Natural Language Processing

Corpora Analysis and Text processing using NLTK.

Lallith Prasath

1. Utilize Python NLTK (Natural Language Tool Kit) Platform and do the following. Install relevant Packages and Libraries

```
!pip install nltk

import nltk
nltk.download('brown')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.8/dist-packages (3.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (from nltk) (1.2.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from nltk) (4.64.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.8/dist-packages (from nltk) (2022.6.2)
Requirement already satisfied: click in /usr/local/lib/python3.8/dist-packages (from nltk) (7.1.2)
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Package brown is already up-to-date!
True
```

Explore Brown Corpus and find the size, tokens, categories

```
from nltk.corpus import brown

print("Categories present in the brown Corpus:\n")
```



```
Categories present in the brown Corpus:

['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion
```

```
print("Number of words in each category:\n")
for category in brown.categories():
    print(category + ': ' + str(len(brown.words(categories=category))))
```

```
Number of words in each category:

adventure: 69342
belles_lettres: 173096
editorial: 61604
fiction: 68488
government: 70117
hobbies: 82345
humor: 21695
learned: 181888
lore: 110299
mystery: 57169
news: 100554
religion: 39399
reviews: 40704
romance: 70022
science_fiction: 14470
```

```
print("Words present in the brown Corpus:\n")
print(brown.words())
```

```
Words present in the brown Corpus:

['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

Find the size of word tokens?

```
print("Size of word tokens in brown Corpus:\n")
word_token = len(brown.words())
print(word_token)
```

Saved successfully!

```
orpus:

1161192
```

Find the size of word types?

```
print("Size of word types in brown Corpus:\n")
word_type = len(set(brown.words()))
print(word_type)
```

```
Size of word types in brown Corpus:

56057
```

Find the size of the category “government”

```
print("The size of word tokens in category 'government' :\n")
govt_token=brown.words(categories='government')
len(govt_token)
```

```
The size of word tokens in category 'government' :

70117
```

```
print("The size of word types in category 'government' :\n")
govt_type = len(set(brown.words(categories='government')))
print(govt_type)
```

```
The size of word types in category 'government' :

8181
```

List the most frequent tokens

```
print("The 35 most frequently occuring tokens in the brown Corpus:\n")
nltk.FreqDist(brown.words()).most_common(35)
```

Saved successfully!

```
g tokens in the brown Corpus:

[('the', 62713),
 (',', 58334),
 ('.', 49346),
 ('of', 36080),
 ('and', 27915),
 ('to', 25732),
 ('a', 21881),
 ('in', 19536),
```

```
('that', 10237),
('is', 10011),
('was', 9777),
('for', 8841),
('``', 8837),
("''", 8789),
('The', 7258),
('with', 7012),
('it', 6723),
('as', 6706),
('he', 6566),
('his', 6466),
('on', 6395),
('be', 6344),
(';', 5566),
('I', 5161),
('by', 5103),
('had', 5102),
('at', 4963),
('?', 4693),
('not', 4423),
('are', 4333),
('from', 4207),
('or', 4118),
('this', 3966),
('have', 3892),
('an', 3542)]
```

Count the number of sentences

```
print("The number of sentences in the brown Corpus:\n")
sentence = len(brown.sents())
print(sentence)
```

```
The number of sentences in the brown Corpus:
```

Saved successfully!

2. Explore the corpora available in NLTK (any two)

• Raw corpus • POS tagged • Parsed • Multilingual aligned • Spoken language • Semantic tagged

2.1 Spoken Language: Switchboard Corpus

The Switchboard corpus, consisting of telephone conversations between speakers of American English, is one of the longest-standing corpora of fully spontaneous speech.

Switchboard is a collection of about 2,400 two-sided telephone conversations among 543 speakers (302 male, 241 female) from all areas of the United States. As such, there have been a range of different sorts of linguistic information annotated on it, including syntax, discourse semantics and prosody which makes it suitable for a variety of natural language processing and speech recognition tasks.

