## glossary

|  |  |
| --- | --- |
| **Glossary/Function** | **Descript** |
| **html** | Hyper Text Markup Language; standard language for creating and structuring web pages. |
| **url** | Shows the location of a resource (web page, file, image, API, etc.). sample: www.google.com |
| **Script** | a set of commands or code written in a programming language or scripting language and executed sequentially by a computer. |
| Package | a collection of functions, datasets, and documentation created to extend the basic capabilities of R |
| install.packages ( n) | For installing the required packages; n = package name |
| Library (n) | Activate package; n = package name |
| # | to descript, make comment, note, explanation |
| %>% | Pipe, connectiong from function to function |
| read.csv2 ( ) | to read file comma separated values (CSV) |
| write.csv2 ( ) | function to save (export) frame data into CSV files, but with different separator and decimal formats, commonly used in Europe and Indonesia |
| Unigram | a model or unit of text analysis that considers each word as a stand-alone unit (token), without regard to the words before or after it. |
| Package {polite} | a tool in the R programming language that makes it easier for us to do web scraping ethically |
| Package {rvest} | R package used for web scraping, which is to easily retrieve (extract) data from web pages. |
| Package {magrittr} | R package that provides the pipe operator (%>%) to make it easier to write chaining code in a more readable way. |
| Package {stringr} | R package that contains a collection of functions for manipulating strings (text) consistently, easily, and efficiently. |
| Package {textmineR} | package in R for text mining and topic modeling designed to make the text analysis process easier, faster, and more structured. |
| Package {tibble} | a package in R that provides a modern data frame format and is more convenient to use than R's default data.frame. This package is part of the tidyverse. |
| Package {topicmodels} | A package in R used for topic modeling, particularly for implementing the Latent Dirichlet Allocation (LDA) and Correlated Topic Model (CTM) methods. This package makes it easy for users to discover hidden topics in a collection of text-based documents. |
| Package {Matrix} | a package in R that is used to create and manage efficient matrices, especially sparse matrices, which are very useful in large data analysis such as text mining, machine learning, and optimization. |
| Package {gglot2} | A package in R for creating data visualizations based on the Grammar of Graphics. This package is a core part of the tidyverse and is widely used to produce flexible, aesthetic, and high-quality graphics. |
| Package {slam} | (Sparse Lightweight Arrays and Matrices) is a package in R that is used to handle sparse matrices efficiently, especially Document-Term Matrix (DTM) and Term-Document Matrix (TDM) in text mining. |
| Package {ldatuning} | An R package used to determine the optimal number of topics in Latent Dirichlet Allocation (LDA) topic modeling. This package helps select the best number of topics (k) by calculating various evaluation metrics on the LDA results. |
| Package {tidytext} | A package in R used for text analysis using a tidy data approach (data in a neat tabular format). This package simplifies the text mining process by leveraging the tidyverse ecosystem, including dplyr, ggplot2, and tidyr. |
| Package {dplyr} | An R package for data manipulation, transformation, and analysis with simple, fast, and consistent syntax. This package is part of the tidyverse ecosystem. |
| Package {lsa} | R package used to perform Latent Semantic Analysis (LSA), a matrix-based text analysis technique for finding latent (hidden) relationships between words and documents. |
| Package {tm} | An R package used for processing, cleaning, and analyzing text before performing further analysis such as clustering, LSA, or LDA. This package is one of the most popular packages for text mining in R. |
| Package {tidyr} | An R package used for tidying and reorganizing data structures for easier analysis. This package is part of the tidyverse, which focuses on data in a "tidy data" format: each column = variable, each row = observation, each cell = single value. |
| Package {scales} | An R package used to organize and format scales in data visualizations, particularly when using ggplot2. This package helps modify the appearance of numbers, colors, labels, dates, percentages, logarithms, and other scale elements for easier reading. |
| tf-idf | TF-IDF (Term Frequency – Inverse Document Frequency) is a method for measuring how important a word is in a document compared to the entire collection of documents (corpus). |
| Document term matrix | representation of text data in the form of a matrix that shows how many times each word (term) appears in each document in a collection of documents (corpus). |

## DATA COLLECTIVE

1. Search “Keyword” selection for human-orangutan interaction (HOI) cases in news media in Indonesia

“Keyword” selection aims to filter out news that is not relevant to the HOI case. Keywords also create effectiveness in searching for news that is in accordance with the purpose of managing HOI cases. The selection of keywords affects the HOI studies being reviewed.

We use the keywords objective:”orangutan”, limit area:”kalimantan”, “indonesia” and time “year”. The word AND is used to combine two different topics that have similar affiliations. Searching with these keywords produces a number of local news focused on kalimantan.

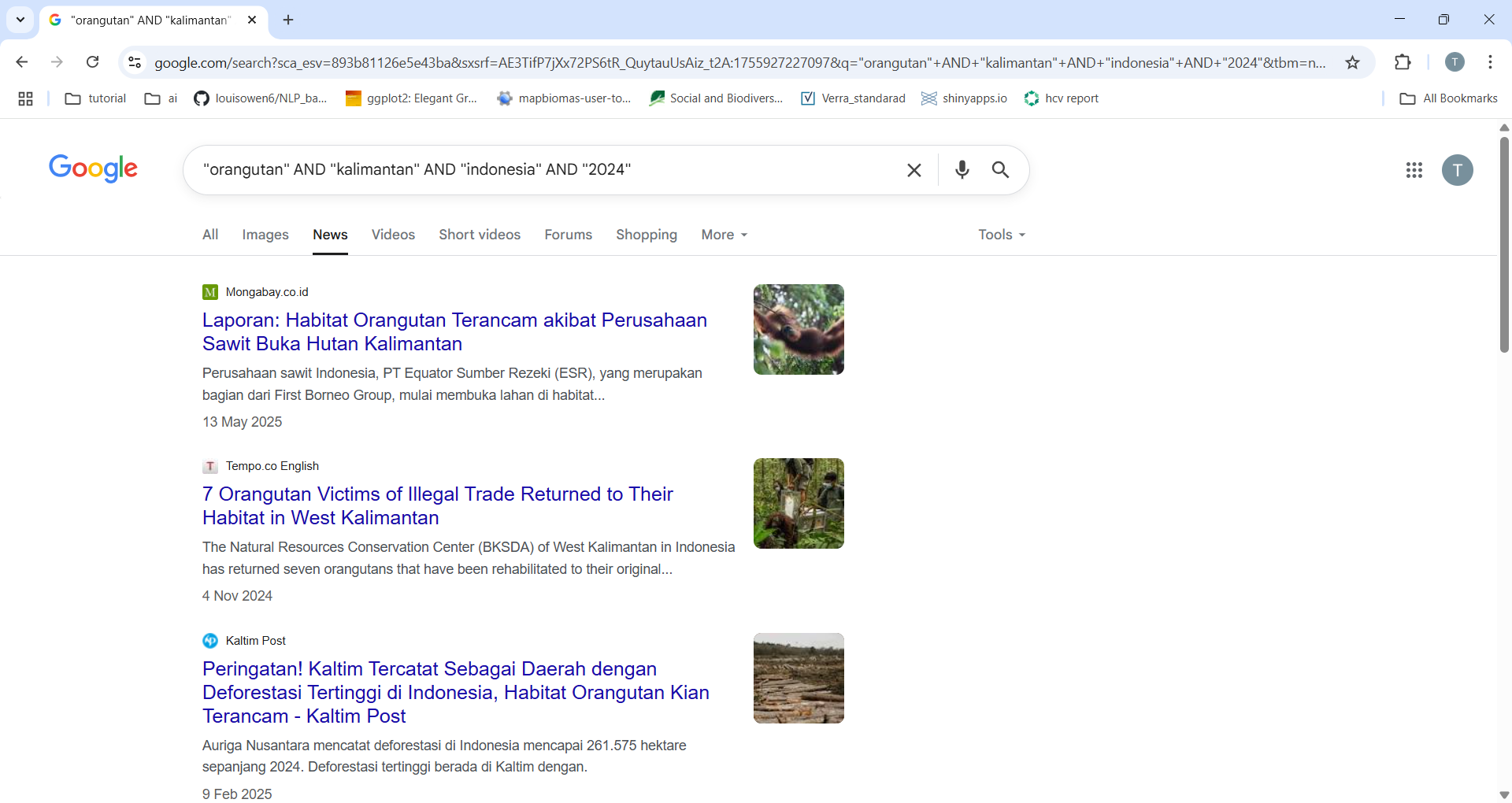


Figure 1 result from searching engine

1. Identify html (element) in news url

Hyper Text Markup Language (html) identification is needed to extract text data needed in the ASF study. HTML identification can use the gadget selector which is an additional extension on Google Chrome. Installing or adding extensions can be done by searching for the "selectorgadget" key and adding the extension.

