LAPORAN TUGAS CRAWLING DAN PREPROCESSING DATA TEXT
MATA KULIAH TEXT MINING & NATURAL LANGUAGE PROCESSING

Tim Penyusun:

1. <5231811022> <Lathif Ramadhan>
2. <5231811029> <Andini Angel Meivita>
3. <5231811033> <Rama Panji Nararendra>
4. <5231811036> <Giffari Riyanda Pradithya>

PROGRAM STUDI SAINS DATA PROGRAM SARJANA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA
2025

1. **Hastag yang digunakan:** #ramadhan
2. **Tuliskan langkah-langkah crawling data text:**
   a. **Tuliskan/screenshotkan codingnya**

      Jawab:

      Link file kodenya:

      https://colab.research.google.com/drive/1wxDKwRGiu_pd9CDx6kNHN6I8RfUdxLrS#scrollTo=o2wMKSTWhoNW

```
# instal Node.js dan npm (Node Package Manager) di sistem
!curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
!sudo apt-get install -y nodejs

# nama file dari data yang berhasil dikumpulkan
data = "ramadan_1.csv"

# kata kunci dari data yang ingin dicari
search_keyword = "Ramadan"

# limit baris pencarian
limit = 200

# jalankan proses crawling tweet dengan bantuan tweet-harvest menggunakan Twitter API token.
!npx --yes tweet-harvest@2.6.1 -o "ramadan_1.csv" -s "{Ramadan}" -l {100} --token "----------------"
```

   b. **Tuliskan/screenshotkan hasil crawlingnya**

      Jawab:

c. Berapa jumlah record yang didapatkan

Jawab: 6543 baris

3. Tuliskan langkah-langkah preprocessing data text:

Link Colab Coding Cleaning:

https://colab.research.google.com/drive/15PBqlsHWNYC5eV-kj9fXwbNN9nC9whRJ?usp=sharing

Link Colab Case Folding, Tokenizing, Filtering, Stemming, dan Simpan Data Bersih:

https://colab.research.google.com/drive/1pBv_cBklF6W0eB2FAZdkWwRCXI3_dIbM?usp=sharing#scrollTo=eW2NnAO87pD8

a. Tuliskan/screenshotkan coding cleaning

Jawab:

```
import pandas as pd
import glob
import os

path = r'/content/' # use your path
from os import listdir
from os.path import isfile, join
onlyfiles = [f for f in listdir(path) if isfile(join(path, f))]
```

```
li = []
# print(all_files)S
for filename in onlyfiles:
            df   =   pd.read_csv(   r'/content/'   +   filename,
index_col=None, header=0)
    li.append(df)

frame = pd.concat(li, axis=0, ignore_index=True)

frame.to_csv("fix_combined_ramadan_real.csv", index=False)

df = pd.read_csv('/content/fix_combined_ramadan_real.csv')
df

df.drop_duplicates()
df.to_csv("/content/fix_combined_ramadan.csv", index=False)
```

Cleaning di preprocessing:

```
df.drop_duplicates(subset = 'Tweet', keep = 'first', inplace
= True)

# Memfilter datanya cuma bahasanya yang English
filt = (df_twit['lang'] == 'en')
df_twit = df_twit.loc[filt, :]
df_twit.reset_index(inplace=True)

#menghilangkan mention/user
def remove_pattern (tweet, pattern):
  r = re.findall(pattern, tweet)
  for i in r:
    tweet = re.sub(i,'', tweet)
  return tweet
df['remove_user']  =  np.vectorize(remove_pattern)(df['teks'],
"@[\w]*")
df['remove_user']
```

b. Tuliskan/screenshotkan Case folding

Jawab:

Case folding adalah proses mengubah semua teks menjadi huruf kecil agar lebih konsisten dalam analisis.

→ Di kode ini, case folding terjadi secara otomatis saat tokenizing

```
#tokenize tweets
        tokenizer    =    TweetTokenizer(preserve_case=True,
strip_handles=True, reduce_len=True)

  tweet_tokens = tokenizer.tokenize(tweet)
  # print(f"Word after tokenizer : {tweet_tokens}")
  tweets_clean = []

  for word in tweet_tokens:
       if (word not in stopwords_english and word not in
emoticons and word not in string.punctuation): # remove
punctuation
      # print(f"Word before stemming: {word}")
      stem_word = stemmer.stem(word) #stemming word
      # print(f"Word after stem: {stem_word}")
      tweets_clean.append(stem_word)
  return tweets_clean
```

Parameter preserve_case=False dalam TweetTokenizer akan secara otomatis mengubah teks menjadi huruf kecil.

Kedua kode ini walaupun fungsinya untuk tokenizing dan stemming tetapi secara tidak langsung mereka juga melakukan case folding, dimana untuk tokenizer preserve_case nya di set ke False untuk di case folding.

Untuk bagian kode yang melakukan case folding di tokenizing adalah

```
tokenizer        =        TweetTokenizer(preserve_case=True,
strip_handles=True, reduce_len=True)
```

```
tweet_tokens = tokenizer.tokenize(tweet)
```

Dan untuk bagian kode yang melakukan case folding di stemming adalah

```
stem_word = stemmer.stem(word)
```

c. Tuliskan/screenshotkan Tokenizing

Tokenizing adalah proses memecah teks menjadi kata-kata atau token.

Jawab:

→ Bagian kode yang melakukan tokenizing:

```
tokenizer = TweetTokenizer(preserve_case=False,
strip_handles=True, reduce_len=True)
tweet_tokens = tokenizer.tokenize(tweet)
```

- `tokenizer.tokenize(tweet)` akan memecah teks menjadi daftar kata-kata individu (token).
- `preserve_case=False` akan membuat kata-kata menjadi huruf kecil (bagian dari case folding juga).

d. Tuliskan/screenshotkan Filtering

Filtering adalah proses membersihkan teks dari karakter atau kata yang tidak diperlukan, seperti angka, URL, tanda baca, stopwords, dan emotikon.

Jawab:

→ Bagian kode yang melakukan filtering:

```
def tweet_clean(tweet):
  #remove angka
  tweet = re.sub('[0-9]+', '', tweet)
  # print(f"ss")
  # remove stock market tickers Like $GE
  tweet = re.sub(r'\$\w*', '', tweet)
  # remove old style retweet text "RT"
  tweet = re.sub(r'RT: [\s]+','', tweet)
```

```
  #remove hyperlinks
  tweet = re.sub(r'https?:\/\/.*[\r\n]*', '', tweet)
  #remove coma
  tweet = re.sub(r',','', tweet)
  # remove hashtags
  # only removing the hash # sign from the word
  tweet = re.sub(r'#','', tweet)
  #Happy Emoticons
  emoticons_happy = set([
   ':-)', ':)', ';)', ':0)', ':]', '3', ':c)', ':>', '=]',
':^)', ':-D', ':D' '8-D', '8D', '-3', '-3', ':-))', ":'-)"
'x-D', 'xD', 'X-D', 'XD' '>:P', 'x-p', 'xp', 'XP', ':-p',
'p', 'p', 'b', 'b', '>:)', '<3' ])
  #Sad Emoticons
  emoticons_sad = set([
   'L', ':-/', '>:/', 'S', '>:', '>:[','',':-(',':[', ':-||',
':-[', ':-<', '=\\', '=/', '>:(', ':(', '>.<', 'c', ':{',
'>:\\', ';(' ])
  #all emoticons (happy + sad)
  emoticons = emoticons_happy.union(emoticons_sad)

  #tokenize tweets
        tokenizer    =    TweetTokenizer(preserve_case=True,
strip_handles=True, reduce_len=True)

  tweet_tokens = tokenizer.tokenize(tweet)
  # print(f"Word after tokenizer : {tweet_tokens}")
  tweets_clean = []

  for word in tweet_tokens:
       if  (word  not  in  stopwords_english  and  word  not  in
emoticons  and  word  not  in  string.punctuation):  #  remove
punctuation
      # print(f"Word before stemming: {word}")
      stem_word = stemmer.stem(word) #stemming word
      # print(f"Word after stem: {stem_word}")
      tweets_clean.append(stem_word)
```

```
  return tweets_clean
df['tweet_clean']    =    df['remove_user'].apply(lambda    x:
tweet_clean(x))

#remove punct
def remove_punct(text):
   text = " ".join([char for char in text if char not in
string.punctuation])
  return text
df['Tweet']      =      df['tweet_clean'].apply(lambda      x:
remove_punct(x))
```

