Report Summary –

Document Clustering and Topic Modeling

Date: 12th May 2018

Submitted by:

Name	SMS ID	SID
Parthasarathi Chatterjee	118284	TA17021

A. Introduction

Reviews on new Ertiga Model of Suzuki India Ltd. have been extracted and gathered from multiple sources in internet and documented in a text file (New_Ertiga_Review.txt) containing 330 text lines

The requirement is to perform text documents clustering for partitioning a collection of text documents in the text file (New_Ertiga_Review.txt) into similar clusters based on the distance or similarity measure as decided by an objective function.

The goal of the document clustering is to minimize intra-cluster distances between documents, while maximizing inter-cluster distances (using an appropriate distance measure between documents). A distance measure (similarity measure) thus lies at the heart of document clustering.

Topic Modeling is performed to analyze large volumes of unlabeled text in the text file as mentioned above. A "topic" consists of a cluster of words that frequently occur together. Using contextual clues, topic models can connect words with similar meanings and distinguish between uses of words with multiple meanings

B. Approach

Given the text file, document clustering is performed using K-Means clustering and Hierarchical clustering

In addition, Topic Modeling is performed using the R Package

(Refer Annexure for the R Codes)

C. Method:Document Clustering using K-Means

Step 1:Creating a corpus, cleaning and pre processing data from the review lines

Reason: Creating a corpus using Bag of Words approach and removing the punctuations, numbers, symbols, etc. and commonly occurring words which provides little meaningful information in the data set.

The functions like stem and tolower is used to stem the words and convert the uppercase letter to lower case respectively.

Reference R Code:

```
library(NLP)
library(plyr)
library(tm)
library(SnowballC)
library(RColorBrewer)
library(wordcloud)
library(cluster)
library(tidytext)
library(tidyverse)
library(topicmodels)
library(factoextra)
setwd("D:R")
options(header=FALSE,stringsAsFactors = FALSE,fileEncoding = "UTF-8")
aa<-readLines("New_Ertiga_Review.txt", n=-1)
aa
ac<-Corpus(VectorSource(aa))
inspect(ac)
ac<-tm_map(ac,removeWords,stopwords("english"))
inspect(ac)
ac<-tm_map(ac,removePunctuation)
inspect(ac)
ac<-tm_map(ac,removeNumbers)
```

inspect(ac)

ac<-tm_map(ac,stripWhitespace)

inspect(ac)

ac<-tm_map(ac,stemDocument,language="english")

inspect(ac)

ac<-tm_map(ac,tolower)

inspect(ac)

ac<-tm_map(ac,removeWords,"the")

inspect(ac)

Outcome/ Inference:

The corpus is created.

Step 2: Visualizing the wordcloud for the New_Ertiga_Review set.

Reason: The wordcloud will display the most frequent words in the review set

Reference R Code:

wordcloud(ac)

Outcome/ Inference:

The wordcloud for the review set is created. The visualization of the New_Ertiga_ReviewWordcloud depicts that in the set most frequent words are good, ertiga, review, comment comfort, bad, etc. The frequency of good term is more than bad term indicating a positive feedback for New Ertiga Model

Fig.: Wordcloud

Step 3:Creating Term-Document matrix (tdm), Document-Term matrix(dtm) and weighted TF-IDF (Term Frequency-Inverse Document Frequncy)

Reason: Creation of tdm / dtm helps to calculated TF-IDF.

TFIDF or term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

The TF-IDF is a measure that considers both the prevalence of a term within a document (TF) and the scarcity of the term over the entire corpus(IDF).TFIDF scores words higher that appear more often in a document but occur less often across all documents in the corpus.

Reference R Code:

```
tdm<-TermDocumentMatrix(ac)
inspect(tdm)
tdm1<-as.matrix(tdm)
tdm1
ftdm<-sort(rowSums(tdm1),decreasing = TRUE)
ftdm
head(ftdm,10)
dtm<-DocumentTermMatrix(ac)
ad<-weightTfIdf(dtm)
inspect(ad)
```

Outcome/ Inference:

The weighted TF-IDF for the individual terms is created

Step 4:Normalizing the TF-IDF values

Reason: Since K- Means clustering technique depends on the distance measurements amongst the records via Euclidean/ Manhattan/Mahalanobis distance etc., it requires the data to be normalized so that no particular variable or subset of variables dominates the analysis

Reference R Code:

```
ae<-as.matrix(ad)
ae
f1<-function(x)
{
  return(sum(x^2)^.5)
}
f2<-function(x)
{
  ae/apply(x,1,f1)
}
kk<-f2(ae)
```

Outcome/ Inference

The TF-IDF values of the individual terms in the review set is normalized by, as per the function mentioned above in R Code.

Step 5:ApplyingAverage Silhouette method to derive the number of optimal clusters (k) via K-Means Clustering

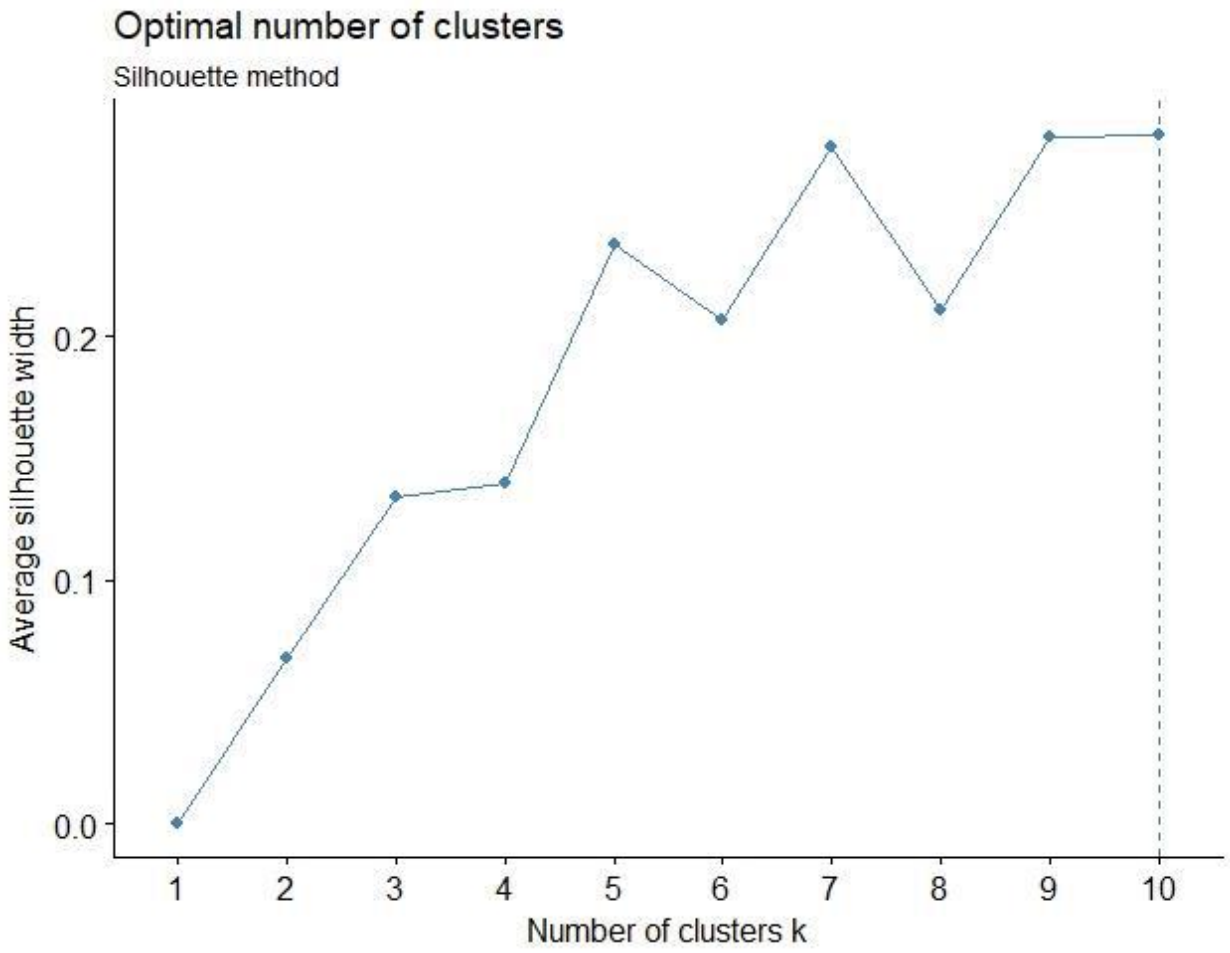
Reason: Average silhouette method computes the average silhouette of observations for different values of k. The optimal number of clusters k is the one that maximize the average silhouette over a range of possible values for k

