**TEXT MINING**

**“Project UAS Kelompok 8”**



**Oleh:**

M. Diaz Fahlifi (2011016310019)

Muhammad Difha Wardana (2011016310014)

Rama Noor Rizki (2011016210004)

**KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN**

**UNIVERSITAS LAMBUNG MANGKURAT**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**PROGRAM STUDI S1 ILMU KOMPUTER**

**BANJARBARU**

**2022**

Tahapan-tahapan yang dilakukan:

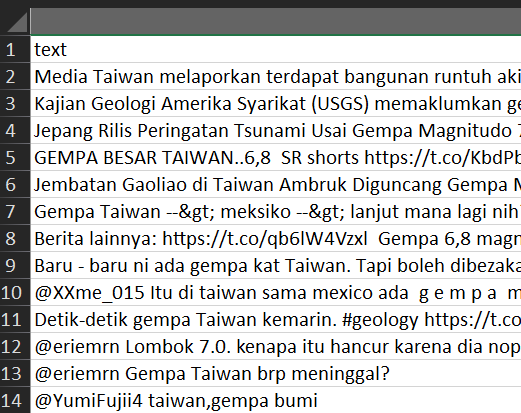
1. **Data Selection**

Hal yang pertama dilakukan adalah dengan mengambil data di **twitter** menggunakan **Library rtweet** yang telah di instal pada **RStudio**. Untuk kata kunci dalam melakukan pencarian yaitu berupa “Gempa”, “Guncangan” , “Taiwan” , “Mexico” , “Getaran”... . Dan untuk Syntax nya yaitu:

|  |
| --- |
| if (!auth\_has\_default() && interactive()) {  auth\_setup\_default()  }  data\_pariaman = search\_tweets(q = "Gempa", n = 500, lang = "id", include\_rts = FALSE)  data\_pariaman = data.frame(data\_pariaman$full\_text)  colnames(data\_pariaman) = "text" |

Untuk menyimpan obyek ke dalam file **Excel** dilakukan dengan fungsi berikut ini

|  |
| --- |
| library(xlsx)  write.xlsx(data\_gempa, "data\_gempa.xlsx", sheetName = "Sheet1", append = FALSE, row.names = FALSE) |



Gambar 1 Obyek pencarian data twitte dari data text yang telah di simpan ke dalam file Excel dengan nama file data\_gempa.xlsx

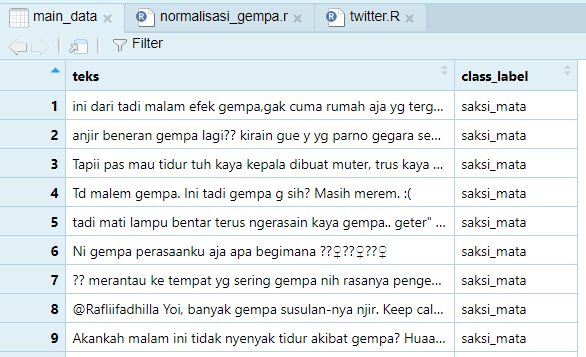
Pencarian data dilakukan secara berulang dengan kata kunci yang berbeda sehingga data yang didapat mencapai **1860** data. Setelah itu dilakukan **Pelabelan** secara manualdengan tiga *class label* yaitu “**saksi\_mata**” , “**bukan\_saksi\_mata**” , dan “**bukan\_gempa**”. Hasil pelabelan diperoleh dengan jumlah yang berbeda pada tiap labelnya, untuk label “saksi\_mata” berjumlah **763** data , “bukan\_saksi\_mata” berjumlah **978** data, dan “bukan\_gempa” berjumlah “**119**” data. Akan tetapi kami hanya mengambil **300** data pada Project Kelompok UAS ini dengan masing-masing label berjumlah **100** data.

