TUGAS 6 Analisis Sentiment dan Data Scrapping

Gojek vs Grab di twitter

Di susun oleh:

Nama: Rizqy Mubarok

NPM: 23.0504.0024

Fakultas Teknik

Universitas Muhammadiyah Magelang 2025

A. Scrapping

1. Untuk scrapping saya menggunakan Library twikit untuk mengambil data dari twitter

Import Library dan token twitter `Twikit` mengharuskan user menginputkan token sehingga tidak bisa anonim

```python
import csv
import nest_asyncio
import asyncio
import random
import math
from twikit import Client, TooManyRequests

nest_asyncio.apply()
# SETUP CLIENT + COOKIE
client = Client("en-US")
client.set_cookies({
    "auth_token": "d125d285011b59146b60cdcd6edd9c82dca055c9",
    "ct0": "59e9745d43723d4cb98f25be06a03b19dc235dbb0f0410ca8b0b49f8231a184b2bc947fcc2fd77668b90979426c200556e89b9d9dbd00
    "twid": "u%3D1645176567038091265",
    "gt": "1989338289653047338",
    "guest_id": "v1%3A175121650454954896",
    "guest_id_ads": "v1%3A175121650454954896",
    "guest_id_marketing": "v1%3A175121650454954896",
    "kdt": "HZdPjfR2pqsvUhxsO5GasalB5x7wYSzJ8sBbSx85",
    "lang": "en",
    "personalization_id": "v1_JcinIxPjnVOOctgz7rKeiQ==",
    "cf_bm": "sjuQ5SgqIGcWGuc4AGCyaC42n3AikDN522y1tic1cGI-1763135431.7418268-1.0.1.1-3bCnpzTDAGCtwt0vvIoOgF_Li.Jdb5N0EUXs
    "cuid": "f6d3d51a5a0345ee9e273cb98b2c5068",
    "att": "1-PTOudQBU2NDJeJZeJ8k2vit0taZstJ225z7kN54g",
    "external_referer": "padhuUp37zhD6%2F29CpQtyhGQCUl05AFo|0|8e8t2xd8A2w%3D",
    "g_state": "{\"i_l\":0,\"i_ll\":1763130464264}",
    "gt": "1989338289653047338",
})
```

Token yang saya pakai Adalah token yang saya ambil dari menu dev tools setelah login emnggunakan akun twitter saya, jadi datanya bisa saya pertanggunga jawabkan.

2. Beberapa parameter yang saya gunakan untuk scrapping

```python
# Config
BATCH_SIZE = 20
TARGET_PER_KEYWORD = 500
CHECKPOINT_EVERY = 50
MIN_DELAY = 10
MAX_DELAY = 20
MAX_RETRIES = 6
CSV_FILENAME = "tweet_gojek_grab.csv"

def write_header_if_needed(filename):
    try:
        with open(filename, "x", encoding="utf-8", newline="") as f:
            writer = csv.writer(f)
            writer.writerow(["keyword", "tweet"])
    except FileExistsError:
        pass

def append_rows(filename, rows):
    with open(filename, "a", encoding="utf-8", newline="") as f:
        writer = csv.writer(f)
        writer.writerows(rows)
```

- Batch size = 20 agar Ketika kode melakukan pengambilan data dalam 1 kali request, maksimal hanya mengambil 20 data untuk menghindari rate limit (too many request).
- Target per keyword = 500 per untuk keyword gojek dan grab masing-masing ada 500 data.
- Min dan max delay untuk menentukan jarak waktu pengiriman request ke twitter, jika terlalu cepat beresiko terkena rate limit.
- Max Retries di gunakan untuk menghentkan kode jika terjadi eror pada jaringan atau twitter menolak respon dari request.

3. Karena twitter mempunyai aturan sangat ketat terkait data pengguna, maka dari itu saya menggunakan paginasi agar Ketika mengakses data dari twitter, tidak langsung mengambil 500 data namun hanya akan mengambil per halaman

```python
# scraping dengan pagination, backoff, dan checkpointing
async def scrape_keyword(keyword, target=TARGET_PER_KEYWORD):
    print(f"Start scraping keyword: {keyword} (target {target})")
    results = []
    cursor = None
    consecutive_empty = 0
    write_header_if_needed(CSV_FILENAME)

    while len(results) < target:
        attempt = 0
        while True:
            try:
                response = await client.search_tweet(
                    f"{keyword} lang:id",
                    product="Latest",
                    count=BATCH_SIZE,
                    cursor=cursor
                )
                tweets = response if response is not None else []
                break

            except TooManyRequests:
                attempt += 1
                backoff = min(60, 2 ** attempt + random.random() * 5)
                print(f"429 Rate limit. Backoff for {backoff:.1f}s (attempt {attempt})")
                await asyncio.sleep(backoff)
                if attempt >= MAX_RETRIES:
                    print("Max retries reached for 429. Exiting this keyword early.")
                    return results

