MEMANFAATKAN API SWAGGER UNTUK ANALISA SENTIMENT PENGGUNA TWITTER

JULIUS RONALD CHRISTANTO

DAFTAR ISI

PENDAHULUAN

- Indonesia merupakan salah satu negara dengan tingkat penggunaan internet yang cukup tinggi di dunia, serta popularitas sosial media yang signifikan, seperti Aplikasi Twitter. Meskipun Aplikasi Twitter tidak sepopuler Aplikasi Facebook atau Aplikasi Instagram, platform ini lebih banyak digunakan untuk berbagi pemikiran, opini, dan informasi singkat serta aktif untuk memberikan komentar. Namun, terdapat juga opini hoax dan komentar yang tidak pantas untuk didengar.

RUMUSAN MASALAH

- Berdasarkan pendahuluan pada slide sebelumnya, terlihat bahwa masyarakat di Indonesia khususnya pengguna Aplikasi Twitter sangat aktif dalam memberikan komentar pada postingan sesorang dengan berbagai jenis karakteristik penulisan dan isi komentar yang berbeda. Namun, dalam komentar yang dikirimkan oleh para pengguna tersebut harus dilakukan pengenalan lebih lanjut terkait pola komentar dan sentiment dari komentar tersebut, apakah masuk ke dalam ranah positif atau negatif.

TUJUAN PENELITIAN

- Membahas dari Rumusan Masalah, adanya Tujuan Penelitian ini untuk melakukan analisis dan penyaringan rata – rata jumlah kata dan sentiment yang diketik oleh para pengguna Aplikasi Twitter, terutama kata – kata yang terdapat sentiment positif atau negatif. Penelitian ini bertujuan untuk memahami pola komentar yang berkaitan dengan sentiment di Aplikasi Twitter.
- Diharapkan hasil Analisis ini dapat berguna bagi berbagai pihak, baik untuk pemahaman opini publik maupun pengembangan komunikasi di masa mendatang.

METODE PENELITIAN

I. METODE CLEANSING DATA

1. Adapun Function yang digunakan untuk membersihkan kata – kata / kalimat yang tidak dibutuhkan sbb :

a. Function Regex

```
# Definisikan Clean Text
def clean_text (twitter_tweets):

    twitter_tweets = re.sub('USER','',twitter_tweets) = menghapus kata user
    twitter_tweets = re.sub('RT','',twitter_tweets) = menghapus kata RT (Retweet)
    twitter_tweets = re.sub('http\S+', '', twitter_tweets) = menghapus kata yang berhubungan dengan URL
    twitter_tweets = re.sub(r'[^a-zA-Z0-9]', ' ', twitter_tweets) = mengganti kata non alpanumberic dengan spasi
    twitter_tweets = re.sub(r'x[a-f0-9a-fA-F]{2}', '', twitter_tweets) = menghpaus unique code jenis X0a, Xbb, dan lainnya
    twitter_tweets = re.sub(r'\b\w\b','', twitter_tweets) = menghapus 1 karakter
    twitter_tweets = re.sub(r'\s+', ' ', twitter_tweets) = menghapus double spacing
    twitter_tweets = re.sub(r'^\d+\s*', '', twitter_tweets) = menghapus angka awal dari awal kalimat
    twitter_tweets = re.sub(r'[^\x08-\x7f]',r'',twitter_tweets) = menghapus kata unique code
    twitter_tweets = re.sub('\?','',twitter_tweets) = menghapus karakter tanda tanya
    twitter_tweets = re.sub('/na','',twitter_tweets) = menghapus kalimat yang terdapat kata ‘/na’
    twitter_tweets = re.sub(r'www\.[^ ]+', '',twitter_tweets) = menghapus URL
    twitter_tweets = twitter_tweets.strip() = menghapus spasi tambahan dari awal dan akhir
    twitter_tweets = twitter_tweets.lower() = mengubah semua huruf menjadi huruf kecil

    return twitter_tweets
```