Reference R Code:`

```
set.seed(1234)
fviz_nbclust(kk, kmeans, method = "silhouette")+
labs(subtitle = "Silhouette method")
```

Outcome/ Inference

The Silhouette method is applied. It is observed from the Silhouette chart that clusters (k=10) is the optimal no of clusters we may select as that is themaximize the average silhouette over a range of possible values for k



Step 6:Creating the K-Means Model

Reason: Using the clusters(k=10), the K-Means Model is created which will be further utilized for the inference or the characteristics of each individual clusters

Reference R Code:

```
ak<-kmeans(kk,10,nstart = 50)
ak
```

ak$size

Outcome/ Inference

K-Means model is applied on the data set and 10 clusters of the following sizes are created:

```
K-means clustering with 10 clusters of sizes 46, 43, 22, 28, 22, 9, 22, 26,
21, 91
```

Step 5:Drawing inference from the Clusters

Reason: The 10 number of clusters can be further analysed. The 10 clusters thus formed where each of the clusters contain closely related documents or documents having the similar type of information.

Reference R Code:

df<-data.frame(text = sapply(ac, as.character), stringsAsFactors = FALSE)

df

aa2<-data.frame(df,ak$cluster)

aa2

cl1<-subset(aa2,aa2$ak.cluster==1)

cl1

cl1d<-Corpus(VectorSource(cl1$text))

inspect(cl1d)

```
cl2<-subset(aa2,aa2$ak.cluster==2)
cl2
cl2d<-Corpus(VectorSource(cl2$text))
inspect(cl2d)
cl3<-subset(aa2,aa2$ak.cluster==3)
cl3
cl3d<-Corpus(VectorSource(cl3$text))
inspect(cl3d)
cl4<-subset(aa2,aa2$ak.cluster==4)
cl4
cl4d<-Corpus(VectorSource(cl4$text))
inspect(cl4d)
cl5<-subset(aa2,aa2$ak.cluster==5)
cl5
cl5d<-Corpus(VectorSource(cl5$text))
inspect(cl5d)
cl6<-subset(aa2,aa2$ak.cluster==6)
cl6
cl6d<-Corpus(VectorSource(cl6$text))
inspect(cl6d)
cl7<-subset(aa2,aa2$ak.cluster==7)
cl7
cl7d<-Corpus(VectorSource(cl7$text))
```

```
inspect(cl7d)

cl8<-subset(aa2,aa2$ak.cluster==8)

cl8

cl8d<-Corpus(VectorSource(cl8$text))

inspect(cl8d)

cl9<-subset(aa2,aa2$ak.cluster==9)

cl9

cl9d<-Corpus(VectorSource(cl9$text))

inspect(cl9d)

cl10<-subset(aa2,aa2$ak.cluster==10)

cl0

cl10d<-Corpus(VectorSource(cl10$text))

inspect(cl10d)
```

Outcome/ Inference:

Clustering algorithms in computational text analysis groups documents into grouping a set of text what are called subsets or clusters where the algorithm's goal is to create internally coherent clusters that are distinct from one another. The output of the individual clusters can be analysed to understand the individual characteristics of each cluster. In addition, a web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group theretrieved documents into a list of meaningful categories.