```

```
                    return results

            except Exception as e:
                attempt += 1
                backoff = min(60, (2 ** attempt) + random.random() * 5)
                print(f"transient error: {e!r}. Backoff {backoff:.1f}s (attempt {attempt})")
                await asyncio.sleep(backoff)
                if attempt >= MAX_RETRIES:
                    print("Max retries reached for errors. Exiting this keyword early.")
                    return results

        if not tweets:
            consecutive_empty += 1
            print(f"iNo tweets returned in this batch (empty count={consecutive_empty}).")
            if consecutive_empty >= 4:
                print("iToo many consecutive empty batches — stopping pagination for this keyword.")
                break
            await asyncio.sleep(random.uniform(MIN_DELAY, MAX_DELAY))
            continue
        else:
            consecutive_empty = 0

        batch_texts = []
        for t in tweets:
            try:
                text = t.text
            except Exception:
                text = str(t)
            batch_texts.append(text)

        rows = []
        for txt in batch_texts:
            results.append(txt)
            rows.append([keyword.replace(" lang:id",""), txt])

        if rows:
            append_rows(CSV_FILENAME, rows)

```

```
        print(f"  got {len(batch_texts)} tweets (total collected: {len(results)}/{target})")

        try:
            next_cursor = getattr(tweets, "next_cursor", None)
            if next_cursor is None and isinstance(response, dict):
                next_cursor = response.get("next_cursor") or response.get("cursor")
        except Exception:
            next_cursor = None

        if not next_cursor:
            if len(results) >= target:
                break
            cursor = None
            print("no next_cursor; will try a few more small batches before stopping.")
        else:
            cursor = next_cursor
        await asyncio.sleep(random.uniform(MIN_DELAY, MAX_DELAY))
        if len(results) >= target:
            break

    print(f"Finished {keyword}: collected {len(results)} tweets.")
    return results[:target]
```

4. Eksekusi scrapping

```python
# main driver
async def main():
    write_header_if_needed(CSV_FILENAME)
    keywords = ["gojek", "grab"]

    final = {}
    for kw in keywords:
        tweets = await scrape_keyword(kw, TARGET_PER_KEYWORD)
        final[kw] = tweets
        cooldown = random.uniform(5, 12)
        print(f"Cooldown {cooldown:.1f}s before next keyword.")
        await asyncio.sleep(cooldown)

    print("All done. Summary:")
    for k, v in final.items():
        print(f" - {k}: {len(v)} tweets collected")
await main()
```

```
Start scraping keyword: gojek (target 500)
  got 19 tweets (total collected: 19/500)
  got 20 tweets (total collected: 39/500)
  got 20 tweets (total collected: 59/500)
  got 20 tweets (total collected: 79/500)
  got 20 tweets (total collected: 99/500)
  got 20 tweets (total collected: 119/500)
  got 20 tweets (total collected: 139/500)
  got 20 tweets (total collected: 159/500)
  got 20 tweets (total collected: 179/500)
  got 20 tweets (total collected: 199/500)
  got 20 tweets (total collected: 219/500)
  got 20 tweets (total collected: 239/500)
  got 20 tweets (total collected: 259/500)
  got 20 tweets (total collected: 279/500)
  got 20 tweets (total collected: 299/500)
  got 20 tweets (total collected: 319/500)
  got 20 tweets (total collected: 339/500)
  got 20 tweets (total collected: 359/500)
  got 20 tweets (total collected: 379/500)
  got 19 tweets (total collected: 398/500)
  got 20 tweets (total collected: 418/500)
  got 20 tweets (total collected: 438/500)
  got 20 tweets (total collected: 458/500)
  got 20 tweets (total collected: 478/500)
  got 20 tweets (total collected: 498/500)
  got 17 tweets (total collected: 515/500)
Finished gojek: collected 515 tweets.
Cooldown 5.1s before next keyword.
Start scraping keyword: grab (target 500)
  got 20 tweets (total collected: 20/500)
  got 20 tweets (total collected: 40/500)
  got 20 tweets (total collected: 60/500)
  got 20 tweets (total collected: 80/500)
  got 20 tweets (total collected: 100/500)
  got 20 tweets (total collected: 120/500)
  got 20 tweets (total collected: 140/500)
  got 20 tweets (total collected: 160/500)
  got 20 tweets (total collected: 180/500)
  got 20 tweets (total collected: 200/500)
  got 20 tweets (total collected: 220/500)
  got 20 tweets (total collected: 240/500)
  got 18 tweets (total collected: 258/500)
  got 20 tweets (total collected: 278/500)
  got 20 tweets (total collected: 298/500)
  got 20 tweets (total collected: 318/500)
  got 20 tweets (total collected: 338/500)
  got 20 tweets (total collected: 358/500)
  got 20 tweets (total collected: 378/500)
  got 20 tweets (total collected: 398/500)
  got 20 tweets (total collected: 418/500)
  got 20 tweets (total collected: 438/500)
  got 20 tweets (total collected: 458/500)
  got 20 tweets (total collected: 478/500)
  got 20 tweets (total collected: 498/500)
  got 20 tweets (total collected: 518/500)
Finished grab: collected 518 tweets.
Cooldown 10.0s before next keyword.
All done. Summary:
 - gojek: 500 tweets collected
 - grab: 500 tweets collected
```

B. Analisis sentiment

1. Import library

```python
import re
import os
import csv
import math
import random
import string
from collections import Counter, defaultdict

import numpy as np
import pandas as pd
from tqdm.auto import tqdm

# NLP
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# sklearn
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, precision_recall_fscore_support

#word2vec
from gensim.models import Word2Vec

# plotting
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from wordcloud import WordCloud, STOPWORDS
```