```
nltk.download('switchboard')
from nltk.corpus import switchboard
```

```
[nltk_data] Downloading package switchboard to /root/nltk_data...
[nltk_data]   Package switchboard is already up-to-date!
```

```
print(switchboard.words()[0:100])
```

```
['Uh', ',', 'do', 'you', 'have', 'a', 'pet', 'Randy', '?', 'Uh', ',', 'yeah', ',', 'currently', 'we', 'have', 'a', 'poodle', '.', 'A', '
```

```
dialogues = switchboard.raw()
print(dialogues[0:3000])
```

```
A.1: Uh/UH ,/, do/VBP you/PRP have/VB a/DT pet/NN Randy/NNP ?/.
B.2: Uh/UH ,/, yeah/UH ,/, currently/RB we/PRP have/VBP a/DT poodle/NN ./.
A.3: A/DT poodle/NN ,/, miniature/JJ or/CC ,/, uh/UH ,/, full/JJ size/NN ?/.
B.4: Yeah/UH ,/, uh/UH ,/, it/PRP 's/BES ,/, uh/UH miniature/JJ ./.
A.5: Uh-huh/UH ./.
B.6: Yeah/UH ./.
A.7: I/PRP read/VBD somewhere/RB that/IN ,/, the/DT poodles/NNS is/VBZ one/CD of/IN the/DT ,/, the/DT most/RBS intelligent/JJ dogs/NNS ,/
B.8: Well/UH ,/, um/UH ,/, I/PRP would/MD n't/RB ,/, uh/UH ,/, I/PRP definitely/RB would/MD n't/RB dispute/VB that/IN ,/, it/PRP ,/, it/P
A.9: Oh/UH ,/, uh-huh/UH ./. So/RB ,/, you/PRP ,/, you/PRP 've/VBP only/RB known/VBN the/DT dog/NN ,/, wh-/XX ,/, how/WRB long/JJ did/VBD
B.10: Well/UH ,/, about/RB a/DT year/NN I/PRP guess/VBP ./.
A.11: Oh/UH ,/, well/UH ,/, uh/UH ,/, is/VBZ it/PRP ,/, uh/UH ,/, uh/UH ,/, how/WRB old/JJ is/VBZ the/DT dog/NN ?/.
two/CD ,/, I/PRP believe/VBP ./.
still/RB just/RB a/DT pup/NN ./.
ah/UH ,/, yeah/UH ./.
A.15: Yeah/UH ,/, I/PRP have/VBP a/DT ,/, uh/UH ,/, well/UH a/DT mutt/NN ,/, myself/PRP ./. I/PRP call/VBP it/PRP a/DT ,/, uh/UH ,/, uh/U
B.16: Okay/UH ./.
A.17: It/PRP 's/BES ,/, uh/UH ,/, part/NN Chow/NNP and/CC part/NN Shepherd/NNP and/CC it/PRP ,/, as/IN I/PRP understand/VBP it/PRP ,/, uh
B.18: Oh/UH ,/, that/DT sounds/VBZ interesting/JJ ./.
A.19: She/PRP has/VBZ the/DT ,/, the/DT color/NN and/CC the/DT black/JJ to-/NN ,/, tongue/NN of/IN a/DT Chow/NNP ,/, but/CC ,/, uh/UH ,/
B.20: Oh/UH ,/, that/DT 's/BES ,/, that/DT 's/BES neat/JJ ./. How/WRB ,/, about/RB how/WRB big/JJ then/RB ?/.
A.21: Oh/UH ,/, she/PRP weighs/VBZ in/RP at/IN about/RB fifty/CD pounds/NNS ,/, so/RB she/PRP 's/BES a/DT medium/JJ size/NN ./.
B.22: Yeah/UH ,/, yeah/UH ./.
```

Saved successfully!

```
A.23: But/CC she/PRP 's/BES big/JJ enough/RB to/TO be/VB intimidating/JJ ,/,
B.24: Most/JJS definitely/RB ./.
A.25: it/PRP is/VBZ a/DT fi/VBN ,/,
```

2.2 POS tagged: CoNLL 2000 Corpus

The CoNLL 2000 corpus is a collection of 2000 documents manually annotated with part-of-speech tags and chunk tags. It contains around 270k words of Wall Street Journal text, divided into training and testing portions, in the IOB format widely used data for noun phrase chunking:

sections 15-18 as training data (211727 tokens) and section 20 as test data (47377 tokens).