C. Method: Document Clustering using Hierarchical Clustering

Step 1 to 4 is same as in case of K-Means Clustering

Step 4: Creating the Hierarchical Clustering Model and selection of the no. of groups/clusters

Reason: The hierarchical clustering model is created. From the resulting dendogram, 10 clusters (k=10) have been selected for further analysis. The method "ward" is selected for the hierarchical clustering and method "euclidean"is selected for the distance calculation

Ward's minimum variance criterion minimizes the total within-cluster variance. To implement this method, at each step find the pair of clusters that leads to minimum increase in total within-cluster variance after merging. This increase is a weighted squared distance between cluster centers. At the initial step, all clusters are singletons (clusters containing a single point).

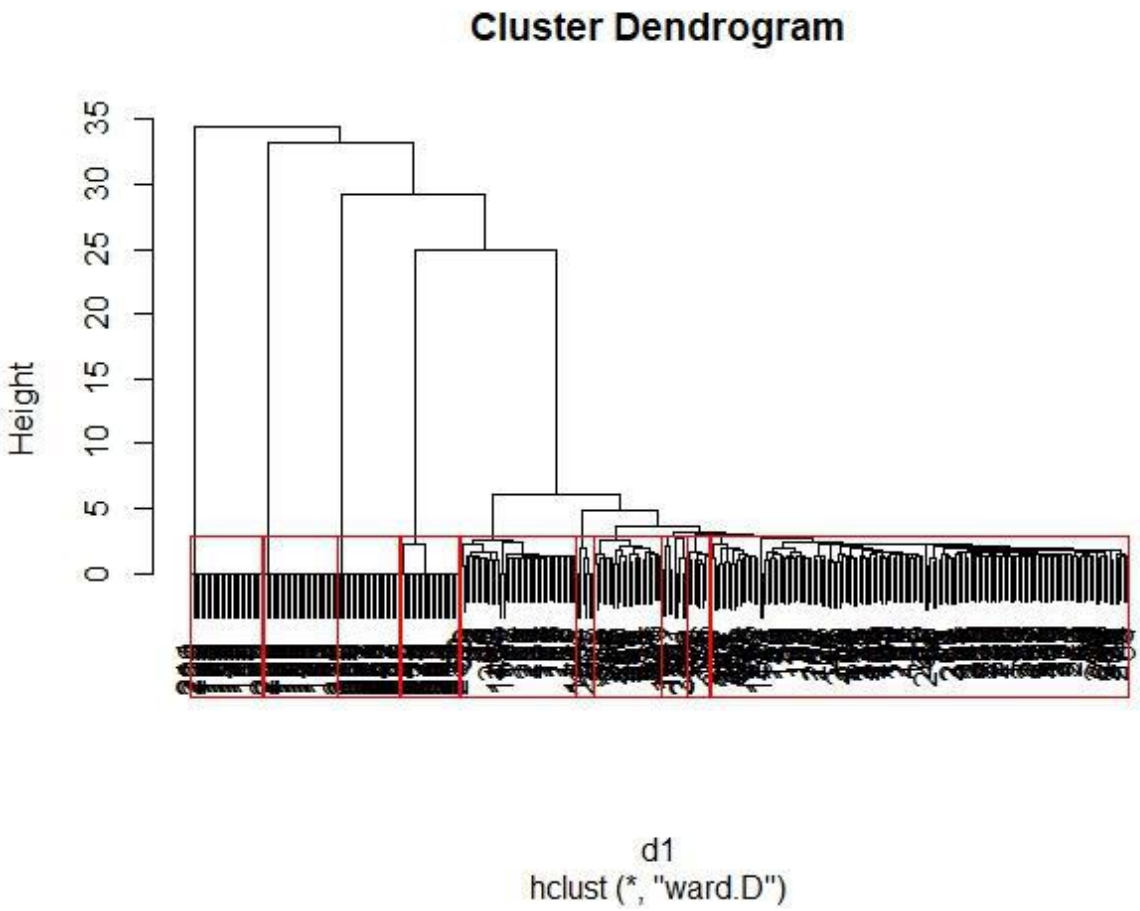
Reference R Code:

```
set.seed(1234)

d1<-dist(kk,method = "euclidean")

d1

hh<-hclust(d1,method = "ward.D")

plot(hh)

rect.hclust(hh,k=10,border="red")

acc<-cutree(hh,10)

table(acc)
```