1. **Preprocessing Data Teks**

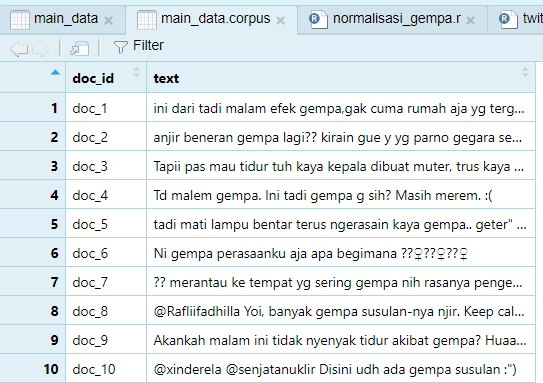
Tahap selanjutnya yaitu dengan melakukan normalisasi teks. Normalisasi teks adalah proses yang berisi tahap-tahap untuk membuat data teks menjadi bersih dan standar agar dapat digunakan oleh tahap text mining selanjutnya. Tahapan-tahapan tersebut berupa menghapus user pada teks, menghapus URL, menghapus hastag, menghapus emoji, mengubah kata menjadi huruf kecil, menghapus karakter spesial, menghapus angka, menghapus stop words, menghapus teks yang mengandung slangword, mengubah menjadi kata dasar, membersihkan spasi berlebih, menghilangkan spasi di awal, dan menghilangkan spasi di awal. Lalu data yang telah dibershikan akan disimpan ke dalam bentuk file **CSV**.

1. Langkah pertama adalah membaca data dari file gempa.xlsx kemudian memuat data teks ke dalam obyek corpus serta mengimport beberapa library yang diperlukan. Berikut adalah kode yang digunakan untuk keperluan itu.

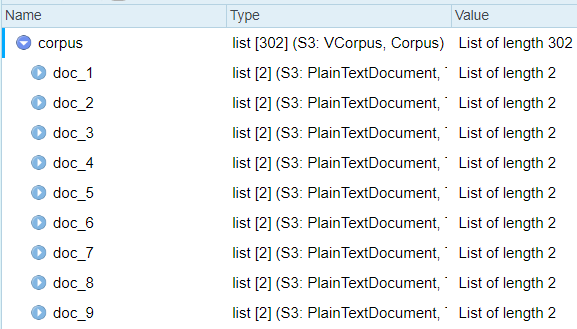
|  |
| --- |
| library(xlsx)  library(tm)  library(katadasaR)  rm(list = ls())  main\_data = read.xlsx("data/gempa.xlsx", sheetName = "Sheet1")  main\_data.corpus = cbind.data.frame(paste0("doc\_",c(1:nrow(main\_data))), main\_data$teks)  colnames(main\_data.corpus) = c("doc\_id", "text") |



Gambar 2 main data



Gambar 3 main\_data\_corpus



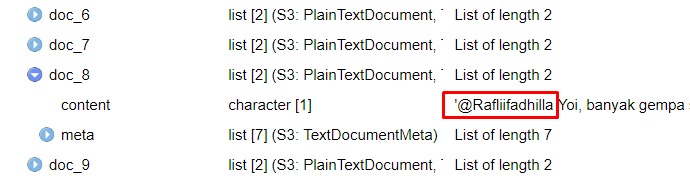
Gambar 4 corpus

1. Langkah selanjutnya yaitu melakukan normalisasi teks.
2. Membersihkan jika ada jarak spasi yang melebihi lebar 2 spasi.

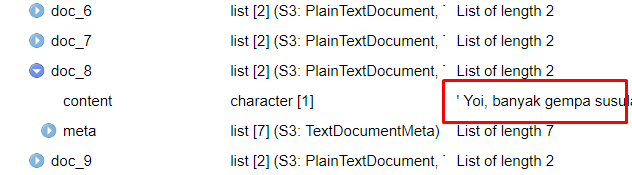
|  |
| --- |
| corpus.processed = tm\_map(corpus, stripWhitespace) |

1. Pada text mining, nama user tidak memiliki makna penting dalam sebuah kalimat sehingga dapat dihapus.

|  |
| --- |
| #menghapus user  removeUser<-function(x) {  return(gsub("@\\w+", "", x))  }  corpus.processed = tm\_map(corpus.processed,content\_transformer(removeUser)) |



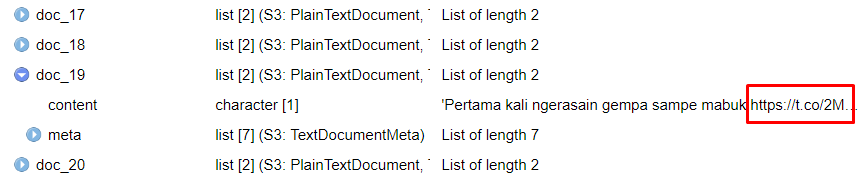
Gambar 5 Sebelum diproses menghapus user.



