

BAB IV

IMPLEMENTASI

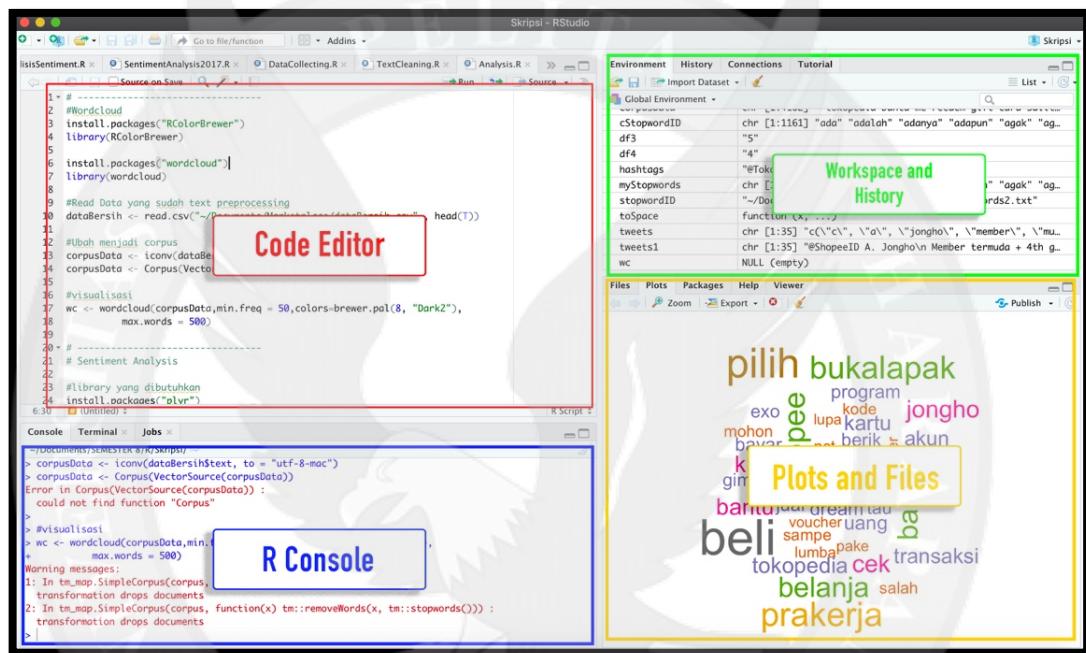
Tahap implementasi sepenuhnya menggunakan *software R studio*, maka dari itu untuk melakukan implementasi penelitian, diperlukan *software R studio*. Terdapat 4 jenis varian *R studio* yang disediakan pada situs resmi *R studio* (rstudio.com). Perbedaan antara 4 jenis *software R studio* dapat dilihat pada gambar 4.1

	RStudio Desktop Open Source License	RStudio Desktop Commercial License	RStudio Server Open Source License	RStudio Server Pro Commercial License
	Free	\$995 /year	Free	\$4,975 /year (5 Named Users)
Integrated Tools for R	✓	✓	✓	✓
Priority Support		✓		✓
Access via Web Browser			✓	✓
Enterprise Security				✓
Project Sharing				✓
Manage Multiple R Sessions & Versions				✓
Admin Dashboard				✓
Load Balancing				✓
Auditing and Monitoring				✓
Data Connectivity				✓
Launcher				✓
Tutorial API				✓
License	AGPL	Commercial	AGPL	Commercial

Gambar 4.1 Jenis-jenis *software R studio*
Sumber: rstudio.com

Berbagai varian *software R studio* yang disediakan sesuai dengan kebutuhan tujuan penggunaan, namun untuk penelitian ini menggunakan *Rstudio Desktop Open Source License* dikarenakan *software tersebut merupakan open source* serta

sudah mendukung fungsi-fungsi yang diperlukan untuk menyelesaikan penelitian ini. Langkah awal yaitu mengunduh dan instal R terlebih dahulu sesuai dengan *operating system* (OS) yang digunakan. Setelah itu, langkah berikutnya yaitu mengunduh dan instal *software* R studio sesuai dengan *operating system* (OS) yang digunakan. Secara umum, *software* R studio terdiri dari 4 bagian seperti terlihat pada gambar 4.2.



Gambar 4.2 Bagian *software* R studio

- *Code Editor*

Area ini merupakan tempat dimana kode yang menggunakan bahasa R ditulis. Untuk menjalankan kode tersebut, cukup pilih baris kode dan tekan ‘Ctrl + Enter’. Selain itu, kode juga dapat dijalankan dengan menekan tombol ‘run’ yang berlokasi di sudut kanan atas bagian *code editor*. Kode yang ditulis pada *code editor* ini dapat disimpan sebagai *R script*. Untuk membuat *R script* baru, cukup menekan tombol *file, new, R script*.

- *R Console*

Area ini menunjukkan output dari kode yang telah dijalankan pada area *code editor*. Selain itu, kode juga bisa dijalankan pada *R console*, namun kode yang dimasukkan langsung di *R console* tidak dapat dilacak nantinya.

- *Workspace and History*

Area ini menampilkan kumpulan elemen eksternal diluar R studio yang ditambahkan. Ini termasuk kumpulan data, variabel, vektor, fungsi, dan lainnya. Area ini biasanya bertujuan untuk memeriksa apakah data telah dimuat dengan benar di R.

- *Plots and Files*

Area ini biasanya menampilkan visualisasi atau grafik yang telah dibuat selama analisis data eksplorasi. Tidak hanya menampilkan visualisasi atau grafik, area ini juga dapat menjalankan *package library* ataupun mencari bantuan dengan dokumentasi resmi R yang tersemat.

Algoritma proses *text mining* dirancang untuk melakukan *clustering* data *tweet* Tokopedia, Shopee, dan Bukalapak. Berikut adalah tahapan proses implementasi *text mining* yang telah dirancang pada metodologi penelitian:

4.1 Authentication

Tahap *authentication* ini merupakan tahap integrasi antara media sosial Twitter dengan *software R studio*. Tahap *authentication* ini menggunakan API. API (*application programming interface*) memungkinkan pengguna untuk dapat mengintegrasikan dua bagian dari aplikasi atau dengan aplikasi yang berbeda dalam waktu yang bersamaan. Beberapa kode yang dibutuhkan berupa:

- API *key*
- API *secret*

- *Access token*
- *Access token secret*

Untuk mendapatkan kode-kode tersebut, terlebih dahulu harus mendaftarkan akun sebagai *developer account*. Setelah melakukan registrasi, langkah berikutnya yaitu membuat aplikasi pada Twitter *developer account*. Twitter akan meminta kita untuk menjelaskan secara lebih detail hal-hal terkait dengan aplikasi. Selanjutnya Twitter akan meninjau, apakah aplikasi disetujui atau tidak. Kode yang akan didapatkan setiap akun dan aplikasinya pun berbeda-beda. Kode digunakan untuk proses integrasi antara Twitter API dengan *software R Studio*, dimana proses integrasi dilakukan dengan menggunakan fungsi ‘`setup_twitter_oauth (api_key, api_secret, access_token, access_token_secret)`’. Berikut tahap-tahap dalam proses *authentication*:

4.1.1 Install Package Library Twitter

Package yang diinstal bernama ‘`twitteR`’.`‘twitteR’` merupakan *package* yang disediakan oleh R Studio yang menyediakan akses ke API Twitter. Proses instalasi *package* ini cukup ketikkan kode berikut pada R Studio:

```
install.packages("twitteR")
```

Gambar 4.3 Kode *install package library twitter*

4.1.2 Load Package Library Twitter

Untuk dapat menggunakan *package* yang telah diinstal pada tahap sebelumnya, agar mendapat akses ke API Twitter, ketikkan kode berikut pada R Studio:

```
library(twitteR)
```

Gambar 4.4 Kode *load package library twitter*

4.1.3 Integrasi

Tahap terakhir dari *authentication* ini adalah integrasi. Tahap ini dapat dilakukan dengan fungsi ‘*setup_twitter_oauth*’ di *software R studio*. Contohnya dapat dilihat seperti pada kode berikut ini:

```
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)
```

Gambar 4.5 Kode integrasi R studio dan twitter

4.2 Data Collecting

Pengambilan data dilakukan secara *real time* dari twitter sebanyak 1500 *tweets* dari Tokopedia, 1500 *tweets* dari Shopee dan 1500 *tweets* dari Bukalapak dengan menggunakan fungsi ‘*searchTwitter*’. Data *tweets* yang diambil pun bukan merupakan *retweet*. Data *tweets* yang diambil merupakan *tweets* berupa *mention* terhadap akun twitter utama ketiga *online marketplace* yang diteliti. Oleh karena pihak twitter hanya memungkinkan untuk mengambil *tweets* dari 7 hari belakang, maka tahap *data collection* pun dibagi menjadi 3 bagian seperti pada tabel 4.1 berikut:

Tabel 4.1 Pembagian *data collecting*

Tanggal Pengambilan	Tokopedia	Shopee	Bukalapak
30 Agustus 2020	500 Tweets	500 Tweets	500 Tweets
6 September 2020	500 Tweets	500 Tweets	500 Tweets
13 September 2020	500 Tweets	500 Tweets	500 Tweets

Penggunaan fungsi ‘*searchTwitter*’ untuk mengambil *tweets* dengan Twitter API dapat dilihat seperti pada kode berikut ini:

```
data1 <- searchTwitter(  
  "@Tokopedia -filter:retweets", n=500, lang="id"  
)  
  
data2 <- searchTwitter(  
  "@ShopeeID -filter:retweets", n=500, lang="id"  
)  
  
data3 <- searchTwitter(  
  "@Bukalapak -filter:retweets", n=500, lang="id"  
)
```

Gambar 4.6 Kode pengambilan *tweets* tahap 1

```
data4 <- searchTwitter(  
  "@Tokopedia -filter:retweets", n=500, lang="id"  
)  
  
data5 <- searchTwitter(  
  "@ShopeeID -filter:retweets", n=500, lang="id"  
)  
  
data6 <- searchTwitter(  
  "@Bukalapak -filter:retweets", n=500, lang="id"  
)
```

Gambar 4.7 Kode pengambilan *tweets* tahap 2

```
data7 <- searchTwitter(  
  "@Tokopedia -filter:retweets", n=500, lang="id"  
)  
  
data8 <- searchTwitter(  
  "@ShopeeID -filter:retweets", n=500, lang="id"  
)  
  
data9 <- searchTwitter(  
  "@Bukalapak -filter:retweets", n=500, lang="id"  
)
```

Gambar 4.8 Kode pengambilan *tweets* tahap 3

Penulisan kode “data1” hingga “data9” diatas merupakan variabel untuk menampung data *tweets* yang telah diambil. Jumlah data *tweets* yang diambil dapat disesuaikan dengan kebutuhan penelitian masing-masing. Penelitian ini mengambil 500 data *tweets* untuk 3 *online marketplace* setiap minggunya yang diambil secara rutin selama 3 minggu, oleh karena itu pada kode diatas n berjumlah 500. Data *tweets* yang diambil pun dapat disesuaikan dengan bahasa yang digunakan. Penelitian ini mengambil data *tweets* berbahasa Indonesia, dimana dapat dilihat dengan penulisan kode lang =“id” diatas. Fungsi “*-filter:retweets*” bertujuan untuk mengambil *tweets* yang bukan merupakan *retweets*.

Setelah data *tweets* berhasil ditampung pada variabel “data1” hingga “data9”, maka data-data yang telah berhasil diambil tersebut dimasukkan ke dalam file berformat CSV. Sebelum dimasukkan ke dalam file, data-data tersebut harus diringkas terlebih dahulu menggunakan fungsi ‘twListToDF’. Fungsi ‘twListToDF’ ini merupakan fungsi yang disediakan oleh *package twitteR*. Fungsi ini berfungsi untuk meringkas informasi tentang daftar tweet yang awalnya berupa *list* ke dalam sebuah *dataframe* agar data menjadi lebih rapi dan dapat disimpan dalam file format CSV. Penggunaan fungsi ini dapat dilihat seperti pada kode berikut ini:

```
tweetsdata1 <- twListToDF(data1)
tweetsdata2 <- twListToDF(data2)
tweetsdata3 <- twListToDF(data3)
tweetsdata4 <- twListToDF(data4)
tweetsdata5 <- twListToDF(data5)
tweetsdata6 <- twListToDF(data6)
tweetsdata7 <- twListToDF(data7)
tweetsdata8 <- twListToDF(data8)
tweetsdata9 <- twListToDF(data9)
```

Gambar 4.9 Kode fungsi *dataframe*

File format CSV dapat digunakan dengan hampir semua program *spreadsheet*, seperti Microsoft Excel atau Google Spreadsheets. Untuk memasukkan data-data tersebut ke dalam file format CSV dapat dilihat seperti pada kode berikut:

```
write.csv(tweetsdata1, file = '~/Documents/Marketplace/TokopediaMinggu1.csv', row.names = F)
write.csv(tweetsdata2, file = '~/Documents/Marketplace/ShopeeMinggu1.csv', row.names = F)
write.csv(tweetsdata3, file = '~/Documents/Marketplace/BukalapakMinggu1', row.names = F)
write.csv(tweetsdata4, file = '~/Documents/Marketplace/TokopediaMinggu2.csv', row.names = F)
write.csv(tweetsdata5, file = '~/Documents/Marketplace/ShopeeMinggu2.csv', row.names = F)
write.csv(tweetsdata6, file = '~/Documents/Marketplace/BukalapakMinggu2.csv', row.names = F)
write.csv(tweetsdata7, file = '~/Documents/Marketplace/TokopediaMinggu3.csv', row.names = F)
write.csv(tweetsdata8, file = '~/Documents/Marketplace/ShopeeMinggu3.csv', row.names = F)
write.csv(tweetsdata9, file = '~/Documents/Marketplace/BukalapakMinggu3.csv', row.names = F)
```

Gambar 4.10 Kode penyimpanan CSV

Fungsi ‘*write.csv*’ diatas berfungsi untuk *export dataframe* ke file format CSV di *software R Studio*. Berikutnya sesuaikan lokasi penyimpanan file sesuai dengan perangkat. Data-data *tweets* berhasil dimasukkan ke dalam file format CSV, namun terpisah. Maka dari itu, perlu digabungkan menjadi 1 file untuk memudahkan proses-proses berikutnya. Pada tahap ini, jumlah file setelah digabungkan berjumlah 4 dengan rincian sebagai berikut:

- File data gabungan, file ini berisikan seluruh file *tweets* Tokopedia, Shopee, dan Bukalapak yang digabungkan.
- File data Tokopedia, file ini berisikan seluruh file *tweets* Tokopedia yang digabungkan.
- File data Shopee, file ini berisikan seluruh file *tweets* Shopee yang digabungkan.
- File data Bukalapak, file ini berisikan seluruh file *tweets* Bukalapak yang digabungkan.

Untuk menggabungkan menjadi 1 file, file-file CSV yang ingin digabung harus dibaca terlebih dahulu oleh *software R Studio* menggunakan fungsi ‘*read*’, dapat dilihat seperti pada kode berikut:

```
dataTokped1<- read.csv("~/Documents/Marketplace/TokopediaMinggu1.csv" , head(T))
dataTokped2<- read.csv("~/Documents/Marketplace/TokopediaMinggu2.csv" , head(T))
dataTokped3<- read.csv("~/Documents/Marketplace/TokopediaMinggu3.csv" , head(T))

dataShopee1<- read.csv("~/Documents/Marketplace/ShoopeMinggu1.csv" , head(T))
dataShopee2<- read.csv("~/Documents/Marketplace/ShoopeMinggu2.csv" , head(T))
dataShopee3<- read.csv("~/Documents/Marketplace/ShoopeMinggu3.csv" , head(T))

dataBukalapak1<- read.csv("~/Documents/Marketplace/BukalapakMinggu1.csv" , head(T))
dataBukalapak2<- read.csv("~/Documents/Marketplace/BukalapakMinggu2.csv" , head(T))
dataBukalapak3<- read.csv("~/Documents/Marketplace/BukalapakMinggu3.csv" , head(T))
```

Gambar 4.11 Kode *read* file CSV

Setelah file tersebut berhasil dibaca oleh *software R Studio* menggunakan fungsi ‘*read*’, langkah selanjutnya yaitu menggabungkan 9 file tersebut dengan fungsi ‘*bind_rows*’, fungsi ‘*bind_rows*’ merupakan fungsi yang disediakan oleh *package dplyr*, oleh karena itu perlu untuk menginstal dan menjalankan *package* tersebut terlebih dahulu, dapat dilihat seperti pada kode berikut:

```
install.packages("dplyr")
library(dplyr)
```

Gambar 4.12 Kode *install package dplyr*

```
dataTokopedia <- bind_rows(dataTokped1,dataTokped2,dataTokped3)

dataShopee <- bind_rows(dataShopee1,dataShopee2,dataShopee3,)

dataBukalapak <- bind_rows(dataBukalapak1,dataBukalapak2,dataBukalapak3)

dataGabungan <- bind_rows(dataTokped1,dataTokped2,dataTokped3,
                           dataShopee1,dataShopee2,dataShopee3,
                           dataBukalapak1,dataBukalapak2,dataBukalapak3)
```

Gambar 4.13 Kode penggabungan file CSV

Langkah terakhir yaitu memasukkan data-data tersebut ke dalam file format CSV lagi dengan menggunakan fungsi ‘*write*’, dapat dilihat pada kode berikut:

```
write.csv(dataTokopedia,"~/Documents/Marketplace/DataTokopedia.csv" , all(T) )  
write.csv(dataShopee,"~/Documents/Marketplace/DataShopee.csv" , all(T) )  
write.csv(dataBukalapak,"~/Documents/Marketplace/DataBukalapak.csv" , all(T) )  
write.csv(dataGabungan,"~/Documents/Marketplace/DataGabungan.csv" , all(T) )
```

Gambar 4.14 Kode penyimpanan file CSV

4.3 Text Preprocessing

Text preprocessing merupakan tahap dimana data-data teks yang tidak terstruktur diubah menjadi teks yang terstruktur sehingga mempermudah proses analisa yang akan dilakukan selanjutnya. Tahap ini memerlukan beberapa *package* antara lain:

- *Package textclean*

Package textclean adalah koleksi *tools* untuk membuat teks menjadi terstruktur dengan baik.

- *Package katadasaR*

Package katadasaR merupakan *package* yang menyediakan fungsi untuk mengambil *stem words* untuk teks Bahasa Indonesia menggunakan algoritma Nazief dan Andriani.

- *Package tokenizers*

Package tokenizers merupakan *package* yang menyediakan fungsi untuk memisahkan kata pada kalimat.

- *Package dplyr*

Package dplyr merupakan *package* yang menyediakan fungsi untuk mengolah *dataframe*.

- *Package tm*

Package tm merupakan *framework* untuk melakukan metode *text mining* dalam R.

- *Package NLP*

Package NLP merupakan *package* yang menyediakan fungsi untuk melakukan NLP (*natural language processing*).

Untuk melakukan proses instalasi *library package* dan menjalankannya dapat dilihat seperti pada kode berikut:

```
install.packages("textclean")
install.packages("katadasaR")
install.packages("tokenizers")
install.packages("dplyr")
install.packages("tm")
install.packages("NLP")

library(textclean)
library(katadasaR)
library(tokenizers)
library(dplyr)
library(tm)
library(NLP)
```

Gambar 4.15 Kode *install package text preprocessing*

Proses *text preprocessing* ini bertujuan untuk mengubah teks pada 4 file yang telah digabungkan sebelumnya menjadi terstruktur. Proses *text preprocessing* ini terdiri dari beberapa tahap yaitu *case folding*, *stemming*, serta *stopwords removing*.

4.3.1 Case Folding

Dalam *text preprocessing*, tahap *case folding* ini merupakan tahap dimana seluruh karakter huruf di dalam dokumen dijadikan *lowercase*, serta membuang seluruh huruf karakter selain huruf a sampai z. Untuk memulai tahap ini, pertama data-data *tweets* yang telah disimpan pada file CSV harus dibaca terlebih dahulu oleh *software R Studio* menggunakan fungsi ‘*read*’, disini variabel ‘*dataTweets1*’

hingga ‘*dataTweets4*’ digunakan untuk menampung data-data *tweets*, dapat dilihat seperti pada kode berikut:

```
dataTweets1<- read.csv("~/Documents/Marketplace/DataGabungan.csv" , head(T))
dataTweets2<- read.csv("~/Documents/Marketplace/DataTokopedia.csv" , head(T))
dataTweets3<- read.csv("~/Documents/Marketplace/DataShopee.csv" , head(T))
dataTweets4<- read.csv("~/Documents/Marketplace/DataBukalapak.csv" , head(T))

dataTweets1 <- dataTweets1$text %>%
  as.character()
head(dataTweets1)
dataTweets2 <- dataTweets2$text %>%
  as.character()
head(dataTweets2)
dataTweets3 <- dataTweets3$text %>%
  as.character()
head(dataTweets3)
dataTweets4 <- dataTweets4$text %>%
  as.character()
head(dataTweets4)
```

Gambar 4.16 Kode *read file text preprocessing*

Setelah file tersebut berhasil dibaca, langkah pertama yaitu menghilangkan simbol ‘\n’. Simbol tersebut merupakan fungsi *enter* pada *keyboard*. Untuk menghilangkan simbol tersebut dapat menggunakan fungsi ‘*gsub()*’, dapat dilihat seperti pada kode berikut:

```
dataTweets1 <- gsub( "\n", " ",dataTweets1)
dataTweets2 <- gsub( "\n", " ",dataTweets2)
dataTweets3 <- gsub( "\n", " ",dataTweets3)
dataTweets4 <- gsub( "\n", " ",dataTweets4)
```

Gambar 4.17 Kode *case folding* tahap 1

Fungsi ‘*gsub()*’ pada kode di atas bertujuan untuk *replace* atau menggantikan *string*. Setelah fungsi tersebut berhasil dijalankan, langkah berikutnya yaitu menghilangkan HTML *markup* dan menghilangkan URL (*uniform resource locator*), URL merupakan istilah yang biasanya digunakan untuk menyebut situs web. Fungsi untuk menghilangkan URL dapat dilihat seperti pada kode berikut:

```

dataTweets1 <- dataTweets1 %>%
  replace_html() %>%
  replace_url()
dataTweets2 <- dataTweets2 %>%
  replace_html() %>%
  replace_url()
dataTweets3 <- dataTweets3 %>%
  replace_html() %>%
  replace_url()
dataTweets4 <- dataTweets4 %>%
  replace_html() %>%
  replace_url()