e. Tuliskan/screenshotkan Stemming

Stemming adalah proses mengubah kata menjadi bentuk dasarnya.

Jawab:

→ Bagian kode yang melakukan stemming menggunakan Sastrawi:

```
stem_word = stemmer.stem(word) #stemming word
```

f. Tuliskan/screenshotkan Simpan data text bersih

Jawab:

```
df.sort_values('Tweet', inplace = True)
df.drop(df.columns[[0,1]], axis = 1, inplace = True)
df.drop_duplicates(subset = 'Tweet', keep = 'first', inplace
= True)
df.to_csv('ramadan_clean_tweet.csv',          encoding='utf8',
index=False)
df.head(10)
```

4. Lampirkan file RAW hasil Crawling dan file bersih hasil preprocessing

Tugas Pembobotan kata menggunakan TF-IDF

1. Lakukan pelabelan pada data text yang sudah bersih, minimal 250 record

Kami pakai library VaderSentiment di python, jadi nanti hasil sentimennya dihitung otomatis.

```
# pip install pandas vaderSentiment
```

```
import pandas as pd

import numpy as np

from vaderSentiment.vaderSentiment import
SentimentIntensityAnalyzer
```

1. Load Dataset

```
df =
pd.read_csv('https://raw.githubusercontent.com/LatiefDataVisionary
/text-mining-and-natural-language-processing-college-task/refs/hea
ds/main/datasets/ramadan_clean_tweet.csv')

df.head()
```

	tweet_clean	Tweet
0	['abraj', 'al', 'bait', 'clock', 'tower', 'bea...	abraj al bait clock tower beams indicating com...
1	['accounts', 'recognised', 'ramadan', 'none', ...	accounts recognised ramadan none recognised be...
2	['admin', 'post', 'peaceful', 'ramadan', 'cele...	admin post peaceful ramadan celebrations east ...
3	['admin', 'post', 'ramadan', 'norway']	admin post ramadan norway
4	['admin', 'post', 'ramadan', 'usual', 'peacefu...	admin post ramadan usual peaceful start englan...

2. Inisialisasi Sentiment Analyzer

```
analyzer = SentimentIntensityAnalyzer()
```

3. Fungsi untuk Analisis Sentimen

```
def get_sentiment(text):
    scores = analyzer.polarity_scores(text)

    # Threshold penentuan sentimen
    if scores['compound'] >= 0.01:
        return 'positive'
    else:
        return 'negative'
```

4. Proses Analisis Sentimen untuk Setiap Tweet

```
df['sentiment'] = df['Tweet'].apply(get_sentiment)
```

5. Tambahkan Skor Sentimen

```
def get_sentiment_scores(text):
    return analyzer.polarity_scores(text)
df['sentiment_scores'] = df['Tweet'].apply(get_sentiment_scores)
```

6. Pisahkan Skor ke Kolom Terpisah

```
df['neg'] = df['sentiment_scores'].apply(lambda x: x['neg'])
df['neu'] = df['sentiment_scores'].apply(lambda x: x['neu'])
df['pos'] = df['sentiment_scores'].apply(lambda x: x['pos'])
df['compound'] = df['sentiment_scores'].apply(lambda x: x['compound'])
df.head()
```

	tweet_clean	Tweet	sentiment	sentiment_scores	neg	neu	pos	compound
0	['abraj', 'al', 'bait', 'clock', 'tower', 'bea...	abraj al bait clock tower beams indicating com...	negative	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	0.000	1.000	0.000	0.0000
1	['accounts', 'recognised', 'ramadan', 'none', ...	accounts recognised ramadan none recognised be...	negative	{'neg': 0.147, 'neu': 0.853, 'pos': 0.0, 'comp...	0.147	0.853	0.000	-0.4767
2	['admin', 'post', 'peaceful', 'ramadan', 'cele...	admin post peaceful ramadan celebrations east ...	positive	{'neg': 0.0, 'neu': 0.714, 'pos': 0.286, 'comp...	0.000	0.714	0.286	0.4939
3	['admin', 'post', 'ramadan', 'norway']	admin post ramadan norway	negative	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	0.000	1.000	0.000	0.0000
4	['admin', 'post', 'ramadan', 'usual', 'peacefu...	admin post ramadan usual peaceful start englan...	positive	{'neg': 0.0, 'neu': 0.775, 'pos': 0.225, 'comp...	0.000	0.775	0.225	0.4939

7. Hapus Kolom Tidak Diperlukan

```
df.drop('sentiment_scores', axis=1, inplace=True)
```

8. Tampilkan Contoh Hasil

```
print(df[['Tweet', 'sentiment', 'compound']].head())
```

```
                                               Tweet sentiment  compound
0  abraj al bait clock tower beams indicating com...  negative    0.0000
1  accounts recognised ramadan none recognised be...  negative   -0.4767
2  admin post peaceful ramadan celebrations east ...  positive    0.4939
3                          admin post ramadan norway  negative    0.0000
4  admin post ramadan usual peaceful start englan...  positive    0.4939
```

```
df.to_csv('ramadan_labeled_sentiment.csv', index=False)
```

```
df = pd.read_csv('ramadan_labeled_sentiment.csv',
usecols=['tweet_clean', 'Tweet', 'sentiment'])
```

```
df.head()
```

	tweet_clean	Tweet	sentiment
0	['abraj', 'al', 'bait', 'clock', 'tower', 'bea...	abraj al bait clock tower beams indicating com...	negative
1	['accounts', 'recognised', 'ramadan', 'none', ...	accounts recognised ramadan none recognised be...	negative
2	['admin', 'post', 'peaceful', 'ramadan', 'cele...	admin post peaceful ramadan celebrations east ...	positive
3	['admin', 'post', 'ramadan', 'norway']	admin post ramadan norway	negative
4	['admin', 'post', 'ramadan', 'usual', 'peacefu...	admin post ramadan usual peaceful start englan...	positive

2. Lakukan pembobotan data text menggunakan TF-IDF yang sudah diberi label

Dalam proyek ini, kita melakukan **analisis sentimen** terhadap tweet bertema **Ramadhan** menggunakan **Natural Language Processing (NLP)**. Fokus utama adalah memproses teks, menerapkan **TF-IDF (Term Frequency - Inverse Document Frequency)** untuk ekstraksi fitur, dan menganalisis kata-kata yang paling berpengaruh dalam dataset.

a. Import Library dan Data

```
[ ] import pandas as pd
    import numpy as np

    dm = pd.read_csv("/content/ramadan_labeled_sentiment.csv", usecols=["tweet_clean", "sentiment"])
    dm.columns = ["tweet_clean", "sentiment"]
    dm.head(10)
```

	tweet_clean	sentiment
0	['abraj', 'al', 'bait', 'clock', 'tower', 'bea...	neutral
1	['accounts', 'recognised', 'ramadan', 'none', ...	negative
2	['admin', 'post', 'peaceful', 'ramadan', 'cele...	positive
3	['admin', 'post', 'ramadan', 'norway']	neutral
4	['admin', 'post', 'ramadan', 'usual', 'peacefu...	positive
5	['alfas', 'inside', 'mega', 'chicken', 'aftern...	neutral
6	['alhamdulillah', 'every', 'lesson', 'every', ...	positive
7	['allah', 'forgive', 'us', 'sins', 'past', 'fu...	positive
8	['allah', 'help', 'brothers', 'gaza', 'lift', ...	positive
9	['allah', 'help', 'us', 'al-aqsa', 'prisons', ...	negative

