UNIVERSITAS SULTAN AGENG TIRTAYASA

UNTIRTA Jawara
Jujur Adil Wibawa Amanah Religius Akuntabel

Data Mining

#11 Meeting

Text Analysis : Tokenisasi, Vektorisasi dan Word Cloud

Ferdian Bangkit Wijaya, S.Stat., M.Si
NIP. 19900520202406100 1

Text Mining

Dalam Data Mining, kita sering bekerja dengan data terstruktur (angka dalam tabel). Namun, diperkirakan lebih dari 80% data di dunia adalah data tidak terstruktur, terutama dalam bentuk teks:

- Ulasan pelanggan (Tokopedia, Google Maps)
- Umpan media sosial (Twitter, Instagram)
- Survei pertanyaan terbuka
- Artikel berita dan email

Analisis Teks (Text Mining) adalah proses mengubah data teks yang tidak terstruktur dan "berantakan" menjadi data terstruktur yang kuantitatif, sehingga kita dapat menemukan pola, tren, dan wawasan tersembunyi.
Tantangan Utama:

- Ambiguitas: Kata "dingin" bisa berarti suhu atau sifat.
- Non-Standar: Typo ("baguss"), slang ("bgt"), singkatan ("yg").
- Konteks: Kata "puas" positif, tapi "tidak puas" sangat negatif.
- Stop Words: Kata-kata umum (seperti "yang", "di", "dan", "ke") sangat sering muncul tetapi tidak memiliki makna analitis.

Tokenisasi → Tidy Text

Untuk mengatasi tantangan ini, kita menggunakan framework Tidy Text. Filosofi utamanya adalah mengubah data teks menjadi format tidy, di mana:
Satu Baris = Satu Unit Analisis (Token)
Token yang paling umum adalah satu kata (unigram). Proses mengubah dokumen (kalimat/paragraf) menjadi token-token individual disebut Tokenisasi.

Perhitungan Manual: Alur Kerja Tokenisasi & Cleaning

Mari kita lakukan proses manual untuk memahami alur kerja tidytext.

Data Mentah (2 Ulasan):
Teks 1: "Pelayanan sangat bagus dan cepat"
Teks 2: "Harga mahal tapi pelayanan jelek"
Kamus Eksternal (Leksikon Stop Words):["dan", "tapi", "sangat"]

Tokenisasi → Tidy Text

Langkah 1: Tokenisasi Kita ubah setiap kalimat menjadi format tidy (satu kata per baris).

Doc ID	Kata
Teks 1	pelayanan
Teks 1	sangat
Teks 1	bagus
Teks 1	dan
Teks 1	cepat
Teks 2	harga
Teks 2	mahal
Teks 2	tapi
Teks 2	pelayanan
Teks 2	jelek

Langkah 2: Stop Word Removal (Menggunakan anti_join)

Kita akan melakukan anti_join. Ini berarti: "Simpan semua kata di data kita yang TIDAK ADA di kamus stop words."

- "sangat" → Dibuang
- "dan" → Dibuang
- "tapi" → Dibuang

Data Setelah Cleaning:

Doc ID	Kata
Teks 1	pelayanan
Teks 1	bagus
Teks 1	cepat
Teks 2	harga
Teks 2	mahal
Teks 2	pelayanan
Teks 2	jelek

Frekuensi & Word Cloud

Langkah 3: Menghitung Frekuensi Sekarang kita bisa count() data bersih kita.

Kata	Frekuensi (n)
pelayanan	2
bagus	1
cepat	1
harga	1
mahal	1
jelek	1

Langkah 4 : Word Cloud

Word Cloud adalah visualisasi paling dasar dari data frekuensi ini. Ukuran kata merepresentasikan frekuensinya.
Meskipun populer, Word Cloud memiliki kelemahan fatal:

- Tidak ada Konteks: Kata "pelayanan" (2 kali) terlihat paling penting. Tapi apakah itu "pelayanan bagus" atau "pelayanan jelek"? Word Cloud tidak bisa membedakannya.
- Masalah Negasi: Frasa "tidak bagus" akan dihitung sebagai "tidak" dan "bagus". Ini secara keliru meningkatkan frekuensi kata "bagus", padahal sentimennya negatif.

Text Mining di R

```
# 1. Install & Load Library
# install.packages(c("dplyr", "tidytext", "janeaustenr", "wordcloud"))
library(dplyr)
library(tidytext)
library(janeaustenr) # Data novel Jane Austen (untuk contoh)
library(wordcloud)   # Untuk plot word cloud

# 2. Siapkan Data Awal (Buku "Emma")
buku_emma <- austen_books() %>%
  filter(book == "Emma") %>%
  select(text) # Hanya ambil kolom teks

# 3. TOKENISASI (Membuat Tidy Text)
# unnest_tokens() mengubah 1 baris paragraf menjadi N baris kata
tidy_emma <- buku_emma %>%
  unnest_tokens(word, text)

# 4. HAPUS STOP WORDS (Menggunakan ANTI_JOIN)
data("stop_words") # Muat kamus stop words bawaan tidytext

tidy_emma_bersih <- tidy_emma %>%
  anti_join(stop_words, by = "word") # Terapkan filter anti_join

# 5. MENGHITUNG FREKUENSI (Data Bersih)
word_counts <- tidy_emma_bersih %>%
  count(word, sort = TRUE)

cat("--- Frekuensi Kata Teratas (Setelah Cleaning) ---\n")
print(head(word_counts, 10))

# 6. MEMBUAT WORD CLOUD
# Plot akan muncul di tab 'Plots' RStudio
with(word_counts,
     wordcloud(words = word,
               freq = n,
               max.words = 100,
               random.order = FALSE,
               colors = RColorBrewer::brewer.pal(8, "Dark2")))
```

Visualisasi Word Cloud di R

Vektorisasi → Tokenisasi ke DTM

Pendekatan Python (scikit-learn)

- Filosofi: Bag-of-Words / Vektorisasi (Data "Lebar").
- Struktur: Setiap baris adalah satu dokumen (ulasan). Setiap kolom adalah satu kata unik.
- Contoh (Data 2 Ulasan): Menghasilkan tabel/matrix dengan 2 baris dan 6 kolom (pelayanan, bagus, cepat, harga, mahal, jelek).
- # Ini adalah Document-Term Matrix (DTM)
- bagus | cepat | harga | jelek | mahal | pelayanan
- ----------
- 0 1 | 1 | 0 | 0 | 0 | 1 <- Teks 1
- 1 0 | 0 | 1 | 1 | 1 | 1 <- Teks 2
- Kekuatan: Sangat efisien dan merupakan format standar yang siap dimasukkan ke dalam model Machine Learning (seperti Naive Bayes, SVM, dll) di scikit-learn.

Text Mining di Python

```
# 1. Import Library
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import pandas as pd
from collections import Counter

# 2. Download data NLTK (cukup 1x saja)
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')

# 3. Data Contoh
data = [
    "Pelayanan sangat bagus dan cepat",
    "Harga mahal tapi pelayanan jelek"
]
```

```
# 4. Cleaning: Hapus stopwords & lowercase
stop_words = set(stopwords.words('indonesian'))
stop_words.update(["dan", "tapi", "sangat"]) # Tambah stopwords custom

teks_bersih = []
for kalimat in data:
 kalimat = kalimat.lower() # Ubah ke huruf kecil
 kata_kata = word_tokenize(kalimat) # Pisahkan jadi kata-kata

 # Ambil kata yg bukan stopwords & hanya huruf
 kata_bersih = [kata for kata in kata_kata
 if kata.isalpha() and kata not in stop_words]

 teks_bersih.append(" ".join(kata_bersih))

print("Teks setelah dibersihkan:")
print(teks_bersih)
```

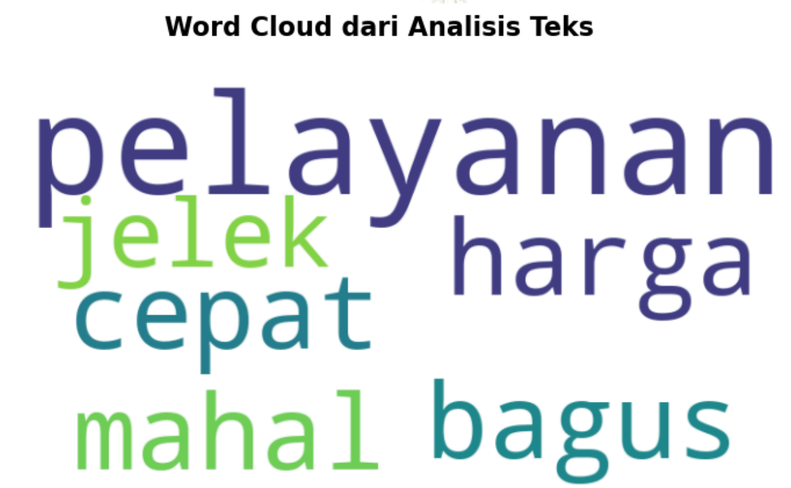
Text Mining di Python

```
# ===== CARA 1: CountVectorizer (untuk Machine Learning) =====
vec = CountVectorizer()
X = vec.fit_transform(teks_bersih)
dtm = pd.DataFrame(X.toarray(),
columns=vec.get_feature_names_out())

print("\n--- Document-Term Matrix (DTM) ---")
print(dtm)
print("\n📝 Penjelasan DTM:")
print("Baris 0 = Teks pertama, Baris 1 = Teks kedua")
print("Kolom = Kata yang muncul")
print("Angka = Frekuensi kata di setiap dokumen")

# Hitung total frekuensi dari DTM
frek_cara1 = dtm.sum().to_dict()
print("\n--- Frekuensi (Cara 1 - DTM) ---")
print(frek_cara1)
```

```
# ===== CARA 2: Counter (untuk Word Cloud) =====
semua_kata = " ".join(teks_bersih) # Gabungkan semua teks
frek_cara2 = Counter(semua_kata.split()) # Hitung frekuensi

print("\n--- Frekuensi (Cara 2 - Counter) ---")
print(frek_cara2)

print("\n✅ KESIMPULAN:")
print("Kedua cara menghasilkan frekuensi yang SAMA!")
print("- Cara 1 (DTM): Cocok untuk machine learning")
print("- Cara 2 (Counter): Cocok untuk Word Cloud (lebih sederhana)")

# Siapkan Word Cloud (akan divisualisasikan di cell berikutnya)
wordcloud = WordCloud(width=800, height=400,
 background_color="white").generate_from_frequencies(frek_cara2)
```

Text Mining di Python

```
# ===== VISUALISASI WORD CLOUD =====

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title("Word Cloud dari Analisis Teks", fontsize=16, fontweight='bold')
plt.axis('off')
plt.show()

print("\n✅ Word Cloud berhasil dibuat!")
print("Kata 'pelayanan' paling besar karena muncul 2x")
```



UNIVERSITAS SULTAN AGENG TIRTAYASA

UNTIRTA Jawara
Jujur Adil Wibawa Amanah Religius Akuntabel

SEE YOU NEXT WEEK !

Ferdian Bangkit Wijaya, S.Stat., M.Si
NIP. 19900520202406100 1
ferdian.bangkit@untirta.ac.id