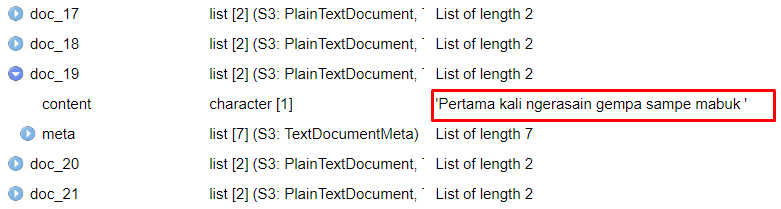
Gambar 6 Sesudah diproses menghapus user.

1. Menghapus URL

|  |
| --- |
| #menghapus URL  cleanURL<-function(x) {  return(gsub("(f|ht)tp(s?)://\\S+", "", x, perl=T))  }  corpus.processed = tm\_map(corpus.processed,content\_transformer(cleanURL)) |



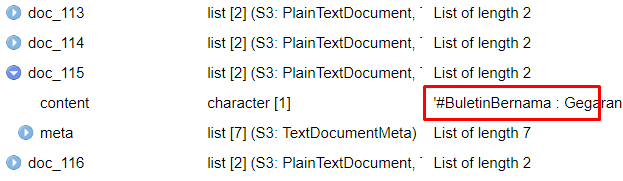
Gambar 7 Sebelum diproses menghapus URL.



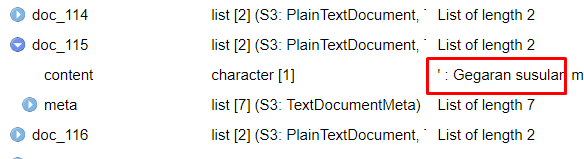
Gambar 8 Sesudah diproses menghapus URL.

1. Menghapus Hashtag

|  |
| --- |
| #menghapus hashtag  cleanHashtag<-function(x) {  return(gsub('#\\S+', '', x))  }  corpus.processed = tm\_map(corpus.processed,content\_transformer(cleanHashtag)) |



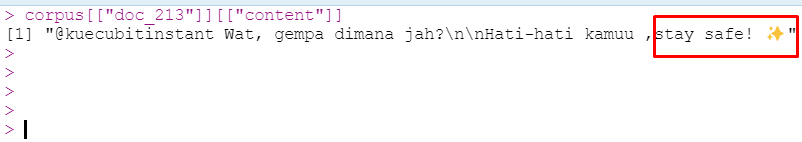
Gambar 9 Sebelum diproses menghapus Hashtag.



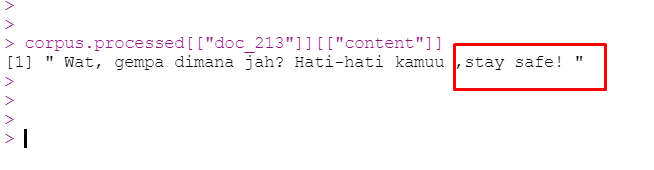
Gambar 10 Sesudah diproses menghapus Hashtag.

1. Menghapus emoji

|  |
| --- |
| #menghapus emoji  cleanEmoji <- function(x){  return(gsub("[^\x01-\x7F]", "", x))  }  corpus.processed = tm\_map(corpus.processed, content\_transformer(cleanEmoji)) |



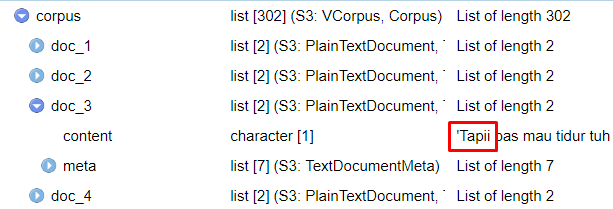
Gambar 11 Sebelum diproses menghapus emoji.



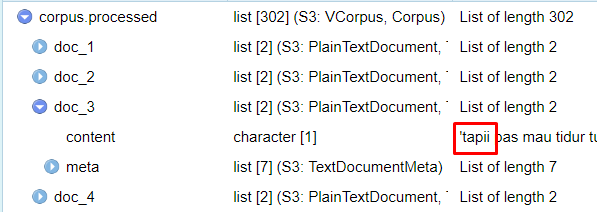
Gambar 12 Sesudah diproses menghapus emoji.

1. Mengubah menjadi huruf kecil

|  |
| --- |
| #mengubah jadi huruf kecil  corpus.processed = tm\_map(corpus.processed,content\_transformer(tolower)) |



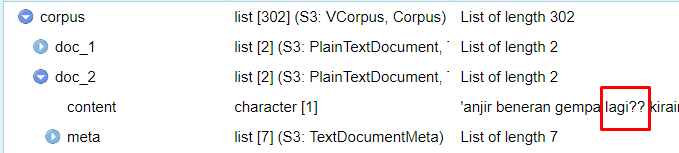
Gambar 13 Sebelum diproses mengubah jadi huruf kecil.