2. Load Dataset

Load Dataset

```python
# load dataset
INPUT_CSV = "tweet_gojek_grab.csv"
df = pd.read_csv(INPUT_CSV)
df = df[['keyword','tweet']].dropna().reset_index(drop=True)
print("Loaded rows:", len(df))
df.head()
```

```
Loaded rows: 1033
```

	keyword	tweet
0	gojek	Lah ini gojek daerah mana nih bawa glock?
1	gojek	@tubirfess kalo ngga mau satu tempat sama mere...
2	gojek	dan ongkir minimum yang diterima driver juga ...
3	gojek	@Ordinaryjhonny baru kmrn nyoba syopifud emg l...
4	gojek	Bangsat, lu kalau mau nyaman ya bawa kendaraan...

Next steps: Generate code with df | New interactive sheet

Dataset yang saya gunakan Adalah dataset hasil scrapping, dengan jumlah data sebanyak 1000 data dan masing-masing keyword ada 500 data.

3. Preprocessing data

```python
# 1. Preprocessing helpers
NORMALIZE_MAP = {
    "gk": "nggak", "ga": "nggak", "g": "nggak",
    "gk2": "nggak", "tdk": "tidak", "td": "tidak",
    "aja": "saja", "klo": "kalau", "kalo": "kalau",
    "jg": "juga", "jd": "jadi", "dgn": "dengan",
    "goblok": "goblok",
    # add more slang/abbrev here
}

# regex patterns
URL_RE = re.compile(r"https?://\S+|www\.\S+")
MENTION_RE = re.compile(r"@\w+")
HASHTAG_RE = re.compile(r"#\w+")
EMOJI_RE = re.compile("["
                      u"\U0001F600-\U0001F64F"
                      u"\U0001F300-\U0001F5FF"
                      u"\U0001F680-\U0001F6FF"
                      u"\U0001F1E0-\U0001F1FF"
                      "]+", flags=re.UNICODE)

PUNCT_TABLE = str.maketrans('', '', string.punctuation)
from nltk.corpus import stopwords
try:
    stop_words = set(stopwords.words('indonesian'))
except:
    stop_words = set(stopwords.words('english'))
```

Preprocessing data saya mulai dengan

- membunag kata-kata yang tidak baku dengan stemming
- menghilangkan karakter-karaktaer unik yaitu emoticon dengan regex
- menormalisasi kata-kata slang menjadi kata baku
- dan melakukan tokenisasi kata

```python

# stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

def normalize_text(s):
    s = str(s)
    s = s.strip()
    s = URL_RE.sub("", s)
    s = MENTION_RE.sub("", s)
    s = HASHTAG_RE.sub(lambda m: m.group(0)[1:], s)
    s = EMOJI_RE.sub("", s)
    tokens_tmp = s.split()
    tokens_tmp = [NORMALIZE_MAP.get(t.lower(), t) for t in tokens_tmp]
    s = " ".join(tokens_tmp)
    s = s.translate(PUNCT_TABLE)
    s = s.lower()
    s = re.sub(r'\s+', ' ', s).strip()
    return s

def preprocess_text(s, do_tokenize=True, do_stopword=True, do_stem=True):
    s = normalize_text(s)
    if not do_tokenize:
        return s
    tokens = word_tokenize(s)
    if do_stopword:
        tokens = [t for t in tokens if t not in stop_words and len(t) > 1]
    if do_stem:
        tokens = [stemmer.stem(t) for t in tokens]
    return tokens

print(preprocess_text("Gojek lemot bgt si update tracking drivernya #gojek 🤦 https://t.co/abc"))

