Analisis Sentimen Film Dune Part 2 di IMDB dengan Model LSTM

1. Scraping

Tahap pertama yaitu perlu melakukan scraping pada film Dune part 2 di IMDB.

```
[ ] file_path = "/content/Dune_ Part Two (2024)- IMDb.html"
    with open(file_path, 'r', encoding='utf-8') as file:
        html_content = file.read()
    soup = BeautifulSoup(html_content, 'html.parser')
```

```
[ ] html = soup.find_all('div', {"class":"review-container"})
    print(len(html))

    1725
```

```
    first_review = html[0]
    first_review.a.text

    ' Long live the fighters\n'
```

```
[ ] reviews = []

    for row in html:
      # review = row.find_all("div", {"class" : "text show-more__control"})
      review = row.a.text
      reviews.append(review)

    reviews
```

- Tahap ini saya menggunakan file html yang sudah saya save lalu menggunakan BeautifulSoup untuk memparsing dari konten HTML yang telah dimasukkan sebelumnya. Setelah berhasil di-parse, kode mencari semua elemen div dengan kelas "review-container" menggunakan metode find_all dari objek BeautifulSoup.
- Setelaj itu, panjang dari list html yang berisi elemen-elemen review-container dicetak untuk mengetahui berapa banyak ulasan yang berhasil diambil dari halaman HTML.
- Kemudian, kode iterasi melalui setiap elemen dalam list html dan mencari elemen anchor (a) di dalamnya untuk mendapatkan teks dari setiap ulasan. Teks ulasan yang ditemukan kemudian ditambahkan ke dalam list reviews.List reviews berisi teks dari semua ulasan yang berhasil diambil dari halaman IMDb tersebut.

2. Membuat DataFrame

```
[ ] df = pd.DataFrame({'Review' : reviews})
    df.to_csv('IMDB_Reviews_Dune.csv', index=False)
```

```
    data = pd.read_csv('IMDB_Reviews_Dune.csv')
    data
```

	Review
0	Long live the fighters\n
1	This is what Hollywood needs!\n
2	Ladies and gentleman.. the PEAK of filmmaking...
3	Arrakis is Real. Believe Me I've Seen It.\n
4	Visual masterpiece, questionable narrative\n

Hasil dari ulasan film Dune part 2 tadi akan dibuat ke dalam bentuk DataFrame. Setelah DataFrame dibuat, saya menyimpanya ke dalam file CSV dengan nama file ‘IMDB_Reviews_Dune’.

Kemudian, karena saya akan menggunakan file tersebut maka menggunakan ’ pd.read_csv()’ untuk membaa file CSV nya dan menyimpannya dalam variabel ’data’.

3. Pra Pemrosesan

Tahap ini dilakukan untuk membersihkan dan mengubah data menjadi bentuk yang bisa di analisis oleh model.

```
data.loc[:, 'Review'] = data['Review'].str.replace('\n', '')
```

```
data.head()
```

	Review
0	Long live the fighters
1	This is what Hollywood needs!
2	Ladies and gentleman.. the PEAK of filmmaking!!
3	Arrakis is Real. Believe Me I've Seen It.
4	Visual masterpiece, questionable narrative

- Tahap ini mengganti setiap baris dalam kolom ’Review’ dari DataFrame data dengan versi yang tidak mengandung karakter baris baru ('\n').

```
def preprocess_data(data):
    # Case Folding
    data['Review'] = data['Review'].str.lower()

    # Remove Number & Punctuation
    data['Review'] = data['Review'].apply(lambda x: re.sub(r'[^\w\s]', '', x))
    data['Review'] = data['Review'].apply(lambda x: re.sub(r'\d+', '', x))

    # Stopword Removal
    stop_words = set(stopwords.words('english'))
    data['Review'] = data['Review'].apply(lambda x: ' '.join([word for word in x.split() if word.lower() not in stop

    # Tokenizing (BERT-Tokenizer)
    tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
    data['Tokenized'] = data['Review'].apply(lambda x: tokenizer.tokenize(x))

    return data

data = preprocess_data(data)
```

```
[89] data
```

	Review	Sentiment	Tokenized
0	long live fighters	Positive	[long, live, fighters]
1	hollywood needs	Neutral	[hollywood, needs]
2	ladies gentleman peak filmmaking	Neutral	[ladies, gentleman, peak, filmmaking]

- Case Folding: Mengubah semua huruf dalam teks menjadi huruf kecil.
- Remove Number & Punctuation: Menghapus angka dan tanda baca dari teks.
- Stopword Removal: Menghapus kata-kata pengisi (stop words) dari teks.
- Membuat Objek Tokenizer: BertTokenizer.from_pretrained('bert-base-uncased') membuat objek tokenizer yang akan digunakan untuk tokenisasi.