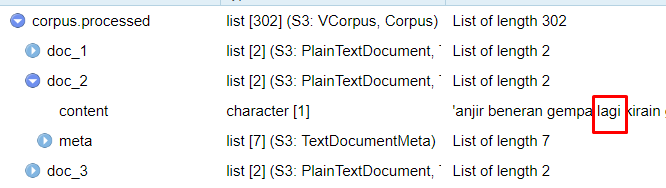
Gambar 14 Sesudah diproses mengubah jadi huruf kecil.

1. Menghapus karakter spesial

|  |
| --- |
| #menghapus tanda baca  corpus.processed = tm\_map(corpus.processed, removePunctuation) |



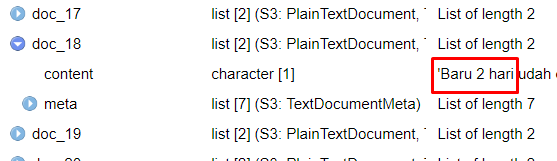
Gambar 15 Sebelum diproses menghapus karakter spesial.



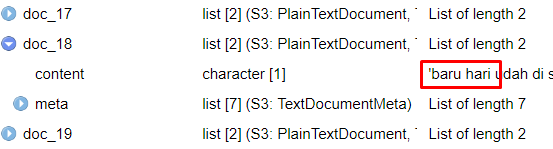
Gambar 16 Sesudah diproses menghapus karakter spesial.

1. Menghapus angka

|  |
| --- |
| #menghapus angka  corpus.processed = tm\_map(corpus.processed, removeNumbers) |



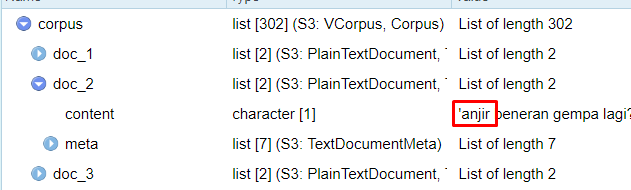
Gambar 17 Sebelum diproses menghapus angka



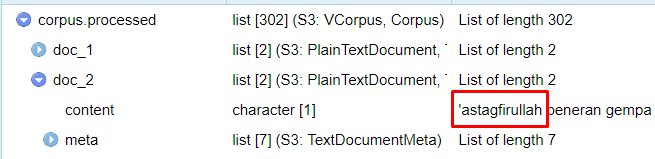
Gambar 18 Sesudah dilakukan proses menghapus angka.

1. Mengubah slangword

|  |
| --- |
| #read slangword  data\_slang <- read.csv("data/slangword.csv")  #menghapus text yang mengandung slangword  getSlangwords<-function(x) {  x <- replace\_internet\_slang(x, slang = paste0("\\b", data\_slang$slang, "\\b"), replacement = data\_slang$formal, ignore.case = TRUE)  return(x)  }  corpus.processed = tm\_map(corpus.processed, content\_transformer(getSlangwords)) |



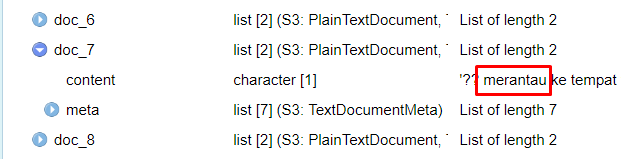
Gambar 19 Sebelum diproses mengubah slangword.



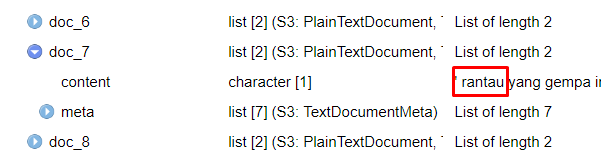
Gambar 20 Sesudah diproses mengubah slangword.

1. Mengubah jadi kata dasar

|  |
| --- |
| #mengubah jadi kata dasar  getKataDasar<-function(x) {  str = unlist(strsplit(stripWhitespace(x), " "))  str<-sapply(str,katadasaR)  str<-paste(str,collapse = " ")  return(str)  }  corpus.processed = tm\_map(corpus.processed,content\_transformer(getKataDasar)) |



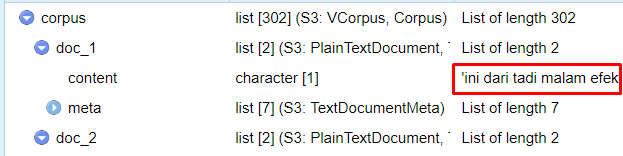
Gambar 21 Sebelum diproses mengubah menjadi kata dasar.