Outcome/ Inference:

Hierarchical model is applied on the data set and 10 agglomerative clusters of the following sizes are created (The R function rect.hclusthas performed the selection of 10 groups marked in red in the dendogram)

```
table(acc)
acc
  1   2   3   4   5   6   7   8   9  10
147  41  24   9   6   8  26  26  22  21
```



<u>Step 5:</u> Drawing inference from the Clusters

Reason : The 10 number of agglomerative clusters are formed where each of the clusters contain closely related documents or documents having the similar type of information.

Reference R Code:

```
df<-data.frame(text = sapply(ac, as.character), stringsAsFactors = FALSE)
df
acc1<-data.frame(df,acc)
acc1
ah1<-subset(acc1,acc1$acc==1)
ah1
cl1d<-Corpus(VectorSource(ah1$text))
inspect(cl1d)
ah2<-subset(acc1,acc1$acc==2)
ah2
cl2d<-Corpus(VectorSource(ah2$text))
inspect(cl2d)
ah3<-subset(acc1,acc1$acc==3)
ah3
cl3d<-Corpus(VectorSource(ah3$text))
inspect(cl3d)
ah4<-subset(acc1,acc1$acc==4)
ah4
cl4d<-Corpus(VectorSource(ah4$text))
inspect(cl4d)
ah5<-subset(acc1,acc1$acc==5)
ah5
cl5d<-Corpus(VectorSource(ah5$text))
inspect(cl5d)
ah6<-subset(acc1,acc1$acc==6)
```

```
ah6
cl6d<-Corpus(VectorSource(ah6$text))
inspect(cl6d)
ah7<-subset(acc1,acc1$acc==7)
ah7
cl7d<-Corpus(VectorSource(ah7$text))
inspect(cl7d)
ah8<-subset(acc1,acc1$acc==8)
ah8
cl8d<-Corpus(VectorSource(ah8$text))
inspect(cl8d)
ah9<-subset(acc1,acc1$acc==9)
ah9
cl9d<-Corpus(VectorSource(ah9$text))
inspect(cl9d)
ah10<-subset(acc1,acc1$acc==10)
ah10
cl10d<-Corpus(VectorSource(ah10$text))
inspect(cl10d)
```

Outcome/ Inference:

Clustering algorithms in computational text analysis groups documents into grouping a set of text what are called subsets or clusters where the algorithm's goal is to create internally coherent clusters that are distinct from one another. The output of the individual clusters can be analysed to understand the individual characteristics of each cluster. In addition, a web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information.

Clustering methods can be used to automatically group theretrieved documents into a list of meaningful categories.

C. Method: Topic Modeling

Topic modelling can be described as a method for finding a group of words (i.e topic) from a collection of documents that best represents the information in the collection.Topic modeling provides tools to automatically organize, search, understand and summarise from vast amount of information. Topic models are statistical methods that examine words from a set of documents, determine the themes over the text, and discover how the themes are associated or change over time. The process of topic modeling can be utilized for the following:

1. **Uncover the hidden topical patterns within a corpus**
2. **Annoate documents according to these topics**
3. **Use annotations to organize, search and summarize texts.**

Step1: Selection of optimal number of topics

Reason:To select the optimal number of topics, Package "ldatuning" is used which realizes 4 metrics to select perfect number of topics for LDA model.The most easy way is to calculate all metrics at once. All existing methods require to train multiple LDA models to select one with the best performance.

