nlp

November 6, 2024

[6]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[22]:
```
import pandas as pd
import joblib
import re
import tensorflow as tf
import pickle
import nltk
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
```

[14]:
```
# Memuat dataset
df = pd.read_csv('/content/drive/MyDrive/dataset/imdb_dataset.csv')
df['sentiment'] = df['sentiment'].astype(str)
df['sentiment'] = df['sentiment'].replace({'Positive': '1', 'Negative': '0'})
df.head()
```

[14]:

	review	sentiment
0	One of the other reviewers has mentioned that …	positive
1	A wonderful little production. The…	positive
2	I thought this was a wonderful way to spend ti…	positive
3	Basically there's a family where a little boy …	negative
4	Petter Mattei's "Love in the Time of Money" is…	positive

[15]:
```
import pandas as pd
import re

# Define the remove_tags function
def remove_tags(string):
    # Handle NaN or non-string values
    if isinstance(string, str):
```

```
        removelist = ""
        result = re.sub('', '', string)  # Remove HTML tags
        result = re.sub('https://.*', '', result)  # Remove URLs
        return result
    else:
        # Return empty string or NaN for non-string values
        return ''  # Or return pd.NA to preserve NaN values


# Memuat dataset
df = pd.read_csv('/content/drive/MyDrive/dataset/imdb_dataset.csv')
df['sentiment'] = df['sentiment'].astype(str)
df['sentiment'] = df['sentiment'].replace({'Positive': '1', 'Negative': '0'})

# Clean the 'review' column
df['review'] = df['review'].apply(lambda cw: remove_tags(cw))
```

[16]:
```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
df['review'] = df['review'].apply(lambda x: ' '.join([word for word in x.
 ↪split() if word not in (stop_words)]))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[17]:
```
# Download 'wordnet' data using nltk.download
nltk.download('wordnet')

w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

def lemmatize_text(text):
    # Convert the input to string if it's not already
    text = str(text)

    st = ""
    for w in w_tokenizer.tokenize(text):
        st = st + lemmatizer.lemmatize(w) + " "
    return st

df['review'] = df.review.apply(lemmatize_text)
df
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[17]:
```
                                                  review sentiment
0      One reviewer mentioned watching 1 Oz episode h…  positive
1      A wonderful little production. <br /><br />The…  positive
2      I thought wonderful way spend time hot summer …  positive
3      Basically there's family little boy (Jake) thi…  negative
4      Petter Mattei's "Love Time Money" visually stu…  positive
…                                                    …         …
49995  I thought movie right good job. It creative or…  positive
49996  Bad plot, bad dialogue, bad acting, idiotic di…  negative
49997  I Catholic taught parochial elementary school …  negative
49998  I'm going disagree previous comment side Malti…  negative
49999  No one expects Star Trek movie high art, fan e…  negative

[50000 rows x 2 columns]
```

[18]:
```
s = 0.0
for i in df['review']:
    word_list = i.split()
    s = s + len(word_list)
print("Average length of each review : ",s/df.shape[0])
pos = 0
for i in range(df.shape[0]):
    if df.iloc[i]['sentiment'] == 'positive':
        pos = pos + 1
neg = df.shape[0]-pos
print("Percentage of reviews with positive sentiment is "+str(pos/df.
 ↪shape[0]*100)+"%")
print("Percentage of reviews with negative sentiment is "+str(neg/df.
 ↪shape[0]*100)+"%")
```

```
Average length of each review :  136.38816
Percentage of reviews with positive sentiment is 50.0%
Percentage of reviews with negative sentiment is 50.0%
```

[19]:
```
from sklearn.preprocessing import LabelEncoder

reviews = df['review'].values
labels = df['sentiment'].values
encoder = LabelEncoder()
encoded_labels = encoder.fit_transform(labels)
```

[20]:
```
train_sentences, test_sentences, train_labels, test_labels =␣
 ↪train_test_split(reviews, encoded_labels, stratify = encoded_labels)
```