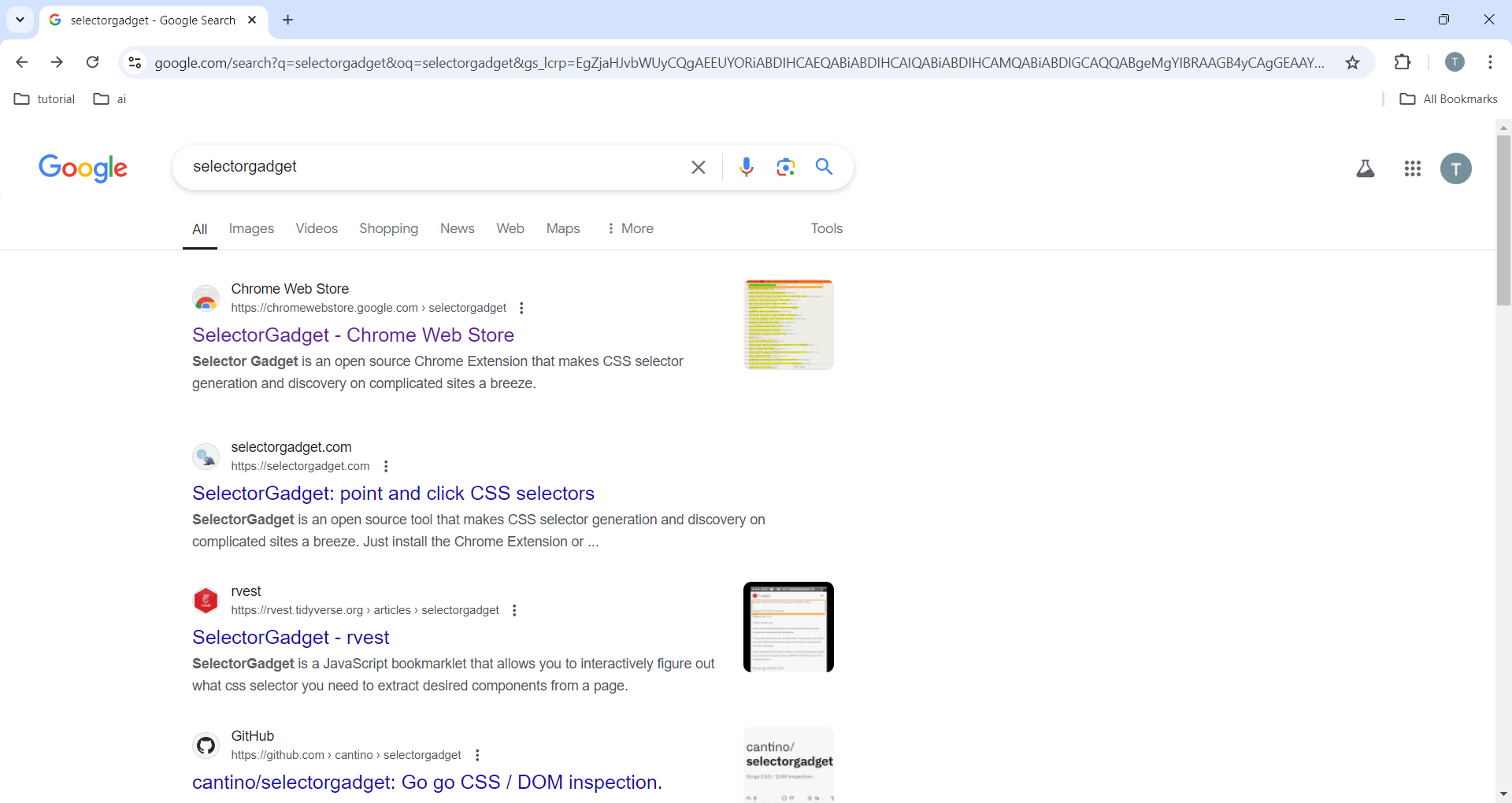


figure 2 selectorgadget from extension google chrome

In addition, html identification can be helped by opening the web element structure by right-clicking on the mouse or keypad and selecting inspect. The identified url gives a sign with a block or appears in the SelectorGadget box. Please note that each news website has different html elements, so you need to be careful to put the url identification results in the Rstudio script.

Sample usage of selectorgadget by opening inspect is in figure 4. The html code related to the title is "h1" from the identification results with selectorgadget which is green and the information seen in the box that appears below and the text "selectorgadget" in the html structure on the right side. There are also data needed in the form of the title, content and publication time of the article which can be further analyzed for text mining.

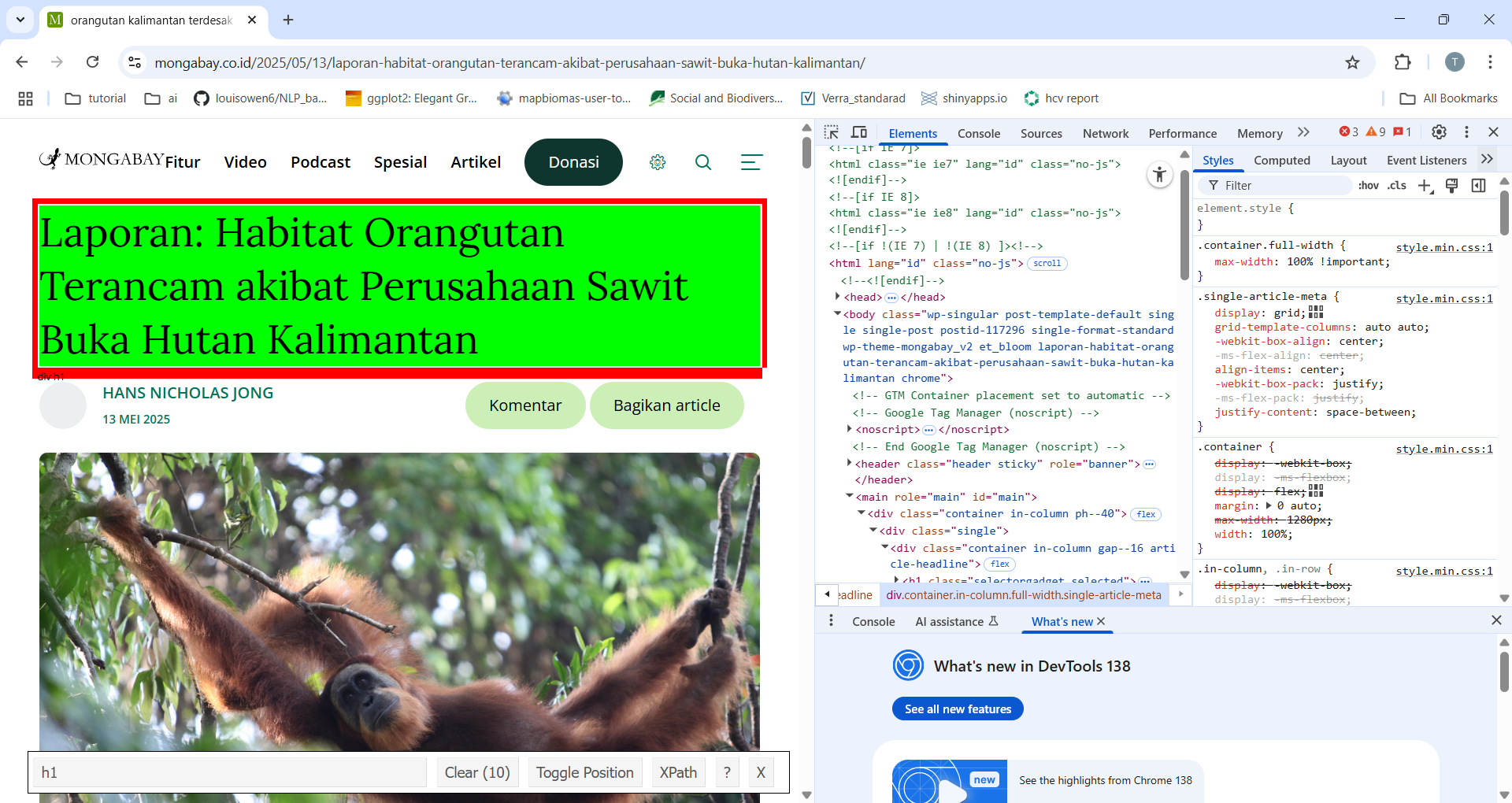


figure 3 view html address

1. Building script/coding for web scraping

The script is created with RStudio using packages (a collection of available functions designed with the R programming language). The packages used are:

* package polite (Perepolkin 2023), Polite package designed to fetch text data from the web by adhering to guidelines, setting the time range for fetching text data, and avoiding the load on the web server.
* package rvest (Wickham 2024), rvest package is a collection of main functions in data retrieval activities on the web.
* package magrittr (bache et al. 2022),magrittr package is used to connect one function (script) to another.
* Packafe stringr (wickham 2023). This package is used to manipulate text data.

The development of this script/coding is done with a combination of these three packages.

Tahapan web scraping:

Step 1. Read excel.csv file containing the collection of news web urls with the script, below:

#1 open file CSV

library(readr)

orangutannews <- read.csv2("D:/011\_git/002\_text\_mining/topic\_modelling/006\_data-tmse\_imo-news-2011-2024.csv")

To see the table content, you can use the View (orangutannews) function. asf is an object assignment or data frame, to call the data source with the previous script function. The content of the table in looks like figure 4.

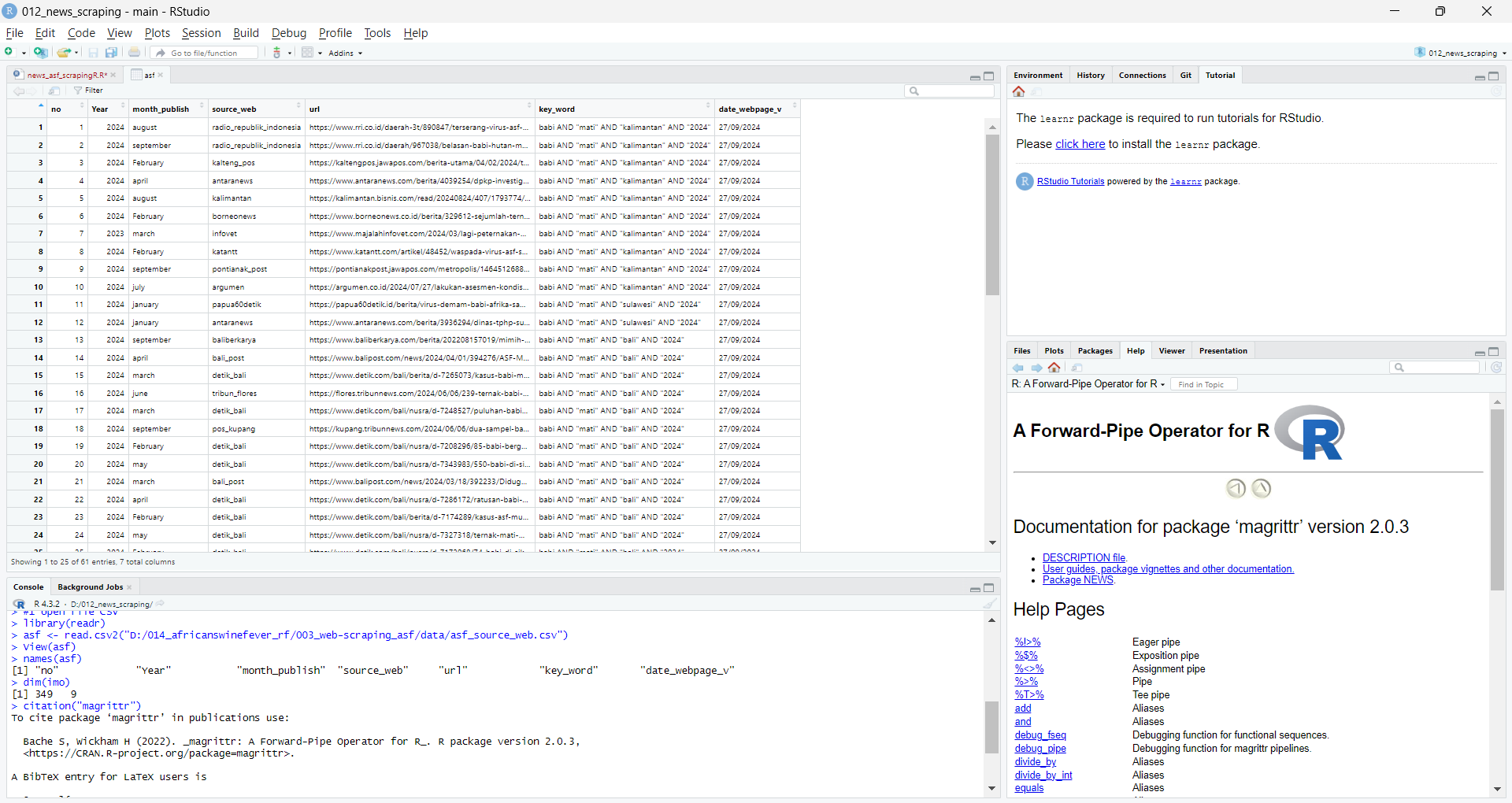


figure 4 Content of table

Step 2.1 Install the packages required for data scraping, then activate the packages.

#2.1 install packages that used

install.packages("polite")

install.packages("rvest")

install.packages("magrittr")

install.packages(“stringr”)

Step 2.2 active packages

#2.2 activate packages that used

library(magrittr)

library(dplyr)

library(rvest)

library(polite)

library(stringr)

step 3.1 creating a new column for scraping results

#3.1 make new coloum for product of scraping

orangutannews$title <- NA #for scraping results of title news

orangutannews$date\_publish <- NA #for scraping results of date

orangutannews$content <- NA #for scraping results of content

The results of creating columns can be seen on the next sheet, as in Figure 6.

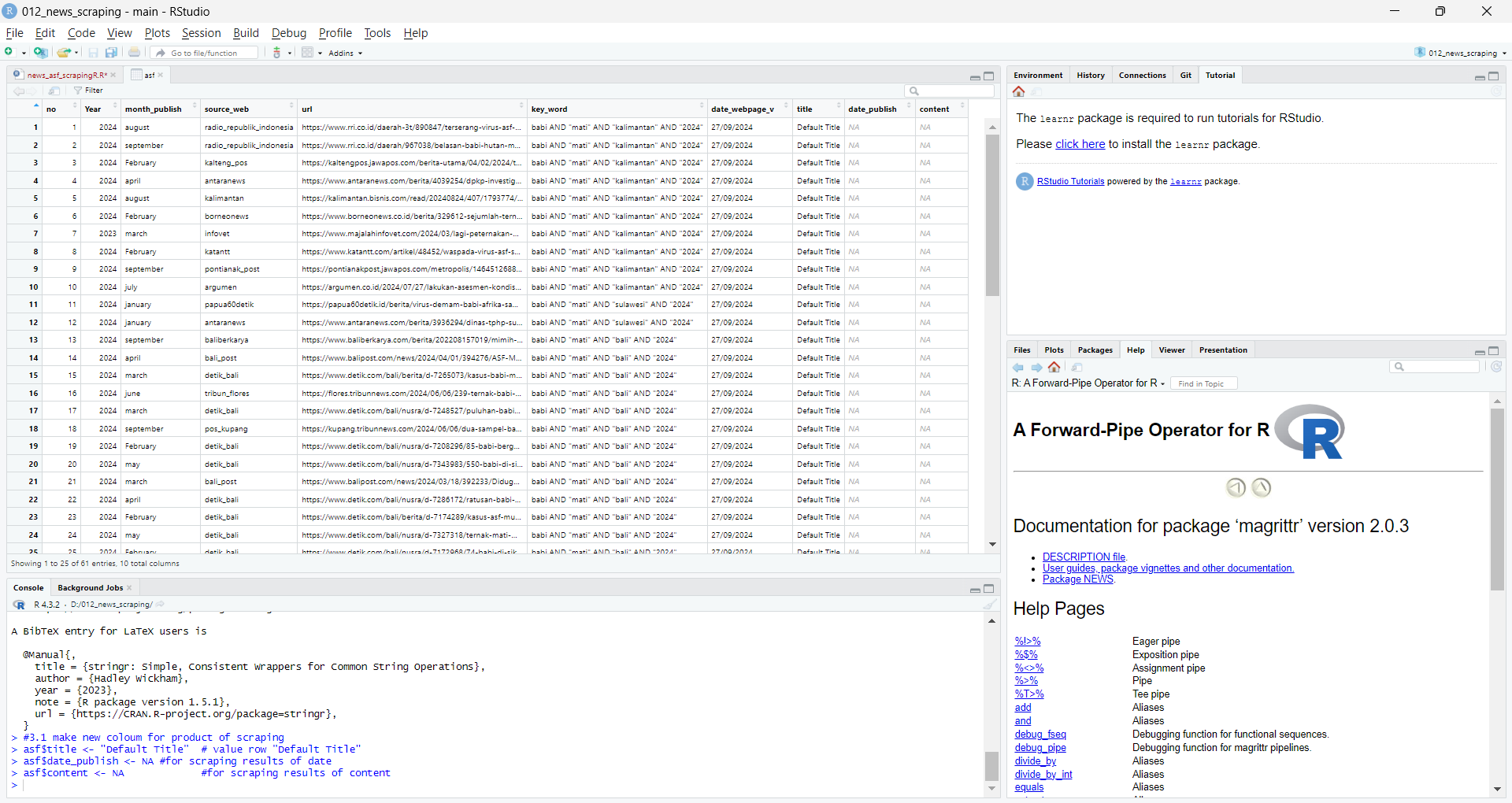


figure 5 new column creation result

Step 3.2. web scraping is done. This stage has several things to note, especially on the html address on the website. In general conditions or without problems, you can use the script below.

This script is intended to perform web scraping on the “url” address in the radio\_republic\_indonesia row in the “source\_web” column with:

* The publication date of the news is scraped with the code "small" and placed in the "date\_publish" column.
* the news title is scraped with the code "#news-title" and placed in the "title" column.
* - news content is scraped and placed in the “content” column

#3.2 scraping based website\_source from orangutannews

for (i in 1:nrow(orangutannews)) {

#for scrape web source from radio\_republik\_indonesia

if(asf$source\_web[i] == "radio\_republik\_indonesia") {

#extract article publication

orangutannews [i,"date\_publish"] <-

polite::bow(orangutannews f[i,"url"]) %>% #introduce yourself to the host

polite::scrape(.) %>% #scrape the content of authorized page

rvest::html\_node("small") %>% #read the codes, take all nodes

rvest::html\_text() %>% #change in to text format

stringr::str\_trim() #remove extra spaces

#extract article title

orangutannews [i,"title"] <-

polite::bow(asf[i,"url"]) %>% #introduce yourself to the host

polite::scrape(.) %>% #scrape the content of authorized page

rvest::html\_node("#news-title") %>% #read the title codes

rvest::html\_text() %>% #change in to text format

stringr::str\_replace\_all(.,"[\r\n\t\"]", "") %>% #remomve line breaks and all symbols

stringr::str\_trim() #remove extra spaces

#extract article content

orangutannews [i,"content"] <-

polite::bow(asf[i,"url"]) %>% #introduce yourself to the host

polite::scrape(.) %>% #scrape the content of authorized page

rvest::html\_node("p") %>% #read the content codes

rvest::html\_text() %>% #change in to text format

stringr::str\_replace\_all(., "[\r\n\t\"]","") %>% #remomve line breaks and all symbols

stringr::str\_trim() %>% #remove extra spaces

paste(collapse="") #combine all sentences

#add sleep time (10 seconds)

Sys.sleep(10)

} else(next)

}

## PRE PROCESSING

The data obtained is non-standard text in the form of upper and lower case letters, text with different vocabulary but the same meaning, and punctuation or numbers that are not meaningful, so pre-processing is necessary to standardize the data. Text standardization can be done multiple times because there is text that was missed in the previous stage.

1. Activate packages

activate the packages required to operate the functions in R for standardization text

library(dplyr) #data manipulation

library(stringr) #texts manipuation

library(tidytext) #data processing

library(magrittr) #pipe acivate (function to function)

library(stringi) #manipulation and process of string

library(purrr) #detection false or NA value

1. standardize text in selected columns

This research aims to reveal latent topics in sentences appearing in news content, so the data that needs to be standardized is in the content column of the previous web scraping results. The script that was built can be seen in the following section.

#standardization of text in columns

orangutannews <- read.csv2("D:/011\_git/002\_text\_mining/topic\_modelling/006\_data-tmse\_imo-news-2011-2024.csv")

content\_imo <- orangutannews$news\_content #retrieve content column as 'content\_imo' object

content\_standard <-

content\_imo %>%

str\_to\_lower() %>% #change text to lowercase

str\_replace\_all("[[:punct:]]", "") %>% #removing punctuation

stri\_replace\_all\_regex("\\d+", "") %>% #deleting numbers with strings

stri\_replace\_all\_regex("[^[:print:]]", "") %>% #removing non-printable characters with stringi

str\_trim() #reduce excess spaces

print(content\_standard)

write.csv2(content, "D:/011\_git/002\_text\_mining/topic\_modelling/006\_data-tmse\_imo-news-2011-2024.csv")

## LDA MODELLING

Topic modeling is used to discover hidden topic patterns within a collection of news content. This concept assumes that each piece of content is composed of a mixture of several topics, and each topic is composed of a specific distribution of words. LDA uses a probabilistic approach to estimate the distribution of topics within news content and the distribution of words within topics. With this method, we can understand the main themes in a collection of text without having to read the entire news content.

1. Activate packages

activate the packages required to operate the functions in R for lda modelling

library(readr) #read data

library(dplyr) #data manipulation

library(stringr) #text manipulation

library(tidytext) #data proces

library(magrittr) #pipe activation (function to function)

library(stringi) #manipulation and string processing

library(purrr) #mistake detection or find NA value

library(tm) #text mining

1. Tokenize unigram

This script is to create a new row containing one term in sequence according to the order in the news content

orangutannews <- read.csv2("D:/011\_git/002\_text\_mining/topic\_modelling/006\_data-tmse\_imo-news-2011-2024.csv")

contentou <- orangutannews$content\_clean

#1st step tokenize unigram

unigram\_ounews <- orangutannews %>%

mutate(line\_id = row\_number()) %>% #create row IDs so that the order is maintained

unnest\_tokens(word, content\_clean, token = "words", to\_lower = TRUE) %>%

group\_by(doc\_id) %>%

mutate(word\_position = row\_number()) %>% #save word position in content

ungroup()

The results of the term-by-term separation per row and column can be seen in the image below

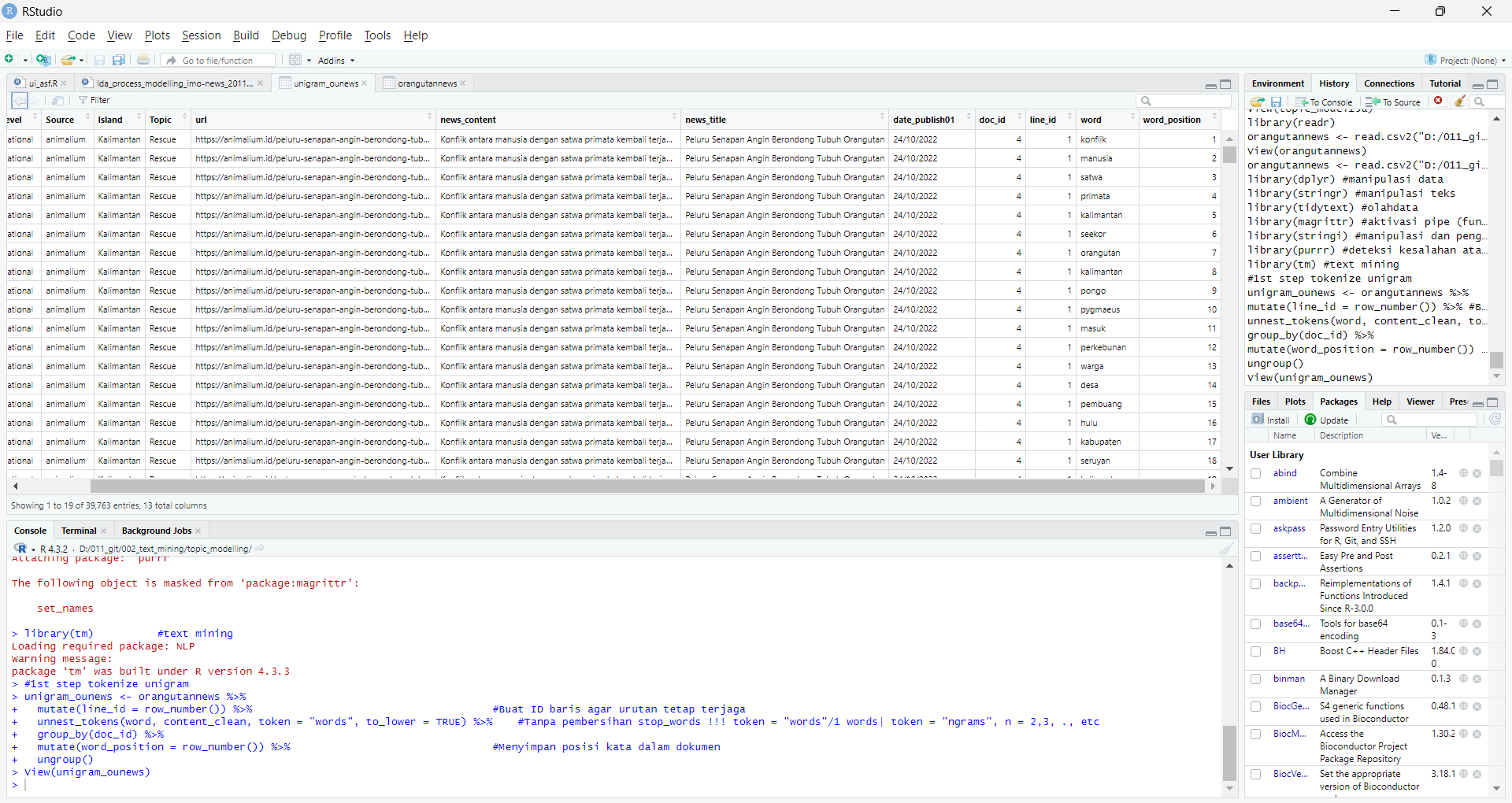


Figure 5 words in each row and column in order

1. remove meaningless terms, such as conjunctions in Indonesian, namely yang = that, di = in, kepada = to, etc. meaningless terms based on the lexicon created by Tala 2016. The way this script works is to filter the word if the word in the same column row as the lexicon word does not mean tala 2016.

#membersihkan kata yang tidak berarti

stopwords\_id <- readLines("D:/011\_git/001\_raw-data/orangutan\_news/ID-Stopwords-master/id.stopwords.02.01.2016.txt")

# Menghapus stopwords dari barisan kata

unigram\_ounews01 <- filter(unigram\_ounews,!(word %in% stopwords\_id))

1. remove part2 after get result from part 1

The result is that the number of words has started to decrease and each vocabulary is easy to observe, so that some meaningless words outside the 2016 Tala lexicon can be detected, such as people's names, abbreviations of location names, names of days and others. The second stage of cleaning up meaningless words based on a review of the words in each news content can be seen in the following box.