```

Gambar 4.18 Kode *case folding* tahap 2

Setelah HTML *markup* dan URL berhasil dihilangkan, langkah berikutnya yaitu menghilangkan *mention* dan *hashtag*. *Mention* biasanya digunakan oleh pengguna twitter untuk menyebut atau memanggil *username* pengguna lain, sedangkan *hashtag* biasanya digunakan oleh pengguna twitter untuk pengelompokan konten. *Hashtag* bisa digunakan untuk ditambahkan pada postingan berupa teks, foto, video, event, dan lain-lain. Fungsi untuk menghilangkan *mention* serta *hashtag* dapat dilihat seperti pada kode berikut:

```

dataTweets1 <- dataTweets1 %>%
  replace_tag(dataTweets1, pattern = "@([A-Za-z0-9_]+)", replacement="")
  replace_hash(dataTweets1, pattern = "#([A-Za-z0-9_]+)", replacement="")
dataTweets2 <- dataTweets2 %>%
  replace_tag(dataTweets2, pattern = "@([A-Za-z0-9_]+)", replacement="")
  replace_hash(dataTweets2, pattern = "#([A-Za-z0-9_]+)", replacement="")
dataTweets3 <- dataTweets3 %>%
  replace_tag(dataTweets3, pattern = "@([A-Za-z0-9_]+)", replacement="")
  replace_hash(dataTweets3, pattern = "#([A-Za-z0-9_]+)", replacement="")
dataTweets4 <- dataTweets4 %>%
  replace_tag(dataTweets4, pattern = "@([A-Za-z0-9_]+)", replacement="")
  replace_hash(dataTweets4, pattern = "#([A-Za-z0-9_]+)", replacement="")

```

Gambar 4.19 Kode *case folding* tahap 3

Setelah *mention* serta *hashtag* berhasil dihilangkan, langkah berikutnya yaitu mengubah kata-kata yang disingkat menjadi kata yang tidak disingkat. Pertama, *lexicon* bahasa Indonesia harus dibaca oleh *software R Studio* terlebih

dahulu menggunakan fungsi ‘*read*’, daftar *lexicon* bahasa Indonesia yang digunakan adalah *Colloquial Indonesian Lexicon* [24]. Dilanjutkan dengan menggunakan fungsi ‘*replace_Internet_slang*’, dapat dilihat seperti pada kode berikut:

```
spell.lex <- read.csv("~/Documents/Marketplace/indonesia-lexicon.csv")

dataTweets1 <- replace_internet_slang(dataTweets1, slang = paste0("\b",
                                                               spell.lex$slang, "\b"),
                                         replacement = spell.lex$formal, ignore.case = TRUE)
dataTweets2 <- replace_internet_slang(dataTweets2, slang = paste0("\b",
                                                               spell.lex$slang, "\b"),
                                         replacement = spell.lex$formal, ignore.case = TRUE)
dataTweets3 <- replace_internet_slang(dataTweets3, slang = paste0("\b",
                                                               spell.lex$slang, "\b"),
                                         replacement = spell.lex$formal, ignore.case = TRUE)
dataTweets4 <- replace_internet_slang(dataTweets4, slang = paste0("\b",
                                                               spell.lex$slang, "\b"),
                                         replacement = spell.lex$formal, ignore.case = TRUE)
```

Gambar 4.20 Kode *case folding* tahap 4

Penulisan kode ‘*spell.lex*’ diatas merupakan variabel penampung untuk *lexicon* bahasa Indonesia yang dibaca oleh *software R Studio*. Langkah berikutnya yaitu menghilangkan tanda baca menggunakan fungsi ‘*strip*’, dapat dilihat seperti pada kode berikut:

```
dataTweets1 <- strip(dataTweets1)
dataTweets2 <- strip(dataTweets2)
dataTweets3 <- strip(dataTweets3)
dataTweets4 <- strip(dataTweets4)
```

Gambar 4.21 Kode *case folding* tahap 5

4.3.2 *Stemming*

Tahap *stemming* merupakan tahap yang bertujuan untuk mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*). Tahap *stemming* ini menggunakan *package katadasaR* [25]. Berikut kode yang digunakan untuk melakukan tahap *stemming* pada *software R Studio*:

```

stemming <- function(x){
  paste(lapply(x,katadasar),collapse = " ")}

dataTweets1 <- lapply(tokenize_words(dataTweets1[]), stemming)
dataTweets2 <- lapply(tokenize_words(dataTweets2[]), stemming)
dataTweets3 <- lapply(tokenize_words(dataTweets3[]), stemming)
dataTweets4 <- lapply(tokenize_words(dataTweets4[]), stemming)

```

Gambar 4.22 Kode *stemming*

4.3.3 Stopwords Removing

Tahap *stopwords removing* merupakan tahap pengambilan kata – kata yang dianggap penting saja. Kata-kata yang berada di dalam daftar *stoplist* akan dihilangkan, sehingga kata-kata dalam dokumen hanya berisi kata-kata yang dianggap penting saja. Langkah pertama yaitu membaca daftar *stoplist* dengan menggunakan fungsi ‘*readLines*’, daftar *stoplist* yang digunakan pada penelitian ini diambil dari jurnal Tala [26], dimana berjumlah 758 *stopwords* dan dikombinasikan dengan *stopwords* lainnya yang diambil pada situs github [27] sehingga berjumlah 833 *stopwords*. Selanjutnya *dataframe* diubah menjadi *corpus*, lalu kata-kata yang berada dalam daftar *stoplist* akan dihilangkan menggunakan fungsi ‘*removeWords*’, dapat dilihat seperti pada kode berikut:

```

myStopwords <- readLines("~/Documents/Marketplace/stopwords.txt")

corpusData <- Corpus(VectorSource(dataTweets1))
corpusData2 <- Corpus(VectorSource(dataTweets2))
corpusData3 <- Corpus(VectorSource(dataTweets3))
corpusData4 <- Corpus(VectorSource(dataTweets4))

dataBersih <- tm_map(corpusData, removeWords, myStopwords)
dataBersih2 <- tm_map(corpusData2, removeWords, myStopwords)
dataBersih3 <- tm_map(corpusData3, removeWords, myStopwords)
dataBersih4 <- tm_map(corpusData4, removeWords, myStopwords)

```

Gambar 4.23 Kode *stopwords removing* tahap 1

Setelah *stopwords* berhasil dihilangkan, tahap *text preprocessing* sudah selesai, namun teks yang sudah terstruktur akan disimpan di dalam file format CSV. Langkah pertama yaitu mengubah data yang berupa *corpus* menjadi *dataframe* dan

memasukkannya ke dalam file dengan menggunakan fungsi ‘*write*’, dapat dilihat seperti pada kode berikut:

```
teksBersih <- data.frame(text = get("content", dataBersih))
teksBersih2 <- data.frame(text = get("content", dataBersih2))
teksBersih3 <- data.frame(text = get("content", dataBersih3))
teksBersih4 <- data.frame(text = get("content", dataBersih4))

write.csv(teksBersih, file = '~/Documents/Marketplace/dataBersih.csv', row.names = F)
write.csv(teksBersih2, file = '~/Documents/Marketplace/dataBersihTokopedia.csv', row.names = F)
write.csv(teksBersih3, file = '~/Documents/Marketplace/dataBersihShopee.csv', row.names = F)
write.csv(teksBersih4, file = '~/Documents/Marketplace/dataBersihBukalapak.csv', row.names = F)
```

Gambar 4.24 Kode *stopwords removing* tahap 2

4.4 Word Cloud Creating

Word cloud merupakan visualisasi teks berdasarkan banyaknya jumlah kata dalam dokumen. Semakin banyak kata dalam dokumen muncul, maka semakin besar juga visualisasi teks yang dimunculkan pada *word cloud*. Langkah pertama yaitu membuka data *tweets* yang sudah terstruktur atau sudah melewati tahap *text preprocessing* menggunakan fungsi ‘*read*’, dapat dilihat seperti pada kode berikut:

```
dataBersih <- read.csv("~/Documents/Marketplace/dataBersih.csv" , head(T))
dataBersih2 <- read.csv("~/Documents/Marketplace/dataBersihTokopedia.csv" , head(T))
dataBersih3 <- read.csv("~/Documents/Marketplace/dataBersihShopee.csv" , head(T))
dataBersih4 <- read.csv("~/Documents/Marketplace/dataBersihBukalapak.csv" , head(T))
```

Gambar 4.25 Kode *word cloud creating* tahap 1

Pada kode diatas, penulisan kode “dataBersih” hingga “dataBersih4” hanya menjadi variabel penampung untuk menampung data-data *tweets*. Langkah berikutnya yaitu mengubah *dataframe* “dataBersih” menjadi *corpus*, dapat dilihat seperti pada kode berikut:

```
corpusData <- iconv(dataBersih$text, to = "utf-8-mac")
corpusData <- Corpus(VectorSource(corpusData))

corpusData2 <- iconv(dataBersih2$text, to = "utf-8-mac")
corpusData2 <- Corpus(VectorSource(corpusData2))

corpusData3 <- iconv(dataBersih3$text, to = "utf-8-mac")
corpusData3 <- Corpus(VectorSource(corpusData3))

corpusData4 <- iconv(dataBersih4$text, to = "utf-8-mac")
corpusData4 <- Corpus(VectorSource(corpusData4))
```

Gambar 4.26 Kode *word cloud creating* tahap 2

Pada kode diatas, penulisan kode “corpusData” hingga “corpusData4” menjadi variabel penampung untuk menampung data *corpus*. Langkah berikutnya yaitu melakukan instalasi dan menjalankan *library package*, dapat dilihat seperti pada kode berikut:

```
install.packages("RColorBrewer")
library(RColorBrewer)

install.packages("wordcloud")
library(wordcloud)

install.packages("tm")
library(tm)
```

Gambar 4.27 Kode *word cloud creating* tahap 3

Package wordcloud merupakan *package* yang disediakan oleh R yang biasanya digunakan untuk merepresentasikan data teks secara visual. Sedangkan *package RcolorBrewer* merupakan *package* yang berisi palet warna yang siap digunakan untuk membuat grafik yang indah. Setelah itu, *word cloud* dapat dibentuk secara otomatis menggunakan fungsi ‘*wordcloud*’, dapat dilihat seperti pada kode berikut:

```
wc <- wordcloud(corpusData,min.freq = 50,colors=brewer.pal(8, "Dark2"),
                 max.words = 500)
wc2 <- wordcloud(corpusData2,min.freq = 20,colors=brewer.pal(8, "Dark2"),
                  max.words = 500)
wc3 <- wordcloud(corpusData3,min.freq = 20,colors=brewer.pal(8, "Dark2"),
                  max.words = 500)
wc4 <- wordcloud(corpusData4,min.freq = 20,colors=brewer.pal(8, "Dark2"),
                  max.words = 500)
```

Gambar 4.28 Kode *word cloud creating* tahap 4

Penulisan kode “min.freq=50” untuk membatasi kata dengan frekuensi dibawah 50 tidak akan divisualisasi pada *word cloud*. Khusus untuk data *tweets* data gabungan antara Tokopedia, Shopee, dan Bukalapak menggunakan 50 sebagai frekuensi minimal dikarenakan jumlah data *tweets* cukup banyak yaitu sebanyak 4500, sedangkan untuk data *tweets* Tokopedia, Shopee, dan Bukalapak menggunakan 20 sebagai frekuensi minimal. Penulisan kode “colors” berfungsi untuk menentukan warna yang akan digunakan pada *word cloud*, terdapat banyak varian warna yang dapat digunakan untuk visualisasi *word cloud*. Disini, warna yang digunakan diambil dari *package RcolorBrewer* yang disediakan oleh R studio. Penulisan kode “max.words=500” merupakan fungsi untuk membatasi jumlah maksimal kata yang muncul dan divisualisasikan pada *word cloud*.



Gambar 4.29 Hasil *word cloud* gabungan



Gambar 4.30 Hasil *word cloud* Tokopedia



Gambar 4.31 Hasil *word cloud* Shopee



Gambar 4.32 Hasil *word cloud* Bukalapak

Tabel 4.2 Rangkuman hasil *word cloud*

Metode	<i>Marketplace</i>		
	Tokopedia	Shopee	Bukalapak
<i>Word cloud Creating</i>	Hasil <i>word cloud creating online marketplace</i> Tokopedia menunjukkan bahwa konsumen <i>online marketplace</i> tersebut banyak membahas mengenai transaksi, dilihat dari kata “beli” dan “belanja” yang mempunyai ukuran terbesar.	Hasil <i>word cloud creating online marketplace</i> Shopee menunjukkan bahwa konsumen <i>online marketplace</i> tersebut banyak membahas mengenai <i>brand ambassador</i> yang menjadi <i>image</i> dari <i>online marketplace</i> tersebut, dilihat dari kata “pilih” dan “jongho” yang mempunyai ukuran terbesar.	Hasil <i>word cloud creating online marketplace</i> Bukalapak menunjukkan bahwa konsumen <i>online marketplace</i> tersebut banyak membahas mengenai transaksi kursus pelatihan keterampilan sebagai bentuk kerja sama dari pemerintah melalui program kartu pra kerja, dilihat dari kata “beli”, “kartu”, “latih” dan “prakerja” yang mempunyai ukuran terbesar.

4.5 Lexicon

Lexicon merupakan metode yang digunakan untuk menilai sentimen atau opini masyarakat mengenai suatu topik tertentu. Konsep yang digunakan pada metode *lexicon* penelitian ini yaitu memberikan nilai pada setiap *tweets* berdasarkan berapa banyak *tweets* tersebut mengandung kata positif atau negatif menggunakan list positif negatif yang telah dibuat sebelumnya pada *Liu's opinion words list* dengan modifikasi dan terjemahan ke bahasa Indonesia [28][29]. Langkah pertama yaitu melakukan instalasi serta menjalankan *package library* yang dibutuhkan untuk melakukan tahap *lexicon*, dapat dilihat seperti pada kode berikut:

```
install.packages("plyr")
install.packages("stringr")

library(plyr)
library(stringr)
```

Gambar 4.33 Kode *lexicon* tahap 1

Package plyr merupakan *package* yang menyediakan fungsi untuk memisahkan, menerapkan serta menggabungkan data. Sedangkan *package stringr*

merupakan *package* yang menyediakan sekumpulan fungsi kohesif yang dirancang agar pengguna dapat bekerja dengan *string* semudah mungkin. Langkah selanjutnya yaitu membuka data *tweets* yang sudah terstruktur atau sudah melewati tahap *text preprocessing* menggunakan fungsi ‘*read*’, dapat dilihat seperti pada kode berikut:

```
dataBersih <- read.csv("~/Documents/Marketplace/dataBersih.csv" , head(T))
dataBersih2 <- read.csv("~/Documents/Marketplace/dataBersihTokopedia.csv" , head(T))
dataBersih3 <- read.csv("~/Documents/Marketplace/dataBersihShopee.csv" , head(T))
dataBersih4 <- read.csv("~/Documents/Marketplace/dataBersihBukalapak.csv" , head(T))

dataBersih <- dataBersih$text %>%
  as.character()
dataBersih2 <- dataBersih2$text %>%
  as.character()
dataBersih3 <- dataBersih3$text %>%
  as.character()
dataBersih4 <- dataBersih4$text %>%
  as.character()
```

Gambar 4.34 Kode *lexicon* tahap 2

Setelah data *tweets* berhasil dibuka, langkah selanjutnya yaitu membuka file *lexicon* positif dan negatif bahasa Indonesia pada *Liu's opinion words list* dengan modifikasi dan terjemahan ke bahasa Indonesia [28][29] menggunakan fungsi ‘*scan*’, dapat dilihat seperti pada kode berikut:

```
lexiconPositif = scan("~/Documents/Marketplace/positive.txt",
                      what = "character", comment.char = ";")
lexiconNegatif = scan("~/Documents/Marketplace/negative.txt",
                      what = "character", comment.char = ";")
```

Gambar 4.35 Kode *lexicon* tahap 3

Pada kode diatas, penulisan kode “lexiconPositif” menjadi variabel untuk menampung *lexicon* dengan sentimen positif, sedangkan kode “lexiconNegatif” menjadi variabel untuk menampung *lexicon* dengan sentimen negatif. Langkah selanjutnya yaitu membuat fungsi untuk melakukan penilaian sentimen, seperti pada kode berikut:

```

nilaiSentimen = function(dataTweet, pos.words, neg.words, .progress = "none")
{
  require(plyr)
  require(stringr)

  scores = laply(dataTweet, function(tweet, pos.words, neg.words) {

    #pisahkan setiap kalimat menggunakan spasi (space delimiter):
    words = unlist(str_split(tweet, "\\s+"))

    #lakukan boolean match dari setiap kata-kata menggunakan pos & neg opinion-lexicon:
    pos.matches = !is.na(match(words, pos.words))
    neg.matches = !is.na(match(words, neg.words))

    #score sentimen = total positive sentiment - total negative:
    score = sum(pos.matches) - sum(neg.matches)

    return(score)
  }, pos.words, neg.words, .progress=.progress)

  #return data frame berisi kalimat beserta sentimennya:
  return(data.frame(text = dataTweet, score = scores))
}

```

Gambar 4.36 Kode *lexicon* tahap 4

Fungsi diatas merupakan modifikasi dari fungsi untuk mendapatkan nilai sentimen pada penelitian serupa berjudul Sentimen Analisis Twitter terhadap Penyelenggaraan Gojek Traveloka Liga 1 Indonesia [30]. Setelah fungsi tersebut dijalankan, langkah selanjutnya yaitu menerapkan fungsi tersebut pada data *tweets*, dapat dilihat seperti pada kode berikut:

```

hasilSentimen = nilaiSentimen(dataBersih, lexiconPositif, lexiconNegatif)
hasilSentimen2 = nilaiSentimen(dataBersih2, lexiconPositif, lexiconNegatif)
hasilSentimen3 = nilaiSentimen(dataBersih3, lexiconPositif, lexiconNegatif)
hasilSentimen4 = nilaiSentimen(dataBersih4, lexiconPositif, lexiconNegatif)

```

Gambar 4.37 Kode *lexicon* tahap 5

Sekarang setiap *tweets* memiliki nilai sentimennya tersendiri, untuk melihat *mean* dari nilai sentimen, dapat dilihat seperti pada kode berikut:

```

mean(hasilSentimen$score)
mean(hasilSentimen2$score)
mean(hasilSentimen3$score)
mean(hasilSentimen4$score)

```

Gambar 4.38 Kode *lexicon* tahap 6

Mean merupakan nilai rata-rata dari sebuah data yang dimiliki. Hasil *mean* yang muncul pada bagian *R console* dari keempat data *tweets* yang dianalisa dapat dilihat pada tabel 4.3.

Tabel 4.3 Nilai mean *lexicon*

Data Tweets	Nilai Mean
Gabungan	-0.121493
Tokopedia	-0.124025
Shopee	-0.08264463
Bukalapak	-0.1296421

Langkah selanjutnya adalah menyimpan data nilai sentimen ke dalam format csv menggunakan fungsi ‘*write*’, dapat dilihat seperti pada kode berikut:

```
write.csv(hasilSentimen, file = "~/Documents/Marketplace/Sentiment.csv")
write.csv(hasilSentimen2, file = "~/Documents/Marketplace/SentimentTokopedia.csv")
write.csv(hasilSentimen3, file = "~/Documents/Marketplace/SentimentShopee.csv")
write.csv(hasilSentimen4, file = "~/Documents/Marketplace/SentimentBukalapak.csv")
```

Gambar 4.39 Kode *lexicon* tahap 7

4.6 Term Weighting

Tahap *term weighting* merupakan tahap dimana pengolahan serta analisis dari data-data *tweets* yang telah melewati tahap *text preprocessing* dilakukan. *Term weighting* ini menggunakan metode *Term Frequency – Inverse Document Frequency* (TF-IDF). Dengan menggunakan metode TF-IDF, dapat diketahui kata (*terms*) apa saja yang dianggap paling penting oleh masyarakat di Internet. Langkah pertama yaitu melakukan instalasi serta menjalankan *package library* yang dibutuhkan untuk melakukan *term weighting*, dapat dilihat seperti pada kode berikut:

```
install.packages("ggplot2")
library(ggplot2)
```

Gambar 4.40 Kode *term weighting* tahap 1

Package library ggplot2 merupakan *package* visualisasi data untuk R dan cukup umum digunakan. Langkah selanjutnya yaitu membuka data *tweets* yang sudah terstruktur atau sudah melewati tahap *text preprocessing* menggunakan fungsi ‘*read*’ dan ubah menjadi *corpus*, dapat dilihat seperti pada kode berikut:

```
dataBersih <- read.csv("~/Documents/Marketplace/dataBersih.csv" , head(T))
dataBersih2 <- read.csv("~/Documents/Marketplace/dataBersihTokopedia.csv" , head(T))
dataBersih3 <- read.csv("~/Documents/Marketplace/dataBersihShopee.csv" , head(T))
dataBersih4 <- read.csv("~/Documents/Marketplace/dataBersihBukalapak.csv" , head(T))

#Ubah menjadi corpus
corpusData <- iconv(dataBersih$text, to = "utf-8-mac")
corpusData <- Corpus(VectorSource(corpusData))

corpusData2 <- iconv(dataBersih2$text, to = "utf-8-mac")
corpusData2 <- Corpus(VectorSource(corpusData2))

corpusData3 <- iconv(dataBersih3$text, to = "utf-8-mac")
corpusData3 <- Corpus(VectorSource(corpusData3))

corpusData4 <- iconv(dataBersih4$text, to = "utf-8-mac")
corpusData4 <- Corpus(VectorSource(corpusData4))
```

Gambar 4.41 Kode *term weighting* tahap 2

Setelah data *tweets* berhasil dibuka dan diubah menjadi *corpus*, langkah selanjutnya yaitu menjalankan fungsi untuk mencari tahu istilah apa yang paling sering muncul menggunakan rumus *term frequency* (TF), dapat dilihat seperti pada kode berikut:

```
tf = TermDocumentMatrix(corpusData,
                        control = list(weighting = weightTf))
freqtf=rowSums(as.matrix(tf))

tf2 = TermDocumentMatrix(corpusData2,
                        control = list(weighting = weightTf))
freqtf2=rowSums(as.matrix(tf2))

tf3 = TermDocumentMatrix(corpusData3,
                        control = list(weighting = weightTf))
freqtf3=rowSums(as.matrix(tf3))

tf4 = TermDocumentMatrix(corpusData4,
                        control = list(weighting = weightTf))
freqtf4=rowSums(as.matrix(tf4))
```

