Muhammad Tahir

K214503

NLP LAB

Tokenization

```
import nltk

nltk.download('punkt_tab')
```

```
text = "I am Muhammad Tahir, and currently in 7th semester"

from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
['I', 'am', 'Muhammad', 'Tahir', ',', 'and', 'currently', 'in', '7th', 'semester']
```

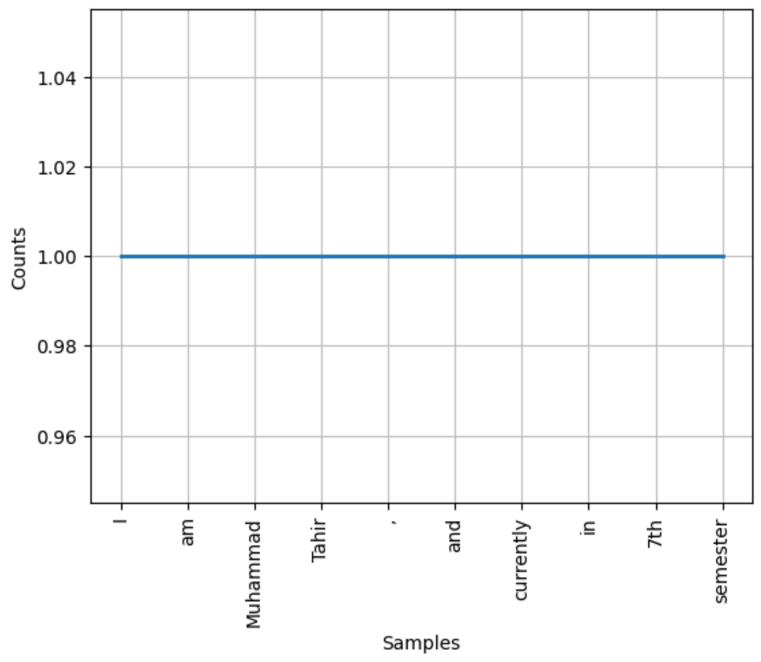
```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
```

```
<FreqDist with 10 samples and 10 outcomes>
```

```
fdist.most_common(2)
```

```
[('I', 1), ('am', 1)]
```

```
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



Tokenize Non-English Languages Text

```
!pip install nltk
import nltk
nltk.download('punkt')
nltk.download('stopwords')

from nltk.tokenize import sent_tokenize
```

```
mytext = "Bonjour M. Adam, comment allez-vous? J'espère que tout va bien. Aujourd'hui est un bon jour."
print(sent_tokenize(mytext, "french"))
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.6)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['Bonjour M. Adam, comment allez-vous?', "J'espère que tout va bien.", "Aujourd'hui est un bon jour."]
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Stopwords

```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{'any', "you've", "isn't", "aren't", 'each', 'll', 'ain', 'no', "hasn't", 'after', 'what', "couldn't", 're', 'same', 'their', 'on',
```

```
filtered_sent=[]
for w in tokenized_word:
    if w not in stop_words:  # Indented this line
        filtered_sent.append(w)  # Indented this line
print("Tokenized Sentence:",tokenized_word)
print("Filterd Sentence:",filtered_sent)
```

```
Tokenized Sentence: ['I', 'am', 'Muhammad', 'Tahir', ',', 'and', 'currently', 'in', '7th', 'semester']
Filterd Sentence: ['I', 'Muhammad', 'Tahir', ',', 'currently', '7th', 'semester']
```

Get Synonyms From WordNet

```
from nltk.corpus import wordnet
syn = wordnet.synsets("pain")
print(syn[0].definition())
print(syn[0].examples())
```

```
a symptom of some physical hurt or disorder
['the patient developed severe pain and distension']
```

```
import nltk

# Download the 'wordnet' data package
nltk.download('wordnet')

from nltk.corpus import wordnet

syn = wordnet.synsets("pain")
print(syn[0].definition())
print(syn[0].examples())
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
a symptom of some physical hurt or disorder
['the patient developed severe pain and distension']
```

Get Antonyms From WordNet

```
from nltk.corpus import wordnet

antonyms = []
for syn in wordnet.synsets("small"):
    # Indent the code block within the outer 'for' loop
    for l in syn.lemmas():
        if l.antonyms():
            antonyms.append(l.antonyms()[0].name())

print(antonyms)
```

```
['large', 'big', 'big']
```

NLTK Word Stemming

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
print(stemmer.stem("working"))
```

```
work
```

Lemmatizing Words Using WordNet

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
print(stemmer.stem("increases"))
# The result is: increas.

# When we lemmatize the same word using NLTK WordNet, the result is increase:
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize('increases'))
```

```
increas
increase
```

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize('playing', pos="v"))
print(lemmatizer.lemmatize('playing', pos="n"))
print(lemmatizer.lemmatize('playing', pos="a"))
print(lemmatizer.lemmatize('playing', pos="r"))
```

```
play
playing
playing
playing
```

Part of speech tagging (POS)

```
import nltk
from nltk.tokenize import word_tokenize

# Download the required resource
nltk.download('averaged_perceptron_tagger_eng')

text = "vote to choose a particular man or a group (party) to represent them in parliament"
tex = word_tokenize(text)  # Tokenize the text
for token in tex:
  print(nltk.pos_tag([token]))
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
[('vote', 'NN')]
[('to', 'TO')]
[('choose', 'NN')]
[('a', 'DT')]
[('particular', 'JJ')]
[('man', 'NN')]
[('or', 'CC')]
[('a', 'DT')]
[('group', 'NN')]
[('(', '(')]
[('party', 'NN')]
[(')', ')')]
[('to', 'TO')]
[('represent', 'NN')]
[('them', 'PRP')]
[('in', 'IN')]
[('parliament', 'NN')]
```

Named entity recognition

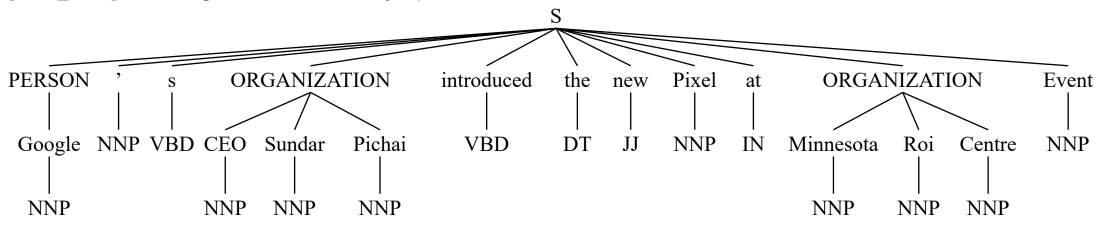
```
!pip install svgling

import nltk
nltk.download('maxent_ne_chunker_tab')
nltk.download('words')
from nltk import ne_chunk # tokenize and POS Tagging before doing chunk

text = "Google’s CEO Sundar Pichai introduced the new Pixel at Minnesota Roi Centre Event" #importing chunk library from nltk
token = word_tokenize(text)
tags = nltk.pos_tag(token)
```

```
chunk = ne_chunk(tags)
chunk
```

```
Collecting svgling
  Downloading svgling-0.5.0-py3-none-any.whl.metadata (7.4 kB)
Collecting svgwrite (from svgling)
  Downloading svgwrite-1.4.3-py3-none-any.whl.metadata (8.8 kB)
Downloading svgling-0.5.0-py3-none-any.whl (31 kB)
Downloading svgwrite-1.4.3-py3-none-any.whl (67 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.1/67.1 kB 4.1 MB/s eta 0:00:00
Installing collected packages: svgwrite, svgling
Successfully installed svgling-0.5.0 svgwrite-1.4.3
[nltk_data] Downloading package maxent_ne_chunker_tab to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker_tab is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
```



Chunking

```
text = "We saw the yellow dog"
token = word_tokenize(text)
tags = nltk.pos_tag(token)
reg = "NP: {<DT>?<JJ>*<NN>}"
a = nltk.RegexpParser(reg)
result = a.parse(tags)
print(result)
```

```
(S We/PRP saw/VBD (NP the/DT yellow/JJ dog/NN))
```

One-hot encoding (CountVectorizing)

```
!pip install scikit-learn

from sklearn.feature_extraction.text import CountVectorizer # Import the CountVectorizer class

corpus = [
'This is the first document.',
'This document is the second document.',
'And this is the third one.',
'Is this the first document?',
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names_out()) # Updated to get_feature_names_out() for newer versions of scikit-learn
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
print(X.toarray())
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
['and' 'document' 'first' 'is' 'one' 'second' 'the' 'third' 'this']
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
sample_text = ["One of the most basic ways we can numerically represent words "
"is through the one-hot encoding method (also sometimes called "
"count vectorizing)."]
# To actually create the vectorizer, we simply need to call fit on the text
# data that we wish to fix
vectorizer.fit(sample_text)
# Now, we can inspect how our vectorizer vectorized the text
# This will print out a list of words used, and their index in the vectors
print('Vocabulary: ')
print(vectorizer.vocabulary_)
```

```
Vocabulary:
{'one': 12, 'of': 11, 'the': 15, 'most': 9, 'basic': 1, 'ways': 18, 'we': 19, 'can': 3, 'numerically': 10, 'represent': 13, 'words'
```

```
# If we would like to actually create a vector, we can do so by passing the
# text into the vectorizer to get back counts
vector = vectorizer.transform(sample_text)

# Our final vector:
print('Full vector: ')
print(vector.toarray())

# Or if we wanted to get the vector for one word:
print('Hot vector: ')
print(vectorizer.transform(['hot']).toarray())

# Or if we wanted to get multiple vectors at once to build matrices
print('Hot, one and Today: ')
print(vectorizer.transform(['hot', 'one', 'of']).toarray())
```

```
Full vector:
[[1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1]]
Hot vector:
[[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
Hot, one and Today:
[[0 0 0 0 0 0 1 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 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]]
```

Example II

```
# We could also do the whole thing at once with the fit_transform method:
print('One swoop:')
new_text = ['Today is the day that I do the thing today, today']
new_vectorizer = CountVectorizer()
print(new_vectorizer.fit_transform(new_text).toarray())
```

```
One swoop:
[[1 1 1 1 2 1 3]]
```

Word Frequencies with TfidfVectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer
# list of text documents
text = ["The quick brown fox jumped over the lazy dog.",
"The dog.",
"The fox"]
# create the transform
vectorizer = TfidfVectorizer()
# tokenize and build vocab
vectorizer.fit(text)

# summarize
print(vectorizer.vocabulary_)
print(vectorizer.idf_)
# encode document
vector = vectorizer.transform([text[0]])
# summarize encoded vector
print(vector.shape)
print(vector.toarray())
```

```
{'the': 7, 'quick': 6, 'brown': 0, 'fox': 2, 'jumped': 3, 'over': 5, 'lazy': 4, 'dog': 1}
[1.69314718 1.28768207 1.28768207 1.69314718 1.69314718 1.69314718
 1.69314718 1.        ]
(1, 8)
[[0.36388646 0.27674503 0.27674503 0.36388646 0.36388646 0.36388646
  0.36388646 0.42983441]]
```