Selanjutnya

METODE PENELITIAN

b. Function Replace Kamus Alay

```
# Fungsi Cleansing Data
def normalize_alay(twitter_tweets):
    path_kamus_alay = '/home/jrjmt/Github_Cleansingdata/24001074-18-jrc-cleansingdata-
gold/Challenge_Gold/docs/new_kamusalay.csv'
    df_kamus_alay = pd.read_csv(path_kamus_alay, encoding='iso-8859-1')

    #iloc = Index location
    #mengambil data pada kolom tertentu dan baris tertentu saja dari keseluruhan data
    alay_dict_map = dict(zip(df_kamus_alay.iloc[:, 0], df_kamus_alay.iloc[:, 1]))
    alay_dict_map
    return ' '.join([alay_dict_map[word] if word in alay_dict_map else word for word in twitter_tweets.split(' ')])
```


Selanjutnya

METODE PENELITIAN

c. Function Input CSV

```
@swag_from("docs/text_processing_file.yml", methods=['POST'])
@app.route('/text-processing-file', methods=['POST'])
def text_processing_file():

    # io.BytesIO = Konversikan kedalam bytes menggunakan ISO-8859-1.
    file = request.files['file']
    contents = file.read()
    df = pd.read_csv(io.BytesIO(contents), encoding='ISO-8859-1')

    # Definsikan Keluaran Direktori dan File Path
    output_directory = '/home/jrjmt/Github_Cleansingdata/24001074-18-jrc-cleansingdata-gold/Challenge_Gold/docs/Result_Data'
    output_file_path = os.path.join(output_directory, 'data_mentah_tweet.csv')

    # Simpan Dataframe ke Format .CSV
    df.to_csv(output_file_path, index=False)

    # Setup Path
    Path_file = '/home/jrjmt/Github_Cleansingdata/24001074-18-jrc-cleansingdata-
gold/Challenge_Gold/docs/Result_Data/data_mentah_tweet.csv'

    data_tweets = pd.read_csv(Path_file, encoding='ISO-8859-1')

    # Konversikan Data Frame ke SQL
    conn = sqlite3.connect ('challenge_data_tweet')
    data_tweets.to_sql('data_tweets', conn, if_exists='append', index = False)
```


Selanjutnya

METODE PENELITIAN

d. Function Cleansing dan membersihkan kata alay

```
# Cleansing
    data_tweets['cleansing_tweets'] = data_tweets['Tweet'].apply(clean_text)

    # Membersihkan kata-kata alay dengan new_kamus_alay
    data_tweets['cleansing_tweets'] = data_tweets['cleansing_tweets'].apply(normalize_alay)

    data_tweets['cleansing_tweets'] = data_tweets['cleansing_tweets'].apply(lambda x: x.lower() if isinstance(x,
str) else x)

    # Konversikan cleansing data ke Format .CSV
    cleansed_file_path = os.path.join(output_directory, 'hasil_cleansing_tweet.csv')
    data_tweets.to_csv(cleansed_file_path, index=False)

    # Menampilkan hasil cleansing data ke API
    result_cleansing_dataraw = data_tweets['cleansing_tweets'].tolist()
```

METODE PENELITIAN

d. Function Cleansing dan membersihkan kata alay

```
# Cleansing
    data_tweets['cleansing_tweets'] = data_tweets['Tweet'].apply(clean_text)

    # Membersihkan kata-kata alay dengan new_kamus_alay
    data_tweets['cleansing_tweets'] = data_tweets['cleansing_tweets'].apply(normalize_alay)

    data_tweets['cleansing_tweets'] = data_tweets['cleansing_tweets'].apply(lambda x: x.lower() if isinstance(x,
str) else x)

    # Konversikan cleansing data ke Format .CSV
    cleansed_file_path = os.path.join(output_directory, 'hasil_cleansing_tweet.csv')
    data_tweets.to_csv(cleansed_file_path, index=False)

    # Menampilkan hasil cleansing data ke API
    result_cleansing_dataraw = data_tweets['cleansing_tweets'].tolist()
```

METODE VISUAL

Menampilkan chart pie presentasi perbandingan antara kata abusive dengan yang tidak mengandung abusive, dengan kode sebagai berikut:

```
#Grafik yang menampilkan perbandingan presentasi tweets yang mengandung Abusive
data_tweets['Abusive'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

Hasil dari kode diatas dapat dilihat dalam **slide Hasil** dan **Gambar 1.** chart pie menampilkan perbandingan presentasi tweets yang mengandung abusive sebanyak 38,3 % dengan tweet yang tidak mengandung abusive sebanyak 61,7 %

Menampilkan chart pie perbandingan presentasi tweets yang mengandung hate speech dengan yang tidak mengandung hate speech sebagai berikut :

```
#Grafik yang menampilkan perbandingan presentasi tweets yang mengandung hatespeech(HS)
data_tweets['HS'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

Hasil dari kode diatas dapat dilihat dalam **slide Hasil** dan **Gambar 2.** chart pie menampilkan menampilkan perbandingan presentasi tweets yang mengandung hate speech sebanyak 42,2 % dengan tweet yang tidak mengandung hate speech sebanyak 57,8%

Selanjutnya

METODE VISUAL

Menampilkan kata yang paling banyak muncul dalam tweet menggunakan wordcloud scrip sebagai berikut :

```
#import Wordcloud
from wordcloud import WordCloud

# Tampilkan data yang sering muncul dan sudah dicleansing di kolom cleansing_tweets , kemudian
gabungkan semuanya menjadi satu string
text = ' '.join(str(tweet)for tweet in data_tweets['cleansing_tweets'])

# Buat wordcloud
wordcloud = WordCloud().generate(text)

# Tampilkan plot
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Hasil dari code diatas dapat dilihat di slide hasil **Gambar 3**. wordcloud menampilkan kata yang lebih sering muncul setelah dilakukan cleansing data. Kata yang lebih banyak muncul yaitu yang, dan, itu, tidak, di, kamu, kalau dan seterusnya hingga kata yang jarang muncul.

Selanjutnya

METODE VISUAL

Menampilkan kata yang paling banyak muncul dalam tweet menggunakan wordcloud scrip sebagai berikut :

```
# Tampilkan data yang sering muncul dan belum dicleansing di kolom 'Tweet', kemudian gabungkan semuanya menjadi satu string
text = ' '.join(str(tweet)for tweet in data_tweets['Tweet'])

# Buat wordcloud
wordcloud = WordCloud().generate(text)

# Tampilkan plot
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

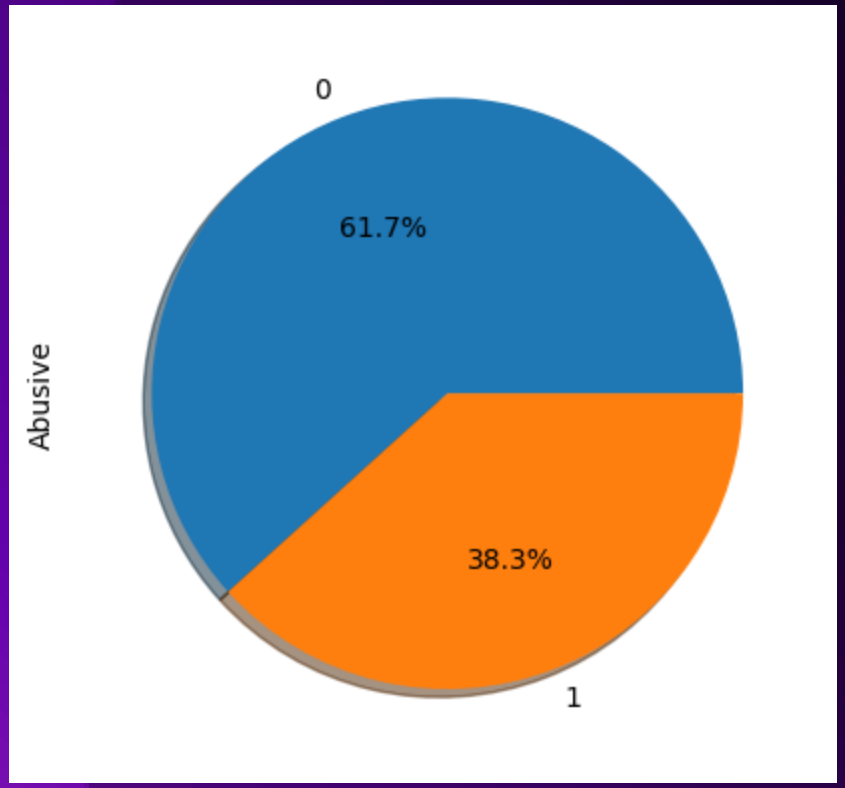
Hasil dari code diatas dapat dilihat di slide hasil **Gambar 4.** wordcloud menampilkan kata yang lebih sering muncul sebelum dilakukan cleansing data. Kata yang lebih banyak muncul yaitu USER, Unique Code, dan, yang, dan seterusnya hingga kata yang jarang muncul.