Gambar 4.42 Kode *term weighting* tahap 3

Setelah fungsi TF tersebut berhasil dilakukan, langkah selanjutnya hanya melakukan visualisasi untuk melihat apa saja 10 kata atau istilah yang paling sering muncul menggunakan *barplot*, dapat dilihat seperti pada kode berikut:

```
VTF <- tail(sort(freqtf),n=10)
barplot(VTF,ylab="Frequency",main="Term Frequency",las=2)

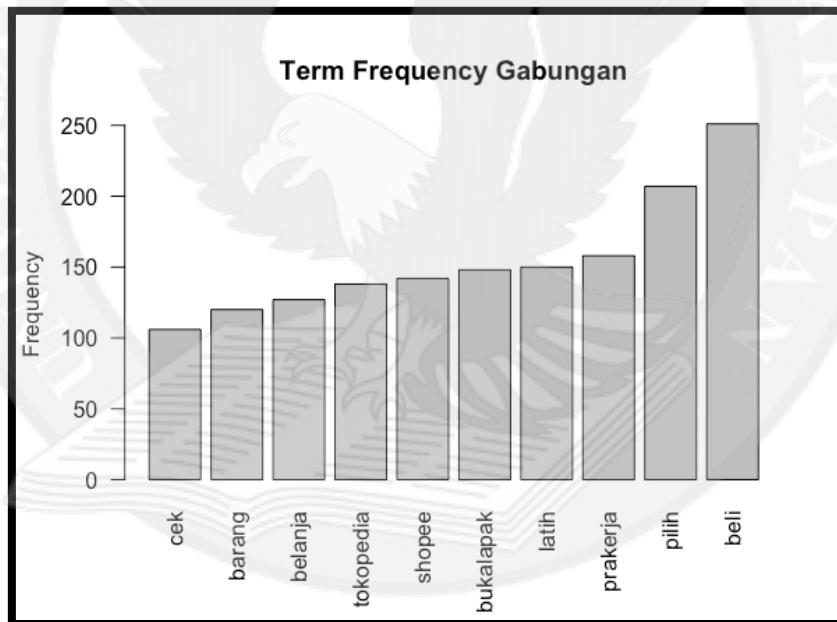
VTF2 <- tail(sort(freqtf2),n=10)
barplot(VTF2,ylab="Frequency",main="Term Frequency",las=2)

VTF3 <- tail(sort(freqtf3),n=10)
barplot(VTF3,ylab="Frequency",main="Term Frequency",las=2)

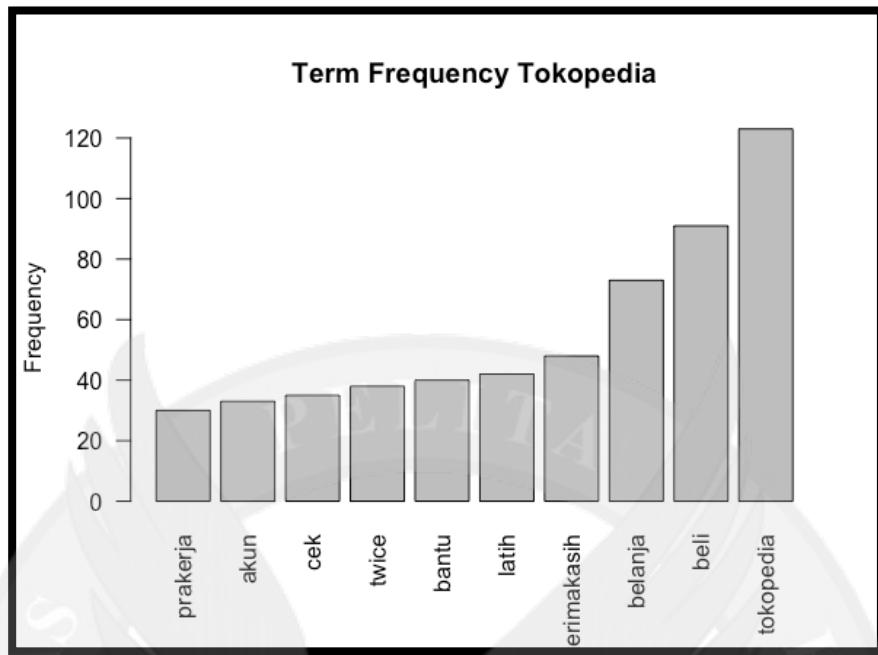
VTF4 <- tail(sort(freqtf4),n=10)
barplot(VTF4,ylab="Frequency",main="Term Frequency",las=2)
```

Gambar 4.43 Kode *term weighting* tahap 4

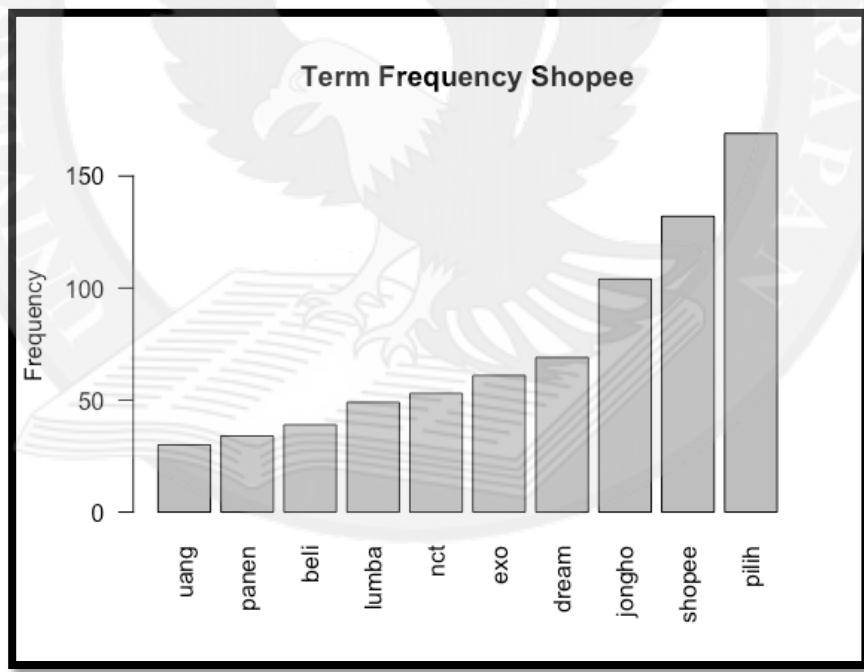
Visualisasi untuk *term frequency* (TF) dapat dilihat pada gambar dibawah.



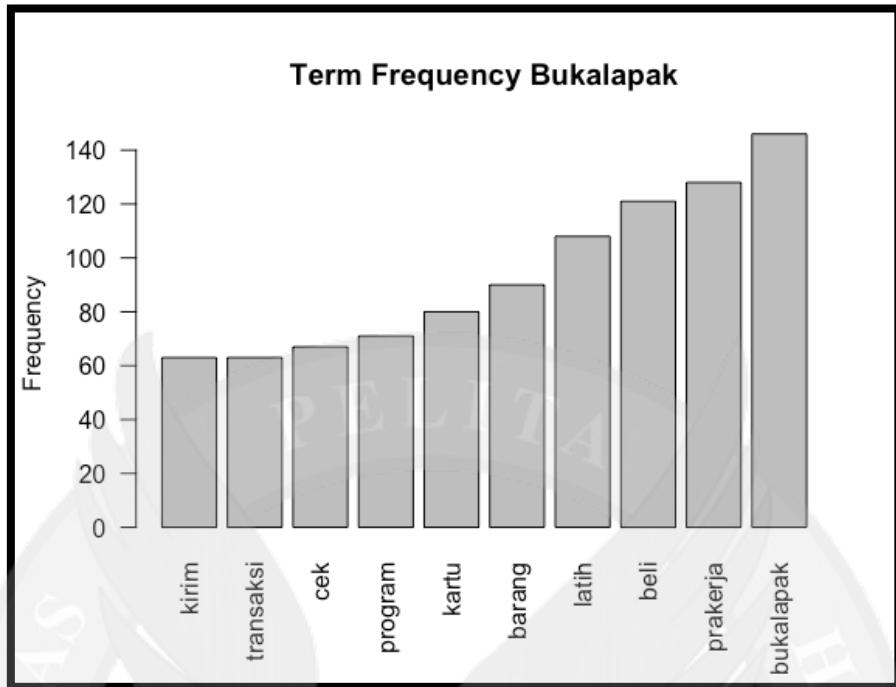
Gambar 4.44 Hasil *term frequency* gabungan



Gambar 4.45 Hasil *term frequency* Tokopedia



Gambar 4.46 Hasil *term frequency* Shopee



Gambar 4.47 Hasil *term frequency* Bukalapak

Dapat dilihat pada visualisasi untuk *term frequency* (TF) selaras seperti pada *word cloud* yang telah dibuat sebelumnya. Langkah selanjutnya yaitu menjalankan fungsi menggunakan rumus TF-IDF untuk tahap pembobotan kata (*term weighting*), dapat dilihat seperti pada kode berikut:

```

tdm = TermDocumentMatrix(corpusData,
                         control = list(weighting = weightTfIdf))
freq=rowSums(as.matrix(tdm))

tdm2 = TermDocumentMatrix(corpusData2,
                         control = list(weighting = weightTfIdf))
freq2=rowSums(as.matrix(tdm2))

tdm3 = TermDocumentMatrix(corpusData3,
                         control = list(weighting = weightTfIdf))
freq3=rowSums(as.matrix(tdm3))

tdm4 = TermDocumentMatrix(corpusData4,
                         control = list(weighting = weightTfIdf))
freq4=rowSums(as.matrix(tdm4))

```

Gambar 4.48 Kode *term weighting* tahap 5

Setelah fungsi TF-IDF tersebut berhasil dijalankan, langkah selanjutnya yaitu melakukan visualisasi untuk 10 kata yang bobotnya mempunyai nilai tertinggi TF-IDF menggunakan *barplot*, dapat dilihat seperti pada kode berikut:

```
VTFIDF <- tail(sort(freq),n=10)
barplot(VTFIDF,ylab="Frequency",main="TF - IDF Gabungan",las=2)

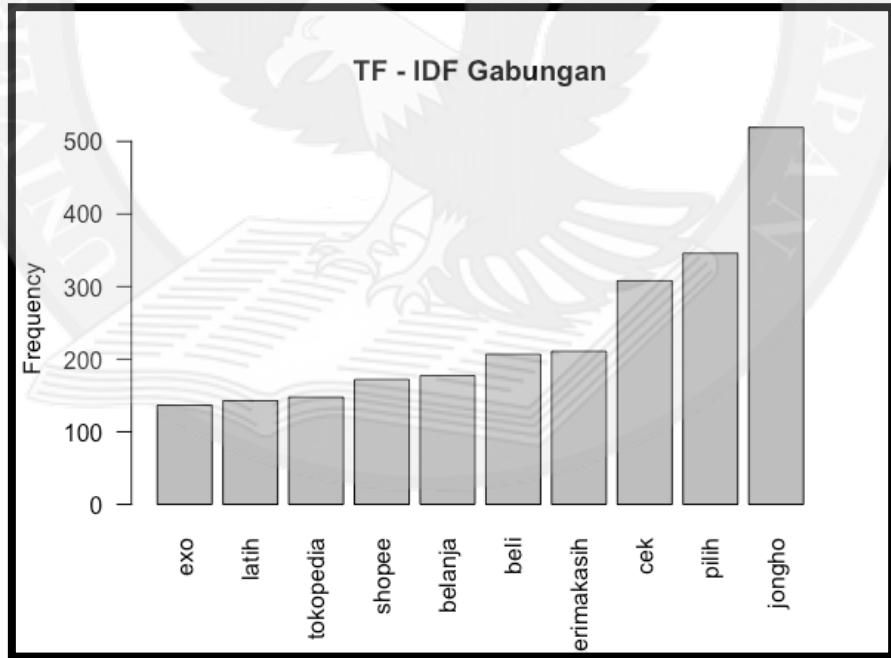
VTFIDF2 <- tail(sort(freq2),n=10)
barplot(VTFIDF2,ylab="Frequency",main="TF - IDF Tokopedia",las=2)

VTFIDF3 <- tail(sort(freq3),n=10)
barplot(VTFIDF3,ylab="Frequency",main="TF - IDF Shopee",las=2)

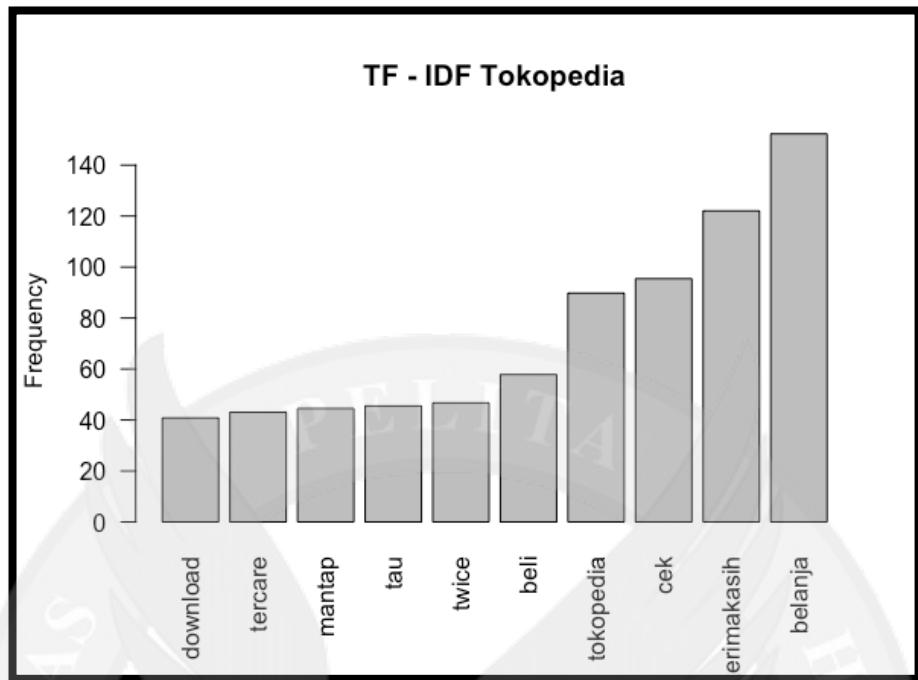
VTFIDF4 <- tail(sort(freq4),n=10)
barplot(VTFIDF4,ylab="Frequency",main="TF - IDF Bukalapak",las=2)
```

