Kumar Gaurav 20122065

Lab 2 NLP

01 sep 2021

In [1]:

```
import os
import nltk
import nltk.corpus
```

In [3]:

```
print(os.listdir(nltk.data.find("corpora")))
```

```
['abc', 'abc.zip', 'alpino', 'alpino.zip', 'biocreative_ppi', 'biocreative_p
pi.zip', 'brown', 'brown.zip', 'brown_tei', 'brown_tei.zip', 'cess_cat', 'ce
ss_cat.zip', 'cess_esp', 'cess_esp.zip', 'chat80', 'chat80.zip', 'city_datab
ase', 'city_database.zip', 'cmudict', 'cmudict.zip', 'comparative_sentence
s', 'comparative_sentences.zip', 'comtrans.zip', 'conll2000', 'conll2000.zi
p', 'conll2002', 'conll2002.zip', 'conll2007.zip', 'crubadan', 'crubadan.zi
p', 'dependency_treebank', 'dependency_treebank.zip', 'dolch', 'dolch.zip',
'europarl_raw', 'europarl_raw.zip', 'floresta', 'floresta.zip', 'framenet_v1
5', 'framenet_v15.zip', 'framenet_v17', 'framenet_v17.zip', 'gazetteers', 'g
azetteers.zip', 'genesis', 'genesis.zip', 'gutenberg', 'gutenberg.zip', 'iee
r', 'ieer.zip', 'inaugural', 'inaugural.zip', 'indian', 'indian.zip', 'jeit
a.zip', 'kimmo', 'kimmo.zip', 'knbc.zip', 'lin_thesaurus', 'lin_thesaurus.zi
p', 'machado.zip', 'mac_morpho', 'mac_morpho.zip', 'masc_tagged.zip', 'movie
_reviews', 'movie_reviews.zip', 'mte_teip5', 'mte_teip5.zip', 'names', 'name
s.zip', 'nombank.1.0.zip', 'nonbreaking_prefixes', 'nonbreaking_prefixes.zi
p', 'nps_chat', 'nps_chat.zip', 'omw', 'omw.zip', 'opinion_lexicon', 'opinio
n_lexicon.zip', 'panlex_swadesh.zip', 'paradigms', 'paradigms.zip', 'pil',
'pil.zip', 'pl196x', 'pl196x.zip', 'ppattach', 'ppattach.zip', 'problem_repo
rts', 'problem_reports.zip', 'product_reviews_1', 'product_reviews_1.zip',
'product_reviews_2', 'product_reviews_2.zip', 'propbank.zip', 'pros_cons',
'pros_cons.zip', 'ptb', 'ptb.zip', 'qc', 'qc.zip', 'reuters.zip', 'rte', 'rt
e.zip', 'semcor.zip', 'senseval', 'senseval.zip', 'sentence_polarity', 'sent
ence_polarity.zip', 'sentiwordnet', 'sentiwordnet.zip', 'shakespeare', 'shak
espeare.zip', 'sinica_treebank', 'sinica_treebank.zip', 'smultron', 'smultro
n.zip', 'state_union', 'state_union.zip', 'stopwords', 'stopwords.zip', 'sub
jectivity', 'subjectivity.zip', 'swadesh', 'swadesh.zip', 'switchboard', 'sw
itchboard.zip', 'timit', 'timit.zip', 'toolbox', 'toolbox.zip', 'treebank',
'treebank.zip', 'twitter_samples', 'twitter_samples.zip', 'udhr', 'udhr.zi
p', 'udhr2', 'udhr2.zip', 'unicode_samples', 'unicode_samples.zip', 'univers
al_treebanks_v20.zip', 'verbnet', 'verbnet.zip', 'verbnet3', 'verbnet3.zip',
'webtext', 'webtext.zip', 'wordnet', 'wordnet.zip', 'wordnet_ic', 'wordnet_i
c.zip', 'words', 'words.zip', 'ycoe', 'ycoe.zip']
```

