NLP Lab1 - 25 Aug 2021

Kumar Gaurav 20122065

Lab Exercises:

1. Write a program to tokenize text
2. Write a program to count word frequency and to remove stop words

In [1]:

```
import nltk
```

In [2]:

```
import nltk.corpus # sample text for performing tokenization
```

In [3]:

```
from nltk.tokenize import word_tokenize
```

In [4]:

```
text = " In Brazil they drive on the right_hand side of the road.Brazil has a large coastli
```

In [5]:

```
token = word_tokenize(text)
```

In [6]:

```
token
```

Out[6]:

```
['In',
 'Brazil',
 'they',
 'drive',
 'on',
 'the',
 'right_hand',
 'side',
 'of',
 'the',
 'road.Brazil',
 'has',
 'a',
 'large',
 'coastline',
 'on',
 'the',
 'eastern',
 'side',
 'of',
 'south',
 'America']
```

In [7]:

```
from nltk.probability import FreqDist
```

In [8]:

```
fdist = FreqDist(token)
fdist
```

Out[8]:

```
FreqDist({'the': 3, 'on': 2, 'side': 2, 'of': 2, 'In': 1, 'Brazil': 1, 'the
y': 1, 'drive': 1, 'right_hand': 1, 'road.Brazil': 1, ...})
```

In [9]:

```
fdist1 = fdist.most_common(10)
fdist1
```

Out[9]:

```
[('the', 3),
 ('on', 2),
 ('side', 2),
 ('of', 2),
 ('In', 1),
 ('Brazil', 1),
 ('they', 1),
 ('drive', 1),
 ('right_hand', 1),
 ('road.Brazil', 1)]
```

stemming

In [10]:

```
from nltk.stem import PorterStemmer
```

In [11]:

```
pst_ = PorterStemmer()
pst_.stem('waiting')
```

Out[11]:

```
'wait'
```

In [12]:

```
pst_.stem('dancing')
```

Out[12]:

```
'danc'
```

In [13]:

```
pst_.stem('copying')
```

Out[13]:

```
'copi'
```

Lemma

In [14]:

```
from nltk.stem import LancasterStemmer
```

In [15]:

```
lst  = LancasterStemmer()
stm = ["giving", "given", "given", "gave"]
for word in stm :
   print(word+ ":" +lst.stem(word))
```

```
giving:giv
given:giv
given:giv
gave:gav
```

Lemmatization

For example, lemmatization would correctly identify the base form of ‘caring’ to ‘care’, whereas, stemming would cutoff the ‘ing’ part and convert it to a car. Lemmatization can be implemented in python by using Wordnet Lemmatizer, Spacy Lemmatizer, TextBlob, Stanford CoreNLP

In [16]:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

In [17]:

```
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
```

```
rocks : rock
corpora : corpus
```

Stop Words “Stop words” are the most common words in a language like “the”, “a”, “at”, “for”, “above”, “on”, “is”,

“all”. These words do not provide any meaning and are usually removed from texts. We can remove these stop words using nltk library

In [18]:

```
# importing stopwords from nltk library
from nltk import word_tokenize
from nltk.corpus import stopwords
a = set(stopwords.words('english'))
text = "Cristiano Ronaldo was born on February 5, 1985, in Funchal, Madeira, Portugal."
text1 = word_tokenize(text.lower())
print(text1)
stopwords = [x for x in text1 if x not in a]
print(stopwords)
```

```
['cristiano', 'ronaldo', 'was', 'born', 'on', 'february', '5', ',', '1985',
',', 'in', 'funchal', ',', 'madeira', ',', 'portugal', '.']
['cristiano', 'ronaldo', 'born', 'february', '5', ',', '1985', ',', 'funcha
l', ',', 'madeira', ',', 'portugal', '.']
```

Part of speech tagging (POS)

Part-of-speech tagging is used to assign parts of speech to each word of a given text (such as nouns, verbs, pronouns, adverbs, conjunction, adjectives, interjection) based on its definition and its context. There are many tools available for POS taggers and some of the widely used taggers are NLTK, Spacy, TextBlob, Standford CoreNLP, etc.

In [19]:

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

```
[('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

It is the process of detecting the named entities such as the person name, the location name, the company name, the quantities and the monetary value.

In [20]:

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

```
The Ghostscript executable isn't found.
See http://web.mit.edu/ghostscript/www/Install.htm (http://web.mit.edu/gho
stscript/www/Install.htm)
If you're using a Mac, you can try installing
https://docs.brew.sh/Installation (https://docs.brew.sh/Installation) then
`brew install ghostscript`
```

Chunking

Chunking means picking up individual pieces of information and grouping them into bigger pieces. In the context of NLP and text mining, chunking means a grouping of words or tokens into chunks.

In [21]:

```
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))
```