Metrics used for Comparison

Arun2010: The measure is computed in terms of symmetric KL-Divergence of salient distributions that are derived from these matrix factor and is observed that the divergence values are higher for non-optimal number of topics (maximize)

CaoJuan2009: method of adaptively selecting the best LDA model based on density.(minimize)

Griffths: To evaluate the consequences of changing the number of topics T, used the Gibbs sampling algorithm to obtain samples from the posterior distribution over z at several choices of T(minimize)

Reference R Code:

```
system.time({
   tunes <- FindTopicsNumber(
dtm = dtm,
     topics = c(2:50),
     metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010"),
     method = "Gibbs",
     control = list(seed = 12345),
mc.cores = 4L,
     verbose = TRUE
   )
 })
FindTopicsNumber_plot(tunes)
```

Outcome/ Inference:

To find the optimal number of topic models, the approach is to find the extremum between the maximize and minimize plot. From the below plot, the conclusion can be made that optimal number of topics is in range 10-13.

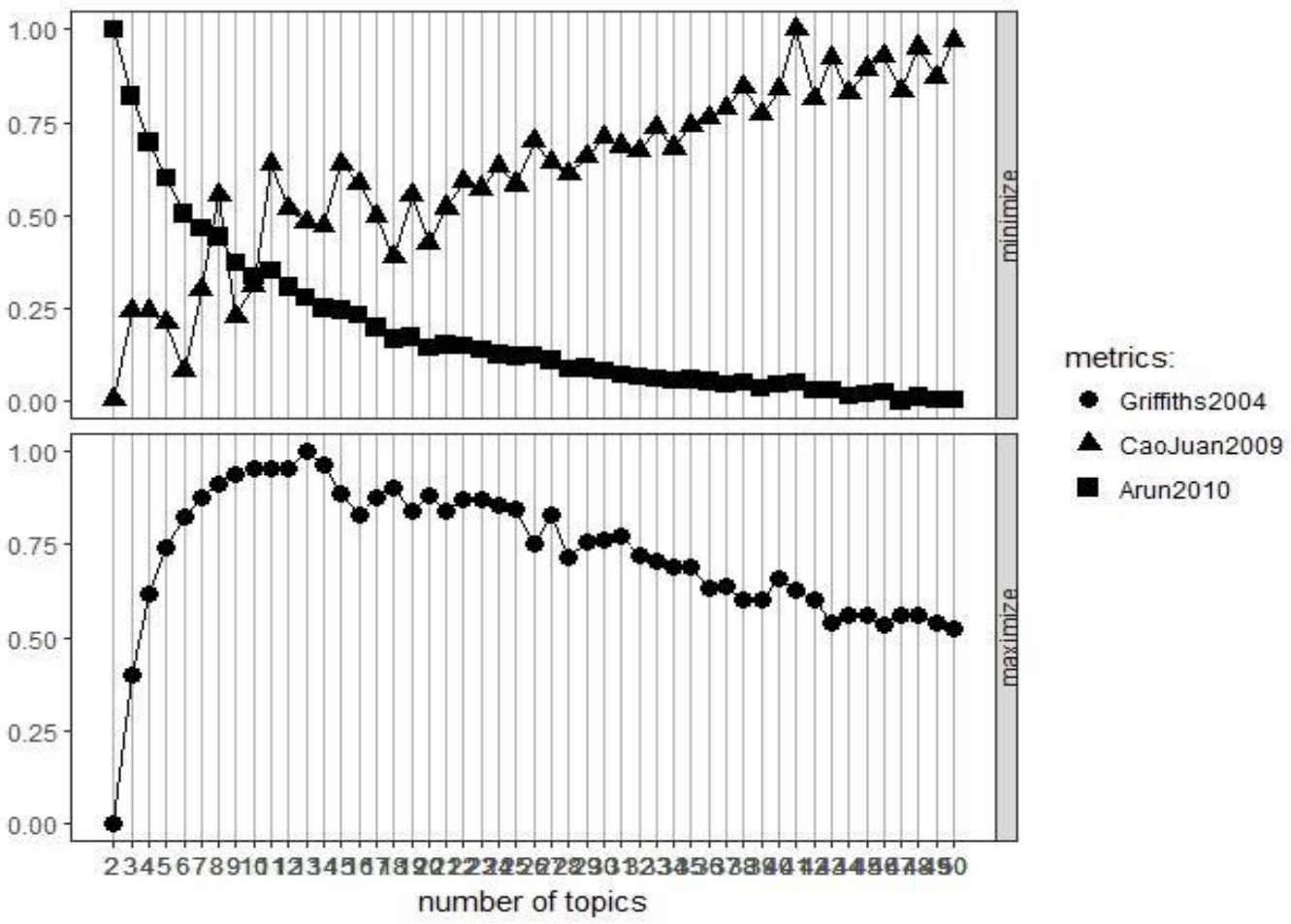


Fig.: Metrics Plot

Step2: Creation of optimal number of topics using LDA

Reason : Creating 10 number of topics having top 10 most informative words using LDA (Latent Dirichlet Allocation) technique