In [4]:

```
nltk.corpus.gutenberg.fileids()
```

Out[4]:

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

In [5]:

```
hamlet=nltk.corpus.gutenberg.words('shakespeare-hamlet.txt')
```

In [8]:

```
hamlet
```

Out[8]:

```
['[', 'The', 'Tragedie', 'of', 'Hamlet', 'by', ...]
```

In [9]:

```
for word in hamlet[:500]:
    print(word, sep='', end='')
```

```
[TheTragedieofHamletbyWilliamShakespeare1599]ActusPrimus.ScoenaPrima.EnterBa
rnardoandFranciscotwoCentinels.Barnardo.Who'sthere?Fran.Nayanswerme:Stand&vn
foldyourselfeBar.LongliuetheKingFran.Barnardo?Bar.HeFran.Youcomemostcarefull
yvponyourhoureBar.'Tisnowstrooktwelue,gettheetobedFranciscoFran.Forthisrelee
femuchthankes:'Tisbittercold,AndIamsickeatheartBarn.HaueyouhadquietGuard?Fra
n.NotaMousestirringBarn.Well,goodnight.IfyoudomeetHoratioandMarcellus,theRiu
alsofmyWatch,bidthemmakehast.EnterHoratioandMarcellus.Fran.IthinkeIhearethe
m.Stand:who'sthere?Hor.FriendstothisgroundMar.AndLeige-mentotheDaneFran.Giue
yougoodnightMar.OfarwelhonestSoldier,whohathrelieu'dyou?Fra.Barnardoha'smypl
ace:giueyougoodnight.ExitFran.Mar.HollaBarnardoBar.Say,whatisHoratiothere?Ho
r.ApeeceofhimBar.WelcomeHoratio,welcomegoodMarcellusMar.What,ha'sthisthingap
pear'dagainetonightBar.IhaueseenenothingMar.Horatiosaies,'tisbutourFantasie,
Andwillnotletbeleefetakeholdofhimtouchingthisdreadedsight,twiceseeneofvs,The
reforeIhaueintreatedhimalongWithvs,towatchtheminutesofthisNight,Thatifagaine
thisApparitioncome,Hemayapproueoureyes,andspeaketoitHor.Tush,tush,'twillnota
ppeareBar.Sitdownea-while,Andletvsonceagaineassaileyoureares,Thataresofortif
iedagainstourStory,WhatwetwoNightshaueseeneHor.Well,sitwedowne,Andletvsheare
BarnardospeakeofthisBarn.Lastnightofall,Whenyondsamestarrethat'sWestwardfrom
thePoleHadmadehiscourset'illumethatpartofHeauenWherenowitburnes,Marcellusand
myselfe,TheBellthenbeatingoneMar.Peace,breaketheeof:EntertheGhost.Lookewhere
itcomesagaineBarn.Inthesamefigure,liketheKingthat'sdeadMar.ThouartaScholler;
speaketoitHoratioBarn.LookesitnotliketheKing?MarkeitHoratioHora.Mostlike:Ith
arrowesmewithfear&wonderBarn.ItwouldbespoketooMar.QuestionitHoratioHor.Whata
rt
```

In [14]:

```
AI="""According to the father of Artificial Intelligence, John McCarthy, it is The science
Artificial Intelligence is a way of making a computer, a computer-controlled robot, or a so
AI is accomplished by studying how human brain thinks, and how humans learn, decide, and wo
Philosophy of AI
While exploiting the power of the computer systems, the curiosity of human, lead him to won
Thus, the development of AI started with the intention of creating similar intelligence in
Goals of AI
To Create Expert Systems – The systems which exhibit intelligent behavior, learn, demonstr,
To Implement Human Intelligence in Machines – Creating systems that understand, think, lear
What Contributes to AI?
Artificial intelligence is a science and technology based on disciplines such as Computer S
Out of the following areas, one or multiple areas can contribute to build an intelligent sy
```

