Exp2 (Implement sentence and word tokenization)

import re

from nltk.tokenize import word\_tokenize,sent\_tokenize

from pathlib import Path

text = Path('myfiles.txt').read\_text()

print("\nWord TOkenization with NLTK:\n")

print(word\_tokenize(text))

print("\nSentence Tokenization with NLTK\n")

print(sent\_tokenize(text))

Exp3 (implement filtration, script validation, stop word removal, stop word removal)

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

f = open("text.txt","r")

#wordtokenizer

for line in f:

print("Tokenization with NLTK \n")

print(word\_tokenize(line))

print("\n")

print("Filteration \n")

bad\_chars = [';',',','!',':','\*','#','<','>','?','@','.']

words = word\_tokenize(line)

print(list(filter(lambda i: i not in bad\_chars,words)))

print("\n")

#stop word removal

print("Stop word removal \n")

stop\_words = set(stopwords.words("English"))

without\_stop\_words = [word for word in words if not word in stop\_words]

with\_stopwords = list(filter(lambda i: i not in bad\_chars, without\_stop\_words))

print(with\_stopwords)

print("\n")

Exp4 (implement, stemming, lemmantization in python)

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from nltk.stem import WordNetLemmatizer

f = open("text.txt","r")

for line in f:

print("Tokenization with NLTK \n")

print(word\_tokenize(line))

print("\n")

print("Filteration \n")

bad\_chars = [';',',','!',':','\*','#','<','>','?','@','.']

words = word\_tokenize(line)

print(list(filter(lambda i: i not in bad\_chars,words)))

print("\n")

print("Length of Filteration:")

print(len(list(filter(lambda i: i not in bad\_chars,words))))

print("\n")#stop word removal

print("Stop word removal \n")

stop\_words = set(stopwords.words("English"))

without\_stop\_words = [word for word in words if not word in stop\_words]

with\_stopwords = list(filter(lambda i: i not in bad\_chars, without\_stop\_words))

print(with\_stopwords)

print("\n")

#Stemming and Lemmatization

ps = PorterStemmer()

l = WordNetLemmatizer()

print("Word \t \tStemming \tLemmatization")

for w in with\_stopwords:

print ("{:<15} {:<15} {:<15}".format( w, ps.stem(w), l.lemmatize(w)))

print("\n")

Exp5 (implement morphological analysis in python)

import nltk

from nltk.stem.wordnet import WordNetLemmatizer

from nltk.corpus import wordnet

from nltk.tokenize import sent\_tokenize, word\_tokenize

from nltk.corpus import stopwords

from textblob import TextBlob

#Morphological Analysis

def get\_wordnet\_pos(word):

tag = nltk.pos\_tag([word])[0][1][0].upper()

tag\_dict = {"J":wordnet.ADJ,

"N":wordnet.NOUN,

"V":wordnet.VERB,

"R":wordnet.ADV}

return tag\_dict.get(tag,wordnet.NOUN)

f=open("Exp5.txt")

stop\_words = set(stopwords.words("English"))

bad\_chars = (';',':','!', "\*",'<','>','#','?','@','p',',','.','(',')','eos')

for line in f:

words = word\_tokenize(line)

print("Word Tokenization\n")

print(words)

print("\n")

#stop word and filterization

without\_stop\_words = [word for word in words if not word in stop\_words]

ws = list(filter(lambda i: i not in bad\_chars,without\_stop\_words))

print("Filtered Words \n")

print(ws)

print("\n")

ws\_tag = nltk.pos\_tag(ws)

wordtaglength = len(ws\_tag)

#Lemmatizer

l = WordNetLemmatizer()

print("WORD","\t\t\t","ROOT WORD(MORPHENE)","\t\t","TAG")

for i in range(wordtaglength):

print(ws[i].ljust(9),"\t\t",l.lemmatize(ws[i],get\_wordnet\_pos(ws[i])).ljust(9),"\t\t",ws\_tag[i])

Exp6 (n-gram)

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from nltk.stem import WordNetLemmatizer

from collections import Counter

f = open("exp 6.txt","r")

for line in f:

l1=[] #Count of bigrams

l2=[] #First word of bigram

l3=[] #Probability of bigram

l4=[] #tuples of bigrams

prod = 1

#wordtokenizer

print("Tokenization with NLTK \n")

print(word\_tokenize(line))

print("\n")

#Filteration

print("Filteration \n")

bad\_chars = [';',',','!',':','\*','#','<','>','?','@','.']

words = word\_tokenize(line)

swords = []

swords = list(filter(lambda i: i not in bad\_chars,words))

print(swords)

print("\n")

pair\_words = []