#remove words included in the unwanted word list

unigram\_ounews02 <- unigram\_ounews01 %>%

filter(!word %in% c("kalbar", "kalteng", "kaltim", "kalsel")) %>%

filter(!word %in% c("senin", "selasa", "rabu", "kamis", "jumat", "sabtu", "minggu")) %>%

filter(!word %in% c("ade", "abduh","aceh","affandi","apaapa","anyin","ardi","astri","afrika","aji","albertus","andika",

"boncel","dendi","fitria","muriansyah","suriani","taufik","samosir","agung","agustinus",

"sustyo","paulus","jamri","ariansyah","arifin","argitoe","sibarani",

"barata","bharata","aan","adi","adirahmanta","adisa","ariyadi",

"azmadi","chadidjah","hartono","hadi","hadiatul","heri","hermayadi","hernowo","idarno","

”iriyoni","iriono","iryono","irwiratno",

"iskandar","purnomo","suciadi","suryadi","supianor","sutiadi","yadi","agus","agungia","ahmad","aini","ali","aliansah","amalia",

"and","anthonius","antoro","anwar","ardhanianto","ari","arsyad","bambang",

"windy","gail","campbell","karmele","llano","martono","masriwiyono","muria", "noor","sadtata","sanchez","seno","sihantoro","siti",

"suryani","susilo","suyono","siregar","simon","silva","siponti","tri","trimarsito","trismanto","yuliana","zola",

"zoya","zulfiqri","yusuf","yudhi","yudi","yudas","zariansyah","yosafat","yohanes",

"yakobus","yansyah","yansen","yayan","yani","yanti","wiwied","wisnu","wiratno","wirendro",

"windaryati","widodo","widiantoro","wicaksono","wibawanto","wawan","utomo",

"utomo","utami","tjahyana","tito","heribertus","joss")) %>%

filter(!word %in% c("januari","februari","maret","april","mei","juni","juli","agustus","september","oktober","november","desember")) %>%

filter(!word %in% c("utara","barat","selatan","timur")) %>%

filter(!word %in% c("wita","wib")) %>%

filter(!word %in% c("si","no","pas")) %>%

filter(!word %in% c("perundang","undangan")) %>%

filter(!word %in% c("nama","bernama","dinamai","namanya")) %>%

filter(!word %in% c("nol","satu","kesatu","dua","kedua","tiga","ketiga","empat","keempat",

"lima","kelima","enam","keenam","tujuh","ketujuh","delapan","kedelapan",

"sembilan","kesembilan","sepuluh","kesepuluh"))

View(unigram\_ounews02)

1. The next stage is changing non-standard words into standard ones, correcting words with typos and combining several words with the same meaning into one word.

# Mengganti tokenisasai 'unigram' tidak baku menjadi baku, dan persamaan kata

unigram\_ounews06 <- unigram\_ounews05 %>%

mutate(word = trimws(word)) %>% # Hapus spasi ekstra

mutate(word = str\_replace\_all(word, "[[:punct:]]", "")) %>% # Hapus tanda baca

mutate(word = str\_replace(word, "^(mentranslokasi|translokasikan|mentranslokasikan|translokasi|tranlokasi|dipindahkan|dipindah)$", "memindahkan")) %>%

mutate(word = str\_replace(word, "^(menghimbau)$", "mengimbau")) %>%

mutate(word = str\_replace(word, "^(ha|hektare)$", "hektar")) %>%

mutate(word = str\_replace(word, "^(cm|sentimeter)$", "centimeter")) %>%

mutate(word = str\_replace(word, "^(kawasan)$", "area")) %>%

mutate(word = str\_replace(word, "^(lantaran)$", "karena")) %>%

mutate(word = str\_replace(word, "^(habituasi)$", "adaptasi")) %>%

mutate(word = str\_replace(word, "^(dikonversi)$", "berubah")) %>%

mutate(word = str\_replace(word, "^(mengajari)$", "mengajarkan")) %>%

mutate(word = str\_replace(word, "^(luka|terluka|lukaluka)$", "cedera")) %>%

mutate(word = str\_replace(word, "^(laporkan|melapor)$", "melaporkan")) %>%

mutate(word = str\_replace(word, "^(memonitor|monitoring|dipantau|pemantauan)$", "memantau")) %>%

mutate(word = str\_replace(word, "^(tersisa)$", "sisa")) %>%

mutate(word = str\_replace(word, "^(membopong)$", "membawa")) %>%

mutate(word = str\_replace(word, "^(ekor|seekor)$", "individu")) %>%

mutate(word = str\_replace(word, "^(melepasliar|meliarkan|lepasliarkan|lepas|dirilis)$", "melepasliarkan")) %>%

mutate(word = str\_replace(word, "^(perjualbelikan|perdagangkan|memperjual|memperdagangkan|memperniagakan)$", "memperjualbelikan")) %>%

mutate(word = str\_replace(word, "^(menaklukan)$", "menaklukkan")) %>%

mutate(word = str\_replace(word, "^(mencarikan|mencariny|pencarian|cari)$", "mencari")) %>%

mutate(word = str\_replace(word, "^(mencengkram)$", "mencengkeram")) %>%

mutate(word = str\_replace(word, "^(mendokumentasikan|catat|tercatat|mencatat|mencatatkan)$", "mendokumentasi")) %>%

mutate(word = str\_replace(word, "^(titip)$", "titipkan")) %>%

mutate(word = str\_replace(word, "^(catatan)$", "dokumentasi")) %>%

mutate(word = str\_replace(word, "^(berisiko|resiko)$", "risiko")) %>%

mutate(word = str\_replace(word, "^(memasuki|masuk|memasukkan|masukkan|masukan)$", "masuk")) %>%

mutate(word = str\_replace(word, "^(berladang)$", "berkebun")) %>%

mutate(word = str\_replace(word, "^(kebunku)$", "kebun")) %>%

mutate(word = str\_replace(word, "^(kebunkebun)$", "perkebunan")) %>%

mutate(word = str\_replace(word, "^(terdeteksi)$", "terpantau")) %>%

mutate(word = str\_replace(word, "^(anakanak|anakan)$", "anak")) %>%

mutate(word = str\_replace(word, "^(mengakibatkan|sebabkan)$", "menyebabkan")) %>%

mutate(word = str\_replace(word, "^(untung)$", "beruntung")) %>%

mutate(word = str\_replace(word, "^(pembakaran)$", "membakar")) %>%

mutate(word = str\_replace(word, "^(penemuan|temukan)$", "menemukan")) %>%

mutate(word = str\_replace(word, "^(engevakuasi)$", "evakuasi")) %>%

mutate(word = str\_replace(word, "^(memakan)$", "makan")) %>%

mutate(word = str\_replace(word, "^(serahkan)$", "menyerahkan")) %>%

mutate(word = str\_replace(word, "^(selamatkan)$", "menyelamatkan")) %>%

mutate(word = str\_replace(word, "^(wargamei)$", "warga mei")) %>%

mutate(word = str\_replace(word, "^(hewan)$", "satwa")) %>%

mutate(word = str\_replace(word, "^(memelihara)$", "pelihara")) %>%

mutate(word = str\_replace(word, "^(undangundang)$", "peraturan")) %>%

mutate(word = str\_replace(word, "^(tewas)$", "mati")) %>%

mutate(word = str\_replace(word, "^(terfragmentasi)$", "fragmentasi")) %>%

mutate(word = str\_replace(word, "^(meningkatdari)$", "meningkat")) %>%

mutate(word = str\_replace(word, "^(berkelamin|berjenis)$", "kelamin")) %>%

mutate(word = str\_replace(word, "^(koordiantor)$", "koordinator")) %>%

mutate(word = str\_replace(word, "^(sosialiasi|sosialisasikan|bersosialisasi|mensosialisasikan|menyosialisasikan)$", "sosialisasi")) %>%

mutate(word = str\_replace(word, "^(pekan|sepekan)$", "")) %>%

mutate(word = str\_replace(word, "^(usia|berusia|seusia)$", "")) %>%

mutate(word = str\_replace(word, "^(bayibayi)$", "bayi")) %>%

mutate(word = str\_replace(word, "^(kronologis)$", "kronologi")) %>%

mutate(word = str\_replace(word, "^(siswasiswa|siswasiswi|siswi)$", "siswa")) %>%

mutate(word = str\_replace(word, "^(bergerakgerak|bergoyang|bergoyanggoyang)$", "bergerak")) %>%

mutate(word = str\_replace(word, "^(berhatihati)$", "waspada")) %>%

mutate(word = str\_replace(word, "^(berdasarkan)$", "")) %>%

mutate(word = str\_replace(word, "^(tribun)$", "")) %>%

mutate(word = str\_replace(word, "^(orangutafan|orangutans)$", "orangutan"))

1. Next, return the words in each row of the column back to their original form in the form of combined news content.

# \*\*Mengembalikan ke Teks Asli Sesuai Urutan Awal\*\*

content\_restored02 <- unigram\_ounews07 %>%

arrange(doc\_id, word\_position) %>% # Mengurutkan kembali berdasarkan posisi kata

group\_by(doc\_id) %>%

summarise(content\_restored = str\_c(word, collapse = " ")) %>%

ungroup()

# Gabungkan kembali hasil tokenisasi ke database awal berdasarkan doc\_id

orangutannews01 <- orangutannews %>%

left\_join(content\_restored02, by = "doc\_id")

1. The word has returned to the original news content, and is now starting to be cleaned of meaningless words. The next step is to change the organization's name to an acronym. This is to prevent the word from being mixed with other words, thus disrupting the text data management process.