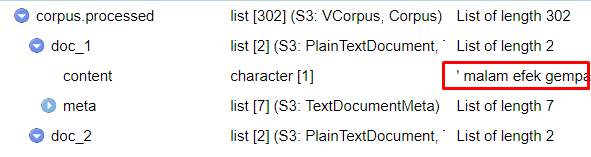
Gambar 22 Sesudah diproses mengubah jadi kata dasar.

1. Menghapus stopwords

|  |
| --- |
| #menghapus stop words  stopwords.id=readLines("data/stopword/tala-masdevid.txt")  corpus.processed = tm\_map(corpus.processed, removeWords, stopwords.id) |



Gambar 23 Sebelum diproses menghapus stopwords.



Gambar 24 Sesudah diproses menghapus stopwords..

1. Membersihkan jarak spasi yang melebihi lebar 2 spasi lagi, karena banyak proses normalisasi pada langkah sebelumnya sehingga memunculkan spasi berlebih.

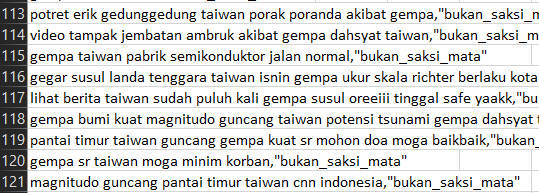
|  |
| --- |
| #membersihkan teks  corpus.processed = tm\_map(corpus.processed, stripWhitespace) |

1. Menghapus spasi di awal teks karena penghapusan user akan memunculkan spasi di awal teks.

|  |
| --- |
| #Menghilangkan Spasi diawal  cleanLeadingWhiteSpace<-function(x) {  return(trimws(x))  }  corpus.processed = tm\_map(corpus.processed, content\_transformer(cleanLeadingWhiteSpace)) |

1. Langkah terakhir pada proses ini yaitu menyimpan data yang telah dilakukan normalisasi ke dalam bentuk file  **CSV** agar dapat dilakukan pada proses selanjutnya

|  |
| --- |
| corpus.df = data.frame(text = unlist(sapply(corpus.processed,'[', "content")), stringsAsFactors = FALSE)  main\_data.processed = cbind.data.frame(corpus.df, main\_data$class\_label)  colnames(main\_data.processed) = c("teks","class\_label")  write.csv(main\_data.processed,"data/gempa\_cleaned.csv", row.names = FALSE) |



Gambar 25 File CSV.

1. **Analisis Teks**

Sebelum data diproses, data teks sebaiknya di lakukan analisis terlebih dahulu, topik teks apa yang dapat dijadikan sebagai acuan dalam class label atau pengklasifikasian. Salah satunya adalah dengan mengetahui jumlah frekuensi kata dalam suatu teks, untuk mempermudah analisis maka dapat digunakan visualisasi dari frekuensi kata dengan Word cloud. Adapaun tahapan pembentukan word cloud sebagai berikut:

1. Muat library yang dibutuhkan dan tambahkan penghapusan cache

|  |
| --- |
| library(wordcloud)  library(tm)  library(RColorBrewer)  rm(list = ls()) |

