Artificial Intelligence and Machine Learning

Documentation

< Worksheet 8 >

Name : Kripa Lama

University Id : 2358546

Course : Artificial Intelligence and Machine Learning

Group : L6CG19

Tutor : Ms. Durga Pokharel

Submitted on : 23rd April, 2025

Part -1 (class work) : Text Pre-processing in NLP.

Exercise

[5] 0s

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, LSTM, Dense, Dropout
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('wordnet')
```

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

Load the Dataset

[6] 3s

```
data = pd.read_csv("/content/drive/MyDrive/Artificial Intelligence and Machine Learning/Week 8/trum_tweet_sentiment_analysis.csv")
data.head()
```

	text	Sentiment
0	RT @JohnLeguizamo: #trump not draining swamp b...	0
1	ICYMI: Hackers Rig FM Radio Stations To Play A...	0
2	Trump protests: LGBTQ rally in New York https:...	1
3	"Hi I'm Piers Morgan. David Beckham is awful b...	0
4	RT @GlennFranco68: Tech Firm Suing BuzzFeed fo...	0

Text Cleaning and Tokenization

```
[7]  # Removing URLs
     def remove_urls(text):
       url_pattern = re.compile(r'https?://\S+|www\.\S+')
       return url_pattern.sub(r'', text)
```

```
[8]  # Remove emojis from the text
     def remove_emoji(string):
       emoji_pattern = re.compile("["
                                  u"\U0001F600-\U0001F64F"  # emoticons
                                  u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                                  u"\U0001F680-\U0001F6FF"  # transport & map symbols
                                  u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                                  u"\U00002702-\U000027B0"
                                  u"\U000024C2-\U0001F251"
                                  "]+", flags=re.UNICODE)
       return emoji_pattern.sub(r' ', string)
```

```
# Number of text preprocessing steps, such as removing user mentions, hashtags, punctuation, and even emojis.

def removeunwanted_characters(document):
  # remove user mentions
  document = re.sub("@[A-Za-z0-9_]+"," ", document)
  # remove hashtags
  document = re.sub("#[A-Za-z0-9_]+","", document)
  # remove punctuation
  document = re.sub("[^0-9A-Za-z ]", "" , document)
  #remove emojis
  document = remove_emoji(document)
  # remove double spaces
  document = document.replace('  ',"")
  return document.strip()
```

Removing Punctuation

```
[10] # Uses a regular expression (r"\w+") to match only word characters (letters, digits, and underscores).

     def remove_punct(text):
       tokenizer = RegexpTokenizer(r"\w+")
       lst=tokenizer.tokenize(' '.join(text))
       return lst
```

```
[11] nltk.download('stopwords')
     from nltk.corpus import stopwords
     from nltk.tokenize import word_tokenize
     stop_words = set(stopwords.words('english'))
     custom_stopwords = ['@', 'RT']
     stop_words.update(custom_stopwords)
```

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

Removing stopwords (the, is, and)

```
def remove_stopwords(text_tokens):
  result_tokens = []
  for token in text_tokens:
    if token not in stop_words:
       result_tokens.append(token)
  return result_tokens
```

Tokenization (Lemmatizationn Text Processing Technique)

```
[13] from nltk.stem import WordNetLemmatizer
     from nltk import word_tokenize,pos_tag
     nltk.download('averaged_perceptron_tagger')
     nltk.download('wordnet')

     def lemmatization(token_text):
       """
       This function performs the lemmatization operations as explained above.
       Input Args:
       token_text: list of tokens.
       Returns:
       lemmatized_tokens: list of lemmatized tokens.
       """
       lemma_tokens = []
       wordnet = WordNetLemmatizer()
       lemmatized_tokens = [wordnet.lemmatize(token, pos = 'v') for token in token_text]

       return lemmatized_tokens
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
[14] lemmatization("Should we go walking or swimming".split())
```

```
['Should', 'we', 'go', 'walk', 'or', 'swim']
```

[15]
```
from nltk.stem import PorterStemmer

def stemming(text):
  """
  This function performs stemming operations.
  Input Args:
  token_text: list of tokenize text.
  Returns:
  stemm_tokes: list of stemmed tokens.
  """
  porter = PorterStemmer()
  stemm_tokens = []
  for word in text:
    stemm_tokens.append(porter.stem(word))
  return stemm_tokens
```

Text Processing Comparison

```
#Test
print("++++++++++++++++++++++++++++++++" "INPUT TOKENS" "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
token_text_test=['Connects','Connecting','Connections','Connected','Connection','Connectings','Connect']
print(token_text_test)
print("++++++++++++++++++" "LEMMATIZED TOKENS" "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
lemma_tokens = lemmatization(token_text_test)
print(lemma_tokens)
print("++++++++++++++++++++" "STEMMED TOKENS" "++++++++++++++++++++++++++++++++++")
stemmed_tokens = stemming(token_text_test)
print(stemmed_tokens)
```