#subtitution name of instantion to akronym

orangutannews02 <- orangutannews01 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "yiari"),

str\_replace(content\_restored, "international animal rescue|inisiasi alam rehabilitasi", ""),

content\_restored))

orangutannews03 <- orangutannews02 %>%

mutate(content\_restored = if\_else(

str\_detect(content\_restored, "bksda"), # Cek apakah ada kata "bksda"

str\_replace\_all(content\_restored,

"badan konservasi sumber daya alam|balai konservasi sumber daya alam|

konservasi sumber daya alam|skw|seksi konservasi wilayah",

""), # Hapus frasa yang sesuai

content\_restored # Jika tidak ada "bksda", tetap pakai teks asli

))

orangutannews04 <- orangutannews03 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "tn"),

str\_replace(content\_restored, "taman nasional", ""),

content\_restored))

orangutannews05 <- orangutannews04 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "bosf"),

str\_replace(content\_restored, "borneo orangutan survival foundation", ""),

content\_restored))

orangutannews06 <- orangutannews05 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "cop"),

str\_replace(content\_restored, "centre orangutan protection|centre for orangutan protection", ""),

content\_restored))

orangutannews07 <- orangutannews06 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "ofi"),

str\_replace(content\_restored, "orangutan foundation international", ""),

content\_restored))

orangutannews08 <- orangutannews07 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "soc"),

str\_replace(content\_restored, "sintang orangutan center", ""),

content\_restored))

orangutannews09 <- orangutannews08 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "yp"),

str\_replace(content\_restored, "yayasan palung", ""),

content\_restored))

orangutannews10 <- orangutannews09 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "fnpf"),

str\_replace(content\_restored, "friends of national park foundation", ""),

content\_restored))

orangutannews11 <- orangutannews10 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "skw"),

str\_replace(content\_restored, "seksi konservasi wilayah", ""),

content\_restored))

orangutannews11 <- orangutannews11 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "wru"),

str\_replace(content\_restored, "wildlife rescue unit", ""),

content\_restored))

orangutannews11 <- orangutannews11 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "pt"),

str\_replace(content\_restored, "perusahaan", ""),

content\_restored))

orangutannews11 <- orangutannews11 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "wssl"),

str\_replace(content\_restored, "wana sawit subur lestari", ""),

content\_restored))

orangutannews11 <- orangutannews11 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "bga"),

str\_replace(content\_restored, "bumitama gunajaya agro", ""),

content\_restored))

orangutannews11 <- orangutannews11 %>%

mutate(content\_restored = ifelse(str\_detect(content\_restored, "orangutan"),

str\_replace(content\_restored, "pongo pygmaeus", ""),

content\_restored))

1. remove regional names

Because we want to find latent topics based on the probability of terms appearing together. terms that contain location names are often mentioned quite frequently in the news, thus becoming a distraction from building latent topics. Therefore, terms that contain regional location names were removed in this LDA modeling session. location names come from a collection of BPS data at the sub-district level throughout Kalimantan.

The script built in this session is that if a terms is found that is the same as the lexicon of regional names in Kalimantan, then remove the terms from the content.

list.files("D:/011\_git/002\_text\_mining/ID-regional-master/")

adm\_kalimantan <- read.csv2("D:/011\_git/002\_text\_mining/ID-regional-master/reg\_kalimantan.csv")

head(adm\_kalimantan)

#Simpan daftar nama kabupaten dari reg\_kalimantan sebagai vektor

nama\_kabupaten <- adm\_kalimantan$nama\_kabupaten\_big %>%

paste(., collapse = "|") #Gabungkan nama kabupaten menjadi pola regex untuk pencarian

orangutannews12 <- orangutannews11 %>%

mutate(content\_restored = str\_replace\_all(content\_restored, nama\_kabupaten, ""))

#Simpan daftar nama kecamatan dari reg\_kalimantan sebagai vektor

nama\_kecamatan <- adm\_kalimantan$nama\_kecamatan\_big %>%

paste(., collapse = "|") #Gabungkan nama kecamatan menjadi pola regex untuk pencarian

orangutannews13 <- orangutannews12 %>%

mutate(content\_restored = str\_replace\_all(content\_restored, nama\_kecamatan, ""))

#Simpan daftar nama desa dari reg\_kalimantan sebagai vektor

nama\_desa <- adm\_kalimantan$nama\_desa\_big %>%

paste(., collapse = "|") #Gabungkan nama kecamatan menjadi pola regex untuk pencarian

orangutannews14 <- orangutannews13 %>%

mutate(content\_restored = str\_replace\_all(content\_restored, nama\_desa, ""))

orangutannews15 <- orangutannews14 %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kabupaten pontianak", "kabupaten mempawah")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kotim", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kutim", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kutai", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kutai timur", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "sangatta", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "tenggarong", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "sampit", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kabupaten", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kotawaringin timur", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kotawaringin barat", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kotawaringin", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "tana paser", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kecamatan siantan", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kecamatan mentawa ketapang", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kecamatan komam", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "pangkalan bun|pangkalanbun", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "ketapang", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "muara teweh", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "sintang", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "palangka raya", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "pontianak", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "sanggau", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "singkawang", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "singapura", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kayong agro lestari", "kal")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "wwfindonesia", "wwf")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "yogyakarta", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kayong", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "mempawah", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "seponti", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "sukadana", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kapuas", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kecamatan", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "areaarea", "area")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "balai", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "konservasi sumber daya alam|skw|wru", "bksda")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "yayasan inisiasi alam rehabilitasi", "yiari")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "orangutan foundation indonesia", "ofi")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "taman nasional", "tn")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "perusahaan", "pt")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "pasal", "peraturan")) %>%

mutate(content\_restored = gsub("\\s+", " ", content\_restored)) # Ubah spasi lebih dari satu jadi satu

mutate(content\_restored = str\_replace\_all(content\_restored, "pangkalan bun|pangkalanbun", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "ketapang", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "muara teweh", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "sintang", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "palangka raya", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "pontianak", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "sanggau", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "singkawang", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "singapura", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kayong agro lestari", "kal")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "wwfindonesia", "wwf")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "yogyakarta", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kayong", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "mempawah", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "seponti", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "sukadana", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kapuas", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "kecamatan", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "areaarea", "area")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "balai", "")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "konservasi sumber daya alam|skw|wru", "bksda")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "yayasan inisiasi alam rehabilitasi", "yiari")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "orangutan foundation indonesia", "ofi")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "taman nasional", "tn")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "perusahaan", "pt")) %>%

mutate(content\_restored = str\_replace\_all(content\_restored, "pasal", "peraturan")) %>%

mutate(content\_restored = gsub("\\s+", " ", content\_restored)) # Ubah spasi lebih dari satu jadi satu

1. calcuate TF-IDF

The steps involved converting the content columns, which were still in the form of sentences or paragraphs, into word-by-word columns. After that, word frequencies were calculated. After the words were counted, the resulting format was changed to a document-terms matrix. After this, the data was ready to be processed for topic modeling.

#step tokenize unigram

unigram\_contentou01 <- orangutannews\_id %>%

mutate(line\_id = row\_number()) %>% #Buat ID baris agar urutan tetap terjaga

unnest\_tokens(word, content\_restored, token = "words", to\_lower = TRUE) %>% #Tanpa pembersihan stop\_words !!! token = "words"/1 words| token = "ngrams", n = 2,3, ., etc

group\_by(doc\_id) %>%

mutate(word\_position = row\_number()) %>% #Menyimpan posisi kata dalam dokumen

ungroup()

unique\_word\_count <- unigram\_contentou01 %>%

distinct(word) %>% # Ambil hanya kata unik

nrow() # Hitung jumlahnya

print(unique\_word\_count)

View(unigram\_contentou01)

#calculate TF-ID

ouword\_counts01 <- unigram\_contentou01 %>%

count(doc\_id, word, sort = TRUE)

# convert result TF-IDF to Document-Term Matrix (DTM) format

dtm\_unigram01 <- ouword\_counts01 %>%

cast\_dtm(doc\_id, word, n)

1. install and activation package for lda modelling process

install.packages("textmineR")

install.packages("tibble") # if tibble has not install

library(textmineR)

library(topicmodels)

library(Matrix)

library(ggplot2)

library(slam)

library(ldatuning)

library(tidytext)

library(tibble)

library(dplyr)

library(lsa)

library(tm) # for DocumentTermMatrix

library(tidyr) # for data manipulation

library(scales)

library(textmineR)

1. lda modelling process and evaluate

The stages of work are to make modelling looping with the number of topics as many as 3 to 20 topics. the settings made in this study, namely seed (1234) which is a random number that is used as a starting point for making random numbers. It is intended that the process results do not change even though they are operated on other computers. the number of iterations or repetitions (2000) in the gibbs sampling model for more stable convergence, burnin (500) to discard the number of initial results so as not to be biased, this is because the initial iteration has not yet obtained stable convergence, so 500 of 2000 is not used and thin (10) to take one sample from 10 repetitions so that the correlation is reduced between samples and improve inference results.

