omkargaikwad23 / **DSBDA** Public

Code Issues Pull requests Actions Projects Wiki Security Insights

main

DSBDA / Assignment07 / **Assignment7.ipynb**

omkargaikwad23 updates

History

1 contributor

620 lines (620 sloc) | 28.7 KB

Name: Omkar Gaikwad
Batch: L1
Roll No: 31126

Problem Statement

Text Analysis

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS tagging, stop words removal, Stemming and Lemmatization.

2. Create representation of document by calculating Term frequency and inverse document frequency.

In [ ]:
```
# Import necessary libraries
import nltk
import pandas as pd
import numpy as np

# Download the required components
# nltk.download()
```

In [ ]:
```
# nltk.download()
```

In [3]:
```
# Load the text from the text files to string
text1 = ""
text2 = ""

with open('testdoc.txt', 'r') as f:
    text1 = f.read()

with open('testdoc2.txt', 'r') as f:
    text2 = f.read()
```

In [4]:
```
# Print a part of the text
print(text1[:100])
```

```
Millions of people in India took part in an annual tree planting drive Sunda
y. More than 250 million
```

In [5]:
```
# Make all the words as lower cased
text1 = text1.lower()
```

Tokenization

Tokenization is the first step when working with language tasks, it simplifies the input data by splitting it into sentences or words, as per the requirement

In [6]:
```
# Tokenize the texts
sentences1 = nltk.sent_tokenize(text1)
words1 = nltk.word_tokenize(text1)

sentences2 = nltk.sent_tokenize(text2)
```

```
words2 = nltk.word_tokenize(text2)

# Print the tokenized output for the first text
print('The number of sentence tokens are: ', len(sentences1))
print('The first sentence is: ', sentences1[0])
print('\nThe number of word tokens are: ', len(words1))
print('The first 5 words are: ', words1[:5])
```

```
The number of sentence tokens are:  20
The first sentence is:  millions of people in india took part in an annual tr
ee planting drive sunday.

The number of word tokens are:  433
The first 5 words are:  ['millions', 'of', 'people', 'in', 'india']
```

POS Tagging and Stop Words removal

In [7]:

```
# Remove stop words in english language like a, an, the
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
print('The number of stopwords in english are: ', len(stop_words))
print('\nThe stop words in english are: ', stop_words)
```

```
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
~\anaconda3\lib\site-packages\nltk\corpus\util.py in __load(self)
     82                 try:
---> 83                     root = nltk.data.find("{}/{}".format(self.subdir,
zip_name))
     84                 except LookupError:

~\anaconda3\lib\site-packages\nltk\data.py in find(resource_name, paths)
    582     resource_not_found = "\n%s\n%s\n%s\n" % (sep, msg, sep)
--> 583     raise LookupError(resource_not_found)
    584

LookupError:
**********************************************************************
  Resource stopwords not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('stopwords')

  For more information see: https://www.nltk.org/data.html

  Attempted to load corpora/stopwords.zip/stopwords/

  Searched in:
    - 'C:\\Users\\omkar madhav gaikwad/nltk_data'
    - 'C:\\Users\\omkar madhav gaikwad\\anaconda3\\nltk_data'
    - 'C:\\Users\\omkar madhav gaikwad\\anaconda3\\share\\nltk_data'
    - 'C:\\Users\\omkar madhav gaikwad\\anaconda3\\lib\\nltk_data'
    - 'C:\\Users\\omkar madhav gaikwad\\AppData\\Roaming\\nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
**********************************************************************


During handling of the above exception, another exception occurred:

LookupError                               Traceback (most recent call last)
<ipython-input-7-d02c71c5a3f0> in <module>
      2 from nltk.corpus import stopwords
      3
```

```
----> 4 stop_words = set(stopwords.words('english'))
      5 print('The number of stopwords in english are: ', len(stop_words))
      6 print('\nThe stop words in english are: ', stop_words)

~\anaconda3\lib\site-packages\nltk\corpus\util.py in __getattr__(self, attr)
    118             raise AttributeError("LazyCorpusLoader object has no attr
ibute '__bases__'")
    119
--> 120         self.__load()
    121         # This looks circular, but its not, since __load() changes ou
r
    122         # __class__ to something new:

~\anaconda3\lib\site-packages\nltk\corpus\util.py in __load(self)
     83                     root = nltk.data.find("{}/{}".format(self.subdir,
zip_name))
     84                 except LookupError:
---> 85                     raise e
     86
     87         # Load the corpus.

~\anaconda3\lib\site-packages\nltk\corpus\util.py in __load(self)
     78         else:
     79             try:
---> 80                 root = nltk.data.find("{}/{}".format(self.subdir, sel
f.__name))
     81             except LookupError as e:
     82                 try:

~\anaconda3\lib\site-packages\nltk\data.py in find(resource_name, paths)
    581     sep = "*" * 70
    582     resource_not_found = "\n%s\n%s\n%s\n" % (sep, msg, sep)
--> 583     raise LookupError(resource_not_found)
    584
    585


LookupError:
**********************************************************************
  Resource stopwords not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('stopwords')

  For more information see: https://www.nltk.org/data.html

  Attempted to load corpora/stopwords

  Searched in:
    - 'C:\\Users\\omkar madhav gaikwad/nltk_data'
    - 'C:\\Users\\omkar madhav gaikwad\\anaconda3\\nltk_data'
    - 'C:\\Users\\omkar madhav gaikwad\\anaconda3\\share\\nltk_data'
    - 'C:\\Users\\omkar madhav gaikwad\\anaconda3\\lib\\nltk_data'
    - 'C:\\Users\\omkar madhav gaikwad\\AppData\\Roaming\\nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
**********************************************************************
```