```

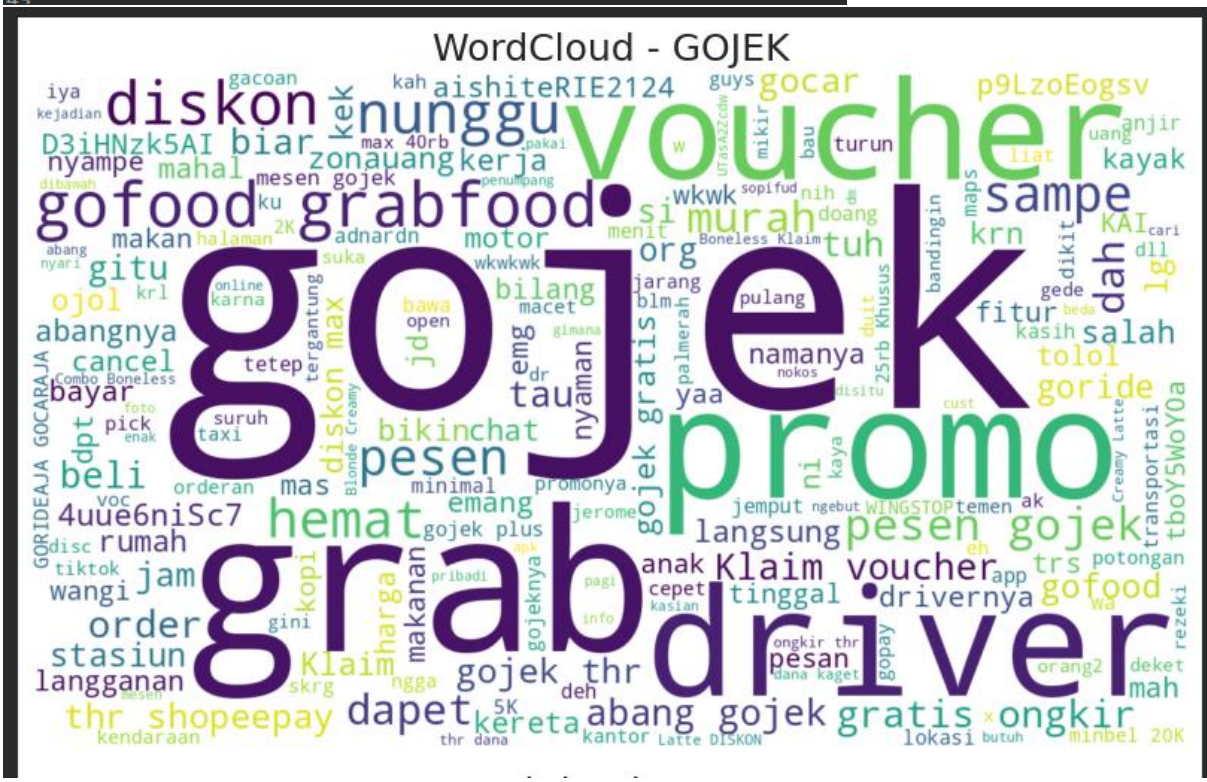
```python
tqdm.pandas()
df['clean_tokens'] = df['tweet'].progress_apply(lambda x: preprocess_text(x))
df['clean_text'] = df['clean_tokens'].apply(lambda toks: " ".join(toks))
```

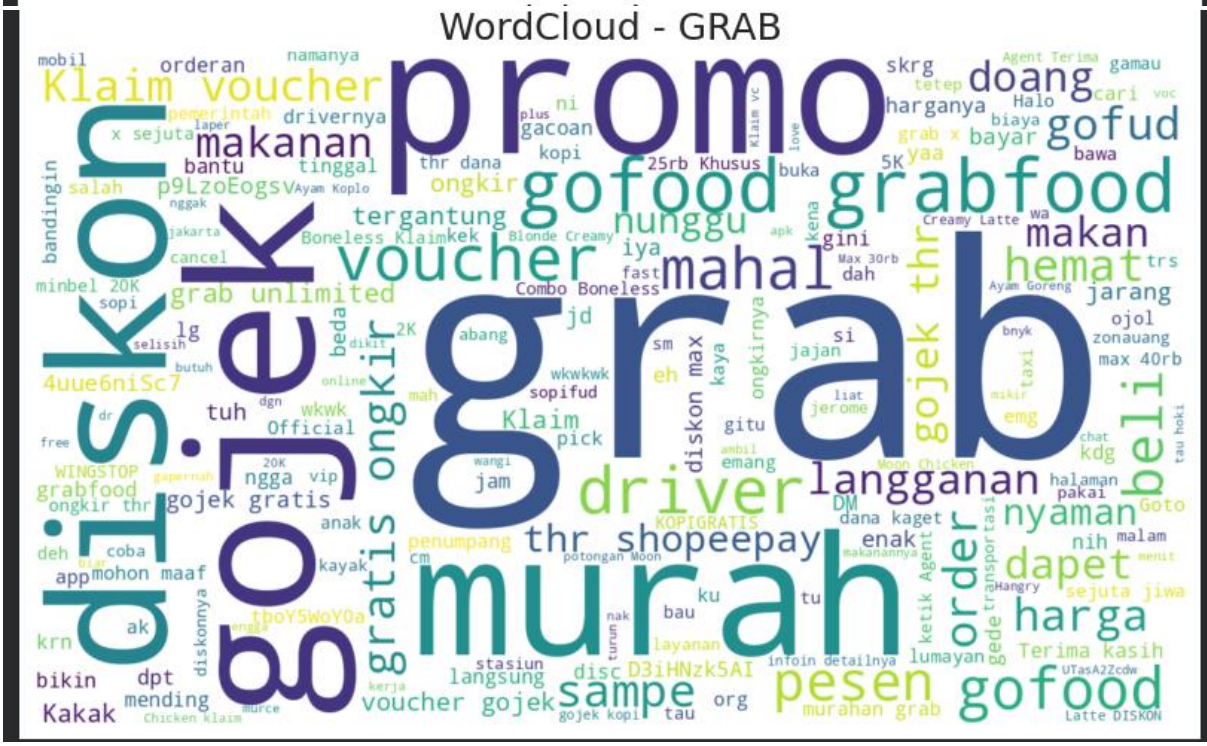
4. Membuat wordcloud

```python
[38]
✓ 5s
# Stopwords dasar
stop_words = set(stopwords.words("indonesian"))
stop_words = stop_words.union(set(stopwords.words("english")))
stop_words = stop_words.union(STOPWORDS)

# Tambahkan stopwords manual
extra_sw = {"rt", "via", "amp", "co", "https", "t", "nya", "gk", "ga", "https", "shoppe", "co", "kalo", "yang", "aja", "pake", "Ordinaryjh
            "shoppefood", "kalo", "aja", "pake", "yang", "udah", "ya", "shopee", "bgt", "go", "Cetepbebek", "kadang", "sih", "banget",
            "gue", "gw", "lu", "klo", "gua", "tp", "pas", "sfood", "food", "resto", "udh", "udah", "Shopeefood", "yg", "gak", "daget",
            "cek", "jg", "abis", "habis", "kl", "kak", "titik", "kode", "jalan", "orang", "lo", "kali", "tubirfess", "akun", "utk",
            "gt", "dh", "seenaknya", "knp"}
stop_words = stop_words.union(extra_sw)

# Fungsi bikin wordcloud
def generate_wordcloud(text, title):
    wc = WordCloud(
        width=1200,
        height=700,
        background_color="white",
        stopwords=stop_words,
        collocations=True
    ).generate(text)

    plt.figure(figsize=(10, 6))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.title(title, fontsize=18)
    plt.show()