[23]:
```
# Hyperparameters of the model
vocab_size = 3000 # choose based on statistics
oov_tok = ''
```

```
embedding_dim = 100
max_length = 200 # choose based on statistics, for example 150 to 200
padding_type='post'
trunc_type='post'
# tokenize sentences
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train_sentences)
word_index = tokenizer.word_index
# convert train dataset to sequence and pad sequences
train_sequences = tokenizer.texts_to_sequences(train_sentences)
train_padded = pad_sequences(train_sequences, padding='post', maxlen=max_length)
# convert Test dataset to sequence and pad sequences
test_sequences = tokenizer.texts_to_sequences(test_sentences)
test_padded = pad_sequences(test_sequences, padding='post', maxlen=max_length)
```

[24]:
```
import tensorflow as tf
from tensorflow import keras

# model initialization
model = keras.Sequential([
    keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    keras.layers.Bidirectional(keras.layers.LSTM(64)),
    keras.layers.Dense(24, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
# compile model
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Menyimpan model
model.save('sentiment_model.h5')

# Menyimpan tokenizer
with open('tokenizer.pkl', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90:
UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.
```

[25]:
```
num_epochs = 5
history = model.fit(train_padded, train_labels,
                    epochs=num_epochs, verbose=1,
                    validation_split=0.1)
```

```
Epoch 1/5
1055/1055             300s 279ms/step
- accuracy: 0.7446 - loss: 0.5002 - val_accuracy: 0.8613 - val_loss: 0.3779
Epoch 2/5
1055/1055             329s 286ms/step
- accuracy: 0.8762 - loss: 0.3071 - val_accuracy: 0.8776 - val_loss: 0.3268
Epoch 3/5
1055/1055             296s 280ms/step
- accuracy: 0.9011 - loss: 0.2487 - val_accuracy: 0.8821 - val_loss: 0.3112
Epoch 4/5
1055/1055             322s 281ms/step
- accuracy: 0.9217 - loss: 0.2046 - val_accuracy: 0.8800 - val_loss: 0.3021
Epoch 5/5
1055/1055             328s 287ms/step
- accuracy: 0.9361 - loss: 0.1683 - val_accuracy: 0.8776 - val_loss: 0.3106
```

[26]:
```
!pip install scikit-learn
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import accuracy_score
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

[27]:
```
prediction = model.predict(test_padded)
# Get labels based on probability 1 if p>= 0.5 else 0
pred_labels = []
for i in prediction:
    if i >= 0.5:
        pred_labels.append(1)
    else:
        pred_labels.append(0)
print("Accuracy of prediction on test set : ",␣
 ↪accuracy_score(test_labels,pred_labels))
```

```
391/391             48s 121ms/step
Accuracy of prediction on test set :  0.87952
```

[28]:
```python
# reviews on which we need to predict
sentence = ["This movie is excellent",
            "The movie was very touching and heart whelming",
            "I have never seen a terrible movie like this",
            "the movie plot is terrible"]
# convert to a sequence
sequences = tokenizer.texts_to_sequences(sentence)
# pad the sequence
padded = pad_sequences(sequences, padding='post', maxlen=max_length)
# Get labels based on probability 1 if p>= 0.5 else 0
prediction = model.predict(padded)
pred_labels = []
for i in prediction:
    if i >= 0.5:
        pred_labels.append(1)
    else:
        pred_labels.append(0)
for i in range(len(sentence)):
    print(sentence[i])
    if pred_labels[i] == 1:
        s = 'Positive'
    else:
        s = 'Negative'
    print("Predicted sentiment : ",s)
```

```
1/1             0s 44ms/step
This movie is excellent
Predicted sentiment :  Positive
The movie was very touching and heart whelming
Predicted sentiment :  Positive
I have never seen a terrible movie like this
Predicted sentiment :  Negative
the movie plot is terrible
Predicted sentiment :  Negative
```