In [ ]:

```
words1 = [word for word in words1 if word not in stop_words]

# Store the first sentence for later comparison
first_sent = sentences1[0]

for i in range(len(sentences1)):
    words = nltk.word_tokenize(sentences1[i])
    words = [word for word in words if word not in stop_words]
    sentences1[i] = ' '.join(words)
```

In [8]:

```
print('After removing stop words, the number of word tokens left are: ', len(
print('\nFirst sentence before removal of stopwords was: ', first_sent)
print('First sentence after removal of stopwords is: ', sentences1[0])
```

```
After removing stop words, the number of word tokens left are:  281

First sentence before removal of stopwords was:  millions of people in india
took part in an annual tree planting drive sunday.
First sentence after removal of stopwords is:  millions people india took par
t annual tree planting drive sunday .
```

Observation

- Earlier the length of word list was 434, but after removal of stopwords the length has reduced to 298. Thus, 136 words have been removed.
- Also, stopwords have been removed from the sentences.

In [9]:

```
# POS tagging (part of speech)

tagged_pairs = []

for i in sentences1:

    #  Using Part of speech tagger
    tagged = nltk.pos_tag(words1)
    tagged_pairs.append(tagged)

print('A few exmaples of tagged words from the first sentence are: ', tagged_
```

```
A few exmaples of tagged words from the first sentence are:  [('millions', 'N
NS'), ('people', 'NNS'), ('india', 'VBP'), ('took', 'VBD'), ('part', 'NN'),
('annual', 'JJ'), ('tree', 'NN'), ('planting', 'VBG'), ('drive', 'JJ'), ('sun
day', 'NN')]
```