Back to Daftar Isi

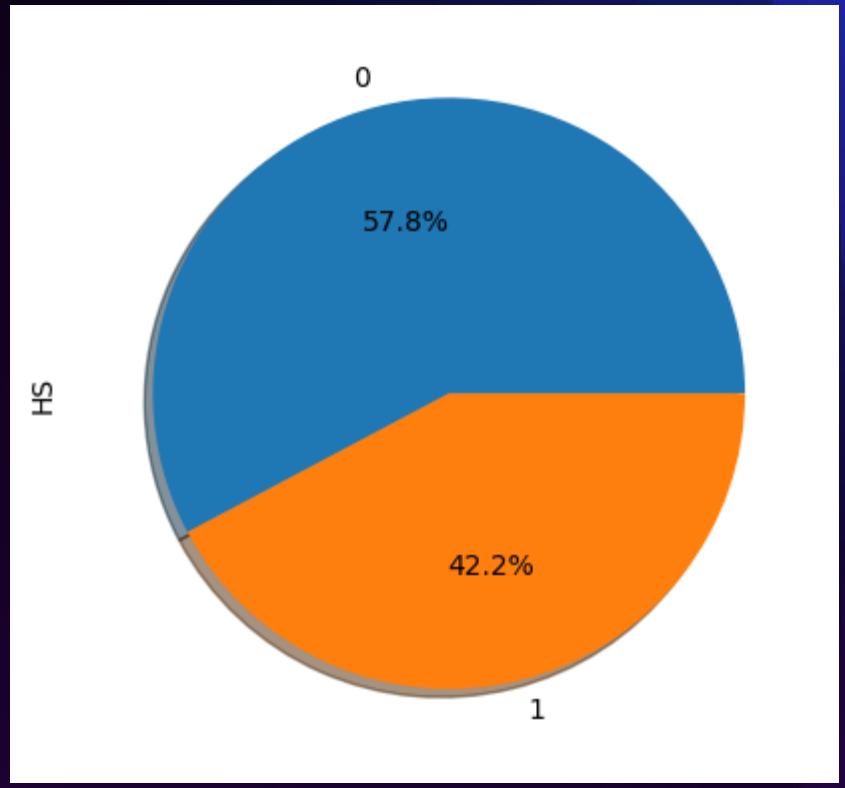
HASIL DAN KESIMPULAN

1. Adapun Hasil dari Cleansing Data sbb :

   a. Pie Chart



Gambar 1. Grafik diatas menampilkan perbandingan presentasi tweets yang mengandung abusive sebanyak 38,3 % dengan tweet yang tidak mengandung abusive sebanyak 61,7 %



Gambar 2. Grafik diatas menampilkan perbandingan presentasi tweets yang mengandung hate speech sebanyak 42,2 % dengan tweet yang tidak mengandung hate speech sebanyak 57,8%

Selanjutnya

HASIL DAN KESIMPULAN

b. Word Cloud

Gambar 3. Grafik diatas menampilkan kata yang lebih sering muncul setelah dilakukan cleansing data. Kata yang lebih banyak muncul yaitu yang, dan, itu, tidak, di, kamu, kalau dan seterusnya hingga kata yang jarang muncul.

Gambar 4. Grafik diatas menampilkan kata yang lebih sering muncul sebelum dilakukan cleansing data. Kata yang lebih banyak muncul yaitu USER, Unique Code, dan, yang, dan seterusnya hingga kata yang jarang muncul.

Back to Daftar Isi