```

Wordcloud saya buat agar memudahkan saya mengidentifikasi kata mana saja yang lebih sering muncul yang nantinya saya gunakan Ketika merancang stopwords.

```python
# Filter tweet berdasarkan keyword

# Tweet yang mengandung "gojek"
df_gojek = df[df["keyword"].str.lower() == "gojek"]
# Tweet yang mengandung "grab"
df_grab = df[df["keyword"].str.lower() == "grab"]

# Gabungkan jadi satu string
text_gojek = " ".join(df_gojek["tweet"].astype(str).tolist())
text_grab = " ".join(df_grab["tweet"].astype(str).tolist())

generate_wordcloud(text_gojek, "WordCloud - GOJEK")
generate_wordcloud(text_grab, "WordCloud - GRAB")

```





5. Labelling dataset

```python
# 2. Lexicon-based sentiment (simple)
LEXICON_CSV = "indonesian_lexicon.csv"
lexicon = {}

if os.path.exists(LEXICON_CSV):
    lex = pd.read_csv(LEXICON_CSV)
    for _, r in lex.iterrows():
        w = str(r['word']).strip().lower()
        s = float(r.get('score', 0))
        lexicon[w] = s
else:
    # fallback minimal lexicon (expand for better results)
    lexicon = {
        "bagus": 1.0, "mantap": 1.0, "mantapjiwa":1.0,"puas": 1.0, "murah": 1.0, "baguslah":1.0,
        "jelek": -1.0, "buruk": -1.0, "payah": -1.0, "kurang": -0.5, "telat": -1.0, "mal": -0.5,
        "lemot": -1.0, "nggak": -0.5, "ga": -0.5,"benci": -1.0, "suka": 1.0, "senang":1.0,
        "marah": -1.0, "tolol": -1.0, "nyaman":1.0,"murah": -1.0, "bagus": 1.0,
        "jelek": -1.0, "buruk": -1.0, "payah": -1.0,"kurang": -0.5, "telat": -1.0, "mal": -0.5,
        "lemot": -1.0, "nggak": -0.5, "ga": -0.5,"benci": -1.0, "suka": 1.0, "senang":1.0,
        "nunggu": -1.0, "tolol": -1.0, "nyaman":1.0,"hemat": 1.0, "promo": 1.0, "diskon": 1.0,
        "voucher": 1.0, "promosi": 1.0, "ramah": 1.0,"goblok": -1.0, "jahat": -1.0, "bangsat": -1.0,
        "tolol": -1.0, "sehat": 1.0, "enak": 1.0,"cape": -1.0, "gaada": -1.0, "gamau": -1.0,
        "fiktif": -1.0, "pukimak": -1.0, "telat": -1.0,"komunikatif": 1.0, "baik": 1.0,  "anjir": -1.0,
        "tidak puas": -1.0, "salah": -1.0, "nyolot": -1.0,"setia": 1.0, "lama": -1.0, "jarang": -1.0,
        "matiin": -1.0, "mati": -1.0, "lambat": -1.0,"parah": -1.0, "parah": -1.0, "mengecewakan": -1.0,
        "mahal": -1.0, "top": 1.0, "terbaik": 1.0, "keren": 1.0, "aman": 1.0, "cepat": 1.0,
        "thr": 1.0, "ragu": -1.0, "jarang": -1.0,"mual": -1.0, "pernah": 1.0, "tidak pernah": -1.0,
        "nggak pernah": -1.0, "ga pernah": -1.0, "ga ada": -1.0,"klaim": 1.0, "gratis": 1.0, "gratislah": 1.0,
        "potongan:": 1.0, "potong": 1.0, "potongin": 1.0,
    }
```

Sama seperti stop words, kata-kata yang saya masukan untuk labelling manual juga berdasarkan kata yang sering muncul di wordcloud.

```python
def lexicon_score(tokens):
    score = 0.0
    for t in tokens:
        score += lexicon.get(t, 0.0)
    return score

def lexicon_label(score, pos_thr=0.5, neg_thr=-0.5):
    if score >= pos_thr:
        return "positive"
    elif score <= neg_thr:
        return "negative"
    else:
        return "neutral"

df['lex_score'] = df['clean_tokens'].apply(lambda toks: lexicon_score(toks))
df['lex_label'] = df['lex_score'].apply(lexicon_label)
print("Lexicon distribution:\n", df['lex_label'].value_counts())
```

Untuk libray yang saya pakai yaitu lexicon.

```
Lexicon distribution:
 lex_label
neutral     436
negative    312
positive    285
Name: count, dtype: int64
```

Hasil akhir labellingnya seperti pada output, namun ini masih mengacu pada 2 keyword tadi.

