Homework Assignment 1: Text Mining

Weerdhawal Chowgule

February 11, 2018

1 Dataset, Programming Language & Packages

As specified in the Assignment 1 question I downloaded the news articles dataset named "20news-18828.tar.gz" from the link provided. The dataset contains 20 Newsgroups dataset and totally contains 18868 articles.
I have used R programming to complete the assignment.version : R version 3.4.1. The different packages used as follows

- **tm** : It is the most widely used text mining package.
- **SnowballC** : An R interface to the C libstemmer library that implements Porter's word stemming algorithm for collapsing words to a common root to aid comparison of vocabulary.
- **gplots & ggplot2** They are the most commonly used packages used for visualization.
- **proxy** This package is been used as it provides methods to calculate cosine and jaccard Similarities.
- **wordcloud** This package helps to create a word cloud as the name suggest.
- **reshape2** This packages is used as it provides a function **melt()**. The melt function takes data in wide format and stacks a set of columns into a single column of data.

2 Data Preprocessing

2.1 Feature Selection

The main goal of this step is to clean the data and create a Document term matrix which contains the articles in rows and terms as columns. The values in the matrix would be the frequency of the occurrence of a term in the respective document/article. To achieve this first we create a data frame of all the words using the function **Corpus**. In that function we recursively

read all the articles by specifying English language in the reader Control attribute as shown on line 17 in 1.

Once the corpus is ready we convert all the values to lower case for easy matching of words that we will perform later.As punctuations,white spaces and numbers are not required we remove all these values using the **tm_map()** function. The function also can remove stop-words from the English dictionary. As there is no exact package/function available to remove just the adverbs so we pass a vector of the adverbs and remove them as last step.

```
15
16 #load the corpus
17 all <- Corpus(DirSource("I:\\Masters\\SPRING 18\\SPATIAL AND TEMPORAL\\spatial\\homework1\\20news-18828\\20news-18828\\",
18                          encoding="UTF-8",recursive=TRUE),readerControl=list(reader=readPlain,language="en"))
19
20 #check values
21 all[[1]]
22
23 #lower case
24 all<- tm_map(all, tolower)
25 #remove punctuations
26 all<-tm_map(all,removePunctuation)
27 #strip extra white spaces
28 all <- tm_map(all,stripwhitespace)
29 #apply Numbers from files
30 all<-tm_map(all,removeNumbers)
31
32 #now remove the standard list of stopwords, like you've already worked out
33 all.nostopwords <- tm_map(all, removeWords, stopwords(kind = "en"))
34 #remove words specified in the document
35 all.nostopwords<-tm_map(all.nostopwords,removeWords,c("on","in","next to","infront of","behind","between","under","through",
36                                                        " around","i","me","my","mine","you","your","yours","he","him","his","she",
37                                                       "her","hers","it","its","we","us","our","ours","they","their","theirs","them","easily",
38                                                       "loudly","quickly","quietly","sadly","silently","slowly","always","frequently","often","once"))
39
40 #making TF matrix
```

Figure 1: Preprocessing Code Snippet

Next, we create term document matrix(TDM) as most of the generic terms have been removed. we also normalize the TDM by specifying it in the command. We also have to check for sparsity,where those terms from TDM are removed which have at least a sparse percentage of empty (i.e., terms occurring 0 times in a document) elements. I.e., the resulting matrix contains only terms with a sparse factor of less than sparse which in our case in 0.99. To make it a Document Term Matrix(DTM) we take a transpose of it.

3 Data Processing

3.1 Feature Selection

Feature selection is the process of selecting a subset of the terms occurring in the training set and using only this subset as features in text classification. Selection of Top-100 words was done by summing the frequency of number from the DTM and sorting them in descending order.3.1 shows the top 10 values and their respective frequencies. Figure 2 is a word cloud of the top 100 words from the DTM that we got. As suggested by the professor a histogram has been plotted. The x-axis represents the words i.e the first words is the rank 1 and so on, where as the y-axis represents the frequency

Word	Frequency
subject	20334
can	15789
will	14507
one	14023
writes	13327
article	11442
dont	9637
like	9493
just	9319
people	9241

Table 1: Top 10 words

of appearance of the term in the entire set of articles. As the dataset of top 100 features would be small for our calculations we select a set of 1000 features randomly from the DTM we have obtained.