Stemming

In [10]:

```
# Performing stemming operation on the text
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

# Store the sentences and words in order to compare later
old_sent = [sent for sent in sentences1]
old_words = [word for word in words1]

# Store the words list to compare later and also store the stemmed words
stemmed = {word : stemmer.stem(word) for word in words1}

# Update the words1 list with stemmed words
words1 = [stemmer.stem(word) for word in words1]

# Stem the words and update the sentence
for i in range(len(sentences1)):
    words = nltk.word_tokenize(sentences1[i])
    words = [stemmer.stem(word) for word in words]
    sentences1[i] = ' '.join(words)

print('First sentence before stemming was: ', old_sent[0])
print('First sentence after stemming is: ', sentences1[0])

# Print the first 15 words before and after stemming
print('\n\nFirst 15 words before and after stemming are: ')
index = 0
for value in stemmed.items():
```

```
for value in stemmed.items():
    print('{0} -> {1}'.format(value[0], value[1]))
    index += 1
    if index >= 15:
        break
```

```
First sentence before stemming was:  millions people india took part annual t
ree planting drive sunday .
First sentence after stemming is:  million peopl india took part annual tree
plant drive sunday .


First 15 words before and after stemming are:
millions -> million
people -> peopl
india -> india
took -> took
part -> part
annual -> annual
tree -> tree
planting -> plant
drive -> drive
sunday -> sunday
. -> .
250 -> 250
million -> million
saplings -> sapl
planted -> plant
```

Lemmatization

In [11]:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

# Store the words list to compare later and also store the lemmatized words
lemmatized = {word : lemmatizer.lemmatize(word) for word in old_words}

# Update the words1 list with stemmed words
words1 = [lemmatizer.lemmatize(word) for word in old_words]

# Lemmatization
# Note that we will perform lemmatization on old sentences which are not stem
for i in range(len(old_sent)):
    words = nltk.word_tokenize(old_sent[i])
    words = [lemmatizer.lemmatize(word) for word in words]
    sentences1[i] = ' '.join(words)

print('First sentence before lemmatization was: ', old_sent[0])
print('First sentence after lemmatization is: ', sentences1[0])

# Print the first 15 words before and after lemmatization
print('\n\nFirst 15 words before and after lemmatization are: ')
index = 0
for value in lemmatized.items():
    print('{0} -> {1}'.format(value[0], value[1]))
    index += 1
    if index >= 15:
        break
```

```
First sentence before lemmatization was:  millions people india took part ann
ual tree planting drive sunday .
First sentence after lemmatization is:  million people india took part annual
tree planting drive sunday .


First 15 words before and after lemmatization are:
millions -> million
```

```
people -> people
india -> india
took -> took
part -> part
annual -> annual
tree -> tree
planting -> planting
drive -> drive
sunday -> sunday
. -> .
250 -> 250
million -> million
saplings -> sapling
planted -> planted
```

Term-Frequency and Inverse Document Frequency

In [12]:

```
# Get the unique words in the word list
word_set = set(words1)
```

In [13]:

```
# First create an index for each word in our word list
index_dict = {}
i = 0
for word in word_set:
    index_dict[word] = i
    i += 1
```

In [14]:

```
# Create a count dictionary to count the number of documents containing the w
def count_dict(sentences):
    word_count = {}
    for word in word_set:
        word_count[word] = 0
        for sent in sentences:
            if word in sent:
                word_count[word] += 1
    return word_count

word_count = count_dict(sentences1)
```

In [15]:

```
# Function to calculate Term frequency (TF)
def term_freq(document, word):
    n = len(document)
    occurance = len([token for token in document if token == word])

    return occurance/n
```

In [16]:

```
# Function to calculate IDF
def inverse_df(word):
    try:
        word_occurance = word_count[word]+1
    except:
        word_occurance = 1
    return np.log(len(sentences1)/word_occurance)
```

In [17]:

```
# Combine Tf and idf functions
def tf_idf(sentence):
    vec = np.zeros((len(word_set),))
    words = nltk.word_tokenize(sentence)
    for word in words:
        tf = term_freq(sentence, word)
        idf = inverse_df(word)
```

```
        idf = inverse_df(word)

        value = tf*idf
        vec[index_dict[word]] = value

    return vec
```

In [23]:

```
# Apply tf-idf encoding to sentences1 corpus
vectors = []
for sent in sentences1:
#     print(sent)
    vec = tf_idf(sent)
    vectors.append(vec)

print(vectors[5])
```

```
281
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
```