Aim: Implement problems on natural language processing - Part of Speech tagging, N-gram & smoothening and Chunking using NLTK

Short notes:

The **Natural Language Toolkit (NLTK)** is a platform used for building programs for text analysis. One of the more powerful aspects of the NLTK module is the Part of Speech tagging.

Part-of-speech (POS) tagging is a process of converting a sentence to forms – list of words, list of tuples (where each tuple is having a form (word, tag)). The tag in case of is a part-of-speech tag, and signifies whether the word is a noun, adjective, verb, and so on.

keywords:

Corpus : Body of text, singular. Corpora is the plural of this.

Lexicon : Words and their meanings.

Token : Each “entity” that is a part of whatever was split up based on rules.

Tags and their meanings

CD cardinal digit

EX existential there (like: “there is” … think of it like “there exists”)

FW foreign word

IN preposition/subordinating conjunction

JJ adjective ‘big’

JJR adjective, comparative ‘bigger’

JJS adjective, superlative ‘biggest’

NN noun, singular ‘desk’

NNS noun plural ‘desks’

NNP proper noun, singular ‘Harrison’

NNPS proper noun, plural ‘Americans’

PDT predeterminer ‘all the kids’

POS possessive ending parent‘s

PRP personal pronoun I, he, she

PRP$ possessive pronoun my, his, hers

RB adverb very, silently,

RBR adverb, comparative better

RBS adverb, superlative best

RP particle give up

N-grams are continuous sequences of words or symbols or tokens in a document. In technical terms, they can be defined as the neighbouring sequences of items in a document.

Steps for n-gram model:

Explore the dataset

Feature extraction

Train-test split

Basic pre-processing

Code to generate N-grams

Creating unigrams

Creating bigrams

Creating trigrams

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

stop_words = set(stopwords.words('english'))

#Dummy text
txt = "Hello. MCA S3 is fantastic. We learn many new concepts and implement them in our pr
"1st of all the data science is a new paper."

# sent_tokenize is one of instances of
# PunktSentenceTokenizer from the nltk.tokenize.punkt module

tokenized = sent_tokenize(txt)
for i in tokenized:

    # Word tokenizers is used to find the words
    # and punctuation in a string
    wordsList = nltk.word_tokenize(i)

    # removing stop words from wordList
    wordsList = [w for w in wordsList if not w in stop_words]

    #  Using a Tagger. Which is part-of-speech
    # tagger or POS-tagger.
    tagged = nltk.pos_tag(wordsList)
```

```
print(tagged)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[('Hello', 'NNP'), ('.', '.')]
[('MCA', 'NNP'), ('S3', 'NNP'), ('fantastic', 'JJ'), ('.', '.')]
[('We', 'PRP'), ('learn', 'VBP'), ('many', 'JJ'), ('new', 'JJ'), ('concepts', 'NNS'),
[('1st', 'CD'), ('data', 'NNS'), ('science', 'NN'), ('new', 'JJ'), ('paper', 'NN'), (
```

N-gram model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use(style='seaborn')

#get the data from https://www.kaggle.com/ankurzing/sentiment-analysis-for-financial-news/

colnames=['Sentiment', 'news']

df=pd.read_csv('all-data.csv',encoding = "ISO-8859-1", names=colnames, header = None)
df.head()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4846 entries, 0 to 4845
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Sentiment  4846 non-null   object
```

```
      1   news       4846 non-null   object
     dtypes: object(2)
     memory usage: 75.8+ KB
```

```
df['Sentiment'].value_counts()
```

```
     neutral     2879
     positive    1363
     negative     604
     Name: Sentiment, dtype: int64
```

```
y=df['Sentiment'].values
y.shape
```

```
     (4846,)
```

```
x=df['news'].values
x.shape
```

```
     (4846,)
```

```
from sklearn.model_selection import train_test_split
(x_train,x_test,y_train,y_test)=train_test_split(x,y,test_size=0.4)
x_train.shape
y_train.shape
x_test.shape
y_test.shape
```

```
     (1939,)
```

```
df1=pd.DataFrame(x_train)
df1=df1.rename(columns={0:'news'})
df2=pd.DataFrame(y_train)
df2=df2.rename(columns={0:'sentiment'})
df_train=pd.concat([df1,df2],axis=1)
df_train.head()
```

```
df3=pd.DataFrame(x_test)
```

```
df3=df3.rename(columns={0:'news'})
df4=pd.DataFrame(y_test)
df4=df2.rename(columns={0:'sentiment'})
df_test=pd.concat([df3,df4],axis=1)
df_test.head()
```

```
#removing punctuations
#library that contains punctuation
import string
string.punctuation
```

```
#defining the function to remove punctuation
def remove_punctuation(text):
  if(type(text)==float):
    return text
  ans=""
  for i in text:
    if i not in string.punctuation:
      ans+=i
  return ans
```

```
#storing the puntuation free text in a new column called clean_msg
df_train['news']= df_train['news'].apply(lambda x:remove_punctuation(x))
df_test['news']= df_test['news'].apply(lambda x:remove_punctuation(x))
df_train.head()
#punctuations are removed from news column in train dataset
```

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

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

```
#method to generate n-grams:
#params:
#text-the text for which we have to generate n-grams
#ngram-number of grams to be generated from the text(1,2,3,4 etc., default value=1)
def generate_N_grams(text,ngram=1):
  words=[word for word in text.split(" ") if word not in set(stopwords.words('english'))]
  print("Sentence after removing stopwords:",words)
  temp=zip(*[words[i:] for i in range(0,ngram)])
  ans=[' '.join(ngram) for ngram in temp]
  return ans
```

```
generate_N_grams("The sun rises in the east",2)
```

```
Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']
['The sun', 'sun rises', 'rises east']
```

```
generate_N_grams("The sun rises in the east",3)
```

```
Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']
['The sun rises', 'sun rises east']
```

```
generate_N_grams("The sun rises in the east",4)
```

```
Sentence after removing stopwords: ['The', 'sun', 'rises', 'east']
['The sun rises east']
```