LAPORAN TUGAS EKSPERIMEN II

Pada tugas eksperimen II ini saya melakukan klasifikasi teks dengan analisis sentiment. Saya menggunakan dataset IMDB reviews yang mengandung 50000 baris data tentang ulasan-ulasan film yang telah dilabeli berdasarkan sentiment yang terkandung didalamnya (positif/negatif). Rasio distribusi kelas artikel bersentimen positif dengan negative pada dataset adalah 50:50, sehingga terdapat sebanyak 25000 data pada masing-masing kelas. Tujuan dilakukan eksperimen ini adalah untuk mengetahui faktor-faktor yang mempengaruhi ulasan memiliki sentimen positive maupun negative. Contoh dataset yang digunakan adalah sebagai berikut.

Reviews	Sentiment
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what happened to me. The first thing that struck me about Oz was its brutality and unflinching scenes of violence, which set in right from the word GO. Trust me, this is not a show for the faint-hearted or timid. This show pulls no…….	Positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his parents are fighting all the time. This movie is slower than a soap opera... and suddenly, Jake decides to become Rambo and kill the zombie. OK, first of all when you're going to make a film you……	Negative

Dalam melakukan eksperimen ini, terdapat lima tahap yang dilakukan, yaitu praproses data, fitur ekstraksi, pemodelan, dan pengujian.

1. Praproses data

Pada bagian ini, hal pertama yang dilakukan adalah normalisasi kata termasuk *case folding*, yaitu mentransformasikan teks menjadi huruf kecil (lower case). berikut adalah data sebelum dan sesudah dilakukan case folding

Sebelum	Sesudah
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what happened to me. The first thing that struck me about Oz was its brutality and unflinching scenes of violence, which set in right from the word GO. Trust me, this is not a show for the faint-hearted or timid. This show pulls no…….	one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened to me. the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint-hearted or timid. this show pulls no…….

Setelah data dilakukan case folding, saya melakukan penghapusan karakter atau tag html. Berikut adalah data sebelum dan sesudah dilakukan penghapusan tag html.

Sebelum	Sesudah
one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened to me. the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint-hearted or timid. this show pulls no…….	one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened to me. the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint-hearted or timid. this show pulls no…….

Dapat dilihat pada tabel diatas, data yang sesudah dilakukan penghapusan tag html tidak memiliki tag “
” dan “</br>”. Selanjutnya ditemukan kata-kata yang disingkat seperti “you’ll” , “should’nt”, “must’ve”, dll. Sehingga hal ini perlu saya tangani dengan cara menormalisasi kata-kata tersebut kebentuk aslinya. Berikut ditunjukkan kondisi data sebelum dan sesudah dilakukan normalisasi.

Sebelum	Sesudah
one of the other reviewers has mentioned that after watching just 1 oz episode you'll be hooked. they are right, as this is exactly what happened to me. the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint-hearted or timid. this show pulls no…….	one of the other reviewers has mentioned that after watching just 1 oz episode you will be hooked. they are right, as this is exactly what happened to me. the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint-hearted or timid. this show pulls no…….

Dari tabel diatas, dapat dilihat pada kondisi data sesudah dilakukan normalisasi, kata”you’ll” telah diubah menjadi “you will”. Selanjutnya, adalah melakukan penghapusan tanda baca pada teks termasuk angka. Pada tabel dibawah ini ditunjukkan kondisi data sebelum dan sesudah dilakukan penghapusan tanda baca.

Sebelum	Sesudah
one of the other reviewers has mentioned that after watching just 1 oz episode you will be hooked. they are right, as this is exactly what happened to me. the first thing that struck me about oz was its brutality and unflinching scenes of violence, which set in right from the word go. trust me, this is not a show for the faint-hearted or timid. this show pulls no…….	one of the other reviewers has mentioned that after watching just oz episode you will be hooked they are right as this is exactly what happened to me the first thing that struck me about oz was its brutality and unflinching scenes of violence which set in right from the word go trust me this is not a show for the faint-hearted or timid this show pulls no…….

Tahap praproses selanjutnya adalah penghapusan stopwords. Stopwords adalah kumpulan kata-kata yang tidak memiliki makna atau konteks seperti “if”, “but”, “then”, dll. Daftar stopwords yang digunakan diambil dari nltk.corpus.stopwords. Berikut ditampilkan kondisi data sebelum dan sesudah dilakukan penghapusan stopwords.

