practical-07

March 8, 2025

Part A

[7]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk import pos_tag
```

[24]:
```
# download below package in not installed

"""
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
nltk.download('wordnet')
"""
```

[24]:
```
"\nnltk.download('punkt')\nnltk.download('averaged_perceptron_tagger')\nnltk.dow
nload('stopwords')\nnltk.download('wordnet')\n"
```

[25]:
```
document = """Natural language processing (NLP) is a subfield of artificial␣
 ↪intelligence (AI) that focuses on the interaction between computers and␣
 ↪humans using natural language. It involves the analysis, understanding, and␣
 ↪generation of human language, enabling machines to process and comprehend␣
 ↪text in a meaningful way. NLP techniques are widely used in various␣
 ↪applications such as sentiment analysis, machine translation, chatbots, and␣
 ↪information retrieval. Preprocessing is an essential step in NLP, which␣
 ↪involves tokenization, part-of-speech tagging, stop words removal, stemming,␣
 ↪and lemmatization."""
```

[26]:
```
# Tokenization

"""
In Python tokenization basically refers to splitting up a larger body of text␣
 ↪into smaller lines, words or even creating words for a non-English language.
"""

tokens = word_tokenize(document)
```

[27]:
```
# POS Tagging

"""
POS Tagging Parts of speech Tagging is responsible for reading the text in a␣
 ↪language and assigning some specific token (Parts of Speech) to each word.
"""

pos_tags = pos_tag(tokens)
```

[28]:
```
# Stop words removal

"""
Stop words removal in Python is a common preprocessing step in Natural Language␣
 ↪Processing (NLP) applications.
Stop words are words that do not add much meaning to a sentence and are␣
 ↪pre-defined and cannot be removed
"""

stop_words = set(stopwords.words('english'))
filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
```

[29]:
```
# Stemming
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
```

[30]:
```
# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]
```

```
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
File ~\AppData\Roaming\Python\Python38\site-packages\nltk\corpus\util.py:84, in␣
 ↪LazyCorpusLoader.__load(self)
     83 try:
---> 84     root = nltk.data.find(f"{self.subdir}/{zip_name}")
     85 except LookupError:

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\data.py:583, in␣
 ↪find(resource_name, paths)
    582 resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 583 raise LookupError(resource_not_found)

LookupError:
**********************************************************************
  Resource omw-1.4 not found.
  Please use the NLTK Downloader to obtain the resource:
```

```
('sentiment', 'NN'), ('analysis', 'NN'), (',', ','), ('machine', 'NN'),
('translation', 'NN'), (',', ','), ('chatbots', 'NNS'), (',', ','), ('and',
'CC'), ('information', 'NN'), ('retrieval', 'NN'), ('.', '.'), ('Preprocessing',
'NNP'), ('is', 'VBZ'), ('an', 'DT'), ('essential', 'JJ'), ('step', 'NN'), ('in',
'IN'), ('NLP', 'NNP'), (',', ','), ('which', 'WDT'), ('involves', 'VBZ'),
('tokenization', 'NN'), (',', ','), ('part-of-speech', 'JJ'), ('tagging', 'NN'),
(',', ','), ('stop', 'VB'), ('words', 'NNS'), ('removal', 'JJ'), (',', ','),
('stemming', 'VBG'), (',', ','), ('and', 'CC'), ('lemmatization', 'NN'), ('.',
'.')]

Filtered Tokens (after stop words removal):
 ['Natural', 'language', 'processing', '(', 'NLP', ')', 'subfield',
'artificial', 'intelligence', '(', 'AI', ')', 'focuses', 'interaction',
'computers', 'humans', 'using', 'natural', 'language', '.', 'involves',
'analysis', ',', 'understanding', ',', 'generation', 'human', 'language', ',',
'enabling', 'machines', 'process', 'comprehend', 'text', 'meaningful', 'way',
'.', 'NLP', 'techniques', 'widely', 'used', 'various', 'applications',
'sentiment', 'analysis', ',', 'machine', 'translation', ',', 'chatbots', ',',
'information', 'retrieval', '.', 'Preprocessing', 'essential', 'step', 'NLP',
',', 'involves', 'tokenization', ',', 'part-of-speech', 'tagging', ',', 'stop',
'words', 'removal', ',', 'stemming', ',', 'lemmatization', '.']

Stemmed Tokens:
 ['natur', 'languag', 'process', '(', 'nlp', ')', 'subfield', 'artifici',
'intellig', '(', 'ai', ')', 'focus', 'interact', 'comput', 'human', 'use',
'natur', 'languag', '.', 'involv', 'analysi', ',', 'understand', ',', 'gener',
'human', 'languag', ',', 'enabl', 'machin', 'process', 'comprehend', 'text',
'meaning', 'way', '.', 'nlp', 'techniqu', 'wide', 'use', 'variou', 'applic',
'sentiment', 'analysi', ',', 'machin', 'translat', ',', 'chatbot', ',',
'inform', 'retriev', '.', 'preprocess', 'essenti', 'step', 'nlp', ',', 'involv',
'token', ',', 'part-of-speech', 'tag', ',', 'stop', 'word', 'remov', ',',
'stem', ',', 'lemmat', '.']
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[31], line 7
      5 print("\nFiltered Tokens (after stop words removal):\n", filtered_tokens)
      6 print("\nStemmed Tokens:\n", stemmed_tokens)
----> 7 print("\nLemmatized Tokens:\n", lemmatized_tokens)

NameError: name 'lemmatized_tokens' is not defined
```

Part B

[32]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[33]:
```
# List of documents
documents = [
    "Natural language processing is a subfield of artificial intelligence.",
    "It focuses on the interaction between computers and humans using natural␣
 ↪language.",
    "NLP techniques are widely used in various applications such as sentiment␣
 ↪analysis and machine translation.",
    "Preprocessing is an essential step in NLP.",
]
```

[34]:
```
# Create an instance of TfidfVectorizer
vectorizer = TfidfVectorizer()
```

[35]:
```
# Fit and transform the documents
tfidf_matrix = vectorizer.fit_transform(documents)
```

[36]:
```
# Get the feature names (terms)
feature_names = vectorizer.get_feature_names_out()
```

[37]:
```
# Print the TF-IDF representation
for i, doc in enumerate(documents):
    print(f"Document {i+1}:")
    for j, term in enumerate(feature_names):
        tfidf_value = tfidf_matrix[i, j]
        if tfidf_value > 0:
            print(f"{term}: {tfidf_value:.4f}")
    print()
```

```
Document 1:
artificial: 0.3817
intelligence: 0.3817
is: 0.3009
language: 0.3009
natural: 0.3009
of: 0.3817
processing: 0.3817
subfield: 0.3817

Document 2:
and: 0.2392
between: 0.3034
computers: 0.3034
focuses: 0.3034
humans: 0.3034
interaction: 0.3034
it: 0.3034
language: 0.2392
```

```
natural: 0.2392
on: 0.3034
the: 0.3034
using: 0.3034

Document 3:
analysis: 0.2686
and: 0.2117
applications: 0.2686
are: 0.2686
as: 0.2686
in: 0.2117
machine: 0.2686
nlp: 0.2117
sentiment: 0.2686
such: 0.2686
techniques: 0.2686
translation: 0.2686
used: 0.2686
various: 0.2686
widely: 0.2686

Document 4:
an: 0.4129
essential: 0.4129
in: 0.3256
is: 0.3256
nlp: 0.3256
preprocessing: 0.4129
step: 0.4129
```

[ ]: