**FINAL PROJECT REPORT**

**ON**

**SUPRISING DISCOVERIES ON HEALTH INFORMATION**

COURSE: Knowledge Discovery in Databases

**TEAM MEMBERS:**

HEMANTH KUMAR KORABOINA 800988741

KAVYA MANDLI 800984226

ANVESH KOTHAPEKKY 800988048

SAI KIRAN MANKENA 800985840

Contents

[DATA CLEANING: 2](#_Toc501028397)

[PROBLEM STATEMENT: 3](#_Toc501028398)

[Languages used 4](#_Toc501028399)

[Operations performed: 4](#_Toc501028400)

[Libraries Used 4](#_Toc501028401)

[Definition of Surprise: 4](#_Toc501028402)

[Cluster 4](#_Toc501028403)

[TidyText 4](#_Toc501028404)

[Ggplot2 4](#_Toc501028405)

[Methods Used for Finding Surprise Element 4](#_Toc501028406)

[TOPIC ANALYSIS APPROACH: 5](#_Toc501028407)

[The task of topic mining. 6](#_Toc501028408)

[TF-IDF: 6](#_Toc501028409)

[Latent Dirichlet allocation (LDA): 7](#_Toc501028410)

[Topic- Probabilities: 8](#_Toc501028411)

[Interpretation: 8](#_Toc501028412)

[Challenge Faced: 10](#_Toc501028413)

[Result: 11](#_Toc501028414)

[MACHINE LEARNING APPROACH: 13](#_Toc501028415)

[SVM Model to get “Surprise” news from news corpus: 13](#_Toc501028416)

[Definition of Surprise: 13](#_Toc501028417)

[Why SVM: 13](#_Toc501028418)

[Structure of Document Term Matrix: 13](#_Toc501028419)

[Implementation and Results: 14](#_Toc501028420)

[APPENDIX: 16](#_Toc501028421)

# **DATA CLEANING:**

Data cleansing is hard to do, hard to maintain, hard to know where to start. There seem to always be errors, dupes, or format inconsistencies. One of the most challenging aspects of data cleansing has got to be maintaining a clean list of data, whether it’s sourced from multiple vendors or manually entered by your hard-working interns, or a combination of both. One mistype could create a whole myriad of problems within your database, and can lead to hours upon hours of manual cleansing that could so easily have been avoided. Now we would probably have heard he quote. “Data Science is 90% cleaning and 10% analysis thereafter".

If we just go through Data lifecycle it can be described in the below flow:

plan -> collect -> assure -> describe -> preserve -> discover -> integrate -> analysis -> report, publication.

The part in between collection and analysis can be broadly referred to as preprocessing, wherever it pertains to preparing for the analysis. So, what is Assurance?? It is the part where data gets validated, verified, and enforced to be consistent.

As we are working on text data (In this case diabetes articles) the first step that should be performed will be Pre-Processing it. From the dataset which we were given we can see there are close to 10,000 files now we assume that that most of the data is available is unstructured and we need to perform all the above data lifecycle steps to get important data from this data.

So, during the pre-processing we try to perform of actions to remove the stop words, White Spaces, Converting the text to lowercase and more because these will not help us in manner to find the pattern which we are looking for. Although we are looking for a pattern which is readily available we are trying to find a surprise factor which is not generally found this way be something which we are not considering or stumbled upon.

This step helps us a bit to reduce amount of data we must go through which will then result in helping us find that surprising factor in the data.

After performing the above steps, we are sure that data is cleaned so we will try to confirm it by comparing the dtm of corpus before the pre-processing and after. So, we were able to see that the number of terms we must go through are reduced this let us know that we were able to complete the pre-processing successfully

# **PROBLEM STATEMENT:**

To find out the surprising elements from the given diabetes text corpus.

**Languages used:** R Language, Python.

**Operations performed:** Topic Analysis, PAM, Cosine Similarity, and Word Cloud.

**Libraries Used:** tm, cluster, topicmodels, tidytext, ggplot2, dplyr.

**Definition of Surprise:**

For our entire analysis we are interested in finding the personalized surprise rather than a generalized surprise. A personalized surprise is nothing but an element which we found may be surprising to us and may not be surprising to others, it mainly depends on the background knowledge of the individual related to diabetes.

We have defined surprise elements as the elements which are present in the corpus and the elements which we don’t have knowledge prior to the data discovery. So, the approach we followed is first categorizing the huge data available to us using iterative topic analysis.

Libraries Used:

**Tm:** This library includes the framework for carrying various text mining applications within R, this library includes various functions for cleaning the data and carrying various other operation on text.

**Cluster:** This library includes various functions for carrying various clustering that is finding groups in the data operation on the corpus like k-means, sk-means etc.

**Topicmodels:** Provides an interface for performing Latent Dirichlet Allocation (LDA) models and Correlated Topics Models (CTM).

**TidyText:** This package is used for performing text mining for word processing and sentiment analysis using 'dplyr', 'ggplot2', and other tidy tools.

**Ggplot2:** ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

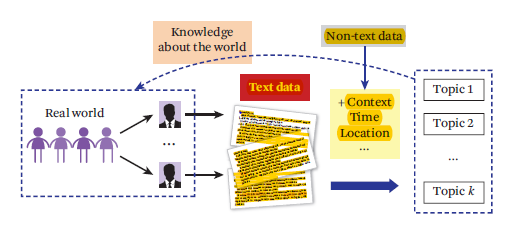
**Dplyr:** A fast, consistent tool for working with data frame like objects, both in memory and out of memory. The dplyr package makes these steps fast and easy:

* By constraining your options, it helps you think about your data manipulation challenges.
* It provides simple “verbs”, functions that correspond to the most common data manipulation tasks, to help you translate your thoughts into code.
* It uses efficient backends, so you spend less time waiting for the computer.

**Methods Used for Finding Surprise Element:**

# **TOPIC ANALYSIS APPROACH:**

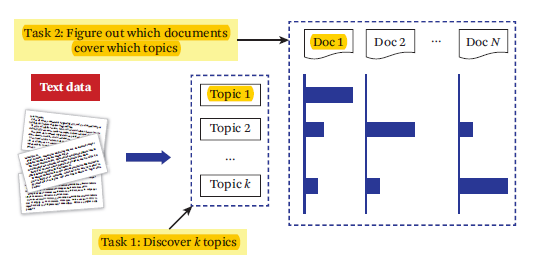
Topic modeling provides an algorithmic solution to managing, organizing and annotating large archival text. The annotations aid you in tasks of information retrieval, classification and corpus exploration. Topic models provide a straightforward way to analyze large volumes of unlabeled text. A "topic" 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. In machine learning and natural language processing topic models are generative models which provide a probabilistic framework for the term frequency occurrences in documents in each corpus.

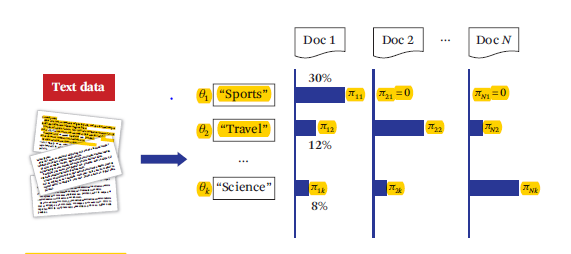


There are generally two different tasks or subtasks:

The first is to discover the k topics from a collection of text

The second task is to figure out which documents cover which topics to what extent.





## **The task of topic mining.**

Input

A collection of N text documents C = {d1, . . . , dN}

Number of topics: k

Output

k topics: {θ1, . . . , θk}

Coverage of topics in each di : {πi1, . . . , πik}

$k

j=1

πij = 1

πij = prob of di covering topic θj

## **TF-IDF:**

Generally, in text mining and natural language processing we always have a question on how to quantify what a document is about. Can we do this by looking at the words that make up the document? One way to approach how important a word can be being its **term frequency (tf),** how frequently a word occurs in a document. There are words in a document, though, that occur many times but may not be important; in English, these are probably words like “the”, “is”, “of”, and so forth. You might take the approach of adding words like these to a list of stop words and removing them before analysis, but it is possible that some of these words might be more important in some documents than others. A list of stop words is not a sophisticated approach to adjusting term frequency for commonly used words.

Another approach is to look at a term’s inverse document frequency (idf), which decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents. This can be combined with term frequency to calculate a term’s tf-idf, the frequency of a term adjusted for how rarely it is used. It is intended to measure how important a word is to a document in a collection (or corpus) of documents. It is a rule-of-thumb or heuristic quantity; while it has proved useful in text mining, search engines, etc., its theoretical foundations are considered less than firm by information theory experts. The inverse document frequency for any given term is defined as

We can use tidy data principles to approach tf-idf analysis and use consistent, effective tools to quantify how important various terms are in a document that is part of a collection.

## **Latent Dirichlet allocation (LDA):**

Latent Dirichlet allocation (LDA) is a particularly popular method for fitting a topic model. It treats each document as a mixture of topics, and each topic as a mixture of words. This allows documents to “overlap” each other in terms of content, rather than being separated into discrete groups, in a way that mirrors typical use of natural language. we can use tidy text principles to approach topic modeling.

Latent Dirichlet allocation is one of the most common algorithms for topic modeling. Without diving into the math behind the model, we can understand it as being guided by two principles.

Every document is a mixture of topics. We imagine that each document may contain words from several topics proportions. For example, in a two-topic model we could say “Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B.”

Every topic is a mixture of words. For example, we could imagine a two-topic model of American news, with one topic for “politics” and one for “entertainment.” The most familiar words in the politics topic might be “President”, “Congress”, and “government”, while the entertainment topic may be made up of words such as “movies”, “television”, and “actor”. Importantly, words can be shared between topics; a word like “budget” might appear in both equally.

LDA is a mathematical method for estimating both at the same time: finding the mixture of words that is associated with each topic, while also determining the mixture of topics that describes each document.

We can use the LDA() function from the topicmodels package, setting k = n, to create a n-topic LDA model.

This function returns an object containing the full details of the model fit, such as how words are associated with topics and how topics are associated with documents.

## **Topic- Probabilities:**

The tidy() method, originally from the broom package (Robinson 2017), for tidying model objects. The tidytext package provides this method for extracting the per-topic-per-word probabilities, called β (“beta”), from the model.

### **Interpretation:**

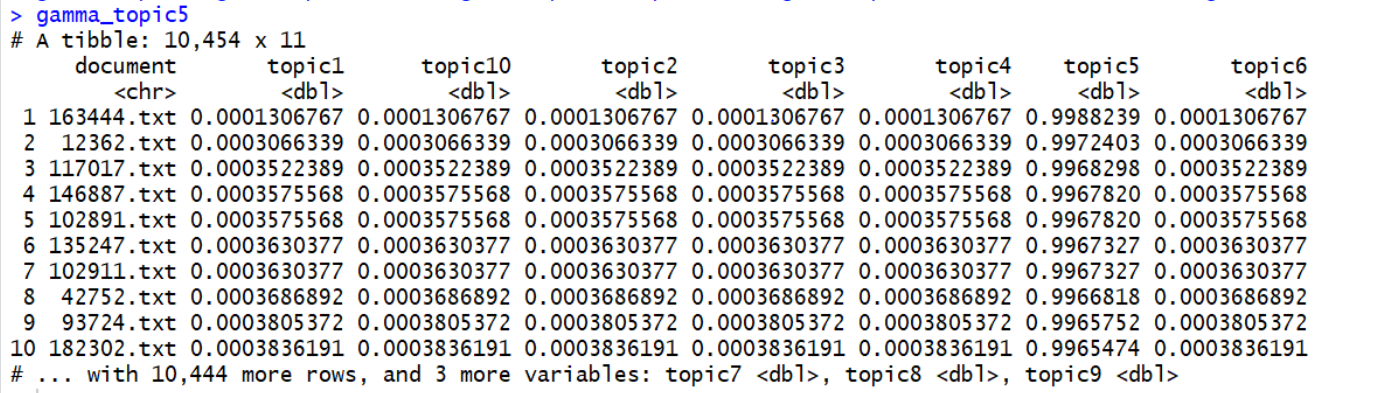
As we have calculated the topic probabilities in each document and word- topic probabilities we can now visualize the top words in a topic and we can know which documents contain what percentage of topics in each document, which we can use to categorize data.

We now visualize top 10 words of each topic which help us in knowing about the topic. We mumbled in the starting on fixing the number of topic but then we have fixed to 10 topics for the whole data set and which shows distinction between several topics in a satisfactory manner. With 10 topics it showed to be like in the output below:

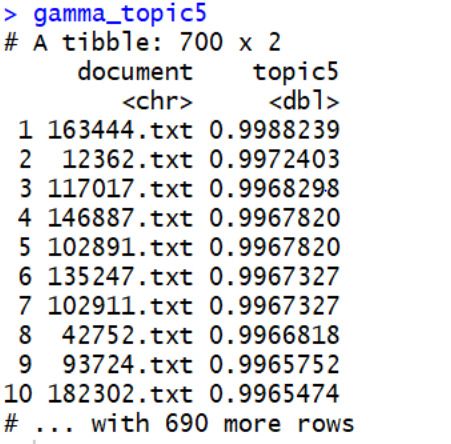
As we can see in the below picture the topic have been well differentiated and we got to know which topic is regarding which subject of corpus. In the below topics we selected topic 5 to further dig into. **A screenshot of a cell phone

Description generated with high confidence**

Then we thought of doing topic analysis on one selected topic. We then calculated gamma spread of each document to know which topic is dominant in which document, it shows out like the picture below:

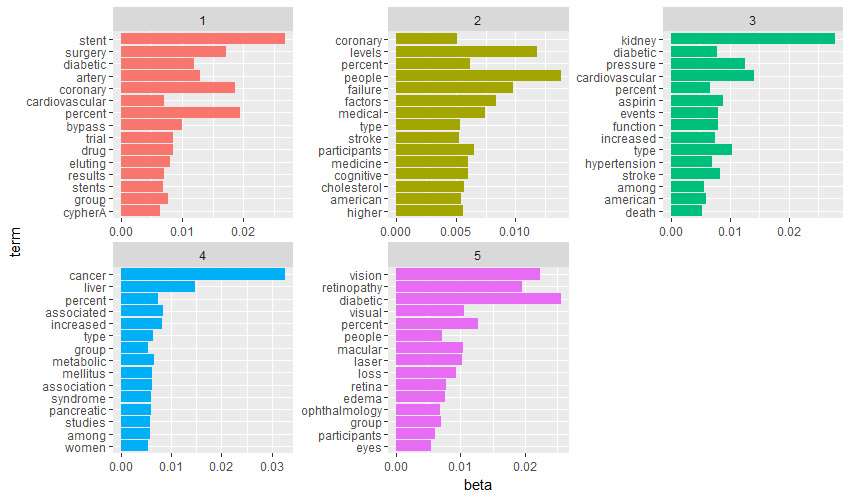


We then need the documents which have 50% of the coverage of the selected topic for this to be done we sorted the table by the column of topic we selected and then extracted the document column and the selected topic column, which we further cut down the rows below which the document probability has less than 50% by which we get to know the documents which are dominant in the selected topic.



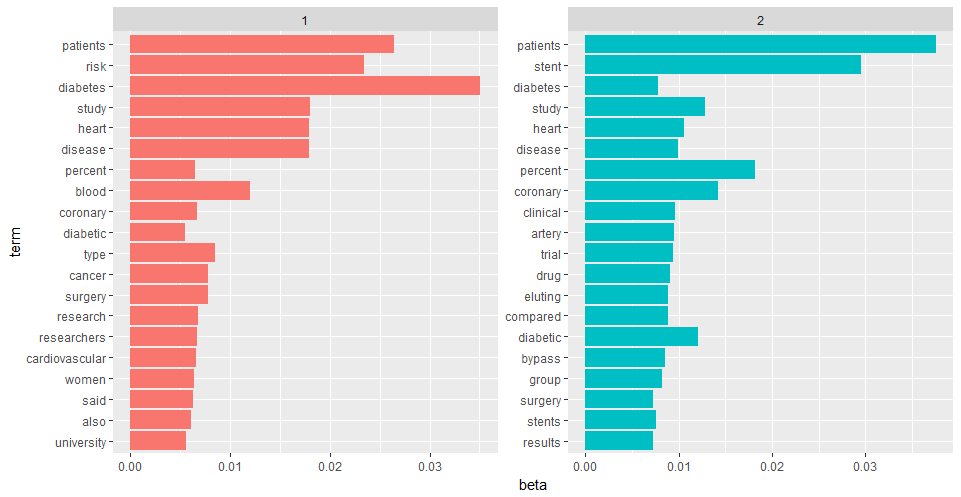
### **Challenge Faced:**

Now to further dig in to the topic we need to form a corpus with the documents which are output in the last step and here we faced a severe problem, we almost stuck there for half a day don’t know how to proceed further but then we have got to know how to add documents by the document name which further lead us to dig our selected topic i.e., topic5 in the previous output.



As we can see here the topics are well defined and they have divided every subject in topic 5 in the correct way. This type of classification doesn’t only help us to analyze documents but if we want to add any document to the corpus we can directly attach it to the set of documents which it is like.

Now we have started to analyze the topic 1 here to further classify data which led us to other set of documents and as they were a small set of documents we have sticked with 2 topics now:

****

### **Result:**

If we observe the above topics we can easily say that the topic 1 contains the words risk, study, heart, coronary, type, surgery, research, cardiovascular, women. Where as the topic 2 contains mainly related bypass surgery and drug related issues. By observing the topic 1 we were fascinated to know that women are more associated to risk and effects related to diabetes than men. But we weren’t sure whether the word women is associated to risk in a positive fashion or in the negative fashion and then we found the mutual information between the words men, risk and women, risk. Then we got to know that the words women and risk has double the mutual information compared to the words men, risk. We further have taken a step further to find documents and read the content of some documents which agrees with our hypothesis. So we are surprised to know that diabetes has more severe effects on heart and other body organs in women as compared to men which was surprising to us as we don’t have any prior knowledge in the medical field of diabetes and doesn’t know that diabetes has different effects on men and women.

We also performed the topic analysis on some other topics but didn’t find any surprising elements there but thought of presenting them here. The outputs are displayed in the below images:

A screenshot of a cell phone

Description generated with high confidence

A screenshot of a cell phone

Description generated with high confidence

# **MACHINE LEARNING APPROACH:**

In this project we developed Topic Modeling and Machine Learning (Support Vector Machine) techniques to identify “surprising” news from a news corpus. “Surprise” is defined as a divergence from an expectation or a low likelihood of an occurrence according to an expected likelihood.

## **SVM Model to get “Surprise” news from news corpus:**

### **Definition of Surprise:**

We extracted less frequency words in each document. We set a threshold value of 30 for the total number of less frequency words in each document. If a document contains more than 30 less frequency words, we define that document as “Surprise”

### **Why SVM:**

We chose SVM over other machine learning algorithms because SVM is Supervised Machine learning model which can be used for both classification and regression SVM models works better when predicting data with more than 4000 dimensions

Produces better accuracy while prediction.

### **Structure of Document Term Matrix:**

We constructed a document term matrix by applying following filters to the corpus.

The word length should be in the size of 5 to 20. Each word in the range of 20 to 60

i.e. all the words that are 20 minimum and 60 at maximum are retrieved from the corpus.

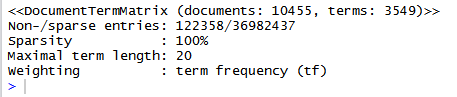


Fig : Summary of Document Term matrix

After applying the above filters on the corpus, we got a document term matrix that contains 10455 rows (Number of documents in the corpus) and 3549 columns(less frequency words in the corpus).

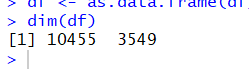


Fig : Dimensions of DTM obtained

### **Implementation and Results:**

Once we had the document term matrix we have appended  a column “rowSums” which contains the total number of less frequency words in each document. Let’s us inspect the column “rowSums”. From the below results we can observe that minimum is 0, 1st Quartile is 8.00,  Median is 14.00 , mean is 16.94  ,and 3rd Quartile is 22 and Maximum is 283.00.

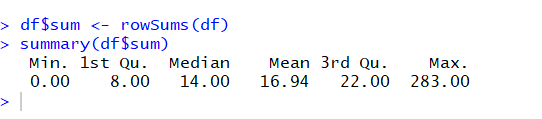


Fig : Summary of rowsums

Therefore, there are few documents which does not contain any less frequency words and maximum number of less frequency words in a document is 283.00.

We played with several values and found the 90% documents contains less than 30 less frequency words. So, we set a threshold of 30. We have set a label surprise to the document term matrix. I.e. If a document has more than 30 less frequency words then the document may contain surprise information otherwise the document does not contain any surprise.

Let us check how many surprised documents are present in our corpus. This can be found by constructing the table of the column “SurpriseVal”.

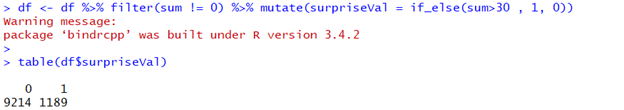


Fig : No. of Surprise Documents classified

From the above, we can observe  that 9214 documents are marked as Non-surprise and 1189 documents are marked as surprise.

Once we have labelled the column , we can split the data into training and testing data sets. We have used 80% of our data for training and 20% for testing. Let us check the dimensions of our training set. From below we can observe that model  used 8324 documents for training and the remaining documents for testing.

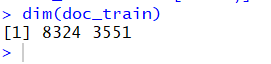


Fig : Dimensions of train data

We have used the SVM machine learning model to train our data set. Once the model is built we have tested on the test dataset. Once we predicted the output we will check the accuracy of our model . i.e. how accurately our model predicted the surprise data. We can check the accuracy by constructing the confusion matrix over our predicted values and the actual values. From the below table we can observe that the accuracy of our model is 91.87%. Hence, we can accept our model.

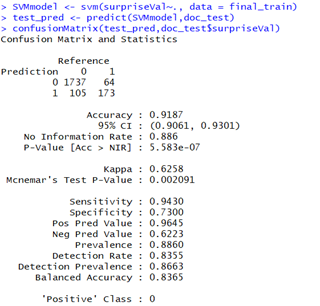


Fig: Confusion Matrix after running SVM model to detect surprise

# **APPENDIX:**

install.packages("tm")

library(tm)

setwd("C:/Users/Hemanth Kumar/Desktop/KDD/Rdata")

text\_corpus<-Corpus(DirSource("diabetes")) # converyting health news in to text corpus

text\_corpus <- tm\_map(text\_corpus, stripWhitespace)

text\_corpus <- tm\_map(text\_corpus, content\_transformer(tolower))

text\_corpus <- tm\_map(text\_corpus, removeWords, stopwords("english"))

text\_corpus

summary(text\_corpus)

#text\_corpus <- tm\_map(text\_corpus, removePunctuation)

#text\_corpus

#stopwords("english")

dtm <- DocumentTermMatrix(text\_corpus) #converts corpus into matrix

dtm

dtm2 <- removeSparseTerms(dtm, sparse=0.95)

dtm2

dtm3 <-DocumentTermMatrix(text\_corpus, control=list(wordLengths=c(4, 20), bounds = list(global = c(5,200)))) # ow we are only converting such words from textcorpus to matrix which are in length from 4 to 20 and having frequency of 5 to 200

dtm3

inspect(dtm)

dtm3 <- weightTfIdf(dtm3, normalize = TRUE)

inspect(dtm)

m3<-as.matrix(dtm)

df3<-as.data.frame(m3)

#topic modeling

install.packages("topicmodels")

library(topicmodels)

dtm3

ap\_lda <- LDA(dtm3, k = 10, control = list(seed = 1234))

ap\_lda

install.packages("tidytext")

library(tidytext)

# Word-Topic probabilities

ap\_topics <- tidy(ap\_lda, matrix = "beta")

ap\_topics

install.packages("ggplot2")

library(ggplot2)

install.packages("dplyr")

library(dplyr)

# Visualization of the top 10 words for each topic

ap\_top\_terms <- ap\_topics %>%

group\_by(topic) %>%

top\_n(15, beta) %>%

ungroup() %>%

arrange(topic, -beta)

ap\_top\_terms %>%

mutate(term = reorder(term, beta)) %>%

ggplot(aes(term, beta, fill = factor(topic))) +

geom\_col(show.legend = FALSE) +

facet\_wrap(~ topic, scales = "free") +

coord\_flip()

#Document-topic probabilities

library(tidytext)

ap\_documents <- tidy(ap\_lda, matrix = "gamma")

ap\_documents

gamma\_spread <- ap\_documents %>%

mutate(topic = paste0("topic", topic)) %>%

spread(topic, gamma)

class(gamma\_spread)

gamma\_topic5<-gamma\_spread[order(gamma\_spread$topic5, rev(gamma\_spread$document), decreasing = TRUE),]

gamma\_topic5<-gamma\_topic5[0:700,c(1,7)]

gamma\_topic5\_1<-gamma\_topic5[c(1)]

topic5list <- as.list(as.data.frame(t(gamma\_topic5\_1)))

write.csv(gamma\_topic5\_1,file = "topic5.csv")

test\_corpus1<-Corpus(DirSource(directory="diabetes",pattern="197207.txt|198713.txt|116951.txt|256432.txt|257008.txt|226968.txt|245141.txt|253608.txt|253242.txt|114552.txt|234887.txt|270197.txt|209752.txt|114054.txt|238828.txt|225121.txt|195317.txt|85065.txt|234015.txt|113241.txt|248673.txt|244152.txt|25248.txt|42225.txt|66350.txt|212395.txt|214347.txt|156295.txt|85876.txt|156086.txt|166691.txt|248275.txt|193191.txt|94068.txt|136167.txt|128574.txt|248300.txt|219579.txt|95023.txt|46452.txt|189476.txt|119568.txt|136171.txt|225209.txt|128766.txt|232605.txt|45407.txt|248368.txt|209593.txt|136398.txt|71610.txt|132935.txt|102025.txt|107364.txt|192812.txt|94338.txt|102016.txt|187878.txt|265188.txt|212429.txt|81317.txt|142041.txt|234143.txt|229730.txt|187151.txt|34961.txt|131070.txt|131050.txt|151465.txt|136259.txt|43884.txt|213380.txt|102496.txt|88605.txt|261299.txt|235105.txt|192834.txt|98399.txt|123456.txt|196311.txt|92450.txt|7699.txt|210734.txt|224503.txt|244369.txt|169614.txt|88812.txt|138646.txt|234638.txt|102207.txt|148874.txt|270458.txt|52009.txt|204139.txt|225735.txt|85278.txt|157642.txt|15263.txt|2167.txt|145061.txt|216894.txt|24700.txt|66799.txt|119724.txt|86237.txt|239123.txt|272569.txt|40890.txt|207026.txt|255364.txt|201308.txt|186511.txt|54729.txt|143563.txt|265756.txt|188510.txt|115426.txt|177603.txt|260807.txt|8270.txt|136463.txt|238235.txt|244378.txt|151892.txt|106195.txt|139931.txt|66739.txt|204952.txt|55175.txt|203235.txt|102732.txt|123453.txt|208185.txt|204057.txt|166889.txt|106420.txt|126514.txt|68765.txt|240930.txt|97277.txt|260829.txt|133173.txt|68878.txt|248331.txt|227796.txt|258099.txt|139629.txt|238821.txt|133560.txt|197678.txt|263796.txt|248374.txt|228702.txt|85603.txt|101975.txt|100108.txt|29481.txt|241273.txt|42806.txt|172561.txt|129470.txt|103730.txt|144688.txt|31294.txt|79502.txt|259194.txt|101437.txt|157869.txt|268430.txt|148864.txt|223403.txt|52911.txt|33155.txt|7560.txt|184670.txt|247440.txt|143499.txt|178091.txt|250349.txt|107947.txt|133440.txt|209017.txt|247845.txt|196672.txt|92196.txt|237980.txt|101993.txt|259543.txt|168162.txt|33264.txt|51446.txt|87393.txt|235362.txt|189477.txt|151126.txt|160450.txt|39356.txt|13209.txt|160362.txt|272147.txt|85659.txt|151505.txt|242949.txt|206379.txt|256096.txt|158.txt|222551.txt|80351.txt|195945.txt|102733.txt|122914.txt|99961.txt|260370.txt|130250.txt|102518.txt|56083.txt|33220.txt|260447.txt|254174.txt|88415.txt|85598.txt|95027.txt|6207.txt|154331.txt|247342.txt|151175.txt|158262.txt|199441.txt|240245.txt|64910.txt|20613.txt|213921.txt|266757.txt|220243.txt|225223.txt|270430.txt|117908.txt|203527.txt|75306.txt|170061.txt|125734.txt|140678.txt|101145.txt|111165.txt|88205.txt|4509.txt|228136.txt|152842.txt|260465.txt|247568.txt|137676.txt|94422.txt|105495.txt|56851.txt|145460.txt|46927.txt|28959.txt|230582.txt|46436.txt|55374.txt|18108.txt|229841.txt|95581.txt|59638.txt|184992.txt|256974.txt|43378.txt|262151.txt|225651.txt|270030.txt|255491.txt|254761.txt|5806.txt|29747.txt|32240.txt|41471.txt|207024.txt|99635.txt|127515.txt|273266.txt|229384.txt|266576.txt|244337.txt|4179.txt|45536.txt|197792.txt|71741.txt|259846.txt|122847.txt|266572.txt|241362.txt|63771.txt|260750.txt|39799.txt|133959.txt|33244.txt|25226.txt|112327.txt|186139.txt|206035.txt|229341.txt|207348.txt|1552.txt|158145.txt|265090.txt|150091.txt|217366.txt|175712.txt|46755.txt|248440.txt|67344.txt|113242.txt|33520.txt|94911.txt|94943.txt|176912.txt|245347.txt|57999.txt|12781.txt|68750.txt|42820.txt|113347.txt|214319.txt|141175.txt|208390.txt|12720.txt|27268.txt|141815.txt|234662.txt|192335.txt|142009.txt|272218.txt|62533.txt|267969.txt|21131.txt|272308.txt|103203.txt|174881.txt|232363.txt|215331.txt|195502.txt|23861.txt|66738.txt|158528.txt|111591.txt|123053.txt|109542.txt|166811.txt|51230.txt|21132.txt|206176.txt|257814.txt|208105.txt|207331.txt|9265.txt|211042.txt|207164.txt|215375.txt|225750.txt|209003.txt|197710.txt|261018.txt|100121.txt|240242.txt|169862.txt|228016.txt|43543.txt|257056.txt|67744.txt|42232.txt|242232.txt|207889.txt|270014.txt|69099.txt|107362.txt|146912.txt|212367.txt"))

test\_corpus1 <- tm\_map(test\_corpus1, stripWhitespace)

test\_corpus1 <- tm\_map(test\_corpus1, content\_transformer(tolower))

test\_corpus1 <- tm\_map(test\_corpus1, removeWords, stopwords("english"))

dtm\_1 <- DocumentTermMatrix(test\_corpus1) #converts corpus into matrix

dtm

dtm2 <- removeSparseTerms(dtm\_1, sparse=0.95)

dtm2

dtm3 <-DocumentTermMatrix(test\_corpus1, control=list(wordLengths=c(4, 20), bounds = list(global = c(5,200)))) # ow we are only converting such words from textcorpus to matrix which are in length from 4 to 20 and having frequency of 5 to 200

dtm3

m3<-as.matrix(dtm\_1)

df3<-as.data.frame(m3)