1. Mengambil data dan simpan ke dalam Corpus

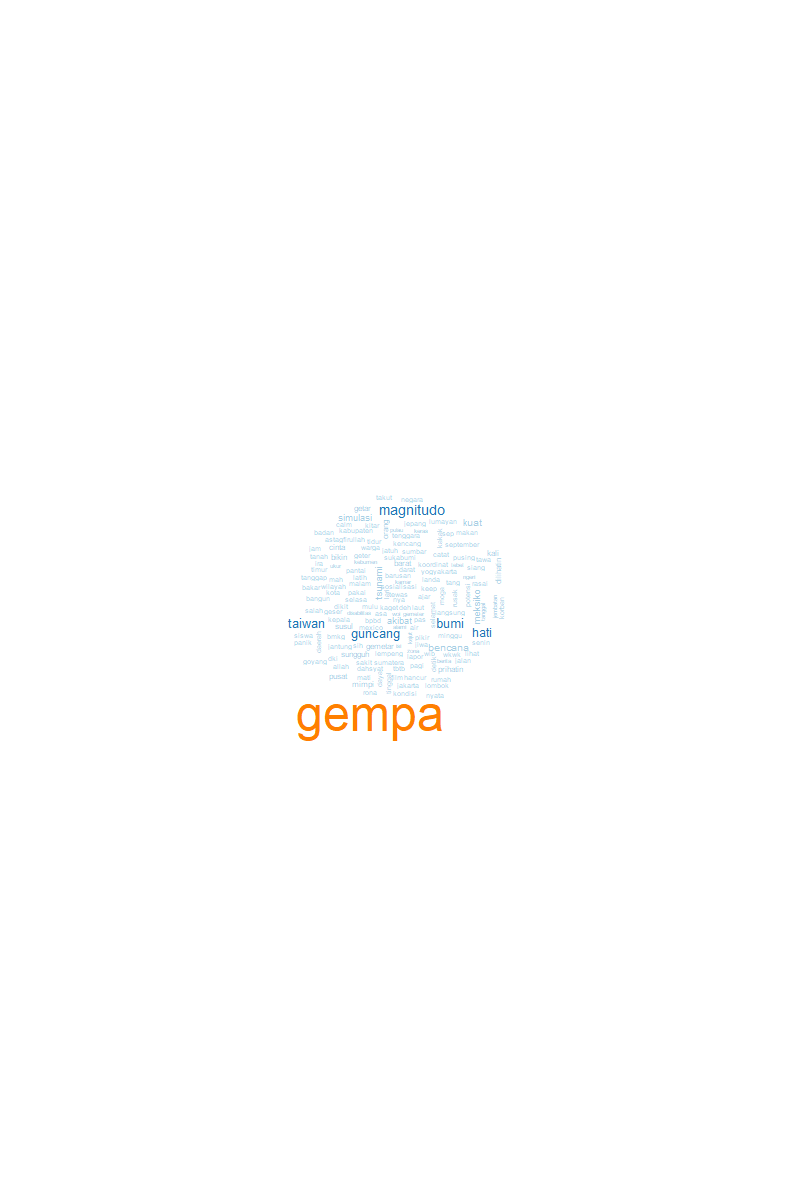
|  |
| --- |
| # TODO 1: Mengambil data  main\_data <- read.csv(  "data-results/data\_gempa\_processed.csv",  stringsAsFactors = FALSE  )  # TODO 2: menyimpan kedalam corpus  main\_data.corpus <- cbind.data.frame(  paste0("doc\_", c(1:nrow(main\_data))),  main\_data$text  )  colnames(main\_data.corpus) <- c("doc\_id", "text")  corpus <- VCorpus(DataframeSource(main\_data.corpus)) |

1. Menghitung jumlah kata per-frekuensi kata

|  |
| --- |
| # TODO 4: Menyimpan TDM kedalam Matrikx  tdm.matrix <- as.matrix(tdm)  # TODO 5: Menghitung frekuensi kemunculan kata dalam seluruh dokumen  term.freq <- rowSums(tdm.matrix)  # TODO 6: Konversi ke dataframe dan mengurutkan data  freq.df <- data.frame(  word = names(term.freq),  frequency = term.freq  )  freq.df <- freq.df[order(freq.df[, 2], decreasing = T), ] |

1. Membuat Word cloud

|  |
| --- |
| # TODO 7: Membuat wordcloud  wordcloud.viz <- wordcloud(  freq.df$word,  freq.df$frequency,  max.words = 150,  colors = brewer.pal(8, "Paired")  ) |



*Gambar 26 Worldcloud*

1. **Ekstraksi Fitur**

Dalam proses ini data tidak terstruktur akan di transformasi menjadi data yang terstruktur. Transformasi ini bertujuan untuk pengelolaan data menjadi model pembelajaran baik klasifikasi, clustering, dan lain-lain. Document Term Matrix dan Term Document Matrix merupakan cara yang dapat digunakan untuk mengubah data teks menjadi data terstruktur. Untuk menghitung jumlah kemunculan data dalam suatu teks digunakan perhitungan pembobotan atau Term Frequency, dalam proyek ini digunakan pembobotan TF dan Jumlah kemunculan data yang muncul akan dihitung per-satu kata atau unigram. Adapun tahapan dari ekstraksi fitur sebagai berikut:

1. Load library tm untuk menggunakan Corpus

|  |
| --- |
| library(tm)  rm(list = ls()) |

1. Baca Data dan buat ke corpus

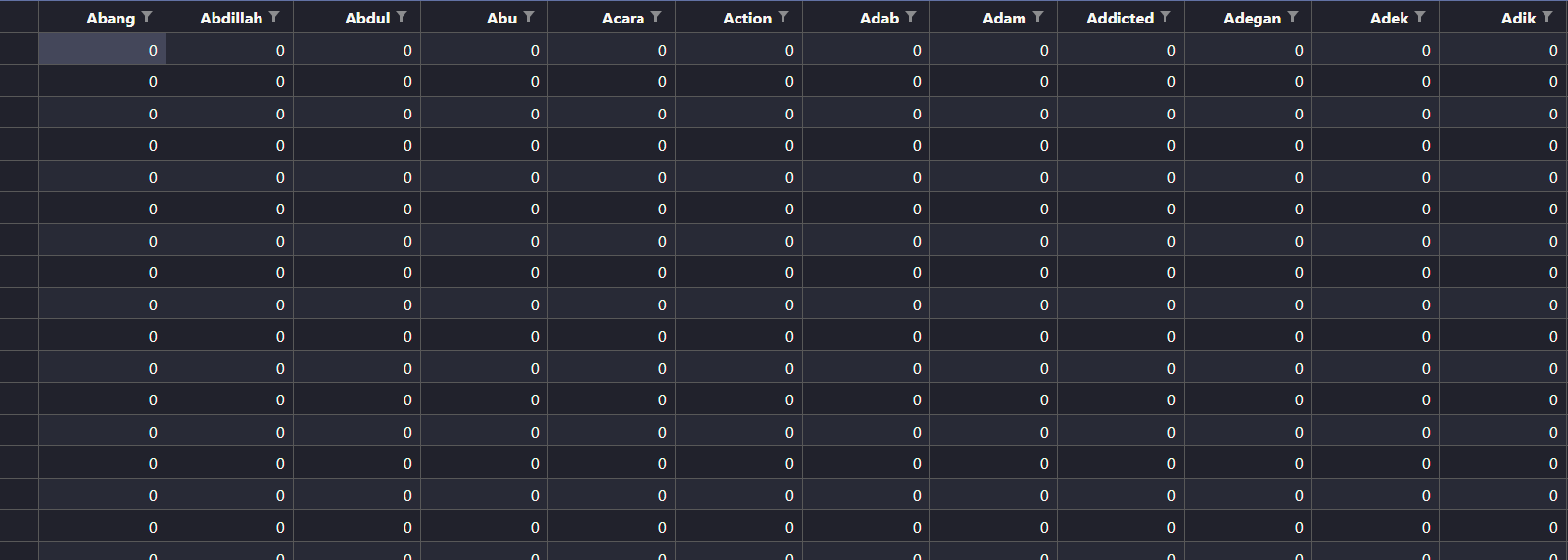
|  |
| --- |
| # TODO: Load Data  main\_data <- read.csv(  "data-results/data\_gempa\_processed.csv",  stringsAsFactors = FALSE  )  main\_data.corpus = cbind.data.frame(  paste0("doc\_",c(1:nrow(main\_data))),  main\_data$text  )  colnames(main\_data.corpus) = c("doc\_id", "text")  corpus <- VCorpus(DataframeSource(main\_data.corpus)) |

1. Transformasi data tidak terstruktur menjadi data terstruktur dengan Document Term Matrix per-satu kata (unigram) dan pembobotan TF-Murni

|  |
| --- |
| # TODO: Transformasi Data terstruktur  dtm <- DocumentTermMatrix(  corpus,  control = list(weighting = weightTf)  )  main\_data.unigram = cbind.data.frame(  as.data.frame(as.matrix(dtm)),  main\_data$class\_label  )  colnames(main\_data.unigram)[ncol(main\_data.unigram)] = "class\_label" |

1. Simpan data ke csv

|  |
| --- |
| # TODO: Simmpan Data  write.csv(main\_data.unigram, "data-results/data-unigram.csv", row.names = FALSE) |

****

*Gambar 27 Hasil Data Terstruktur*

1. **Membangun Model KNN**

Pemodelan menggunakan KNN digunakan untuk data terstruktur dengan pembelajaran terawasi atau Supervised Learning. Dalam proyek ini membangun model menggunakan algoritma KNN yang bertujuan untuk menentukan hasil class label berdasarkan teks yang diinputkan. Untuk dapat menggunakan model KNN digunakan library Class. Adapun tahapan sebagai berikut:

1. Muat library dan hapus data cache

|  |
| --- |
| library(class)  library(gmodels)  rm(list = ls()) |

1. Membaca data dan memasukan ke dalam corpus

|  |
| --- |
| data = read.csv(  "data-results/data-unigram.csv",  stringsAsFactors = FALSE  ) |

1. Memasukan data class label

|  |
| --- |
| row\_labels <- data[, ncol(data)] |

1. Membuat index data yang digunakan mengambil data training dan testing dengan ukuran split data 8:2

|  |
| --- |
| set.seed(123)  size <- floor(0.8 \* nrow(data))  train\_ind <- sample(  seq\_len(nrow(data)),  size = size,  ) |

1. Split data train dan data testing

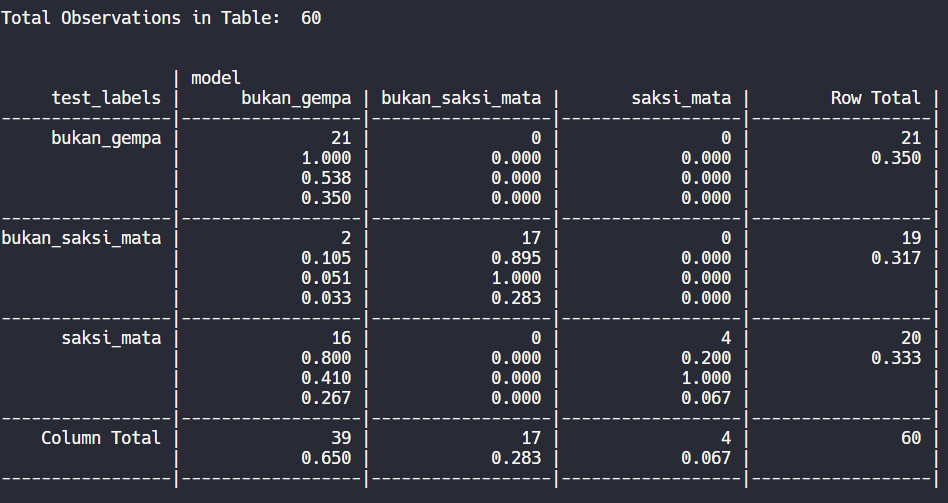
|  |
| --- |
| train\_labels <- data[train\_ind, ncol(data)]  test\_labels <- row\_labels[-train\_ind]  data\_train <- data[train\_ind, 1:(ncol(data)-1)]  data\_test <- data[-train\_ind, 1:(ncol(data)-1)] |

1. Membangun Model KNN dengan jumlah kluster 13

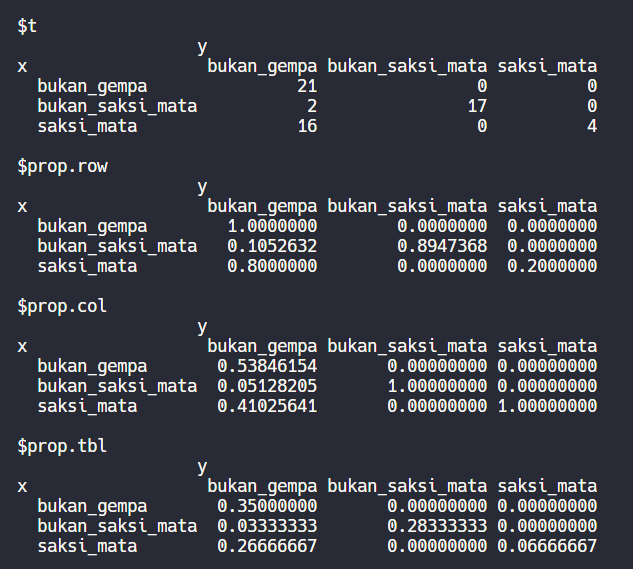
|  |
| --- |
| model <- knn(  train = data\_train,  test = data\_test,  cl = train\_labels,  k = 13  ) |

1. Evaluasi dengan cross table

|  |
| --- |
| ct = CrossTable(  x = test\_labels,  y = model,  prop.chisq = FALSE  )  print(ct) |



*Gambar 28 Hasil Evaluasi Confusion Matrix dengan Cross Table*



*Gambar 29 Hasil Evaluasi Confusion Matrix dengan Cross Table*

1. Cek akurasi

|  |
| --- |
| cm <- table("Predictions" = model, Actual = test\_labels)  accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  acc <- accuracy(cm)  print(acc) |



*Gambar 30 Hasil Akurasi*