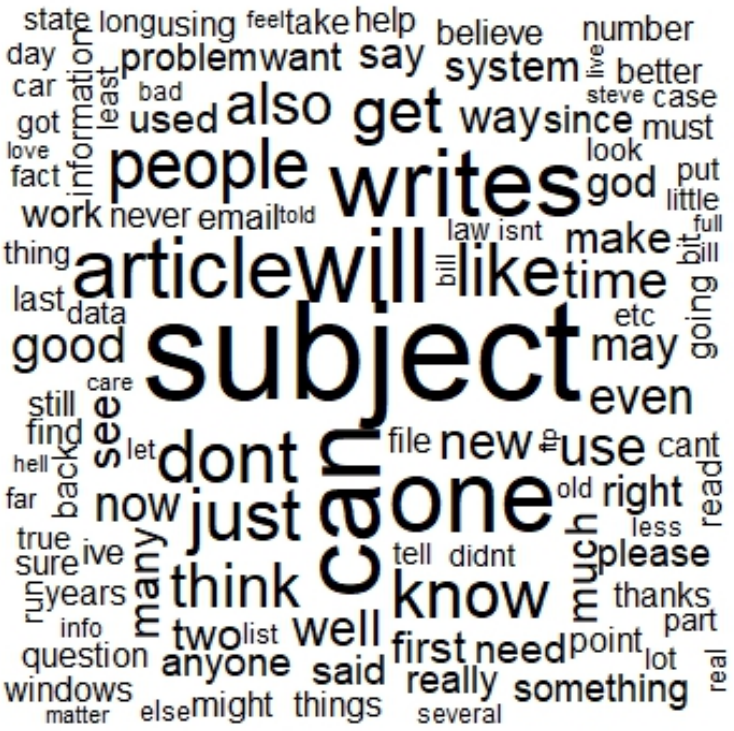

Figure 2: Word Cloud of Top 100

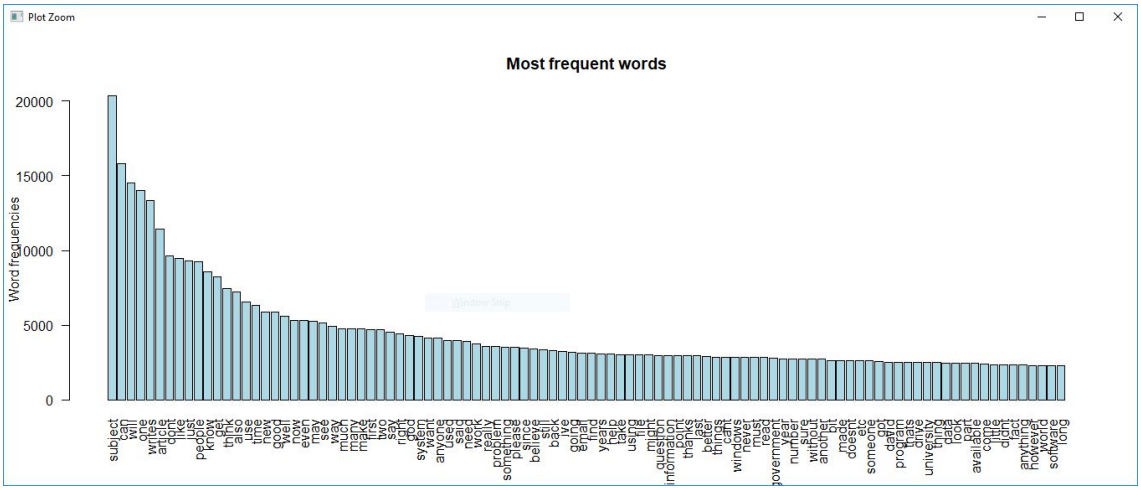

Figure 3: Most frequent Words Histogram

3.2 Similarities

The three standard metrics used to calculate the similarities are Euclidean, Cosine and Jaccard Similarities. In R we compute the distance matrix by using the **dist()** function and setting the method as euclidean, cosine and jaccard respectively. The heat maps of these similarities are as shown in Figure 4,5&6 respectively. As we have selected the data randomly we see that the euclidean heat-map shows a lot of low variance i.e it is not consistent where as the other two are comparatively consistent.