Sebelum	Sesudah
one of the other reviewers has mentioned that after watching just oz episode you will be hooked they are right as this is exactly what happened to me the first thing that struck me about oz was its brutality and unflinching scenes of violence which set in right from the word go trust me this is not a show for the faint-hearted or timid this show pulls no…….	one reviewers mentioned watching oz episode hooked right exactly happened first thing struck oz brutality unflinching scenes violence set right word go trust show faint hearted timid show pulls…..

Tahap selanjutnya adalah melakukan *stemming*. Stemming adalah mengubah kata yang memiliki imbuhan Kembali ke bentuk aslinya, contohnya seperti “mentioned” menjadi “mention” dan “watching” menjadi “watch”. berikut ditampilkan kondisi data sebelum dan sesudah dilakukan *stemming*.

Sebelum	Sesudah
one reviewers mentioned watching oz episode hooked right exactly happened first thing struck oz brutality unflinching scenes violence set right word go trust show faint hearted timid show pulls…..	one review mention watch oz episode hook right exact happen first thing struck oz brutal unflinch scene violence set right word go trust show faint heart timid show pull…..

Tahap praproses selanjutnya adalah menghapus kata-kata yang terlalu sering digunakan dan dianggap tidak berpengaruh terhadap sentiment suatu review. Kata yang dihapus meliputi

'movi', 'film', 'one', 'would', 'time', 'make', 'charact', 'see', 'get', 'even', 'stori', 'scene', 'show' dan 'look' (akan ditampilkan pada lampiran).

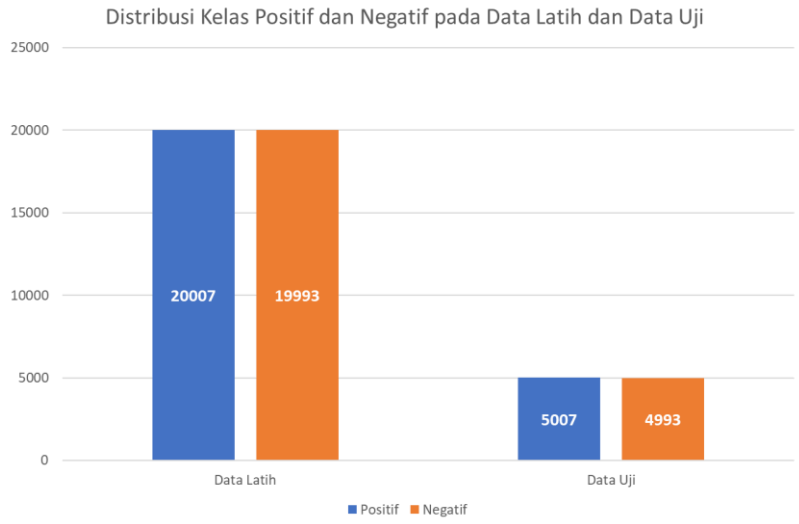
2. Ekstraksi Fitur

Ekstraksi fitur digunakan untuk mengubah data hasil praproses menjadi fitur-fitur sehingga dapat diteruskan ke pemodelan. Saya menggunakan word indexing yang disediakan oleh keras untuk merepresentasikan fitur-fitur pada korpus menjadi sebuah vektor. Pada eksperimen ini, saya membatasi jumah maksimum fitur yang digunakan pada korpus sebesar 10000, sedangkan panjang maksimum vektor representasi sebesar 1000. Berikut ditunjukkan hasil ekstraksi fitur pada gambar dibawah ini.

```
array([[   0,    0,    0, ...,  425, 3290,  369],
       [   0,    0,    0, ...,  256,    3,  130],
       [   0,    0,    0, ...,   88,   12,   97],
       ...,
       [   0,    0,    0, ..., 5872, 3140, 1035],
       [   0,    0,    0, ..., 1749, 1350,  304],
       [   0,    0,    0, ..., 1679,  870,  543]])
```

3. Pemodelan

sebelum memasuki pemodelan dataset dibagi menjadi data pelatihan dan data uji. Model klasifikasi dilatih menggunakan data pelatihan dan kemudian dievaluasi menggunakan data uji. Rasio pemisahan data latih dan data uji yang ditentukan adalah 80:20. Dengan demikian jumlah data latih dan data uji masing-masing adalah 40000 dan 10000. Adapun distribusi kelas pada data latih dan data uji di perlihatkan pada gambar dibawah ini.



Untuk membangun model klasifikasi, saya menggunakan metode Long Short Term Memory (LSTM). arsitektur yang digunakan ditunjukkan pada tabel dibawah ini.