The results of the LDA modelling process with made 3 to 20 topics are evaluated by coherence score. Coherence score shows how reasonable and meaningful the topics are based on the co-occurrence relationship of terms probabilities in each topic

# lda modelling and calculate coherence score untuk k-n sampai k-n

# Suppose dtm\_unigram01 is an existing DocumentTermMatrix object

# Convert DocumentTermMatrix (DTM) to sparse matrix (dgCMatrix)

dim(dtm\_unigram01)

# convert DTM to sparse matrix

dtm\_unigram01\_sparse <- as.matrix(dtm\_unigram01) #Change to normal matrix first

dtm\_unigram01\_sparse <- Matrix(dtm\_unigram01\_sparse, sparse = TRUE) #convert to dgCMatrix

dim(dtm\_unigram01\_sparse)

# make sure the matrix is ​​correct

row\_sums <- rowSums(as.matrix(dtm\_unigram01\_sparse))

sum(row\_sums == 0) #Count the number of empty documents

# Dataframe to store results

coherence\_results <- tibble()

# Loop for varying number of topics

for (k in 3:20) { # for the number of topics you want to create

cat("Running LDA for k =", k, "\n")

# Menjalankan LDA dengan metode Gibbs

lda\_modelgibbs <- LDA(dtm\_unigram01\_sparse, k = k, method = "Gibbs",

control = list(seed = 1234,

iter = 2000,

burnin = 500,

thin = 10))

# \*\*1. Extract topic-terms distribution (phi)\*\*

phi <- as.matrix(lda\_modelgibbs@beta) # Taking beta from LDA model

# \*\*2. Make sure the phi column matches the DTM \*\*

colnames(phi) <- colnames(dtm\_unigram01\_sparse) # Customize column names

# \*\*3. Extract topics from the model \*\*

topics\_gibbs <- tidy(lda\_modelgibbs, matrix = "beta")

# \*\*4. Ambil 10 kata utama per topik\*\*

top\_terms <- topics\_gibbs %>%

group\_by(topic) %>%

slice\_max(beta, n = 10) %>%

ungroup()

# \*\*5. Konversi ke list untuk coherence\*\*

top\_terms\_list <- split(top\_terms$term, top\_terms$topic)

# \*\*6. Hitung coherence score\*\*

coherence\_score <- CalcProbCoherence(phi = phi, dtm = dtm\_unigram01\_sparse, M = 10)

# \*\*7. Simpan hasil coherence\*\*

coherence\_results <- bind\_rows(coherence\_results, tibble(k = k, coherence = mean(coherence\_score)))

# \*\*2. Make sure the phi column matches the DTM \*\*

colnames(phi) <- colnames(dtm\_unigram01\_sparse) # Customize column names

# \*\*3. Extract topics from the model \*\*

topics\_gibbs <- tidy(lda\_modelgibbs, matrix = "beta")

# \*\*4. Take 10 main words per topic \*\*

top\_terms <- topics\_gibbs %>%

group\_by(topic) %>%

slice\_max(beta, n = 10) %>%

ungroup()

# \*\*5. Convert to list for coherence \*\*

top\_terms\_list <- split(top\_terms$term, top\_terms$topic)

# \*\*6. Calculate the coherence score \*\*

coherence\_score <- CalcProbCoherence(phi = phi, dtm = dtm\_unigram01\_sparse, M = 10)

# \*\*7. Save coherence results\*\*

coherence\_results <- bind\_rows(coherence\_results, tibble(k = k, coherence = mean(coherence\_score)))

write.csv2(coherence\_results,"D:/011\_git/002\_text\_mining/topic\_modelling/evaluation\_topic\_results-rv08.csv")

1. data top 10 main terms visualization

The results of the management obtained a variety of topics from the number of latent topics as many as 3 to 20. After looking at the coherence score and showing that the latent topic was 11 which was considered good compared to the others, the number of latent topics as many as 11 was selected. After that, the data was visualized. The visualization that will be displayed is 10 main terms in each topic.

# Determine the maximum beta value of all data as a reference for the X-axis.

max\_beta <- max(top\_terms\_gibbs11$beta)

top\_terms\_gibbs11 %>%

# Sort terms by beta within each topic

group\_by(topic) %>%

arrange(desc(beta)) %>%

ungroup() %>%

mutate(word = reorder\_within(term, beta, topic)) %>%

ggplot(aes(x = beta, y = word)) +

geom\_col(fill = "grey60", show.legend = FALSE) + # grey colour

facet\_wrap(~ topic, scales = "free\_y") + # only y is free, x is consistent

scale\_y\_reordered() +

scale\_x\_continuous(

limits = c(0, max\_beta), #all graphs use the same x scale

breaks = seq(0, max\_beta, length.out = 5), # even intervals

labels = scales::number\_format(accuracy = 0.01)

) +

labs(

title = "",

x = "Term Probability",

y = "Term"

) +

theme\_minimal() +

theme(

strip.text = element\_text(size = 12), #font size of title topic

axis.title.x = element\_text(size = 12, color = "black", face = "bold"),

axis.title.y = element\_text(size = 12, color = "black", face = "bold"),

axis.text.x = element\_text(size = 11, angle = 0, hjust = 1),

axis.text.y = element\_text(size = 11)

)

write.csv2(top\_terms\_gibbs11,"D:/011\_git/002\_text\_mining/topic\_modelling/005-output-top\_terms\_gibbs-011.csv")

Result:

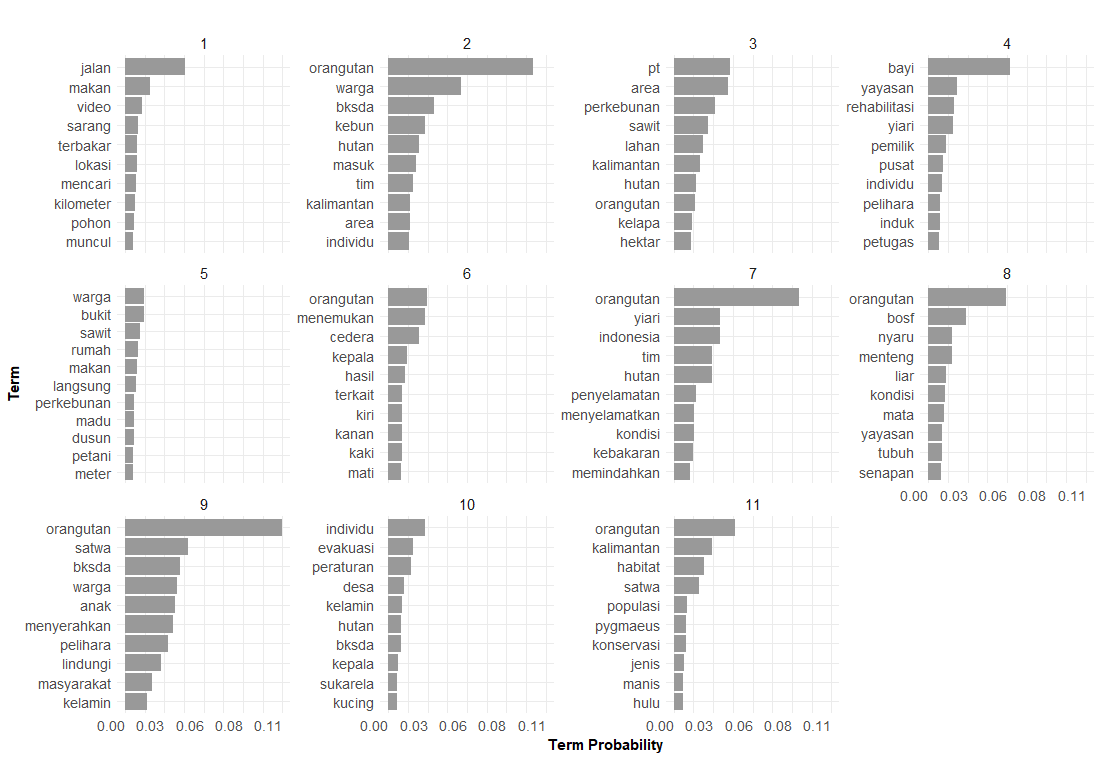


Figure 6 visualisation term probability per topic

1. Calculate similarity jaccard between terms of topics

This calculation demonstrates that the words forming each topic do not exhibit significant overlap or overfitting. When overlap exists, topic similarity is quantified using an appropriate similarity measure. This approach also ensures that each topic remains distinct and independent, minimizing potential bias.