#topic modeling

dtm3

ap\_lda <- LDA(dtm3, k = 5, control = list(seed = 1234))

ap\_lda

library(tidytext)

# Word-Topic probabilities

ap\_topics <- tidy(ap\_lda, matrix = "beta")

ap\_topics

# Visualization of the top 10 words for each topic

ap\_top\_terms <- ap\_topics %>%

group\_by(topic) %>%

top\_n(15, beta) %>%

ungroup() %>%

arrange(topic, -beta)

ap\_top\_terms %>%

mutate(term = reorder(term, beta)) %>%

ggplot(aes(term, beta, fill = factor(topic))) +

geom\_col(show.legend = FALSE) +

facet\_wrap(~ topic, scales = "free") +

coord\_flip()

#Document-topic probabilities

ap\_documents <- tidy(ap\_lda, matrix = "gamma")

ap\_documents

gamma\_spread <- ap\_documents %>%

mutate(topic = paste0("topic", topic)) %>%

spread(topic, gamma)

gamma\_spread

class(gamma\_spread)

gamma\_topic5<-gamma\_spread[order(gamma\_spread$topic1, rev(gamma\_spread$document), decreasing = TRUE),]

gamma\_topic5<-gamma\_topic5[0:145,c(1,2)]

gamma\_topic5\_1<-gamma\_topic5[c(1)]

class(gamma\_topic5\_1)

topic5list <- as.list(as.data.frame(t(gamma\_topic5\_1)))

write.csv(gamma\_topic5\_1,file = "heart\_organ.csv")

gamma\_topic\_vision<-gamma\_spread[order(gamma\_spread$topic3, rev(gamma\_spread$document), decreasing = TRUE),]

gamma\_topic\_vision$topic3

gamma\_topic\_vision<-gamma\_topic\_vision[0:175,c(1,4)]

gamma\_topic5\_3<-gamma\_topic\_vision[c(1)]

class(gamma\_topic5\_1)

topic5list <- as.list(as.data.frame(t(gamma\_topic5\_3)))

write.csv(gamma\_topic5\_3,file = "vision\_organ.csv")

#### organ topic modelling

test\_corpus1<-Corpus(DirSource(directory="diabetes",pattern="265188.txt|260807.txt|151465.txt|248374.txt|151505.txt|209003.txt|248331.txt|260829.txt|229841.txt|45536.txt|266576.txt|248300.txt|248440.txt|248368.txt|209017.txt|204139.txt|113241.txt|128574.txt|184670.txt|136463.txt|68878.txt|247568.txt|101145.txt|57999.txt|69099.txt|248275.txt|137676.txt|4179.txt|87393.txt|257056.txt|107362.txt|102025.txt|102016.txt|234887.txt|235105.txt|261299.txt|166811.txt|186511.txt|85876.txt|214319.txt|253242.txt|136171.txt|136167.txt|88205.txt|207348.txt|39356.txt|169862.txt|225651.txt|68750.txt|207331.txt|207889.txt|151892.txt|7560.txt|105495.txt|99961.txt|225750.txt|24700.txt|158528.txt|123861.txt|193191.txt|245141.txt|266572.txt|116951.txt|160362.txt|160450.txt|238828.txt|51446.txt|254158.txt|29481.txt|195317.txt|125734.txt|213921.txt|234638.txt|267969.txt|176912.txt|235362.txt|244378.txt|102207.txt|242949.txt|156083.txt|256432.txt|111165.txt|141175.txt|270458.txt|232363.txt|212395.txt|67744.txt|259194.txt|122847.txt|187151.txt|241273.txt|169614.txt|175712.txt|80351.txt|113242.txt|259846.txt|212429.txt|85278.txt|158145.txt|240242.txt|272218.txt|234662.txt|187878.txt|260465.txt|247845.txt|186237.txt|131158.txt|45806.txt|142041.txt|119724.txt|64910.txt|31294.txt|39799.txt|144688.txt|32240.txt|208185.txt|248158.txt|174158.txt|103203.txt|42225.txt|7699.txt|6207.txt|42232.txt|126514.txt|146158.txt|71158.txt|258099.txt|89158.txt|129265.txt|260750.txt|25248.txt|267344.txt|204952.txt|107364.txt|239123.txt|130250.txt|261018.txt|33244.txt|155374.txt|5806.txt|244152.txt|248673.txt|206035.txt|123453.txt|206176.txt|145460.txt|33155.txt|156295.txt|25226.txt|88605.txt|88812.txt|148874.txt|187560.txt|203527.txt|211042.txt|15263.txt|237980.txt|234015.txt|196672.txt|120613.txt|43884.txt|85065.txt|150091.txt|138646.txt|263796.txt|54729.txt|240890.txt|244337.txt|115426.txt|207024.txt|168162.txt|133560.txt|223403.txt|8270.txt|27268.txt"))

text\_corpus1 <- tm\_map(test\_corpus1, stripWhitespace)

text\_corpus1 <- tm\_map(text\_corpus1, content\_transformer(tolower))

text\_corpus1 <- tm\_map(text\_corpus1, removeWords, stopwords("english"))

dtm <- DocumentTermMatrix(text\_corpus1) #converts corpus into matrix

dtm

dtm2 <- removeSparseTerms(dtm, sparse=0.95)

dtm2

dtm3 <-DocumentTermMatrix(text\_corpus1, control=list(wordLengths=c(4, 20), bounds = list(global = c(5,200)))) # ow we are only converting such words from textcorpus to matrix which are in length from 4 to 20 and having frequency of 5 to 200

dtm3

m3<-as.matrix(dtm)

df3<-as.data.frame(m3)