Layer	Neurons
Embedding	-
LSTM	64
Dense	256
Dropout	0.1
Dense	128
Dense	1

Disamping itu, kami menggunakan optimizer yang disediakan oleh Keras backend, yaitu Adam. Proses pelatihan dilakukan sebanyak 10 kali iterasi (epochs) dengan jumlah batch size yang digunakan adalah 32. Hasil pelatihan model dengan metode LSTM nampaknya memiliki akurasi yang sangat tinggi, model ini berhasil mengklasifikasikan ulasan kedalam sentimen positif dan negative secara benar sebanyak 39.444 ulasan atau 98,6%.

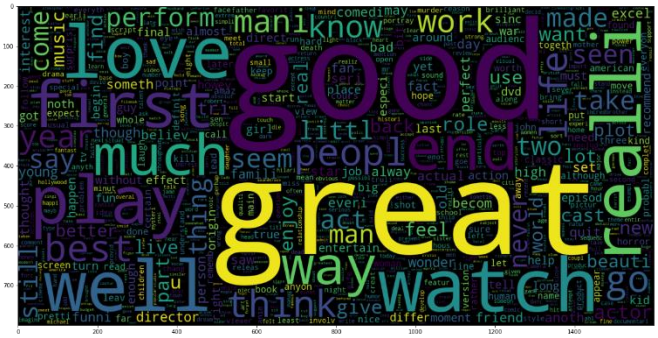
4. Pengujian

Seperti yang sudah dijelaskan sebelumnya, dalam menguji kelayakan model yang telah dilatih, saya menggunakan data uji sebanyak 10000 records dengan distribusi kelas positif dan negative masing-masing adalah 5007 dan 4993. Hasil pengujian menunjukkan model ini menghasilkan akurasi pengujian sebesar 86,7%, yang artinya model yang dibangun dapat diterapkan untuk semua dataset ulasan film. Berikut ditunjukkan matriks konfusi sebagai hasil evaluasi performansi dari model.

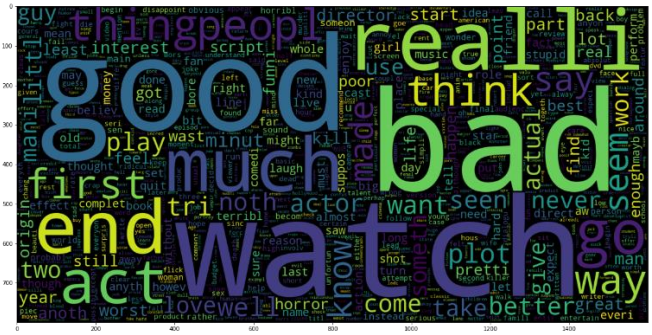
	Predicted Positive	**Predicted Negative**
Actual Positive	43,74%	6,33%
Actual negative	6,96%	42,97%

Dari tabel diatas, jumlah data sentimen positif yang diprediksi positif (TP) adalah 4374 atau 43,74%. Jumlah data sentimen positif yang diprediksi negatif (FN) adalah 633 atau 6,33%. Jumlah data sentimen negatif yang diprediksi negatif (TN) adalah 42,97 atau 4297%. Jumlah data sentimen negatif yang diprediksi positif (FP) adalah 696 atau 69,6%. Dengan kata lain, model mengklasifikasikan 5070 data positif (TP + FP) dan 4930 data negatif (TN + FN). Setelah mengetahui data-data yang terklasifikasi positif dan negative, selanjutnya akan ditunjukkan faktor-faktor apa saja yang mempengaruhi nilai sentimen suatu ulasan dengan menggunakan Wordcloud. Dengan wordcloud, kita dapat mengetahui kata-kata apa saja yang sering digunakan pada ulasan bersentimen positif dan negative. Ditunjukkan wordcloud data bersentimen positif dan negative dibawah ini.

Wordcloud data kelas positif



Wordcloud data kelas negatif



Terlihat dari gambar diatas bahwa faktor-faktor yang mempengaruhi ulasan memiliki sentimen positif adalah adalah kata “good”, “great”, “love”, “well”, “much”, dan “realli” sedangkan faktor-faktor yang mempengaruhi ulasan memiliki sentimen positif adalah adalah kata “good”, “bad”, “much”, “realli”, “watch”, dan “end”. Dari kedua wordcloud terlihat beberapa kata yang terdapat pada 2 kelas, yaitu “good”, “realli”, dan “much”. Dari yang telah saya analisis kata-kata tersebut memang mungkin ada dia du akelas, seperti “good” bisa di kelas positif karena memiliki makna bagus, namun di kelas positif juga bisa apabila “not” tidak terhapus pada stopwords. Sehingga pada kelas negative kata “good” mungkin artinya “not good”. Hal ini juga berlaku untuk word “realli”, kata tersebut merupakan suatu keterangan hiperbola untuk melebih-lebih kan sesuatu, salah satunya “really good” atau “really bad”, sehingga kata tersebut dapat berada di dua kelas.