4. Analisis Sentimen

```
[91] def analyze_sentiment(text):
        blob = TextBlob(text)
        sentiment = blob.sentiment.polarity
        if sentiment > 0:
            return 'Positive'
        elif sentiment < 0:
            return 'Negative'
        else:
            return 'Neutral'

    data['Sentiment'] = data['Review'].apply(analyze_sentiment)
```

```
[92] data
```

	Review	Sentiment	Tokenized
0	long live fighters	Positive	[long, live, fighters]
1	hollywood needs	Neutral	[hollywood, needs]
2	ladies gentleman peak filmmaking	Neutral	[ladies, gentleman, peak, filmmaking]

- Fungsi analyze_sentiment di atas menggunakan TextBlob untuk melakukan analisis sentimen pada teks yang diberikan.
- Lalu menghitung sentimen polaritas menggunakan TextBlob, TextBlob memiliki metode sentiment, yang menghitung polaritas teks, yaitu seberapa positif atau negatifnya teks tersebut. Nilai polaritas berkisar dari -1 (sangat negatif) hingga 1 (sangat positif).

5. Membuat label X dan y

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
X = data['Tokenized'].apply(lambda x: tokenizer.convert_tokens_to_ids(x))

X = pad_sequences(X)
y = data['Sentiment']
y_numeric = y.map({'Positive': 0, 'Negative': 1, 'Neutral': 2})
```

- Pertama yaitu konversi Token ke ID dimana data['Tokenized'].apply(lambda x: tokenizer.convert_tokens_to_ids(x)) mengaplikasikan metode convert_tokens_to_ids dari objek tokenizer untuk setiap daftar token dalam kolom 'Tokenized' DataFrame data. Ini mengonversi setiap token menjadi ID token yang sesuai.
- Padding Sequences: pad_sequences(X) dilakukan untuk memastikan semua token sequence memiliki panjang yang sama.
- Mengambil Label Sentimen: Label sentimen dari kolom 'Sentiment' DataFrame data diambil ke dalam variabel y. Kemudian, label sentimen ini diubah menjadi numerik menggunakan mapping label 'Positive' ke 0, 'Negative' ke 1, dan 'Neutral' ke 2.

6. Model LSTM

```
[95] X_train, X_test, y_train, y_test = train_test_split(X, y_numeric, test_size=0.2, random_state=42)
```

```
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.vocab), output_dim=64))
model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(3, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Training the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1)

# Evaluating the model
y_pred_prob = model.predict(X_test)
y_pred = np.argmax(y_pred_prob, axis=1)

# Compute metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average=None)
recall = recall_score(y_test, y_pred, average=None)
f1 = f1_score(y_test, y_pred, average=None)

print(classification_report(y_test, y_pred, target_names=['Negative', 'Neutral', 'Positive']))
```

Model LSTM

- Model dimulai dengan Sequential, yang merupakan model neural network.
- Lapisan pertama adalah Embedding layer, yang bertanggung jawab untuk mengonversi input (token-ID) menjadi vektor embedding.

- Lapisan kedua adalah LSTM (Long Short-Term Memory) layer, yang merupakan jenis lapisan rekursif yang mampu mengingat informasi jangka panjang dan jangka pendek dari urutan data.
- Lapisan ketiga adalah Dense layer dengan fungsi aktivasi softmax, yang menghasilkan probabilitas untuk setiap kelas sentimen.

7. Sanity Check

```
MAX_TOKEN_LEN = 256
def preprocess_text(text, tokenizer, max_token_len):
    tokens = tokenizer.encode(text, max_length=max_token_len, truncation=True)
    return np.array(tokens, dtype='int32').reshape(1, -1)

def predict(text):
    text_tokens = preprocess_text(text, tokenizer, MAX_TOKEN_LEN)

    y_pred_prob = model.predict(text_tokens)
    y_pred = np.argmax(y_pred_prob, axis=1)[0]

    labels_reverse = {0: 'Negative', 1: 'Neutral', 2: 'Positive'}
    sentiment_label = labels_reverse[y_pred]

    return sentiment_label
```

```
text = "amazing graphics special effects movie"
predicted_sentiment = predict(text)
print("Predicted Sentiment:", predicted_sentiment)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 250ms/step
Predicted Sentiment: Negative
```

- Disini menggunakan dua fungsi yaitu fungsi preprocess_text dan predict.
- preprocess_text: Fungsi ini bertanggung jawab untuk melakukan pra-pemrosesan teks yang diberikan sebelum melakukan prediksi sentimen menggunakan model.
- predict: Fungsi ini bertanggung jawab untuk melakukan prediksi sentimen berdasarkan teks yang diberikan. Dengan demikian, fungsi predict dapat digunakan untuk memprediksi sentimen dari teks yang diberikan.

Latar belakang

Jaringan saraf berulang (RNN) adalah bentuk jaringan Syaraf Tiruan yang dapat mengingat rangkaian pola masukan dengan panjang sembarang dengan menangkap koneksi antar tipe data sekuensial. Namun, karena kegagalan gradien stokastik, RNN tidak dapat mendeteksi ketergantungan jangka panjang dalam rangkaian yang panjang. Beberapa model RNN baru, terutama LSTM, diusulkan untuk mengatasi masalah ini. Jaringan LSTM adalah ekstensi RNN yang dirancang untuk mempelajari data sekuensial (temporal) dan koneksi jangka panjangnya dengan lebih tepat daripada RNN standar. Model ini biasanya digunakan dalam aplikasi pembelajaran mendalam seperti perkiraan stok, pengenalan suara, pemrosesan bahasa alami, dll.

Menurut beberapa penelitian LSTM paling populer untuk menangani klasifikasi sentimen. LSTM diusulkan oleh Hoch Reiter dan Schmid Huber pada tahun 1997 dan disempurnakan serta dipopulerkan oleh banyak orang dalam karya berikutnya. Mereka bekerja dengan sangat baik pada berbagai jenis masalah dan sekarang banyak digunakan. LSTM secara eksplisit dirancang untuk mengabaikan masalah ketergantungan jangka panjang. Mengingat informasi untuk waktu yang lama pada dasarnya merupakan perilaku default mereka, bukan sesuatu yang sulit mereka pelajari. Semua jaringan saraf berulang berbentuk rantai modul jaringan saraf berulang. Pada level RNN, modul berulang ini memiliki struktur yang sangat sederhana, seperti lapisan tanh tunggal.

Hasil evaluasi

Ini merupakan classification report yang mencakup beberapa metrik evaluasi yang penting untuk mengevaluasi kinerja model klasifikasi, khususnya untuk tugas klasifikasi sentimen teks.

	precision	recall	f1-score	support
Negative	0.91	0.92	0.92	176
Neutral	0.48	0.41	0.44	37
Positive	0.83	0.86	0.84	132
accuracy			0.84	345
macro avg	0.74	0.73	0.73	345
weighted avg	0.83	0.84	0.84	345

Precision: Precision adalah rasio dari true positive (TP) terhadap semua prediksi positif yang dilakukan oleh model. Dalam konteks ini:

- Precision untuk kelas 'Negative' adalah 0.91, yang berarti dari semua prediksi yang dilabeli sebagai 'Negative' oleh model, 91% di antaranya benar-benar 'Negative'.
- Precision untuk kelas 'Neutral' adalah 0.48, yang berarti dari semua prediksi yang dilabeli sebagai 'Neutral' oleh model, 48% di antaranya benar-benar 'Neutral'.
- Precision untuk kelas 'Positive' adalah 0.83, yang berarti dari semua prediksi yang dilabeli sebagai 'Positive' oleh model, 83% di antaranya benar-benar 'Positive'.

Recall: Recall adalah rasio dari true positive (TP) terhadap semua kasus yang benar-benar positif dalam data. Dalam konteks ini:

- Recall untuk kelas 'Negative' adalah 0.92, yang berarti dari semua sampel 'Negative' dalam data, model berhasil mengklasifikasikan 92% di antaranya dengan benar.

- Recall untuk kelas 'Neutral' adalah 0.41, yang berarti dari semua sampel 'Neutral' dalam data, model hanya berhasil mengklasifikasikan 41% di antaranya dengan benar.
- Recall untuk kelas 'Positive' adalah 0.86, yang berarti dari semua sampel 'Positive' dalam data, model berhasil mengklasifikasikan 86% di antaranya dengan benar.

F1-score: F1-score adalah harmonic mean dari precision dan recall. Dalam konteks ini:

- F1-score untuk kelas 'Negative' adalah 0.92, yang merupakan harmonic mean dari precision dan recall untuk kelas 'Negative'.
- F1-score untuk kelas 'Neutral' adalah 0.44.
- F1-score untuk kelas 'Positive' adalah 0.84.

Accuracy: Accuracy adalah rasio dari prediksi yang benar (positif dan negatif) terhadap total jumlah prediksi. Dalam konteks ini, akurasi adalah 0.84, yang berarti model berhasil memprediksi dengan benar sekitar 84% dari semua sampel.

Macro avg dan Weighted avg: Macro avg adalah rata-rata dari semua kelas, di mana setiap kelas diberi bobot yang sama. Weighted avg adalah rata-rata yang memperhitungkan jumlah sampel untuk setiap kelas. Dalam konteks ini, rata-rata tersebut adalah 0.73 (macro avg) dan 0.84 (weighted avg).

Link GitHub: https://github.com/nazhifahnaurah/dibimbing/tree/main/NLP_day%2023