Reference R Code:

```
# function to get & plot the most informative terms by a specificed number

# of topics, using LDA

top_terms_by_topic_LDA<- function(ac, # should be a columm from a dataframe

                                  plot = T, # return a plot? TRUE by defult
```

```
number_of_topics = 10)
{
  # create a corpus (type of object expected by tm) and document term matrix
  Corpus <- Corpus(VectorSource(ac)) # make a corpus object
  DTM <- DocumentTermMatrix(Corpus) # get the count of words/document

  # remove any empty rows in our document term matrix (if there are any
  # we'll get an error when we try to run our LDA)
unique_indexes<- unique(DTM$i) # get the index of each unique value
  DTM <- DTM[unique_indexes,] # get a subset of only those indexes

  # preform LDA & get the words/topic in a tidy text format
lda<- LDA(DTM, k = number_of_topics, control = list(seed = 1234))
  topics <- tidy(lda, matrix = "beta")

  # get the top ten terms for each topic
top_terms<- topics  %>% # take the topics data frame and..
group_by(topic) %>% # treat each topic as a different group
top_n(10, beta) %>% # get the top 10 most informative words
ungroup() %>% # ungroup
arrange(topic, -beta) # arrange words in descending informativeness

  # if the user asks for a plot (TRUE by default)
```