KESIMPULAN

Dari hasil dan pembahasan, terdapat tiga hal yang dapat disimpulkan, pertama model klasifikasi sentimen ulasan film dari IMDB dataset menggunakan metode LSTM menghasilkan akurasi yang sangat tinggi, yaitu 98,6% dari sisi pelatihan dan 86,7% dari sisi pengujian. Hal ini memberikan alasan yang kuat bahwa model yang dibangun layak digunakan untuk semua dataset dalam hal memprediksi sentimen yang terkandung dalam sebuah ulasan film. Lalu yang kedua, dari hasil performansi yang ditunjukkan *confusion matrix*, terdapat sebanyak 5070 data yang terprediksi positif dan 4030 data yang terprediksi negative, hal ini menunjukkan bahwa hasil prediksi di setiap kelas memiliki distribusi yang cukup seimbang. Lalu yang terakhir, dapat disimpulkan bahwa faktor yang mempengaruhi ulasan memiliki sentimen positif adalah “good”, “great”, “love”, “well”, “much”, dan “realli”. Sedangkan faktor yang mempengaruhi ulasan memiliki sentimen negative adalah “good”, “bad”, “much”, “realli”, “watch”, dan “end”.

REFERENSI

N. Lakshmipathi, “Sentiment Analysis of IMDB Movie Reviews,” 2020. https://www.kaggle.com/code/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews (accessed Jul. 07, 2022).

I. Bernando, “Stemming Text with NLTK,” May 3, 2021. https://towardsdatascience.com/stemming-corpus-with-nltk-7a6a6d02d3e5 (accessed Jul. 07, 2022).

LINK VIDIO PRESENTASI DAN DEMO KODINGAN :

https://youtu.be/KUslEDI2Z-Y

LAMPIRAN

1. Kodingan

1.1. Data Collection

```
import pandas as pd

# read the dataset
df = pd.read_csv('IMDB Dataset.csv')

# map positive and negative classes tot 1 and 0
df['sentiment'] = df['sentiment'].replace(['positive','negative'],[1,0])
```

1.2. Data Preprocessing

```
import string
import nltk
from bs4 import BeautifulSoup
import re

# Setting punctuation
PUNCT_TO_REMOVE = string.punctuation
PUNCT_TO_REMOVE += '0123456789'
PUNCT_TO_REMOVE = PUNCT_TO_REMOVE.replace("'","").replace(".","")

# Setting English stopwords
stopword_list=nltk.corpus.stopwords.words('english')

# Custom fuction to remove the html strips
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

# Custom fuction to remove text withing square bracket
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)

# custom function to remove punctuation
def remove_punctuation(text):
    return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))

# custom function to remove stopwords
def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in
    stopword_list])


# Stemming the text
```

```
def simple_stemmer(text):
    stem = nltk.stem.SnowballStemmer(language='english')
    text = ' '.join([stem.stem(word) for word in text.split()])
    return text

# custom function to change apostrophe/short words
def decontracted(text):
    # specific
    text = re.sub(r"won\'t", "will not", text)
    text = re.sub(r"can\'t", "can not", text)

       # general
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    return text

# Remove Frequent Word
def remove_freqwords(text):
    return " ".join([word for word in str(text).split() if word
    not in FREQWORDS])

# Case Folding
df['review'] = df['review'].str.lower()
# remove html strips
df["review"] = df["review"].apply(lambda text: strip_html(text))
# remove square brackets
df["review"] = df["review"].apply(lambda text:
               remove_between_square_brackets(text))
# remove punctuation
df["review"] = df["review"].apply(lambda text: remove_punctuation(text))
# change apostrophe/short words
df["review"] = df["review"].apply(lambda text: decontracted(text))
# correcting text

for i in range(50000):
    # correcting dot placement
    df['review'][i] = df['review'][i].replace(".",". ").replace("  "," ")
    # remove apostrophe
    df['review'][i] = df['review'][i].replace("'","")
    # correcting multiple white spaces
    df['review'][i] = re.sub(' +',' ', df['review'][i])
    # correcting whitespace in the first and last part of the text
```

```
    df['review'][i] = df['review'][i].strip()

PUNCT_TO_REMOVE += "."

# remove punctuation
df["review"] = df["review"].apply(lambda text: remove_punctuation(text))

for i in range(50000):
    # correcting multiple white space
    df['review'][i] = re.sub(' +',' ', df['review'][i])

# Remove Stopwords
df["review"] = df["review"].apply(lambda text: remove_stopwords(text))
#Apply function on review column
df['review']=df['review'].apply(lambda text: simple_stemmer(text))

from collections import Counter

# Check number of unique feature
cnt = Counter()
for text in df["review"].values:
    for word in text.split():
        cnt[word] += 1

most_common_words = cnt.most_common(20)

# remove the word 'like','good','well','much', 'bad' from the list that will
be used to remove the most frequent word
indices = {3,6,11,14,16,18}

most_common_words = [v for i, v in enumerate(most_common_words) if i not in
indices]

FREQWORDS = []
for i in most_common_words:
    FREQWORDS.append(i[0])

# remove frequent words
df["review"] = df["review"].apply(lambda text: remove_freqwords(text))
```

1.3. Ekstraksi Fitur

```
from tensorflow.keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence

max_words = 10000
max_len = 1000
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(df['review'].values)
```

```
sequences = tok.texts_to_sequences(df['review'].values)
sequences_matrix = sequence.pad_sequences(sequences,maxlen=max_len)
```

1.4. Data Splitting

```
x_train = sequences_matrix[:40000]
y_train = df['sentiment'][:40000]

x_test = sequences_matrix[40000:]
y_test = df['sentiment'][40000:]
```

1.5. Data Modelling

```
from tensorflow.keras.layers import Embedding, Dense, LSTM, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

# create the model
embedding_vecor_length = 32

model = Sequential()
model.add(Embedding(max_words, embedding_vecor_length, input_length=max_len))
model.add(LSTM(64))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer=Adam(), metrics=
['accuracy']
history = model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=10,
batch_size=32, verbose=1)
```

1.6. Model Evaluation

```
model.evaluate(x_test,y_test)
```

1.7. Lihat Faktor di tiap kelas

```
from wordcloud import WordCloud
processedtext1 = list(df_test_positive['review'])

wc1 = WordCloud(max_words = 1000 , width = 1600 , height = 800,
              collocations=False).generate(" ".join(processedtext1))
plt.figure(figsize = (20,20))
plt.imshow(wc)

processedtext2 = list(df_test_negative['review'])

wc2 = WordCloud(max_words = 1000 , width = 1600 , height = 800,
              collocations=False).generate(" ".join(processedtext2))
plt.figure(figsize = (20,20))
plt.imshow(wc)
```

2. Hasil Kodingan

2.1. Hasil training

```
Epoch 1/10
1250/1250 [==============================] - 275s 215ms/step - loss: 0.3461 - accuracy: 0.8484 - val_loss: 0.2826 -
val_accuracy: 0.8837
Epoch 2/10
1250/1250 [==============================] - 258s 206ms/step - loss: 0.2323 - accuracy: 0.9110 - val_loss: 0.2956 -
val_accuracy: 0.8733
Epoch 3/10
1250/1250 [==============================] - 251s 201ms/step - loss: 0.1835 - accuracy: 0.9292 - val_loss: 0.3610 -
val_accuracy: 0.8725
Epoch 4/10
1250/1250 [==============================] - 241s 193ms/step - loss: 0.1439 - accuracy: 0.9465 - val_loss: 0.3570 -
val_accuracy: 0.8726
Epoch 5/10
1250/1250 [==============================] - 258s 207ms/step - loss: 0.1137 - accuracy: 0.9590 - val_loss: 0.3624 -
Epoch 6/10
1250/1250 [==============================] - 218s 174ms/step - loss: 0.0908 - accuracy: 0.9683 - val_loss: 0.5309 -
val_accuracy: 0.8656
Epoch 7/10
1250/1250 [==============================] - 277s 222ms/step - loss: 0.0707 - accuracy: 0.9755 - val_loss: 0.5593 -
val_accuracy: 0.8615
Epoch 8/10
1250/1250 [==============================] - 247s 198ms/step - loss: 0.0569 - accuracy: 0.9811 - val_loss: 0.6095 -
val_accuracy: 0.8625
Epoch 9/10
1250/1250 [==============================] - 252s 202ms/step - loss: 0.0456 - accuracy: 0.9850 - val_loss: 0.6625 -
val_accuracy: 0.8637
Epoch 10/10
1250/1250 [==============================] - 244s 195ms/step - loss: 0.0428 - accuracy: 0.9861 - val_loss: 0.6569 -
val_accuracy: 0.8671
```

2.2. Hasil Pengujian

```
313/313 [==============================] - 26s 82ms/step - loss: 0.6569 - accuracy: 0.8671

[0.6568961143493652, 0.8671000003814697]
```

2.3. Hasil Konfusi Matriks