training_Data
74  test_Data <- TrumpResults$Tagging[161:200] #test set
75  test_Data
76  #document term matrix TF-IDF
77  DTM_TRAIN_TFIDF <- DocumentTermMatrix(Tweets_Pre_Processed, control = list(weighting = weightTfIdf))
78  DTM_TRAINING_SET <- DTM_TRAIN_TFIDF[1:160,]
79  DTM_TRAINING_SET
80
81  DTM_TEST_SET <- DTM_TRAIN_TFIDF[161:200,]
82  DTM_TEST_SET
83
84  # use the NB classifier with Laplace smoothing
85  NAIVE_BAYES_RESULT <- naiveBayes(as.matrix(DTM_TRAINING_SET), training_Data, laplace=1)
86  NAIVE_BAYES_RESULT
87
88  # predict with testdata
89  PREDICTED_RESULT <- predict (NAIVE_BAYES_RESULT,as.matrix(DTM_TEST_SET))
90  PREDICTED_RESULT
91
92  #Confusion Matrix for Naive Bayes
93  xtab <- table( "Actual" = test_Data, "Predictions"= PREDICTED_RESULT)
94
95  #xtab
96  confMatrixNB <- confusionMatrix(xtab)
97  confMatrixNB
98
99
100 #apply svm and confusion matrix matrix
101 SVM_TRAIN <- svm(as.matrix(DTM_TRAINING_SET), training_Data, type='C-classification',kernel="radial", cost=10, gamma=0.5)
102 SVM_TRAIN
103 #Prediction Result
104 SVM_PREDICT <- predict(SVM_TRAIN, as.matrix(DTM_TEST_SET))
105 SVM_PREDICT
106
107 #Confusion Matrix
108 xSVM <- table("Actual" = test_Data, "Predictions"= SVM_PREDICT)
109 xSVM
110 confMatrixSVM <- confusionMatrix(xSVM)
111 confMatrixSVM
112
```

Figure 42: Task3 - part2

```
113
114 #Performance for Naive Bayes - Precison, Recall, Fmeasure
115 n1 = sum(xtab) # number of instances
116 nc1 = nrow(xtab) # number of classes
117 diag1 = diag(xtab) # number of correctly classified instances per class
118 rowsums1 = apply(xtab, 1, sum) # number of instances per class
119 colsums1 = apply(xtab, 2, sum) # number of predictions per class
120 p1 = rowsums1 / n1 # distribution of instances over the actual classes
121 q1 = colsums1 / n1 # distribution of instances over the predicted classes
122
123 accuracy1 = sum(diag1) / n1
124
125 precisionNB = diag1 / colsums1
126 recallNB = diag1 / rowsums1
127 f1NB = 2 * precisionNB * recallNB / (precisionNB + recallNB)
128 data.frame(precisionNB, recallNB, f1NB, accuracy1)
129
130
131
132 #Performance for SVM - Precison, recall, fmeasure
133
134 n2 = sum(xSVM) # number of instances
135 nc2 = nrow(xSVM) # number of classes
136 diag2 = diag(xSVM) # number of correctly classified instances per class
137 rowsums2 = apply(xSVM, 1, sum) # number of instances per class
138 colsums2 = apply(xSVM, 2, sum) # number of predictions per class
139 p2 = rowsums2 / n2 # distribution of instances over the actual classes
140 q2 = colsums2 / n2 # distribution of instances over the predicted classes
141
142 accuracy2 = sum(diag2) / n2
143
144 precisionSVM = diag2 / colsums2
145 precallSVM = diag2 / rowsums2
146 f1SVM = 2 * precisionSVM * precallSVM / (precisionSVM + precisionSVM)
147 data.frame(precisionSVM, recallSVM, f1SVM, accuracy2)
148
149
```

Figure 43: Task3 - part3