Gambar 4.49 Kode *term weighting* tahap 6

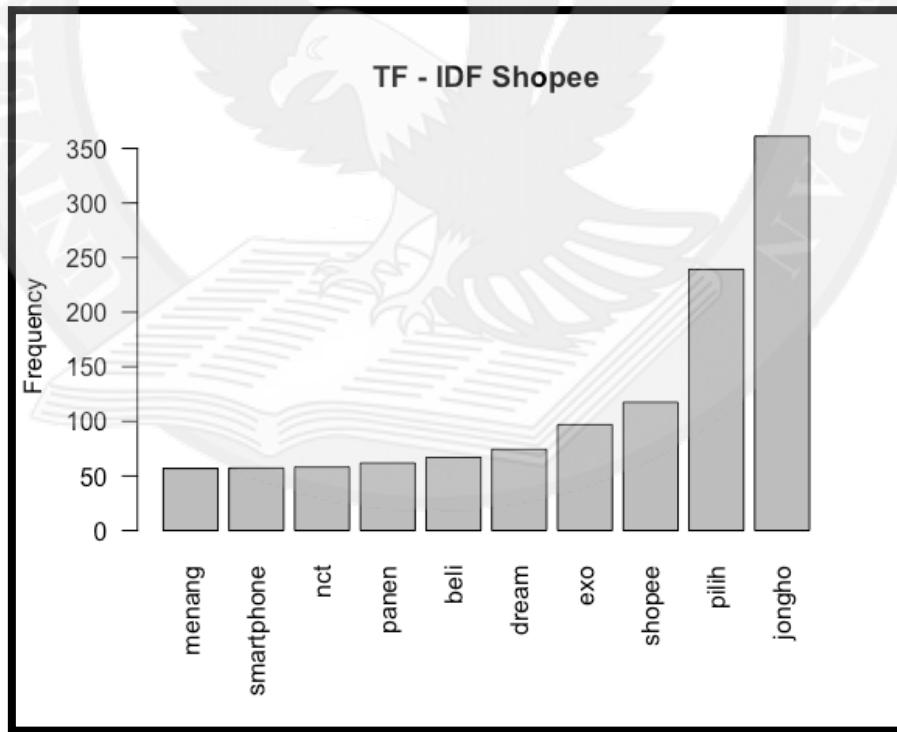
Visualisasi untuk TF-IDF dapat dilihat pada gambar dibawah.



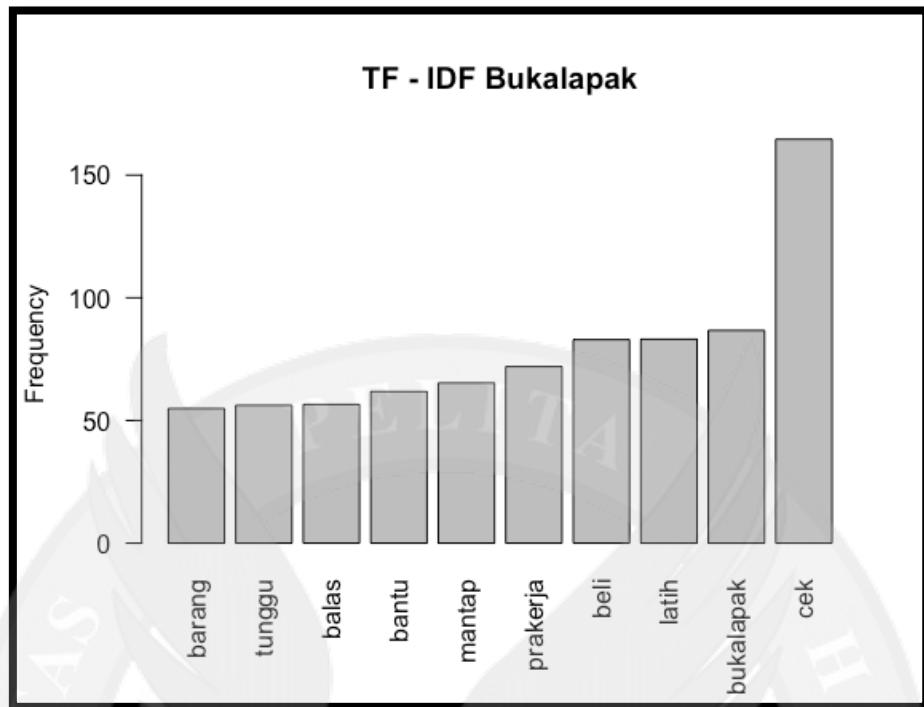
Gambar 4.50 Hasil TF-IDF gabungan



Gambar 4.51 Hasil TF-IDF Tokopedia



Gambar 4.52 Hasil TF-IDF Shopee



Gambar 4.53 Hasil TF-IDF Bukalapak

Tabel 4.4 Rincian hasil *term weighting*

Metode	<i>Marketplace</i>		
	Tokopedia	Shopee	Bukalapak
<i>Term Weighting</i>	Hasil <i>term weighting</i> online marketplace Tokopedia menunjukkan bahwa konsumen online marketplace tersebut banyak membahas mengenai transaksi, dilihat dari kata “belanja”, “terimakasih”, “cek”, “Tokopedia” dan “beli” yang menjadi kata dengan frekuensi terbesar.	Hasil <i>term weighting</i> online marketplace Tokopedia menunjukkan bahwa konsumen online marketplace tersebut banyak membahas mengenai transaksi, dilihat dari kata “jongho”, “pilih”, “Shopee”, “exo” dan “dream” yang menjadi kata dengan frekuensi terbesar.	Hasil <i>term weighting</i> online marketplace Bukalapak menunjukkan bahwa konsumen online marketplace tersebut banyak membahas mengenai transaksi, dilihat dari kata “cek”, “Bukalapak”, “latih”, “beli” dan “prakerja” yang menjadi kata dengan frekuensi terbesar.

4.7 Clustering

Tahap *clustering* merupakan tahap pengelompokan secara otomatis. Tahap *clustering* ini menggunakan metode algoritma *k-means clustering* dengan alasan *k-means clustering* ini dapat diaplikasikan dengan cukup mudah serta memiliki hasil yang baik. Langkah pertama yaitu melakukan instalasi serta menjalankan *package*

library yang dibutuhkan untuk melakukan tahap *clustering* ini, dapat dilihat seperti pada kode berikut:

```
install.packages("cluster")
library(cluster)
```

Gambar 4.54 Kode *clustering* tahap 1

Package cluster merupakan *package* yang menyediakan fungsi untuk melakukan metode *clustering analysis*. Setelah *package* berhasil diinstal dan dijalankan, langkah selanjutnya yaitu melakukan *clustering* dengan *k-means* menggunakan fungsi ‘*dist*’ dan juga ‘*kmeans*’ dan diakhiri dengan melakukan visualisasi dapat dilihat seperti pada kode berikut:

```
clus <- dist(VTFIDF, method="euclidian")
hasilCluster <- kmeans(VTFIDF, 3)
clusplot(as.matrix(clus), hasilCluster$cluster, color=T, lines=0, label=4,sub="", 
main="K Means Gabungan")

clus2 <- dist(VTFIDF2, method="euclidian")
hasilCluster2 <- kmeans(VTFIDF2, 3)
clusplot(as.matrix(clus2), hasilCluster2$cluster, color=T, lines=0, label=4,sub="", 
main="K Means Tokopedia")

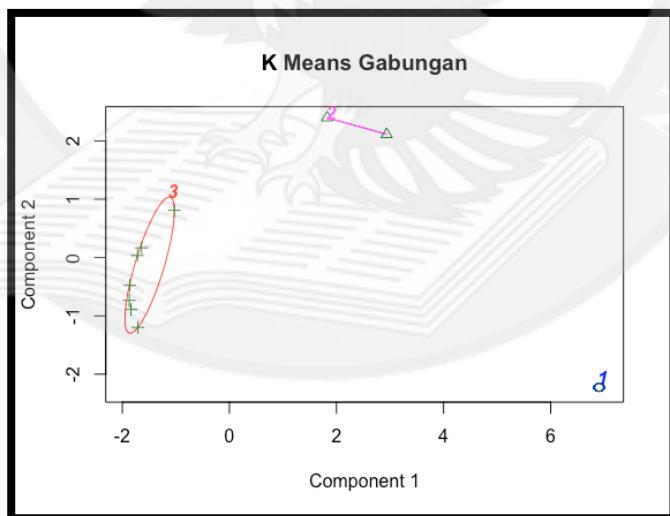
clus3 <- dist(VTFIDF3, method="euclidian")
hasilCluster3 <- kmeans(VTFIDF3, 3)
clusplot(as.matrix(clus3), hasilCluster3$cluster, color=T, lines=0, label=4,sub="", 
main="K Means Shopee")

clus4 <- dist(VTFIDF4, method="euclidian")
hasilCluster4 <- kmeans(VTFIDF4, 3)
clusplot(as.matrix(clus4), hasilCluster4$cluster, color=T, lines=0, label=2,sub="", 
main="K Means Bukalapak")
```