#topic modeling

ap\_lda <- LDA(dtm3, k = 2, control = list(seed = 1234))

ap\_lda

install.packages("tidytext")

library(tidytext)

# Word-Topic probabilities

ap\_topics <- tidy(ap\_lda, matrix = "beta")

ap\_topics

# Visualization of the top 10 words for each topic

ap\_top\_terms <- ap\_topics %>%

group\_by(topic) %>%

top\_n(20, beta) %>%

ungroup() %>%

arrange(topic, -beta)

ap\_top\_terms %>%

mutate(term = reorder(term, beta)) %>%

ggplot(aes(term, beta, fill = factor(topic))) +

geom\_col(show.legend = FALSE) +

facet\_wrap(~ topic, scales = "free") +

coord\_flip()

#Document-topic probabilities

ap\_documents <- tidy(ap\_lda, matrix = "gamma")

ap\_documents

gamma\_spread <- ap\_documents %>%

mutate(topic = paste0("topic", topic)) %>%

spread(topic, gamma)

gamma\_spread

class(gamma\_spread)

gamma\_topic5<-gamma\_spread[order(gamma\_spread$topic2, rev(gamma\_spread$document), decreasing = TRUE),]

gamma\_topic5<-gamma\_topic5[0:40,c(1,3)]

gamma\_topic5\_1<-gamma\_topic5[c(1)]

class(gamma\_topic5\_1)

topic5list <- as.list(as.data.frame(t(gamma\_topic5\_1)))

write.csv(gamma\_topic5\_1,file = "heart\_organ\_2.csv")

### topic modelling on diabetic bypass surgery stent

test\_corpus2<-Corpus(DirSource(directory="diabetes",pattern="95581.txt|99635.txt|56083.txt|63771.txt|46755.txt|29747.txt|46927.txt|94911.txt|94943.txt|43543.txt|95027.txt|133173.txt|67344.txt|132935.txt|248440.txt|18108.txt|186139.txt|226968.txt|197678.txt|227796.txt|9265.txt|87158.txt|55175.txt|61552.txt|184670.txt|197792.txt|131070.txt|102733.txt|51446.txt|114552.txt|66799.txt|5806.txt|102732.txt|146158.txt|245141.txt|89158.txt|244152.txt|225735.txt|95023.txt|122914.txt"))

text\_corpus2 <- tm\_map(test\_corpus2, stripWhitespace)

text\_corpus2 <- tm\_map(text\_corpus2, content\_transformer(tolower))

text\_corpus2 <- tm\_map(text\_corpus2, removeWords, stopwords("english"))

dtm <- DocumentTermMatrix(text\_corpus2) #converts corpus into matrix

dtm

dtm2 <- removeSparseTerms(dtm, sparse=0.95)

dtm2

dtm3 <-DocumentTermMatrix(text\_corpus2, control=list(wordLengths=c(4, 20), bounds = list(global = c(5,200)))) # ow we are only converting such words from textcorpus to matrix which are in length from 4 to 20 and having frequency of 5 to 200

dtm3

m3<-as.matrix(dtm)

df3<-as.data.frame(m3)

#topic modeling

ap\_lda <- LDA(dtm3, k = 2, control = list(seed = 1234))

ap\_lda

# Word-Topic probabilities

ap\_topics <- tidy(ap\_lda, matrix = "beta")

ap\_topics

# Visualization of the top 10 words for each topic

ap\_top\_terms <- ap\_topics %>%

group\_by(topic) %>%

top\_n(20, beta) %>%

ungroup() %>%

arrange(topic, -beta)

ap\_top\_terms %>%

mutate(term = reorder(term, beta)) %>%

ggplot(aes(term, beta, fill = factor(topic))) +

geom\_col(show.legend = FALSE) +

facet\_wrap(~ topic, scales = "free") +

coord\_flip()

#Document-topic probabilities

ap\_documents <- tidy(ap\_lda, matrix = "gamma")

ap\_documents

gamma\_spread <- ap\_documents %>%

mutate(topic = paste0("topic", topic)) %>%

spread(topic, gamma)

gamma\_spread

class(gamma\_spread)

gamma\_topic5<-gamma\_spread[order(gamma\_spread$topic2, rev(gamma\_spread$document), decreasing = TRUE),]

gamma\_topic5<-gamma\_topic5[0:40,c(1,3)]

gamma\_topic5\_1<-gamma\_topic5[c(1)]

class(gamma\_topic5\_1)

topic5list <- as.list(as.data.frame(t(gamma\_topic5\_1)))

write.csv(gamma\_topic5\_1,file = "heart\_organ\_2.csv")

#finding documents

m<-as.matrix(dtm)

memory.limit()

dtm

m5<-m[,"women"]

m5<-as.matrix(m5)

m6<- as.matrix(m5[which(m5[,]>0),])

m6

inspect(m6)