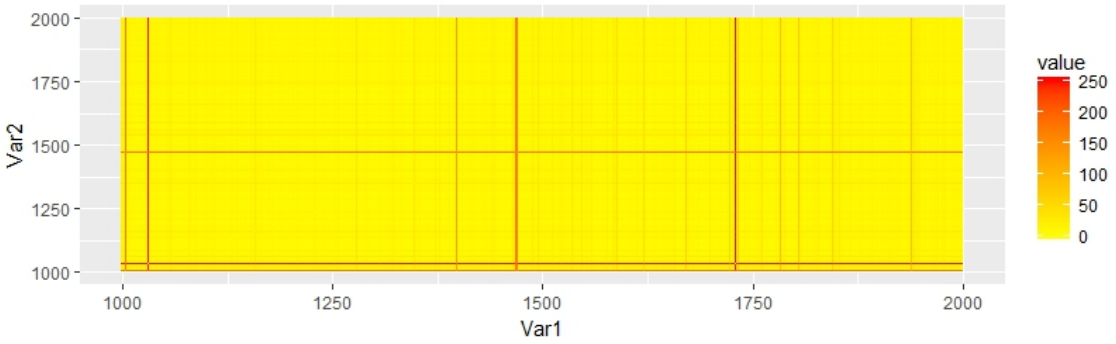

Figure 4: Heat Map Euclidean Similarity

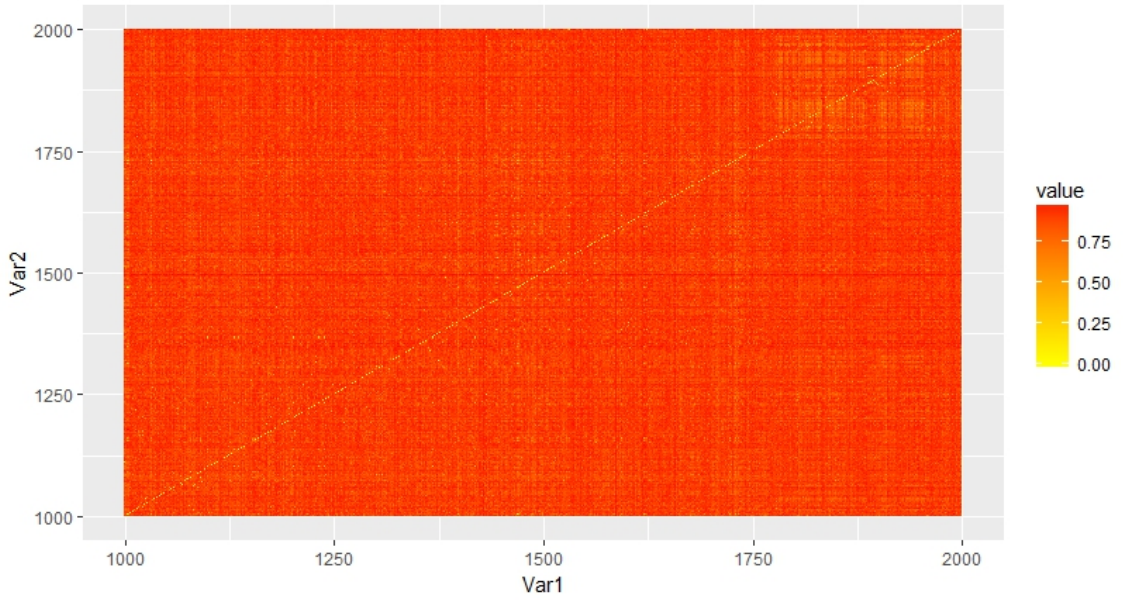

Figure 5: Heat Map Cosine Similarity