b. Menggabungkan string list menjadi string

```
import ast

def join_text_list(texts):
    texts = ast.literal_eval(texts)
    return ' '.join([text for text in texts])
dm["tweet_join"] = dm["tweet_clean"].apply(join_text_list)

dm["tweet_join"].head()
```

	tweet_join
0	abraj al bait clock tower beams indicating com...
1	accounts recognised ramadan none recognised be...
2	admin post peaceful ramadan celebrations east ...
3	admin post ramadan norway
4	admin post ramadan usual peaceful start englan...

dtype: object

c. Menghitung TF-IDF

Menghitung TF-IDF menggunakan TfidfVectorizer Untuk menghitung TF-IDF menggunakan Scikit-Learn, dapat dilakukan dengan cara berikut,

```
from sklearn.feature_extraction.text import TfidfVectorizer

# banyaknya term yang akan digunakan,
# di pilih berdasarkan top max_features
# yang diurutkan berdasarkan term frequency seluruh corpus
max_features = 10000

# Feature Engineering
print ("------- TF-IDF on Tweet data -------")

tf_idf = TfidfVectorizer(max_features=max_features, binary=True)
tf_idf.fit(dm["tweet_join"])  # Melakukan fitting terlebih dahulu
tfidf_mat = tf_idf.transform(dm["tweet_join"]).toarray()

print("TF-IDF ", type(tfidf_mat), tfidf_mat.shape)
```

```
------- TF-IDF on Tweet data -------
TF-IDF  <class 'numpy.ndarray'> (836, 3524)
```

Hasil dari tfidf menggunakan dataframe

d. Hasil Matriks TF-IDF

Hasil dari tfidf menggunakan dataframe

```
[ ]  import pandas as pd

     # Mengambil nama fitur (term)
     feature_names = tf_idf.get_feature_names_out()

     # Membuat DataFrame
     df_tfidf = pd.DataFrame(tfidf_mat, columns=feature_names)

     # Menampilkan DataFrame
     display(df_tfidf.head(20))
```

	aa	aameen	aamiin	aaron	aas	abandoned	abandoning	abducted	abdul	abdullah	...	zahra	zakat	zarafshan	zardari	zaria	zayn	zazzau	zealand	zelensky	zumrat
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

20 rows × 3524 columns

melihat list top 50 term yang meiliki TF-IDF terbesar

Baris mewakili tiap dokumen/data, sedangkan kolom menunjukkan semua kata yang ada di dalam korpus. Jika sebuah kata muncul dalam suatu dokumen, maka pada perpotongan baris dan kolom tersebut akan terisi nilai TF-IDF-nya.

e. Menampilkan ke dalam list

50 term teratas dengan total TF-IDF tertinggi.

```
terms = tf_idf.get_feature_names_out()

# sum tfidf frequency of each term through documents
sums = tfidf_mat.sum(axis=0)

# connecting term to its sums frequency
data = []
for col, term in enumerate(terms):
    data.append((term, sums[col] ))

ranking = pd.DataFrame(data, columns=['term','rank'])
ranking.sort_values('rank', ascending=False)
```

	term	rank
2480	ramadan	54.951769
2000	month	20.407185
2021	mubarak	16.447736
1913	may	16.075416
109	allah	15.525257
...	...	...
2681	rx	0.176130
1828	location	0.176130
1687	keyless	0.176130
93	akpabio	0.176130
3441	wike	0.176130

3524 rows × 2 columns

```
data = pd.read_csv("/content/ramadan_clean_tweet.csv")
data.to_excel("ramadan_clean_tweet_excel.xlsx", index=False)
```