In [15]:

```
type(AI)
```

Out[15]:

```
str
```

In [12]:

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

In [16]:

```
AI_tokens=word_tokenize(AI)
AI_tokens
```

Out[16]:

```
['According',
 'to',
 'the',
 'father',
 'of',
 'Artificial',
 'Intelligence',
 ',',
 'John',
 'McCarthy',
 ',',
 'it',
 'is',
 'The',
 'science',
 'and',
 'engineering',
 'of',
```

In [17]:

```
len(AI)
```

Out[17]:

```
1581
```

In [19]:

```
len(AI_tokens)
```

Out[19]:

```
285
```

In [20]:

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

In [21]:

```
for word in AI_tokens:
    fdist[word.lower()]+=1
fdist
```

Out[21]:

```
FreqDist({',': 32, 'of': 14, 'the': 13, 'and': 12, '.': 9, 'a': 9, 'to': 7,
'intelligence': 6, 'intelligent': 6, 'ai': 6, ...})
```

Tokenization

Bigrams-Tokens of two consecutive written words knows as Bigram Trigram-Tokens of three consecutive written words known as Trigram Ngram- Tokens of any numbers of consecutive written words known as Ngram

In [23]:

```
from nltk.util import bigrams,trigrams,ngrams
```

In [24]:

```
string = " The best and most beautiful things in the world cannot be seen or even touched -
quote_tokens = nltk.word_tokenize(string)
quote_tokens
```

...

In [25]:

```
quote_bigrams=list(nltk.bigrams(quote_tokens))
quote_bigrams
```

Out[25]:

```
[('The', 'best'),
 ('best', 'and'),
 ('and', 'most'),
 ('most', 'beautiful'),
 ('beautiful', 'things'),
 ('things', 'in'),
 ('in', 'the'),
 ('the', 'world'),
 ('world', 'can'),
 ('can', 'not'),
 ('not', 'be'),
 ('be', 'seen'),
 ('seen', 'or'),
 ('or', 'even'),
 ('even', 'touched'),
 ('touched', '-'),
 ('-', 'they'),
 ('they', 'must'),
```

In [26]:

```
quote_bigrams=list(nltk.trigrams(quote_tokens))
quote_bigrams
```

Out[26]:

```
[('The', 'best', 'and'),
 ('best', 'and', 'most'),
 ('and', 'most', 'beautiful'),
 ('most', 'beautiful', 'things'),
 ('beautiful', 'things', 'in'),
 ('things', 'in', 'the'),
 ('in', 'the', 'world'),
 ('the', 'world', 'can'),
 ('world', 'can', 'not'),
 ('can', 'not', 'be'),
 ('not', 'be', 'seen'),
 ('be', 'seen', 'or'),
 ('seen', 'or', 'even'),
 ('or', 'even', 'touched'),
 ('even', 'touched', '-'),
 ('touched', '-', 'they'),
 ('-', 'they', 'must'),
 ('they', 'must', 'be'),
```

In [27]:

```
quote_ngrams=list(nltk.ngrams(quote_tokens,4))
quote_ngrams
```

Out[27]:

```
[('The', 'best', 'and', 'most'),
 ('best', 'and', 'most', 'beautiful'),
 ('and', 'most', 'beautiful', 'things'),
 ('most', 'beautiful', 'things', 'in'),
 ('beautiful', 'things', 'in', 'the'),
 ('things', 'in', 'the', 'world'),
 ('in', 'the', 'world', 'can'),
 ('the', 'world', 'can', 'not'),
 ('world', 'can', 'not', 'be'),
 ('can', 'not', 'be', 'seen'),
 ('not', 'be', 'seen', 'or'),
 ('be', 'seen', 'or', 'even'),
 ('seen', 'or', 'even', 'touched'),
 ('or', 'even', 'touched', '-'),
 ('even', 'touched', '-', 'they'),
 ('touched', '-', 'they', 'must'),
 ('-', 'they', 'must', 'be'),
 ('they', 'must', 'be', 'felt'),
```