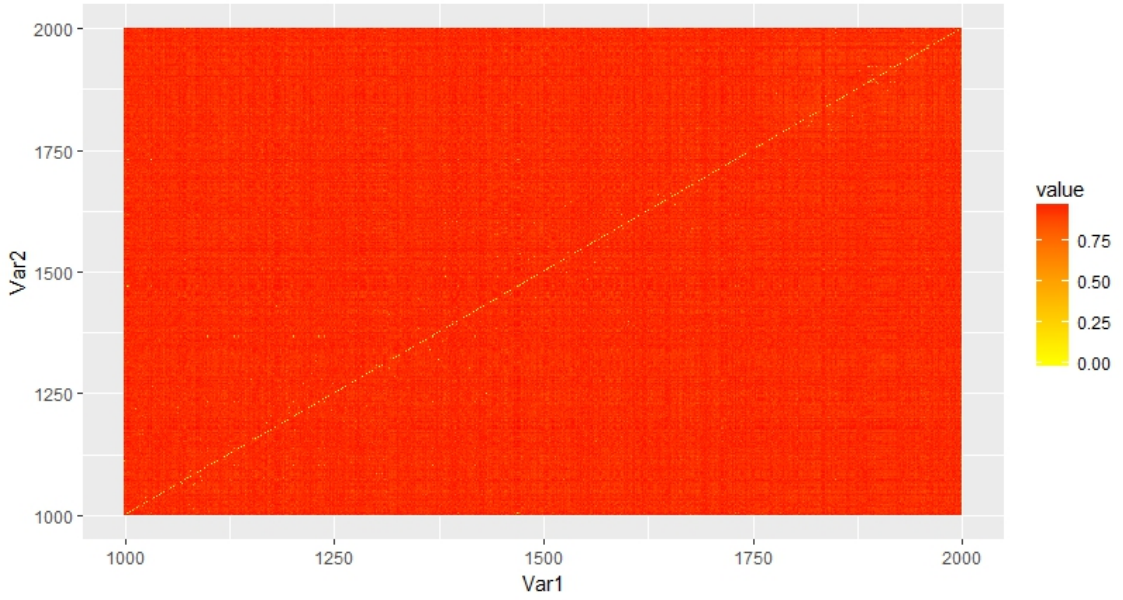

Figure 6: Heat Map Jaccard Similarity

In the next step when linear regression was applied to the distance matrix we got the output as in Figure 7,8 &9