```
nltk.download('conll2000')
nltk.download('universal_tagset')

from nltk.corpus import conll2000

word_tokens = conll2000.words()[0:50]
print(word_tokens)
```

```
['Confidence', 'in', 'the', 'pound', 'is', 'widely', 'expected', 'to', 'take', 'another', 'sharp', 'dive', 'if', 'trade', 'figures', 'fo
[nltk_data] Downloading package conll2000 to /root/nltk_data...
[nltk_data]   Package conll2000 is already up-to-date!
```

```
sentences = conll2000.chunked_sents()
print(sentences)
```

```
[Tree('S', [Tree('NP', [('Confidence', 'NN')]), Tree('PP', [('in', 'IN')]), Tree('NP', [('the', 'DT'), ('pound', 'NN')]), Tree('VP', [('
```

```
from nltk.corpus import conll2000, switchboard
print(conll2000.tagged_sents(tagset='universal')[0:30])
```

Saved successfully!

```
[('Confidence', 'NOUN'), ('in', 'ADP'), ('the', 'DET'), ('pound', 'NOUN'), ('is', 'VERB'), ('widely', 'ADV'), ('expected', 'VERB'), ('to
[nltk_data] Downloading package universal_tagset to /root/nltk_data...
[nltk_data]   Package universal_tagset is already up-to-date!
```

3. Create a text corpus with a minimum of 200 words (unique content). Implement the following text processing.

```
from nltk.corpus import PlaintextCorpusReader
from google.colab import drive
drive.mount('/content/gdrive')

path = 'gdrive/My Drive/NLP'
file_type = '.*\.txt'

corpus = PlaintextCorpusReader(path, file_type)
print(corpus.fileids())
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
['wine.txt']
```

```
txt = corpus.raw('wine.txt')
print(txt)
```

```
According to experts, the wine is differentiated according to its smell, flavor, and color, but we are not a wine expert to say that wine
The excellence of New Zealand Pinot noir wines is well-known worldwide. We utilised 18 Pinot noir wine samples with 54 different characte
```

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
                          is already up-to-date!
                        e omw-1.4 to /root/nltk_data...
                          is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
True
```

Saved successfully!

Word Segmentation

```
from nltk.tokenize import word_tokenize

words = word_tokenize(txt)
print(words)
```

```
['According', 'to', 'experts', ',', 'the', 'wine', 'is', 'differentiated', 'according', 'to', 'its', 'smell', ',', 'flavor', ',', 'and',
```

Sentence Segmentation

```
from nltk.tokenize import sent_tokenize

sentences = sent_tokenize(txt)
print(sentences)
```

```
['According to experts, the wine is differentiated according to its smell, flavor, and color, but we are not a wine expert to say that w
```

Convert to Lowercase

```
lowercase_words = [word.lower() for word in words]
print(lowercase_words)
```

```
['according', 'to', 'experts', ',', 'the', 'wine', 'is', 'differentiated', 'according', 'to', 'its', 'smell', ',', 'flavor', ',', 'and',
```

```
from nltk.corpus import stopwords

nglish'))
filtered_words = [word for word in lowercase_words if word not in stop_words]
print(filtered_words)
```

```
['according', 'experts', ',', 'wine', 'differentiated', 'according', 'smell', ',', 'flavor', ',', 'color', ',', 'wine', 'expert', 'say',
```

Saved successfully!

Stemming

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(word) for word in filtered_words]
print(stemmed_words)
```

```
['accord', 'expert', ',', 'wine', 'differenti', 'accord', 'smell', ',', 'flavor', ',', 'color', ',', 'wine', 'expert', 'say', 'wine', 'g
```

Lemmatization

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_words]
print(lemmatized_words)
```

```
['according', 'expert', ',', 'wine', 'differentiated', 'according', 'smell', ',', 'flavor', ',', 'color', ',', 'wine', 'expert', 'say',
```

Part of speech tagger

```
from nltk.tag import pos_tag

tagged_words = pos_tag(lemmatized_words)
print(tagged_words)
```

```
[('according', 'VBG'), ('expert', 'NN'), (',', ','), ('wine', 'NN'), ('differentiated', 'VBD'), ('according', 'VBG'), ('smell', 'NN'), (
```

Saved successfully!

Saved successfully!