In [28]:

```
from nltk.stem import PorterStemmer
pst=PorterStemmer()
```

In [29]:

```
pst.stem("having")
```

Out[29]:

```
'have'
```

In [30]:

```
words_to_stem = ["give", "giving" ,"given" , "gave"]
for words in words_to_stem:
    print(words+ ":" +pst.stem(words))
```

```
give:give
giving:give
given:given
gave:gave
```

lemmatization

In [32]:

```
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
word_len = WordNetLemmatizer()
for words in words_to_stem:
    print(words+ ":" +word_len.lemmatize(words))
```

```
give:give
giving:giving
given:given
gave:gave
```

Stop word

In [33]:

```
from nltk.corpus import stopwords
```

In [34]:

```
stopwords.words('english')
```

Out[34]:

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
```

In [35]:

```
len(stopwords.words('english'))
```

Out[35]:

```
179
```

Top 10 frequency of stop word

In [39]:

```
fdist.most_common(10)
```

Out[39]:

```
[(',', 32),
 ('of', 14),
 ('the', 13),
 ('and', 12),
 ('.', 9),
 ('a', 9),
 ('to', 7),
 ('intelligence', 6),
 ('intelligent', 6),
 ('ai', 6)]
```

regular expression_

In [45]:

```
import re
punctuation = re.compile(r'[-.?!,:;()[0-9]]')
```

In [47]:

```
post_punctuation=[]
for words in AI_tokens:
    word=punctuation.sub("",words)
    if len(word)>0:
        post_punctuation.append(word)
```

In [48]:

```
post_punctuation
```

Out[48]:

```
['According',
 'to',
 'the',
 'father',
 'of',
 'Artificial',
 'Intelligence',
 ',',
 'John',
 'McCarthy',
 ',',
 'it',
 'is',
 'The',
 'science',
 'and',
 'engineering',
 'of',
```

In [49]:

```
len(post_punctuation)
```

Out[49]:

```
285
```

pos

part of speech

In [50]:

```
sent= "The best and most beautiful things in the world cannot be seen or even touched but m
sent_tokens = word_tokenize(sent)
```

In [51]:

```
for token in sent_tokens:
    print(nltk.pos_tag([token]))
```

```
[('The', 'DT')]
[('best', 'JJS')]
[('and', 'CC')]
[('most', 'JJS')]
[('beautiful', 'NN')]
[('things', 'NNS')]
[('in', 'IN')]
[('the', 'DT')]
[('world', 'NN')]
[('can', 'MD')]
[('not', 'RB')]
[('be', 'VB')]
[('seen', 'VBN')]
[('or', 'CC')]
[('even', 'RB')]
[('touched', 'VBN')]
[('but', 'CC')]
[('must', 'MD')]
[('be', 'VB')]
```

In [52]:

```
sent2= "john is eating a delicious cake"
sent_tokens = word_tokenize(sent2)
for token in sent_tokens:
    print(nltk.pos_tag([token]))
```

```
[('john', 'NN')]
[('is', 'VBZ')]
[('eating', 'VBG')]
[('a', 'DT')]
[('delicious', 'JJ')]
[('cake', 'NN')]
```

Named entity Recognition

In [53]:

```
from nltk import ne_chunk
```

In [56]:

```
NE_sent =  "The US President stays in the WHITE HOUSE"
```

In [57]:

```
NE_tokens= word_tokenize(NE_sent)
NE_tags = nltk.pos_tag(NE_tokens)
```

In [58]:

```
NE_NER = ne_chunk(NE_tags)
print(NE_NER)
```

```
(S
  The/DT
  (ORGANIZATION US/NNP)
  President/NNP
  stays/VBZ
  in/IN
  the/DT
  (FACILITY WHITE/NNP HOUSE/NNP))
```

In [ ]:

```
# SYNTAX=Priciples + rules + process
```

Chunking_

In [60]:

```
new = "The big cat ate the little mouse who was after fresh cheese"
new_tokens = nltk.pos_tag(word_tokenize(new))
new_tokens
```

Out[60]:

```
[('The', 'DT'),
 ('big', 'JJ'),
 ('cat', 'NN'),
 ('ate', 'VBD'),
 ('the', 'DT'),
 ('little', 'JJ'),
 ('mouse', 'NN'),
 ('who', 'WP'),
 ('was', 'VBD'),
 ('after', 'IN'),
 ('fresh', 'JJ'),
 ('cheese', 'NN')]
```

In [61]:

```
grammer_np = r"NP:{<DT>?<JJ>*<NN>}"
```

In [62]:

```
chunk_parser = nltk.RegexpParser(grammer_np)
```

In [63]:

```
chunk_result = chunk_parser.parse(new_tokens)
chunk_result
```

```
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`
```

Write a program to tokenize non_English Languages

Some terms that will be frequently used are :

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.

For examples, each word is a token when a sentence is “tokenized” into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph. So basically tokenizing involves splitting sentences and words from the body of the text.

Different Methods to Perform Tokenization in Python

Tokenization using Python split() Function Tokenization using Regular Expressions Tokenization using NLTK Tokenization using Spacy Tokenization using Keras Tokenization using Gensim

Tokenize non_english _ Language

In [65]:

```
from nltk.tokenize import sent_tokenize
mytext = "Bonjour M. Adam, comment allez-vous? J'espère que tout va bien. Aujourd'hui est u
```

In [66]:

```
print(sent_tokenize(mytext,"french"))
```

```
['Bonjour M. Adam, comment allez-vous?', "J'espère que tout va bien.", "Aujo
urd'hui est u"]
```

In [67]:

```
text = '''
NLTK ist Open Source Software. Der Quellcode wird unter den Bedingungen der Apache License
Die Dokumentation wird unter den Bedingungen der Creative Commons-Lizenz Namensnennung - Ni
abgeleiteten Werke 3.0 in den Vereinigten Staaten verteilt.
'''
print("\nOriginal string:")
print(text)
from nltk.tokenize import sent_tokenize
token_text = sent_tokenize(text, language='german')
print("\nSentence-tokenized copy in a list:")
print(token_text)
print("\nRead the list:")
for s in token_text:
    print(s)
```

```
Original string:

NLTK ist Open Source Software. Der Quellcode wird unter den Bedingungen de
r Apache License Version 2.0 vertrieben.
Die Dokumentation wird unter den Bedingungen der Creative Commons-Lizenz N
amensnennung - Nicht kommerziell - Keine
abgeleiteten Werke 3.0 in den Vereinigten Staaten verteilt.


Sentence-tokenized copy in a list:
['\nNLTK ist Open Source Software.', 'Der Quellcode wird unter den Bedingu
ngen der Apache License Version 2.0 vertrieben.', 'Die Dokumentation wird
unter den Bedingungen der Creative Commons-Lizenz Namensnennung - Nicht ko
mmerziell - Keine \nabgeleiteten Werke 3.0 in den Vereinigten Staaten vert
eilt.']

Read the list:
```

In [68]:

```
quote_tokens = nltk.word_tokenize(text)
quote_tokens
```

Out[68]:

```
['NLTK',
 'ist',
 'Open',
 'Source',
 'Software',
 '.',
 'Der',
 'Quellcode',
 'wird',
 'unter',
 'den',
 'Bedingungen',
 'der',
 'Apache',
 'License',
 'Version',
 '2.0',
 'vertrieben',
```

iNLTK- Hindi, Punjabi, Sanskrit, Gujarati, Kannada, Malyalam, Nepali, Odia, Marathi, Bengali, Tamil, Urdu Indic NLP Library- Assamese, Sindhi, Sinhala, Sanskrit, Konkani, Kannada, Telugu, StanfordNLP- Many of the above languages

In [*]:

```
pip install inltk
```

In [*]:

```
import inltk
```