6. Ekstraksi fitur dengan TF-IDF

```python
# 3. Feature extraction
# BoW and TF-IDF
tfidf = TfidfVectorizer(max_features=8000, ngram_range=(1,2))
X_tfidf = tfidf.fit_transform(df['clean_text'])

# optionally Word2Vec (train on our token lists)
w2v_model = Word2Vec(sentences=df['clean_tokens'].tolist(), vector_size=100, window=5, min_count=2, workers=2, epochs=10)
def doc_vector(tokens):
    vecs = []
    for t in tokens:
        if t in w2v_model.wv:
            vecs.append(w2v_model.wv[t])
    if not vecs:
        return np.zeros(w2v_model.vector_size)
    return np.mean(vecs, axis=0)
X_w2v = np.vstack([doc_vector(toks) for toks in df['clean_tokens']])
```

7. Split data dan training dengan beberapa model

```python
# 4. Supervised ML
df_ml = df.copy()
df_ml = df_ml[df_ml['lex_label'].isin(['positive','negative','neutral'])].reset_index(drop=True)
y = df_ml['lex_label'].map({'negative':0,'neutral':1,'positive':2}).values
X = tfidf.transform(df_ml['clean_text'])  # use TF-IDF features; or use X_w2v

# train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# models to try
models = {
    "MultinomialNB": MultinomialNB(),
    "LogisticRegression": LogisticRegression(max_iter=200),
    "LinearSVC": LinearSVC(max_iter=2000)
}

results = {}
for name, model in models.items():
    print("Training", name)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, target_names=['neg','neu','pos'])
    print(name, "accuracy:", acc)
    print(report)
    results[name] = {
        "model": model,
        "accuracy": acc,
        "report": report,
        "y_pred": y_pred
    }
best_name = max(results.keys(), key=lambda k: results[k]['accuracy'])
best_model = results[best_name]['model']
print("Best model:", best_name, results[best_name]['accuracy'])
```

Hasil:

I. Model multinominal

```
Training MultinomialNB
MultinomialNB accuracy: 0.7439613526570048
              precision    recall  f1-score   support

         neg       0.76      0.67      0.71        63
         neu       0.65      0.85      0.74        87
         pos       0.97      0.67      0.79        57

    accuracy                           0.74       207
   macro avg       0.80      0.73      0.75       207
weighted avg       0.78      0.74      0.75       207
```

II. Model logistic regression

```
Training LogisticRegression
LogisticRegression accuracy: 0.7729468599033816
              precision    recall  f1-score   support

         neg       0.86      0.67      0.75        63
         neu       0.67      0.93      0.78        87
         pos       1.00      0.65      0.79        57

    accuracy                           0.77       207
   macro avg       0.84      0.75      0.77       207
weighted avg       0.82      0.77      0.77       207
```

III. Model linearSVC

```
Training LinearSVC
LinearSVC accuracy: 0.8115942028985508
              precision    recall  f1-score   support

         neg       0.84      0.76      0.80        63
         neu       0.74      0.89      0.81        87
         pos       0.93      0.75      0.83        57

    accuracy                           0.81       207
   macro avg       0.84      0.80      0.81       207
weighted avg       0.82      0.81      0.81       207

Best model: LinearSVC 0.8115942028985508
```

8. Training dengan keseluruhan data menggunakan model terbaik

```python
# menggunakan model terbaik untuk prediksi ke seluruh dataset
X_all = tfidf.transform(df['clean_text'])
y_all_pred = best_model.predict(X_all)
inv_map = {0:'negative',1:'neutral',2:'positive'}
df['ml_label'] = [inv_map[int(i)] for i in y_all_pred]

#membandingkan lexicon dengan ml counts
print("Lexicon counts:\n", df.groupby('keyword')['lex_label'].value_counts().unstack(fill_value=0))
print("\nML counts:\n", df.groupby('keyword')['ml_label'].value_counts().unstack(fill_value=0))

#Menyimpan csv dengan label
OUT_CSV = "tweet_gojek_grab_labeled.csv"
df.to_csv(OUT_CSV, index=False)
print("Saved labeled CSV:", OUT_CSV)
```

Hasil:

```
Lexicon counts:
 lex_label  negative  neutral  positive
keyword
gojek           146      232       137
grab            166      204       148

ML counts:
 ml_label  negative  neutral  positive
keyword
gojek           140      246       129
grab            166      207       145
Saved labeled CSV: tweet_gojek_grab_labeled.csv
```

