practical-7

March 10, 2024

```
[ ]: import nltk
     from nltk.tokenize import word_tokenize
     from nltk.corpus import stopwords
     from nltk.stem import PorterStemmer, WordNetLemmatizer
     from nltk import pos_tag
     from nltk.corpus import wordnet as wn
     from sklearn.feature_extraction.text import TfidfVectorizer
     import re
```

```
[ ]: def preprocess_text(text):

         text = text.lower()
         text = re.sub(r'[^\w\s]', ' ', text)
         return text
```

```
[ ]: text = "i am a student.hello!! there is a session going onn."
```

```
[ ]: preprocessed_document = preprocess_text(text)
     text
```

```
[ ]: 'i am a student.hello!! there is a session going onn.'
```

```
[ ]: nltk.download('punkt')
     def tokenize_text(text):
         tokens = word_tokenize(text)
         return tokens
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
[ ]: tokens = tokenize_text(preprocessed_document)
     tokens
```

```
[ ]: ['i',
      'am',
      'a',
      'student',
      'hello',
      'there',
      'is',
      'a',
      'session',
      'going',
      'onn']
```

```
[ ]: def pos_tag_tokens(tokens):
         pos_tags = pos_tag(tokens)
         return pos_tags
```

```
[ ]: nltk.download('averaged_perceptron_tagger')
     pos_tags = pos_tag_tokens(tokens)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

```
[ ]: pos_tags
```

```
[ ]: [('i', 'NN'),
      ('am', 'VBP'),
      ('a', 'DT'),
      ('student', 'NN'),
      ('hello', 'NN'),
      ('there', 'EX'),
      ('is', 'VBZ'),
      ('a', 'DT'),
      ('session', 'NN'),
      ('going', 'VBG'),
      ('onn', 'NN')]
```

```
[ ]: def remove_stop_words(tokens):
         stop_words = set(stopwords.words('english'))
         filtered_tokens = [word for word in tokens if word.lower() not in␣
      ↪stop_words]
         return filtered_tokens
```

```
[ ]: nltk.download('stopwords')
     filtered_tokens = remove_stop_words(tokens)
```

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

```
[ ]: filtered_tokens
```

```
[ ]: ['student', 'hello', 'session', 'going', 'onn']
```

[ ]:
```python
def stem_tokens(tokens):
    stemmer = PorterStemmer()
    stemmed_tokens = [stemmer.stem(word) for word in tokens]
    return stemmed_tokens
```

[ ]:
```python
stemmed_tokens = stem_tokens(filtered_tokens)
```

[ ]:
```python
stemmed_tokens
```

[ ]:
```
['student', 'hello', 'session', 'go', 'onn']
```

[ ]:
```python
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data…
```

[ ]:
```
True
```

[ ]:
```python
def lemmatize_tokens(tokens):
    lemmatizer = WordNetLemmatizer()

    def get_wordnet_pos(treebank_tag):
        if treebank_tag.startswith('J'):
            return wn.ADJ
        elif treebank_tag.startswith('V'):
            return wn.VERB
        elif treebank_tag.startswith('N'):
            return wn.NOUN
        elif treebank_tag.startswith('R'):
            return wn.ADV
        else:
            return None

    pos_tags = pos_tag(tokens)
    lemmatized_tokens = []
    for word, pos in pos_tags:
        wordnet_pos = get_wordnet_pos(pos) or wn.NOUN
        lemmatized_tokens.append(lemmatizer.lemmatize(word, pos=wordnet_pos))
    return lemmatized_tokens
```

[ ]:
```python
lemmatized_tokens = lemmatize_tokens(tokens)
```

[ ]:
```python
lemmatized_tokens
```

[ ]:
```
['i',
 'be',
 'a',
 'student',
 'hello',
 'there',
 'be',
 'a',
 'session',
 'go',
 'onn']
```

[ ]:
```python
def get_tfidf_representation(documents):
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
    return tfidf_matrix
```

[ ]:
```python
tfidf_matrix = get_tfidf_representation([text])
```

[ ]:
```python
tfidf_matrix
```

[ ]:
```
<1x8 sparse matrix of type '<class 'numpy.float64'>'
        with 8 stored elements in Compressed Sparse Row format>
```

[ ]:
```python
print("Original Tokens:")
print(tokens)
print("\nPOS Tagging:")
print(pos_tags)
print("\nFiltered Tokens after Stop Words Removal:")
print(filtered_tokens)
print("\nStemmed Tokens:")
print(stemmed_tokens)
print("\nLemmatized Tokens:")
print(lemmatized_tokens)
print("\nTF-IDF Representation:")
print(tfidf_matrix)
```

```
Original Tokens:
['i', 'am', 'a', 'student', 'hello', 'there', 'is', 'a', 'session', 'going',
'onn']

POS Tagging:
[('i', 'NN'), ('am', 'VBP'), ('a', 'DT'), ('student', 'NN'), ('hello', 'NN'),
('there', 'EX'), ('is', 'VBZ'), ('a', 'DT'), ('session', 'NN'), ('going',
'VBG'), ('onn', 'NN')]

Filtered Tokens after Stop Words Removal:
['student', 'hello', 'session', 'going', 'onn']

Stemmed Tokens:
['student', 'hello', 'session', 'go', 'onn']
```

```
Lemmatized Tokens:
['i', 'be', 'a', 'student', 'hello', 'there', 'be', 'a', 'session', 'go', 'onn']

TF-IDF Representation:
  (0, 4)        0.35355339059327373
  (0, 1)        0.35355339059327373
  (0, 5)        0.35355339059327373
  (0, 3)        0.35355339059327373
  (0, 7)        0.35355339059327373
  (0, 2)        0.35355339059327373
  (0, 6)        0.35355339059327373
  (0, 0)        0.35355339059327373
```