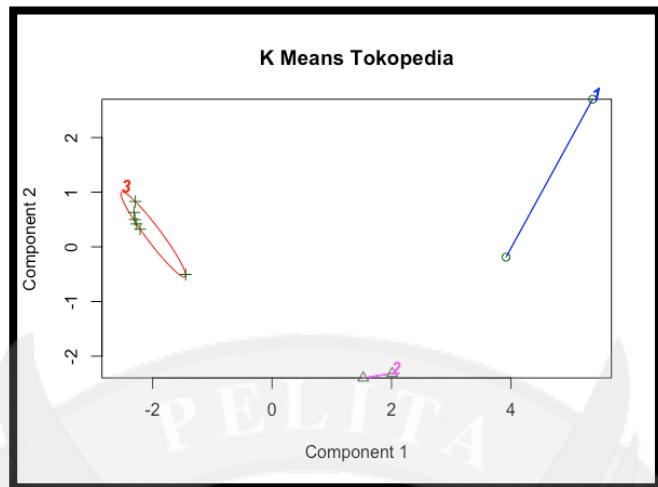
Gambar 4.55 Kode *clustering* tahap 2

Penulisan fungsi ‘*dist*’ diatas berfungsi untuk menghitung dan mengembalikan matriks jarak yang dihitung dengan menggunakan ukuran jarak yang ditentukan untuk menghitung jarak antara baris matriks data. Sedangkan penulisan kode “VTFIDF” hingga “VTFIDF4” merupakan objek yang akan dihitung menggunakan fungsi ‘*dist*’. “VTFIDF” hingga “VTFIDF4” merupakan *data frame* TF-IDF yang telah dibuat pada tahap sebelumnya. *Method* yang tersedia pada fungsi ‘*dist*’ cukup banyak, contohnya seperti *maximum*, *manhattan*, *binary*, *euclidian*, dan masih banyak lagi. Pada fungsi ini menggunakan metode *euclidian*

karena metode tersebut merupakan *method* yang paling sederhana serta tujuan dari *method* tersebut yaitu menghitung jarak antar vektor *dataframe*. Penulisan angka 3 pada baris fungsi ‘kmeans’ bertujuan untuk menentukan berapa *cluster* yang diinginkan. Penelitian ini membagi *cluster* menjadi 3 berdasarkan frekuensi munculnya kata (jarang, sedang, dan sering). Hal tersebut menjadikan alasan mengapa “VTFIDF” hingga “VTFIDF4” dijadikan sebagai objek *clustering*. Sedangkan untuk penulisan fungsi ‘clusplot’, penulisan kode “as.matrix(clus)” merupakan objek *data matrix* yang diperlukan untuk melakukan fungsi *clustering* ini. Penulisan kode “hasilCluster\$cluster” merupakan panjang vektor yang merepresentasikan *clustering* dari “hasilCluster\$cluster”. Penulisan kode “color=T” bertujuan agar warna dalam visualisasi *cluster* tidak dalam warna *default* agar visualisasi dapat terlihat lebih baik. Sedangkan penulisan kode “lines=0” bertujuan agar tidak ada garis koneksi antar *cluster*. Penulisan kode “label=2” bertujuan agar *cluster* memiliki label penanda masing-masing. Sedangkan penulisan kode “main” seperti pada umumnya pada R studio yaitu bertujuan untuk menentukan judul pada visualisasi yang dilakukan.



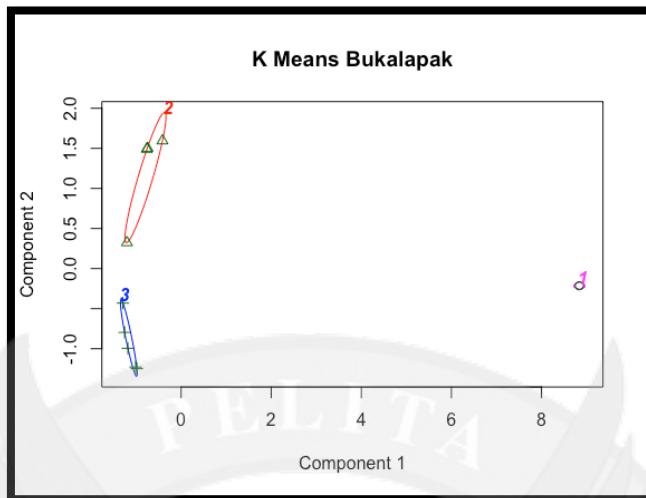
Gambar 4.56 Hasil *K-means* topik gabungan



Gambar 4.57 Hasil *K-means* topik Tokopedia



Gambar 4.58 Hasil *K-means* topik Shopee



Gambar 4.59 Hasil *K-means* topik Bukalapak

Setelah visualisasi berhasil dibuat, selanjutnya adalah melakukan evaluasi untuk melihat seberapa baik akurasi dari *clustering* yang telah dilakukan, dapat dilihat seperti pada kode berikut:

```
> hasilCluster$betweenss/hasilCluster$totss*100
[1] 95.3652
> hasilCluster2$betweenss/hasilCluster2$totss*100
[1] 95.32189
> hasilCluster3$betweenss/hasilCluster3$totss*100
[1] 91.51515
> hasilCluster4$betweenss/hasilCluster4$totss*100
[1] 97.89755
```

Gambar 4.60 Evaluasi *K-means* topik

Hasil evaluasi yang muncul pada bagian *R console* dari keempat data *tweets* yang dianalisa dapat dilihat pada tabel 4.5.

Tabel 4.5 Hasil evaluasi akurasi *K-means* topik

Data Tweets	Evaluasi Akurasi
Gabungan	95%
Tokopedia	95%
Shopee	91%
Bukalapak	97%

Berikutnya untuk *clustering* penilaian sentimen dapat dimulai dengan menggunakan fungsi ‘*kmeans*’ pada nilai sentimen yang telah dibuat pada tahap *lexicon* sebelumnya, dapat dilihat seperti pada kode berikut:

```
sentimentClus <- kmeans(hasilSentimen$score, 3)
sentimentClus2 <- kmeans(hasilSentimen2$score, 3)
sentimentClus3 <- kmeans(hasilSentimen3$score, 3)
sentimentClus4 <- kmeans(hasilSentimen4$score, 3)
```

Gambar 4.61 Kode *clustering* tahap 3

Setelah fungsi tersebut berhasil dijalankan, selanjutnya dilakukan dengan pengambilan jumlah isi *cluster*, dapat dilihat seperti pada kode berikut:

```
SentimentClusSize <- sentimentClus$size
SentimentClusSize2 <- sentimentClus2$size
SentimentClusSize3 <- sentimentClus3$size
SentimentClusSize4 <- sentimentClus4$size
```

Gambar 4.62 Kode *clustering* tahap 4

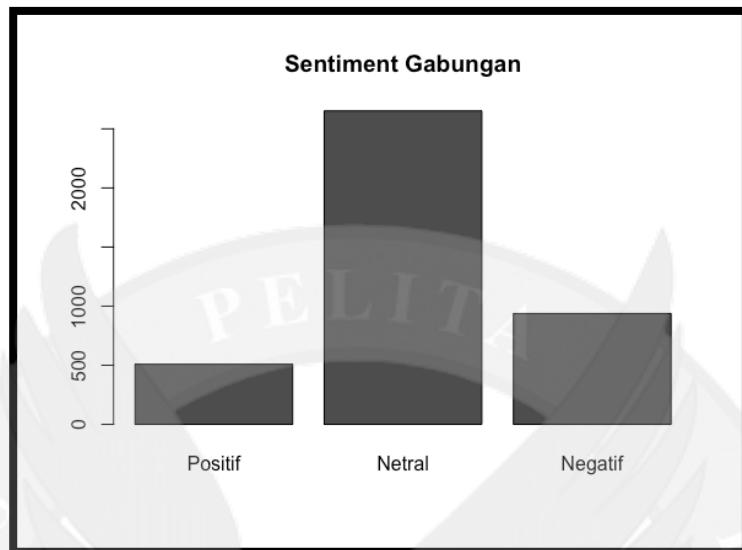
Setelah jumlah isi *cluster* berhasil didapatkan, selanjutnya dilanjutkan dengan mengubahnya menjadi *dataframe* dan divisualisasikan dengan *barplot*, dapat dilihat seperti pada kode berikut:

```
DFSentiment <- data.frame(SentimentClusSize,sentimentLabel)
DFSentiment2 <- data.frame(SentimentClusSize2,sentimentLabel)
DFSentiment3 <- data.frame(SentimentClusSize3,sentimentLabel)
DFSentiment4 <- data.frame(SentimentClusSize4,sentimentLabel)

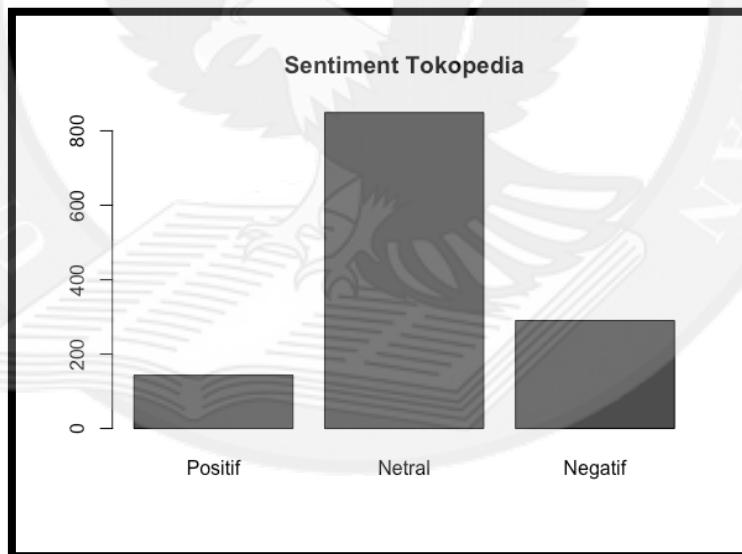
barplot(t(as.matrix(DFSentiment)),names.arg = sentimentLabel, main="Sentiment")
barplot(t(as.matrix(DFSentiment2)),names.arg = sentimentLabel, main="Sentiment")
barplot(t(as.matrix(DFSentiment3)),names.arg = sentimentLabel, main="Sentiment")
barplot(t(as.matrix(DFSentiment4)),names.arg = sentimentLabel, main="Sentiment")
```

Gambar 4.63 Kode *clustering* tahap 5

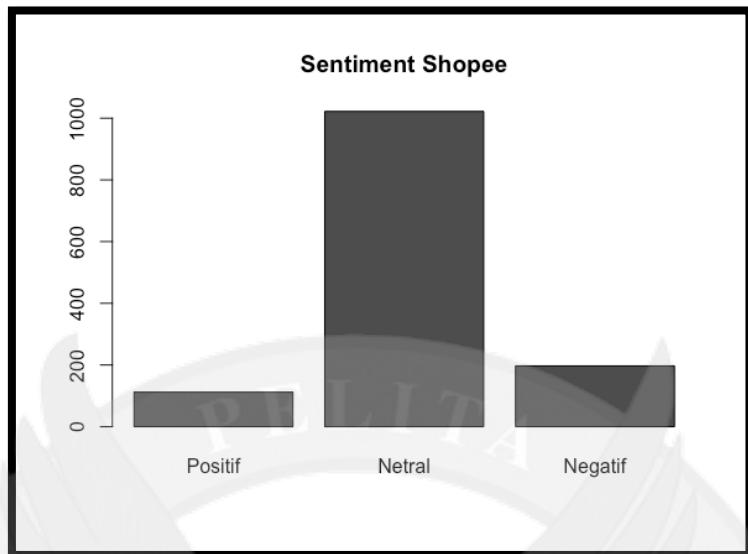
Visualisasi untuk *clustering* sentimen dapat dilihat pada gambar dibawah.



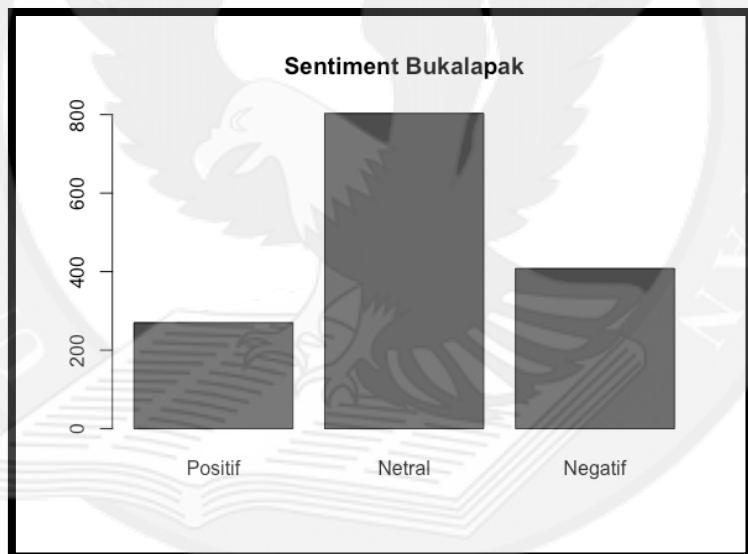
Gambar 4.64 Hasil *K-Means* sentimen gabungan



Gambar 4.65 Hasil *K-Means* sentimen Tokopedia



Gambar 4.66 Hasil *K-Means* sentimen Shopee



Gambar 4.67 Hasil *K-Means* sentimen Bukalapak

Setelah visualisasi berhasil dibuat, selanjutnya adalah melakukan evaluasi untuk melihat seberapa baik akurasi dari *clustering* yang telah dilakukan, dapat dilihat seperti pada kode berikut:

```
> sentimentClus$betweenss/sentimentClus$totss*100
[1] 84.70287
> sentimentClus2$betweenss/sentimentClus2$totss*100
[1] 88.57839
> sentimentClus3$betweenss/sentimentClus3$totss*100
[1] 85.18947
> sentimentClus4$betweenss/sentimentClus4$totss*100
[1] 82.70211
```

Gambar 4.68 Evaluasi *K-means* sentimen

Hasil evaluasi yang muncul pada bagian *R console* dari keempat data *tweets* yang dianalisa dapat dilihat pada tabel 4.6.

Tabel 4.6 Hasil evaluasi akurasi *K-means* sentimen

Data Tweets	Evaluasi Akurasi
Gabungan	84%
Tokopedia	88%
Shopee	81%
Bukalapak	82%

4.8 Analisa

Hasil-hasil yang didapatkan antar *online marketplace* ini dapat dikatakan cukup unik karena memiliki perbedaan yang cukup terlihat satu sama lainnya. Setelah menganalisa data topik-topik yang paling banyak dibahas oleh konsumen *online marketplace* di media sosial twitter, topik-topik yang banyak dibahas oleh konsumen di media sosial twitter tersebut tentu harus menjadi hal yang patut diperhatikan. Memperhatikan komentar, ulasan pada media sosial kini telah menjadi salah satu hal yang harus dilakukan oleh pebisnis terutama jika bisnisnya sudah dikenal banyak oleh masyarakat. Penelitian lainnya yang berjudul *The Effect of Online Customer Review Toward Purchase Intention* [31] meneliti apakah

review konsumen *online* mempengaruhi keputusan konsumen untuk membeli produk. Penelitian tersebut mengemukakan bahwa *online customer review* memiliki pengaruh yang signifikan terhadap niat beli konsumen. Penelitian lainnya yang berjudul *The Impact of Online Consumer Reviews Dimension on Online Purchase Intentions In Tokopedia* [32] juga mengemukakan bahwa *review* yang relevan bisa mempengaruhi niat beli konsumen. Dengan kata lain, *review* positif akan meningkatkan niat seseorang untuk membeli dimana hal tersebut juga akan meningkatkan *brand* atau *image* dari *online marketplace* secara tidak langsung.

Berdasarkan hasil *word cloud* yang dibuat pada tahap *word cloud creating*, dapat dilihat perbedaan yang cukup signifikan diantara tiga *online marketplace* tersebut, yaitu Tokopedia, Shopee dan Bukalapak. Pada visualisasi *word cloud* Tokopedia, dapat dilihat bahwa kata belanja dan beli merupakan kata dengan ukuran terbesar, dimana artinya kata tersebut merupakan kata yang paling sering muncul. Sedangkan pada visualisasi *word cloud* Shopee, konsumen pada media sosial twitter lebih dominan membahas mengenai grup *music entertainment* yang berasal dari negeri Korea Selatan, hal ini cukup banyak dibahas saat ini terutama setelah Tokopedia beberapa saat lalu mengumumkan BTS (grup *music entertainment* yang berasal dari negeri Korea Selatan) sebagai *brand ambassador* mereka [33]. Sedangkan pada visualisasi *word cloud* Bukalapak, konsumen pada media sosial twitter banyak membahas mengenai transaksi dan program kartu prakerja, hal tersebut dikarenakan pada masa tahap *data collecting*, Bukalapak melakukan kerjasama dengan pemerintah untuk memberikan bantuan kepada masyarakat yang membutuhkan sehingga kartu tersebut bisa digunakan untuk membeli *voucher* belajar di Bukalapak [34].

Berdasarkan hasil dari *term weighting* yang menggunakan metode TF-IDF, ketiga *online marketplace* ini juga memiliki hasil yang berbeda. Hasil dari *term weighting* ini tidak jauh berbeda dengan *word cloud* pada tahap sebelumnya, dikarenakan kedua metode ini memiliki konsep yang mirip namun perbedaannya terletak pada visualisasi yang digunakan. Sedangkan untuk hasil dari *clustering* topik yang menggunakan metode *k-means clustering* dapat dilihat pada tabel 4.7.

Tabel 4.7 Rincian hasil *clustering* topik

<i>Marketplace</i>	<i>Cluster</i>		
	1	2	3
Tokopedia	belanja, terimakasih	cek, Tokopedia	beli, twice, tau, mantap, tercare, download
Shopee	jongho, pilih	Shopee, exo	dream, beli, panen, nct, smartphone, menang
Bukalapak	cek	Bukalapak, latih, beli, prakerja	mantap, bantu, balas, tunggu, barang

Hasil *clustering* topik pada tabel 4.7 terbagi menjadi 3 sesuai dengan frekuensi munculnya kata/*term* tersebut. *Clustering* tersebut dilakukan secara otomatis menggunakan metode *k-means*. Topik-topik yang banyak dibicarakan oleh masyarakat di media sosial tentu menjadi suatu hal yang harus diperhatikan, dan bisa saja menjadi salah satu *success factor*. Namun, jika berbicara mengenai *success factor*, tentu faktor-faktor yang banyak dibahas oleh masyarakat pada media sosial tidak dapat mencakup keseluruhan *success factor*, karena *success factor* sendiri terdiri dari banyak aspek dan mencakup lingkup yang sangat luas. Contohnya seperti faktor internal maupun eksternal yang tentunya sama sekali tidak dibahas oleh masyarakat di media sosial. Seperti yang dibahas sebelumnya, *Success factor* adalah faktor-faktor yang dapat membuat bisnis, ataupun karir mencapai titik kesuksesan. Namun, topik-topik yang banyak dibahas oleh konsumen tersebut dapat mempengaruhi niat konsumen dalam membeli [31] [32]. Hal tersebut menjadi penting mengingat pembelian merupakan salah satu pemasukan untuk *online marketplace*, serta apabila konsumen memutuskan untuk melakukan transaksi pada salah satu *online marketplace* berarti *online marketplace* tersebut berhasil menjadi pilihan *online marketplace* terbaik menurut konsumen dibandingkan dengan sekian banyaknya *online marketplace* saat ini. Hasil *clustering* untuk *online marketplace* Tokopedia menunjukkan bahwa kata “belanja” dan “terimakasih” menjadi kata dengan frekuensi muncul yang paling sering, dan kata “cek” dan “Tokopedia” menjadi kata dengan frekuensi muncul sedang, dan kata “beli”, “twice”, “tau”, “mantap”, “tercare” dan “download” menjadi kata dengan frekuensi muncul agak

rendah. Hal ini menunjukkan bahwa banyak konsumen *online marketplace* Tokopedia yang membahas mengenai transaksi yang dilakukan pada *online marketplace* tersebut, konsumen melakukan transaksi melalui *marketplace* dikarenakan konsumen tersebut sudah mempunyai kepercayaan(*trust*) terhadap *online marketplace* tersebut. Pada penelitian lainnya yang membahas *success factor* berjudul *Lesson from Tokopedia.com: E-Commerce Success Factor Analysis* [35], menyimpulkan bahwa kesuksesan *e-commerce* berasal dari pembentukan *trust* dan *loyalty*. Penelitian lainnya yang berjudul *The Key Success Factors In E-Marketplace Implementation: A Systematic Literature Review* [36] juga menjadikan *trust* sebagai salah satu hasil *key success factors* pada penelitian tersebut. *Trust* menjadi faktor terpenting karena tanpa *trust*, *online marketplace* tidak akan ada [34]. Sedangkan hasil *clustering* untuk *online marketplace* Shopee menunjukkan bahwa kata “jongho” serta “pilih” menjadi kata dengan frekuensi muncul yang paling sering, dan kata “Shopee” serta “exo” menjadi kata dengan frekuensi muncul sedang, dan kata “dream”, “beli”, “panen”, “nct”, “smartphone”, “menang” menjadi kata dengan frekuensi muncul agak rendah. Hal ini menunjukkan bahwa banyak konsumen *online marketplace* Shopee yang membahas mengenai *brand ambassador marketplace* tersebut. Salah satu alasan mengapa konsumen *marketplace* Shopee banyak membahas mengenai *k-pop* adalah karena *marketplace* Shopee pada saat yang berdekatan dengan tahap *data collecting* mengadakan *giveaway* ketika konsumennya melakukan *vote* dan membagikannya di media sosial. Dari topik tersebut dapat dilihat bahwa teknik pemasaran dengan melibatkan nama-nama artis ternama bisa sangat berpengaruh terhadap minat konsumen, terbukti dari banyaknya konsumen yang membahas terkait hal ini di media sosial sehingga menjadikan *marketplace* Shopee makin banyak dikenal di banyak kalangan. Mengundang atau bekerja sama dengan artis ternama dapat meningkatkan minat konsumen, apalagi negara Indonesia menempati peringkat ketiga pada daftar negara dengan jumlah cuitan *K-Pop* terbanyak di twitter menurut situs kpopchart [37]. Hal tersebut menjadi salah satu penyebab mengapa grup *music entertainment* yang berasal dari negeri Korea Selatan cukup banyak dibahas oleh konsumen *online marketplace* Shopee di media sosial twitter.



Gambar 4.68 Daftar negara dengan ciutan *kpop* terbanyak di twitter

Sumber: situs kpopchart [37]

Sedangkan hasil *clustering* topik untuk *online marketplace* Bukalapak menunjukkan bahwa kata “cek” menjadi kata dengan frekuensi muncul yang paling sering, dan kata “Bukalapak”, “latih”, “beli” serta “prakerja” menjadi kata dengan frekuensi muncul sedang, dan kata “mantap”, “bantu”, “balas”, “tunggu”, serta “barang” menjadi kata dengan frekuensi muncul agak rendah. Hal tersebut menunjukkan bahwa banyak konsumen *online marketplace* Bukalapak yang membahas mengenai pelatihan program kartu pra kerja. Hal tersebut dikarenakan Bukalapak menjadi salah satu mitra platform digital yang menjalin kerjasama dengan penyedia prakerja sebagai penyedia layanan pembelian pelatihan. Bentuk kerjasama Bukalapak dan pemerintah untuk memberikan bantuan kepada masyarakat yang membutuhkan sehingga kartu tersebut bisa digunakan untuk membeli voucher pelatihan di Bukalapak. Hal lainnya yang banyak dibahas oleh konsumen *online marketplace* adalah mengenai keluhan kepada *customer service* dari Bukalapak untuk membela keluhan yang ada di media sosial twitter terlihat dari kata “cek” menjadi kata dengan frekuensi tertinggi.

Sedangkan untuk hasil *clustering* sentimen ketiga *online marketplace* yang diteliti, ketiga *online marketplace* ini memiliki hasil yang cukup serupa yaitu hasil sentimen didominasi oleh sentimen netral jika dilihat pada visualisasi.