9. Visualisasi

```python
# 6. Evaluation helper
def evaluate_with_gold(y_true, y_pred, labels=['negative','neutral','positive']):
    print(classification_report(y_true, y_pred, labels=labels))
    cm = confusion_matrix(y_true, y_pred, labels=[0,1,2])
    sns.heatmap(cm, annot=True, fmt="d", xticklabels=labels, yticklabels=labels)
    plt.xlabel("pred")
    plt.ylabel("true")
    plt.show()
```

```python
#Hitung jumlah
count_df = df.groupby(['keyword','ml_label']).size().unstack(fill_value=0)

#Hitung persentase
percent_df = count_df.div(count_df.sum(axis=1), axis=0) * 100

print("COUNT PER SENTIMENT")
display(count_df)

print("\n PERCENTAGE PER SENTIMENT (%) ")
display(percent_df.round(2))

# --- Plot ---
plt.figure(figsize=(10,4))
ax = count_df.plot(kind='bar', stacked=False)

plt.title("Sentiment Counts per Keyword (ML Labels)")
plt.xlabel("Keyword")
plt.ylabel("Count")

# tampilkan nilai di atas bar
for p in ax.patches:
    ax.annotate(
        str(p.get_height()),
        (p.get_x() + p.get_width()/2, p.get_height()),
        ha='center', va='bottom', fontsize=8
    )

plt.tight_layout()
plt.show()
```

```python

# Plot Persentase
plt.figure(figsize=(10,4))
ax2 = percent_df.plot(kind='bar', stacked=False)

plt.title("Sentiment Percentage per Keyword (ML Labels)")
plt.xlabel("Keyword")
plt.ylabel("Percentage (%)")

# tampilkan angka persentasenya
for p in ax2.patches:
    ax2.annotate(
        f"{p.get_height():.1f}%",
        (p.get_x() + p.get_width()/2, p.get_height()),
        ha='center', va='bottom', fontsize=8
    )

plt.tight_layout()
plt.show()

```

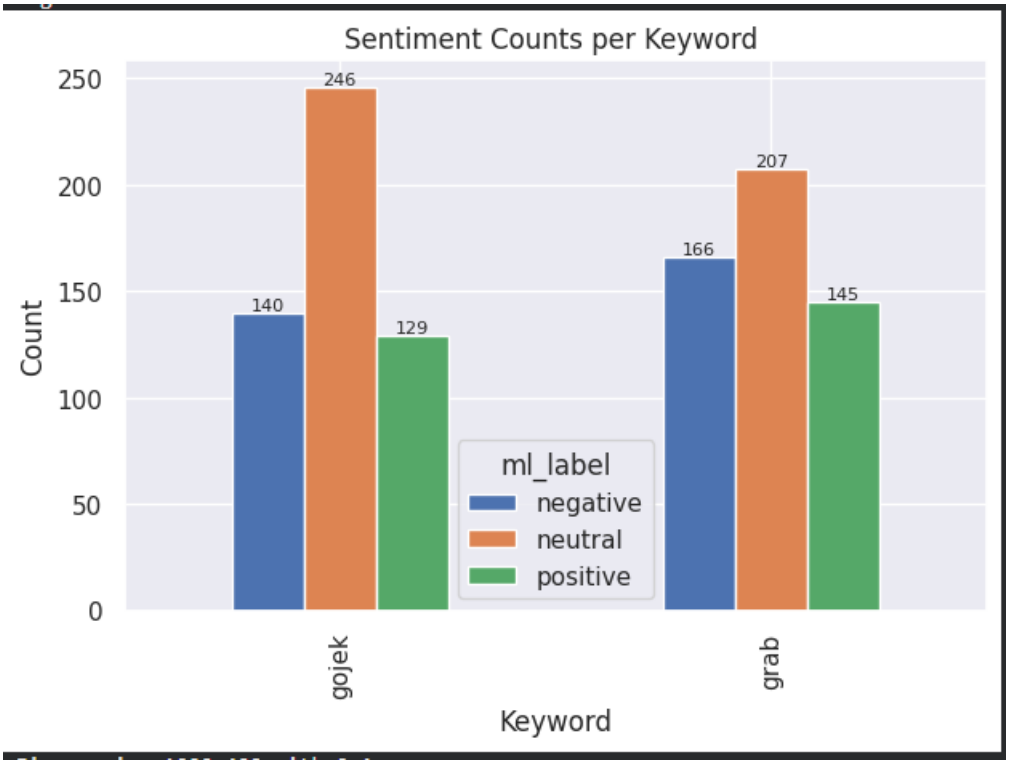
Hasil:

COUNT PER SENTIMENT

keyword	negative	neutral	positive
gojek	140	246	129
grab	166	207	145

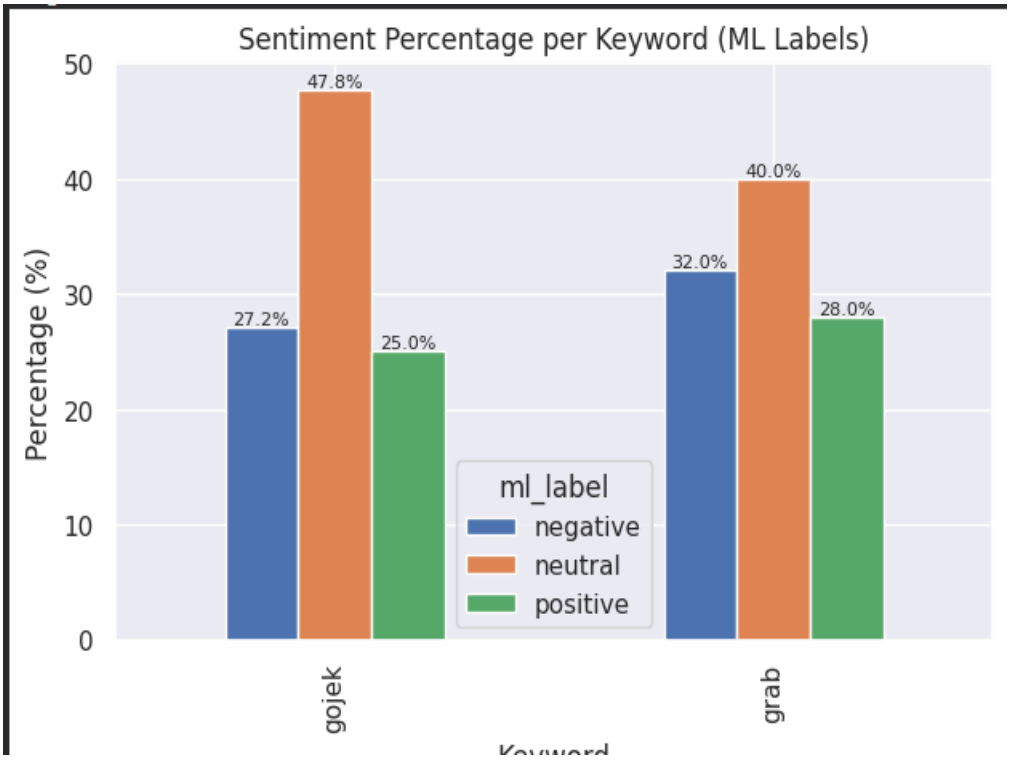
PERCENTAGE PER SENTIMENT (%)

keyword	negative	neutral	positive
gojek	27.18	47.77	25.05
grab	32.05	39.96	27.99



Gojek memiliki sentiment netral tertinggi, ini terjadi karena banyak kata-kata pada keyword gojek jarang muncul di wordcloud sehingga tidak terinput di dalam list labelling manual.

Sentiment negative dan positif dua-duanya paling banyak di sentiment grab.



Hasil prediksi antara model machine learning dan lexicon tidak terlalu signifikan walaupun menggunakan pendekatan yang berbeda, sentiment netral masih berada di keyword gojek.

Analisis hasil

No	Tweet asli	Setelah prepocessing	Prediksi model	Analisa manual	Cocok/tidak
1.	Lah ini gojek daerah mana nih bawa glock?	gojek daerah nih bawa glock	netral	netral	Tidak cocok
2.	bangun pagi ga sudi naik gojek juga ga sudi huf	bangun pagi nggak sudi gojek nggak sudi huf	negative	negative	cocok
3.	hallo!! aku masih open nokos ridi banyak yak, gojek, grab, wa, shopee, tokopedia yukk sung ke tele/ wa fastrespon yak✨ #zonauang	hallo open nokos ridi yak gojek grab wa shopee tokopedia yukk sung tele wa fastrespon yak zonauang	netral	netral	cocok
4.	@laalaafun @cassiocpeia @gtauadasiapadmn Rating akun kita as costumer ka Kalo di gojek driver bisa ngasih rating ke penumpang dan cuma bisa diliat driver	rating akun costumer ka gojek driver ngasih rating tumpang liat driver	netral	netral	Tidak cocok
5.	INI AMANG GOJEK NYA SPEK RACING KAHHH	amang gojek spek racing kahhh	netral	netral	cocok
6.	gaada dan belom ada sih yag grab	gaada bom sih yag grab	negativ	negativ	cocok

7.	for me to grab it sambil bilang ""istriku"	grab bilang istri	netral	netral	cocok
8.	Mari beli makanan enak sekaligus memajukan UMKM 😋😋😋😋https://t.co/ox1dSFWiZV	mari beli makan enak maju umkm	positiv	positiv	cocok
9.	@tebakaksyapa bisa order disini, ayam goreng soto jeniper by hangry <3https://t.co/Rh0adWiylL	order ayam goreng soto jeniper hangry lt3	Netral	netral	cocok
10.	knp grab di makassar kasar banget kah, i literally cried bcs sekasar itu tadi	knp grab makassar kasar banget kah literally cried bcs kasar	negativ	negativ	cocok

. Refleksi dan Diskusi

1. Apakah model lebih baik menggunakan data mentah atau yang sudah distemming?
   Model lebih baik menggunakan data yang sudah di stemming, karena data mentah lebih sulit di kenali oleh model
2. Apa yang menyebabkan model salah memprediksi tweet netral?
   Karena kata-kata pada tweet netral belum di definisikan menjadi sentiment positif atau negative.
3. Jika dataset diperbesar 10x lipat, bagaimana pengaruhnya terhadap akurasi model?
   Akurasi model belum tentu lebih baik karena tantangan terbesarnya ada di prepocessing
4. Bagaimana jika digunakan model *deep learning* seperti IndoBERT?
   Mungkin bisa jadi opsi yang lebih baik jika preprocessing benar-benar di lakukan dengan teliti dan cermat.

Link repository dan dataset Klik Di sini