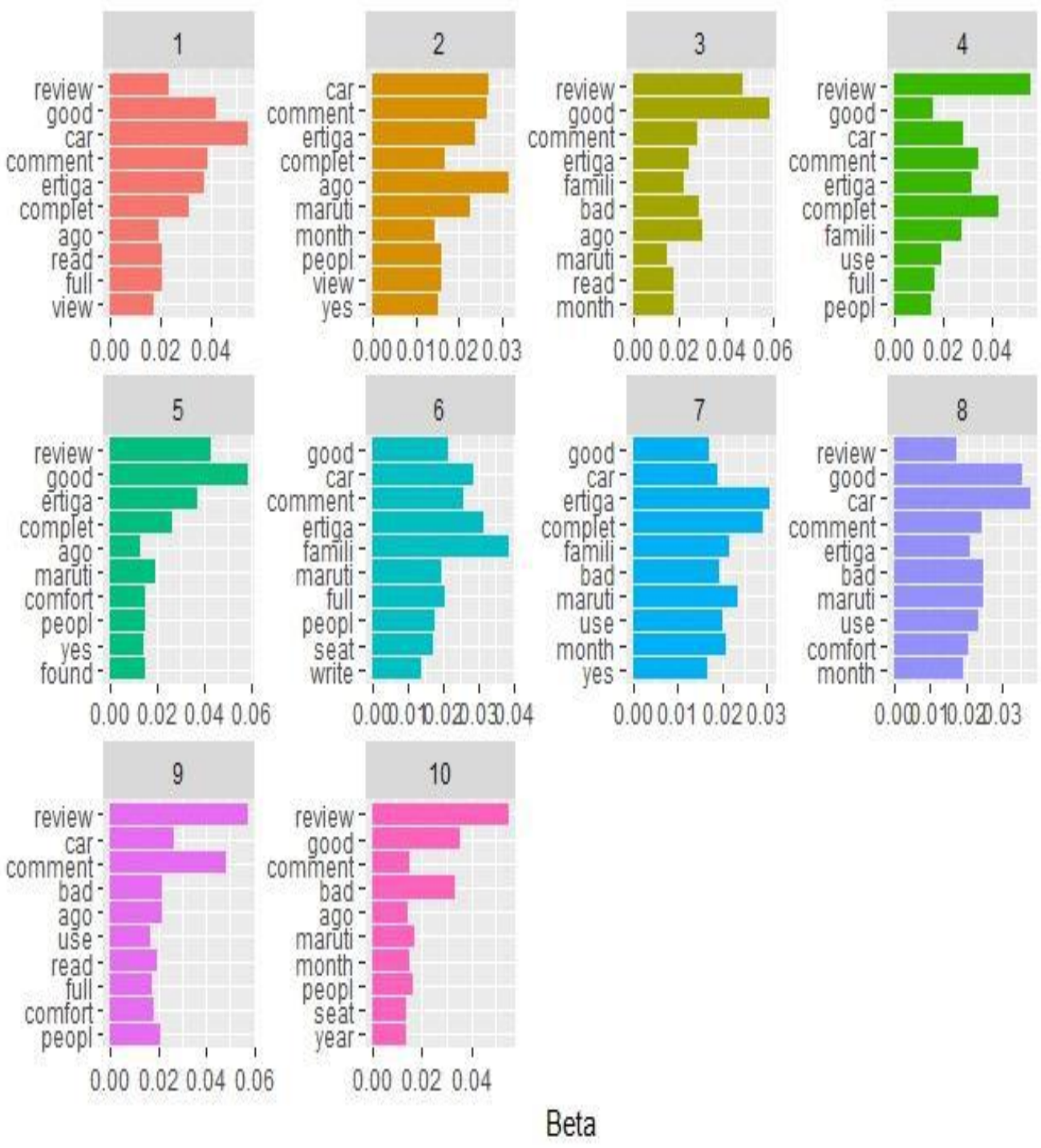
```
if(plot == T){
    # plot the top ten terms for each topic in order
top_terms %>% # take the top terms
mutate(term = reorder(term, beta)) %>% # sort terms by beta value
ggplot(aes(term, beta, fill = factor(topic))) + # plot beta by theme
geom_col(show.legend = FALSE) + # as a bar plot
facet_wrap(~ topic, scales = "free") + # which each topic in a seperate plot
labs(x = NULL, y = "Beta") + # no x label, change y label
coord_flip() # turn bars sideways
}else{
    # if the user does not request a plot
    # return a list of sorted terms instead
    return(top_terms)
  }
}
ss<-top_terms_by_topic_LDA(ac, number_of_topics = 10)
ss
```

Outcome/ Inference:

10 number of topics having top 10 most informative words is created randomly drawn from a corpus where each topic contains a list of the most important words from the vocabulary.The process of topic modeling can be utilized for the following:

1. **Uncover the hidden topical patterns within a corpus**
2. **Annotate documents according to these topics**
3. **Use annotations to organize, search and summarize texts.**

The ten topics thus created comes with its own explanatory power guided by the terms with highest beta in each topic.The goal of LDA is to infer the underlying topics, topic proportions and topic assignments for every document



ANNEXURE

1. Working Data Set

New_Ertiga_Review.txt

2. R Code

R_Code