ymlktvll9

April 29, 2025

Part A

[1]:
```
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 sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import numpy as np
```

[2]:
```
"""
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')  # This was missing in the original code
"""
```

[2]:
```
"\nnltk.download('punkt')\nnltk.download('averaged_perceptron_tagger')\nnltk.dow
nload('stopwords')\nnltk.download('wordnet')\nnltk.download('omw-1.4')  # This
was missing in the original code\n"
```

[3]:
```
print("PART A: TEXT PREPROCESSING")
print("-" * 50)
```

```
PART A: TEXT PREPROCESSING
--------------------------------------------------
```

[4]:
```
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."""
```

```
[5]: print("Original Document:")
     print(document)
     print("\n" + "-" * 50)
```

```
Original 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.

--------------------------------------------------
```

```
[6]: # Step 1: Tokenization
     tokens = word_tokenize(document)
     print("\nStep 1: Tokenization")
     print(f"Total tokens: {len(tokens)}")
     print("First 15 tokens:", tokens[:15])
     print("\n" + "-" * 50)
```

```
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
Cell In[6], line 2
      1 # Step 1: Tokenization
----> 2 tokens = word_tokenize(document)
      3 print("\nStep 1: Tokenization")
      4 print(f"Total tokens: {len(tokens)}")

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\nltk\tokenize\__init__.
 ↪py:142, in word_tokenize(text, language, preserve_line)
    127 def word_tokenize(text, language="english", preserve_line=False):
    128     """
    129     Return a tokenized copy of *text*,
    130     using NLTK's recommended word tokenizer
   (…)
    140     :type preserve_line: bool
    141     """
--> 142     sentences = [text] if preserve_line else␣
 ↪sent_tokenize(text, language)
    143     return [
    144         token for sent in sentences for token in␣
 ↪_treebank_word_tokenizer.tokenize(sent)
    145     ]
```

```
File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\nltk\tokenize\__i it__.
 ↪py:119, in sent_tokenize(text, language)
    109 def sent_tokenize(text, language="english"):
    110     """
    111     Return a sentence-tokenized copy of *text*,
    112     using NLTK's recommended sentence tokenizer
   (…)
    117     :param language: the model name in the Punkt corpus
    118     """
--> 119     tokenizer = _get_punkt_tokenizer(language)
    120     return tokenizer.tokenize(text)

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\nltk\tokenize\__i it__.
 ↪py:105, in _get_punkt_tokenizer(language)
     96 @functools.lru_cache
     97 def _get_punkt_tokenizer(language="english"):
     98     """
     99     A constructor for the PunktTokenizer that utilizes
    100     a lru cache for performance.
   (…)
    103     :type language: str
    104     """
--> 105     return PunktTokenizer(language)

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\nltk\tokenize\pun t.
 ↪py:1744, in PunktTokenizer.__init__(self, lang)
   1742 def __init__(self, lang="english"):
   1743     PunktSentenceTokenizer.__init__(self)
-> 1744     self.load_lang(lang)

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\nltk\tokenize\pun t.
 ↪py:1749, in PunktTokenizer.load_lang(self, lang)
   1746 def load_lang(self, lang="english"):
   1747     from nltk.data import find
-> 1749     lang_dir = find(f"tokenizers/punkt_tab/{lang}/")
   1750     self._params = load_punkt_params(lang_dir)
   1751     self._lang = lang

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\nltk\data.py:
 ↪579, in find(resource_name, paths)
    577 sep = "*" * 70
    578 resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579 raise LookupError(resource_not_found)
```

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

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

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

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - 'C:\\Users\\Hp/nltk_data'
    - 'C:\\Users\\Hp\\AppData\\Local\\Programs\\Python\\Python311\\nltk_data'
    - 'C:
↪\\Users\\Hp\\AppData\\Local\\Programs\\Python\\Python311\\share\\nltk_data'
    - 'C:
↪\\Users\\Hp\\AppData\\Local\\Programs\\Python\\Python311\\lib\\nltk_data'
    - 'C:\\Users\\Hp\\AppData\\Roaming\\nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
**********************************************************************
```

[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:

  >>> import nltk
  >>> nltk.download('omw-1.4')

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

  Attempted to load corpora/omw-1.4.zip/omw-1.4/

  Searched in:
    - 'C:\\Users\\UNIQUE/nltk_data'
    - 'D:\\Python\\nltk_data'
    - 'D:\\Python\\share\\nltk_data'
    - 'D:\\Python\\lib\\nltk_data'
    - 'C:\\Users\\UNIQUE\\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)
Cell In[30], line 3
      1 # Lemmatization
      2 lemmatizer = WordNetLemmatizer()
----> 3 lemmatized_tokens = [lemmatizer.lemmatize(token) for token in␣
 ↪filtered_tokens]

Cell In[30], line 3, in <listcomp>(.0)
      1 # Lemmatization
      2 lemmatizer = WordNetLemmatizer()
----> 3 lemmatized_tokens = [lemmatizer.lemmatize(token) for token in␣
 ↪filtered_tokens]

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\stem\wordnet.py:45, i␣
 ↪WordNetLemmatizer.lemmatize(self, word, pos)
     33 def lemmatize(self, word: str, pos: str = "n") -> str:
     34     """Lemmatize `word` using WordNet's built-in morphy function.
     35     Returns the input word unchanged if it cannot be found in WordNet.
     36
   (…)
     43     :return: The lemma of `word`, for the given `pos`.
     44     """
---> 45     lemmas = wn._morphy(word, pos)
     46     return min(lemmas, key=len) if lemmas else word

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\corpus\util.py:121, i␣
 ↪LazyCorpusLoader.__getattr__(self, attr)
    118 if attr == "__bases__":
    119     raise AttributeError("LazyCorpusLoader object has no attribute␣
 ↪'__bases__'")
--> 121 self.__load()
    122 # This looks circular, but its not, since __load() changes our
    123 # __class__ to something new:
    124 return getattr(self, attr)

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\corpus\util.py:89, in
 ↪LazyCorpusLoader.__load(self)
     86             raise e
     88 # Load the corpus.
---> 89 corpus = self.__reader_cls(root, *self.__args, **self.__kwargs)
     91 # This is where the magic happens!  Transform ourselves into
     92 # the corpus by modifying our own __dict__ and __class__ to
     93 # match that of the corpus.
     95 args, kwargs = self.__args, self.__kwargs

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\corpus\reader\wordnet
 ↪py:1176, in WordNetCorpusReader.__init__(self, root, omw_reader)
```

```
   1172     warnings.warn(
   1173         "The multilingual functions are not available with this Wordnet␣
 ↪version"
   1174     )
   1175 else:
-> 1176     self.provenances = self.omw_prov()
   1178 # A cache to store the wordnet data of multiple languages
   1179 self._lang_data = defaultdict(list)

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\corpus\reader\wordnet
 ↪py:1285, in WordNetCorpusReader.omw_prov(self)
   1283 provdict = {}
   1284 provdict["eng"] = ""
-> 1285 fileids = self._omw_reader.fileids()
   1286 for fileid in fileids:
   1287     prov, langfile = os.path.split(fileid)

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\corpus\util.py:121, in␣
 ↪LazyCorpusLoader.__getattr__(self, attr)
    118 if attr == "__bases__":
    119     raise AttributeError("LazyCorpusLoader object has no attribute␣
 ↪'__bases__'")
--> 121 self.__load()
    122 # This looks circular, but its not, since __load() changes our
    123 # __class__ to something new:
    124 return getattr(self, attr)

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\corpus\util.py:86, in␣
 ↪LazyCorpusLoader.__load(self)
     84             root = nltk.data.find(f"{self.subdir}/{zip_name}")
     85         except LookupError:
---> 86             raise e
     88 # Load the corpus.
     89 corpus = self.__reader_cls(root, *self.__args, **self.__kwargs)

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\corpus\util.py:81, in␣
 ↪LazyCorpusLoader.__load(self)
     79 else:
     80     try:
---> 81         root = nltk.data.find(f"{self.subdir}/{self.__name}")
     82     except LookupError as e:
     83         try:

File ~\AppData\Roaming\Python\Python38\site-packages\nltk\data.py:583, in␣
 ↪find(resource_name, paths)
    581 sep = "*" * 70
    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:

  >>> import nltk
  >>> nltk.download('omw-1.4')

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

  Attempted to load corpora/omw-1.4

  Searched in:
    - 'C:\\Users\\UNIQUE/nltk_data'
    - 'D:\\Python\\nltk_data'
    - 'D:\\Python\\share\\nltk_data'
    - 'D:\\Python\\lib\\nltk_data'
    - 'C:\\Users\\UNIQUE\\AppData\\Roaming\\nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
**********************************************************************
```

[31]:
```python
# Print the results
print("Original Document:\n", document)
print("\nTokens:\n", tokens)
print("\nPOS Tags:\n", pos_tags)
print("\nFiltered Tokens (after stop words removal):\n", filtered_tokens)
print("\nStemmed Tokens:\n", stemmed_tokens)
print("\nLemmatized Tokens:\n", lemmatized_tokens)
```

```
Original 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.

Tokens:
 ['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', '.']

POS Tags:
 [('Natural', 'JJ'), ('language', 'NN'), ('processing', 'NN'), ('(', '('),
('NLP', 'NNP'), (')', ')'), ('is', 'VBZ'), ('a', 'DT'), ('subfield', 'NN'),
('of', 'IN'), ('artificial', 'JJ'), ('intelligence', 'NN'), ('(', '('), ('AI',
'NNP'), (')', ')'), ('that', 'WDT'), ('focuses', 'VBZ'), ('on', 'IN'), ('the',
'DT'), ('interaction', 'NN'), ('between', 'IN'), ('computers', 'NNS'), ('and',
'CC'), ('humans', 'NNS'), ('using', 'VBG'), ('natural', 'JJ'), ('language',
'NN'), ('.', '.'), ('It', 'PRP'), ('involves', 'VBZ'), ('the', 'DT'),
('analysis', 'NN'), (',', ','), ('understanding', 'NN'), (',', ','), ('and',
'CC'), ('generation', 'NN'), ('of', 'IN'), ('human', 'JJ'), ('language', 'NN'),
(',', ','), ('enabling', 'VBG'), ('machines', 'NNS'), ('to', 'TO'), ('process',
'VB'), ('and', 'CC'), ('comprehend', 'VB'), ('text', 'NN'), ('in', 'IN'), ('a',
'DT'), ('meaningful', 'JJ'), ('way', 'NN'), ('.', '.'), ('NLP', 'NNP'),
('techniques', 'NNS'), ('are', 'VBP'), ('widely', 'RB'), ('used', 'VBN'), ('in',
'IN'), ('various', 'JJ'), ('applications', 'NNS'), ('such', 'JJ'), ('as', 'IN'),
('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_token
      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
```

```
[ ]:
```