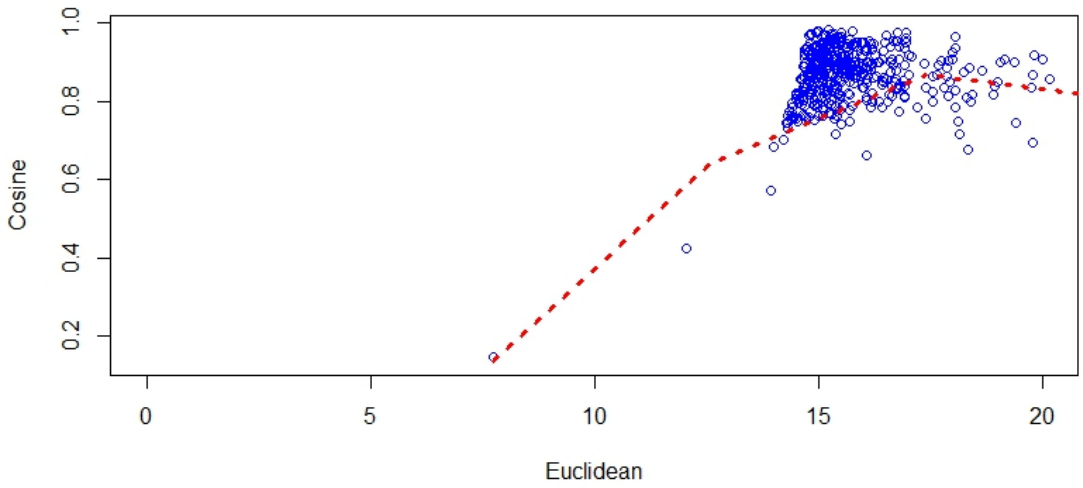

Figure 7: Cosine Vs Euclidean

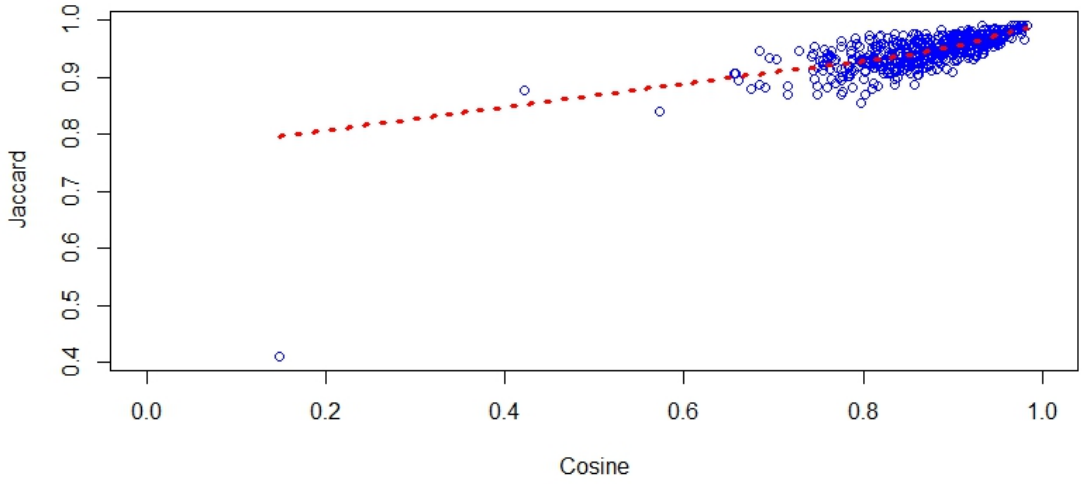

Figure 8: Jaccard Vs Cosine

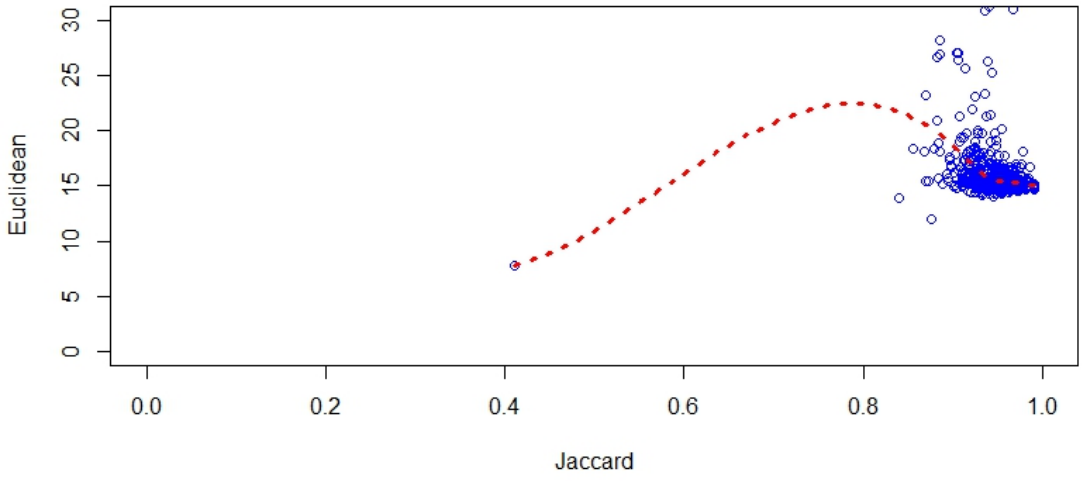

Figure 9: Euclidean Vs Jaccard

3.3 Standard Deviation

Figure 10 represents the standard deviation of all the three metrics with respect to the features. We can see that Euclidean tends to increase exponentially where as Jaccard and Cosine have same values. From these two we can suggest that they are similar to each other.

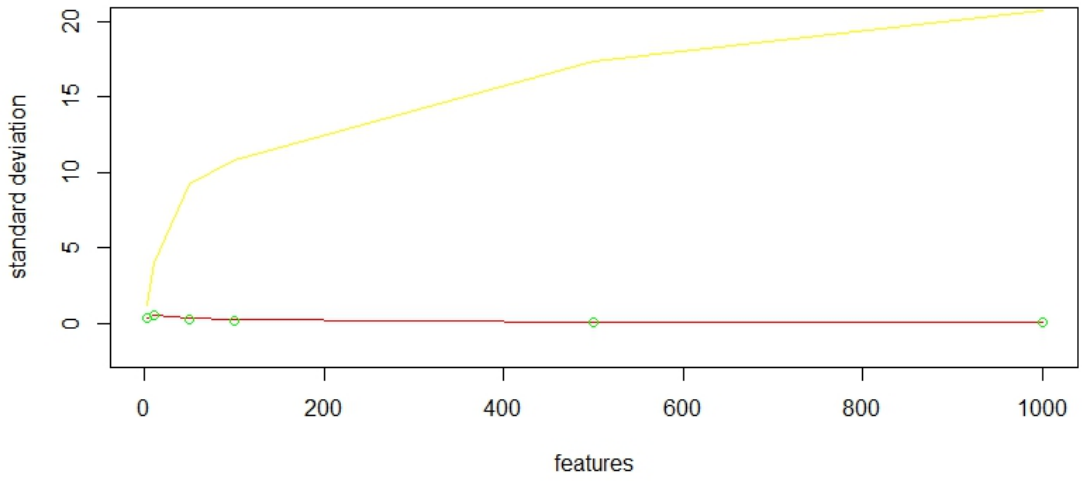

Figure 10: Standard deviation Vs No. of Feature

4 Conclusion

From the last Figure 11 we can see that there are no more similarities between different articles. So there is a dilemma which similarity metric is better. But from the Standard Deviation plot we can conclude that Jaccard and Cosine are much similar.

```
> tail(trial_melted_cos[order(trial_melted_cos$value),],10)
       Var1 Var2     value
403991 1587 1403 0.9976766
587991 1403 1587 0.9976766
497369 1872 1496 0.9983729
873369 1496 1872 0.9983729
4996   1991 1004 0.9984925
991996 1004 1991 0.9984925
471462 1991 1470 0.9984948
992462 1470 1991 0.9984948
497084 1587 1496 0.9986463
588084 1496 1587 0.9986463
> trial_melted_jac <- melted_jac_d
> tail(trial_melted_jac[order(trial_melted_jac$value),],10)
       Var1 Var2     value
32747  1714 1032 0.9985856
692406 1714 1691 0.9985856
714747 1032 1714 0.9985856
715406 1691 1714 0.9985856
178909 1730 1178 0.9985876
730909 1178 1730 0.9985876
32211  1178 1032 0.9985896
178211 1032 1178 0.9985896
178870 1691 1178 0.9985896
691870 1178 1691 0.9985896
> trial_melted_euc <- melted_eud_d
> tail(trial_melted_euc[order(trial_melted_euc$value),],10)
       Var1 Var2    value
178909 1730 1178 250.8725
730909 1178 1730 250.8725
715445 1730 1714 250.8745
731445 1714 1730 250.8745
275005 1730 1274 250.8785
731005 1274 1730 250.8785
497227 1730 1496 250.9522
731227 1496 1730 250.9522
588318 1730 1587 250.9641
731318 1587 1730 250.9641
```

Figure 11: Article similarity pair