# Get Top-N terms per topic from the topics\_gibbs11 object

top\_n <- 10

top\_terms\_jaccard <- topics\_gibbs11 %>%

group\_by(topic) %>%

slice\_max(beta, n = top\_n) %>%

summarise(terms = list(term)) %>%

arrange(topic)

# Create an empty matrix to store Jaccard

n\_topics <- nrow(top\_terms\_jaccard)

jaccard\_matrix <- matrix(0, nrow = n\_topics, ncol = n\_topics)

rownames(jaccard\_matrix) <- paste0( 1:n\_topics)

colnames(jaccard\_matrix) <- paste0( 1:n\_topics)

# calcuate Jaccard similarity

for (i in 1:n\_topics) {

for (j in 1:n\_topics) {

terms\_i <- top\_terms\_jaccard$terms[[i]]

terms\_j <- top\_terms\_jaccard$terms[[j]]

intersect\_len <- length(intersect(terms\_i, terms\_j))

union\_len <- length(union(terms\_i, terms\_j))

jaccard\_matrix[i, j] <- intersect\_len / union\_len

}

}

# View dan Simpan

View(jaccard\_matrix)

write.csv2(jaccard\_matrix, "D:/011\_git/002\_text\_mining/topic\_modelling/jaccard\_sim\_matrix-011.csv")

# Buat heatmap

pheatmap(

jaccard\_matrix,

color = colorRampPalette(c("white", "grey", "black"))(100),

main = paste("Jaccard Similarity"),

display\_numbers = FALSE,

legend = TRUE,

cluster\_rows = FALSE,

cluster\_cols = FALSE,

angle\_col = 0,

fontsize = 11,

fontsize\_row = 11,

fontsize\_col = 11,

border\_color = NA

)

# calcuate Jaccard similarity

for (i in 1:n\_topics) {

for (j in 1:n\_topics) {

terms\_i <- top\_terms\_jaccard$terms[[i]]

terms\_j <- top\_terms\_jaccard$terms[[j]]

intersect\_len <- length(intersect(terms\_i, terms\_j))

union\_len <- length(union(terms\_i, terms\_j))

jaccard\_matrix[i, j] <- intersect\_len / union\_len

}

}

# View dan Simpan

View(jaccard\_matrix)

write.csv2(jaccard\_matrix, "D:/011\_git/002\_text\_mining/topic\_modelling/jaccard\_sim\_matrix-011.csv")

# Buat heatmap

pheatmap(

jaccard\_matrix,

color = colorRampPalette(c("white", "grey", "black"))(100),

main = paste("Jaccard Similarity"),

display\_numbers = FALSE,

legend = TRUE,

cluster\_rows = FALSE,

cluster\_cols = FALSE,

angle\_col = 0,

fontsize = 11,

fontsize\_row = 11,

fontsize\_col = 11,

border\_color = NA

)

1. Jaccard similarity visualization

#create heatmap for jaccard similarity

pheatmap(

jaccard\_matrix,

color = colorRampPalette(c("white", "grey", "black"))(100),

main = paste("Jaccard Similarity"),

display\_numbers = FALSE,

legend = TRUE,

cluster\_rows = FALSE,

cluster\_cols = FALSE,

angle\_col = 0,

fontsize = 11,

fontsize\_row = 11,

fontsize\_col = 11,

border\_color = NA

)

Result:

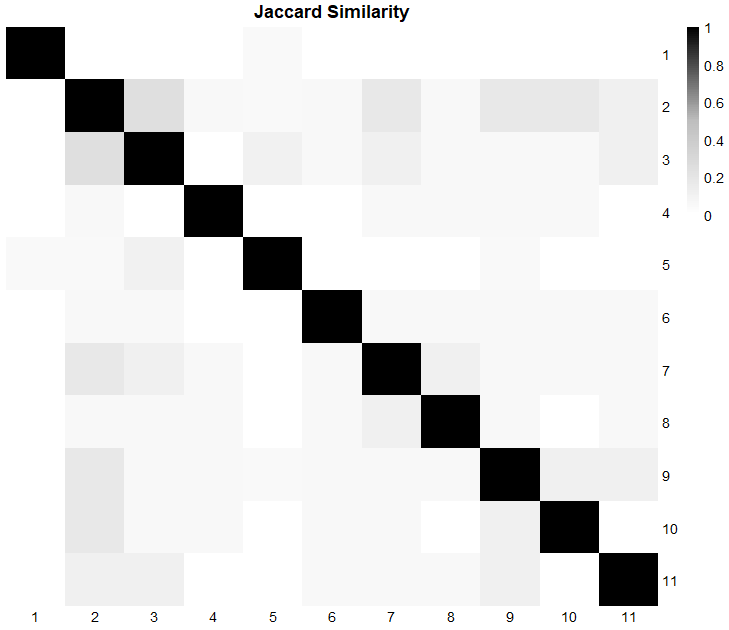


Figure 7 heatmap jaccard similarity

## SENTIMENT ANALYSIST

1. Activate package

activate the packages required to operate the functions in R for sentiment analysist

library(magrittr)

library(dplyr)

library(tidyr)

library(stringdist) #calculate similarity terms

1. Sentiment analysist

This sentiment analysis uses the Indonesian lexicon that has been created by previous research, so that the sentiment analysis is carried out by matching words in the positive and negative categories. The first stage is matching words that are similar without measurement.

topic\_model11 <- read.csv2("D:/006\_pascasarjana/008\_thesis/003\_jurnal/002\_latent\_topic-sentiment\_imo\_news/005-output-top\_terms\_topic\_modelling-gibbs-011.csv")

sentimentpost\_id <- readLines("D:/011\_git/002\_text\_mining/sentiment/fajri-sentiment-lexicon/2018-fajri-positive.tsv")

sentimentneg\_id <- readLines("D:/011\_git/002\_text\_mining/sentiment/fajri-sentiment-lexicon/2018-fajri-negative.tsv")

#Ubah file sentiment ke bentuk data frame

positive\_words <- data.frame(words = sentimentpost\_id, stringsAsFactors = FALSE)

negative\_words <- data.frame(words = sentimentneg\_id, stringsAsFactors = FALSE)

head(positive\_words)

# \*\*1️⃣ Memisahkan kata dan skor dari format tab ('\t')\*\*

# Pastikan 'words' dalam positive\_words dan negative\_words memiliki format yang benar

positive\_words <- positive\_words %>%

separate(words, into = c("word", "score"), sep = "\t", convert = TRUE)

head(positive\_words)

negative\_words <- negative\_words %>%

separate(words, into = c("word", "score"), sep = "\t", convert = TRUE)

head(negative\_words)

###identify based real word###

#### Count sentiment per topic ####

head(positive\_words)

head(negative\_words)

head(topic\_model11)

#change name of coloumn term into word on topic\_model19

topic\_model11 <- topic\_model11 %>%

rename(word = term)

#give "positive" lable if term was found in positive category

topic\_model11 <- topic\_model11 %>%

mutate(sentiment = ifelse(word %in% positive\_words$word, "positive", NA))

#give "negative" lable only if sentiment coloum still ‘NA’ and term was found in negative lexicon

topic\_model11 <- topic\_model11 %>%

mutate(sentiment = ifelse(is.na(sentiment) & word %in% negative\_words$word, "negative", sentiment))

#create coloumn score

topic\_model11 <- topic\_model11 %>%

mutate(score = NA\_real\_) # Pastikan bertipe numerik

# Lakukan join dengan positive\_words (gunakan suffix untuk mencegah bentrok nama)

topic\_model11 <- topic\_model11 %>%

left\_join(positive\_words, by = "word", suffix = c("", "\_pos")) %>%

mutate(

# Jika sentiment sudah 'positive' dan kata ada dalam positive\_words, isi score

score = ifelse(sentiment == "positive" & !is.na(score\_pos), as.numeric(score\_pos), score)

) %>%

select(-score\_pos) # Hapus kolom tambahan dari join agar lebih rapi

# Lakukan join dengan negative\_words (gunakan suffix agar tidak terjadi konflik nama)

topic\_model11 <- topic\_model11 %>%

left\_join(negative\_words, by = "word", suffix = c("", "\_neg")) %>%

mutate(

score = ifelse(sentiment == "negative" & !is.na(score\_neg), as.numeric(score\_neg), score)

) %>%

select(-score\_neg) # Hapus kolom tambahan dari join

Step 2: Match based on term distance calculate

Words in the Indonesian sentiment (inset) lexicon are calculated for similarity to those obtained from the 10 main words for each topic. Words with a distance of more than 95% (0.05) are labeled as positive or negative, as provided in the inset lexicon.

####identify word based approoached JW####

# \*\*Identification positive terms\*\*

topic\_model11 <- topic\_model11 %>%

rowwise() %>%

mutate(

match\_word\_pos = list(positive\_words$word[

stringdist(term, positive\_words$word, method = "jw") <= 0.05

]),

sentiment = ifelse(length(unlist(match\_word\_pos)) > 0, "positive", NA),

words = ifelse(length(unlist(match\_word\_pos)) > 0, paste(unlist(match\_word\_pos), collapse = ","), NA),

score = ifelse(length(unlist(match\_word\_pos)) > 0,

sum(as.numeric(positive\_words$score[positive\_words$word %in% unlist(match\_word\_pos)]), na.rm = TRUE),

NA)

) %>%

ungroup() %>%

select(-match\_word\_pos)

# \*\* identification negative terms if not have 'positive' lable\*\*

topic\_model11 <- topic\_model11 %>%

rowwise() %>%

mutate(

match\_word\_neg = list(negative\_words$word[which(stringdist(term, negative\_words$word, method = "jw") <= 0.05)]),

words = ifelse(is.na(sentiment) & length(match\_word\_neg) > 0, paste(match\_word\_neg, collapse = ","), words),

sentiment = ifelse(is.na(sentiment) & !is.na(words) & words != "", "negative", sentiment),

score = ifelse(sentiment == "negative", sum(as.numeric(negative\_words$score[negative\_words$word %in% match\_word\_neg])), score)

) %>%

ungroup() %>%

select(-match\_word\_neg) #delete help coloumn

write.csv2(topic\_model11,"D:/006\_pascasarjana/008\_thesis/003\_jurnal/002\_latent\_topic-sentiment\_imo\_news/008-output-sentiment\_terms-gibbs-011.csv")

The results of the term analysis for each topic can be seen using Microsoft Excel and can be calculated using the Excel formula, namely the sum (= sum) of terms in the positive and negative categories.