```
++++++++++++++++++++++++++++++++INPUT TOKENS++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
['Connects', 'Connecting', 'Connections', 'Connected', 'Connection', 'Connectings', 'Connect']
++++++++++++++++++LEMMATIZED TOKENS++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
['Connects', 'Connecting', 'Connections', 'Connected', 'Connection', 'Connectings', 'Connect']
++++++++++++++++++++STEMMED TOKENS++++++++++++++++++++++++++++++++++
['connect', 'connect', 'connect', 'connect', 'connect', 'connect', 'connect']
```

Converting all characters in the input text to lowercase

[17]
```
def lower_order(text):
  """
  This function converts all the text in input text to lower order.
  Input Args:
  token_text : input text.
  Returns:
  small_order_text : text converted to small/lower order.
  """
  small_order_text = text.lower()
  return small_order_text

# Test:
sample_text = "This Is some Normalized TEXT"
sample_small = lower_order(sample_text)
print(sample_small)
```

```
this is some normalized text
```

[18]
```
data = pd.read_csv("/content/drive/MyDrive/Artificial Intelligence and Machine Learning/Week 8/trum_tweet_sentiment_analysis.csv")
data.head()
```

	text	Sentiment
0	RT @JohnLeguizamo: #trump not draining swamp b...	0
1	ICYMI: Hackers Rig FM Radio Stations To Play A...	0
2	Trump protests: LGBTQ rally in New York https:...	1
3	"Hi I'm Piers Morgan. David Beckham is awful b...	0
4	RT @GlennFranco68: Tech Firm Suing BuzzFeed fo...	0

Data Cleaning

```
[19] # Data cleaning operation targeting the "text" column
     data_cleaning = data["text"].dropna()
```

```
[20] data_cleaning[0]
```

```
'RT @JohnLeguizamo: #trump not draining swamp but our taxpayer dollars on his trips to advertise his properties! @realDonaldTrump\x85 https://t.co/gFBvUkMX9z'
```

```
[21] def text_cleaning_pipeline(dataset, rule = "lemmatize"):
       """
       This...
       """
       # Convert the input to small/lower order.
       data = lower_order(dataset)
       # Remove URLs
       data = remove_urls(data)
       # Remove emojis
       data = remove_emoji(data)
       # Remove all other unwanted characters.
       data = removeunwanted_characters(data)
       # Create tokens.
       tokens = data.split()
       # Remove stopwords:
       tokens = remove_stopwords(tokens)
       if rule == "lemmatize":
         tokens = lemmatization(tokens)
       elif rule == "stem":
         tokens = stemming(tokens)
       else:
         print("Pick between lemmatize or stem")

       return " ".join(tokens)
```

```
[22] sample = "Hello @gabe_flomo 👋🏾, I still want us to hit that new sushi spot??? LMK when you're free cuz I can't go this or next weekend since I'll be swimmir
     print(text_cleaning_pipeline(sample))
```

```
hello still want us hit new sushi spot lmk youre free cuz cant go next weekend since ill swim
```

```
[23] test = data["text"][0]
```

```
[24] print(text_cleaning_pipeline(test))
```

```
rtnot drain swamp taxpayer dollars trip advertise properties
```

```
[25] cleaned_tokens = data["text"].apply(lambda dataset: text_cleaning_pipeline(dataset))
```

Train-Test Split

```
[26] data['cleaned_text'] = data['text'].apply(lambda dataset: text_cleaning_pipeline(dataset))

     X_train, X_test, y_train, y_test = train_test_split(data['cleaned_text'], data['Sentiment'], test_size=0.2, random_state=42)
```

```
[27] print(data.columns)
```

```
Index(['text', 'Sentiment', 'cleaned_text'], dtype='object')
```

TF-IDF Vectorization

28s

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

Model Training and Evaluation

16s

```
[29] from sklearn.linear_model import LogisticRegression

     model = LogisticRegression()
     model.fit(X_train_tfidf, y_train)
     y_pred = model.predict(X_test_tfidf)
```

0s

```
[30] from sklearn.metrics import classification_report

     print("Classification Report:")
     print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.96      0.94    248563
           1       0.90      0.86      0.88    121462

    accuracy                           0.92    370025
   macro avg       0.92      0.91      0.91    370025
weighted avg       0.92      0.92      0.92    370025
```

Predict a Single Tweet

0s

```
[32] tweet = data["text"][89]

     cleaned_tweet = text_cleaning_pipeline(tweet)
     tweet_tfidf = vectorizer.transform([cleaned_tweet])
     prediction = model.predict(tweet_tfidf)[0]

     sentiment = "Positive" if prediction == 1 else "Negative"
     print(f"Tweet: {tweet}")
     print(f"Predicted Sentiment: {sentiment}")
```

```
Tweet: RT @KeithOlbermann: The Deputy Editorial Page editor of The Wall Street Journal and I agree on the only word that fits @realDonaldTrump  https://t.co/F7Z3LA6Om7
Predicted Sentiment: Positive
```