#Bigram generation

for i in range(len(swords)-1):

pair\_words.append((swords[i],swords[i+1]))

print("The bi-grams are:")

print(pair\_words)

print("\n")

cnt = dict(Counter(pair\_words))

print("Count of occurances of bigrams")

for pair, number in cnt.items():

l1.append(number)

l2.append(pair[0])

l4.append(pair)

print(pair, ":", number)

print("\n")

print("Probability of the bigrams")

for x in range(len(l2)):

p = swords.count(l2[x])

l3.append(round(l1[x]/p,3))

prod = l3[x]\*prod

print(l4[x], ":", l3[x])

print("\n")

print("Probability of the sentence.")

print(prod)

Exp7 (POS tagging)

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import sent\_tokenize, word\_tokenize

from nltk.util import ngrams

from nltk.lm.preprocessing import pad\_both\_ends

from collections import Counter

import numpy as np

import pandas as pd

import string

text = '''Marry Jane can see Will.

Spot will see Mary.

Will Jane spot Mary?

Mary will pat Spot.'''

sent\_text = nltk.sent\_tokenize(text)

tagged\_sents = []

tags\_transitions = []

uni\_tags = []

for sent in sent\_text:

tokenized\_words = word\_tokenize(sent)

tokens = list(filter(lambda token: token not in string.punctuation, tokenized\_words))

tokens = list(x.lower() for x in tokens)

tagger = nltk.pos\_tag(tokens)

tagged\_sents.extend(tagger)

tags\_transition = [tup[1] for tup in tagger]

uni\_tags.extend(tags\_transition + ['<s>', '</s>'])

tags\_transitions.extend(list(ngrams(pad\_both\_ends(tags\_transition, n=2), n=2)))

print("Tagged Sentences :\n",tagged\_sents)

tagged\_words = [ tup for tup in tagged\_sents ]

count\_tagged\_words = Counter(tagged\_words)

tags = list({tag for word, tag in tagged\_words})

vocabs = {word for word, tag in tagged\_words}

print("\nTags: ",tags)

print("\nVocabs: ",vocabs)

em = pd.DataFrame({tag: [] for tag in ["Words"]+tags})

print("\nEMISSION")

for vocab in vocabs:

em.loc[vocab] = [vocab] + [count\_tagged\_words[vocab, tag] for tag in tags]

em.set\_index('Words')

print("\nFrequency :\n",em)

tag\_freq\_em = Counter(elem[1] for elem in tagged\_sents)

for vocab in vocabs:

for tag in tags:

em.at[vocab, tag] /= tag\_freq\_em[tag]

print("\nProbability :\n",em)

print("\nTRANSITION")

tags\_trans\_freq = Counter(tags\_transitions)

tr = pd.DataFrame({tag: [] for tag in ["Tags"]+tags+["</s>"]})

for tag\_row in ["<s>"]+tags:

tr.loc[tag\_row] = [tag\_row] + [tags\_trans\_freq[tag\_row, tag\_col] for tag\_col in tags+["</s>"]]

tr.set\_index('Tags')

print("\nFrequency :\n",em)

tag\_freq\_tr = Counter(uni\_tags)

for tag\_row in ["<s>"]+tags:

for tag\_col in tags+["</s>"]:

tr.at[tag\_row, tag\_col] /= max(1,tag\_freq\_tr[tag\_row])

tr.set\_index('Tags')

print("\nProbability :\n",em)

wrong\_tag = [('will', 'MD'),

('can', 'VB'),

('spot', 'NN'),

('mary', 'NN')]

wrong\_tags = [tup[1] for tup in wrong\_tag]

wrong\_tags\_pairs = list(ngrams(pad\_both\_ends(wrong\_tags, n=2), n=2))

print("\nWrong Tags Pairs : ", wrong\_tags\_pairs)

prob = 1

for pair in wrong\_tags\_pairs:

prob \*= tr.at[pair[0], pair[1]]

print("\nProbability of Correct Sentence:",prob)

Exp8 (Chunking)

import nltk

nltk.download('state\_union')

from nltk.corpus import state\_union

from nltk.tokenize import PunktSentenceTokenizer

train\_text = state\_union.raw("2005-GWBush.txt")

custom\_sent\_tokenizer = PunktSentenceTokenizer(train\_text)

tokenized = custom\_sent\_tokenizer.tokenize("Manchester United Football Club is a professional football club based in Old Trafford")

def process\_content():

try:

for i in tokenized:

words = nltk.word\_tokenize(i)

tagged = nltk.pos\_tag(words)

chunkGram = r"""Chunk: {<RB.?>\*<VB.?>\*<NNP>+<NN>?}"""

chunkParser = nltk.RegexpParser(chunkGram)

chunked = chunkParser.parse(tagged)

chunked.draw()

except Exception as e:

print(str(e))

process\_content()

Exp9 (NER)

import spacy

from spacy import displacy

from collections import Counter

import en\_core\_web\_sm

nlp = en\_core\_web\_sm.load()

doc = nlp(text)

l1 = list([(X.text, X.label\_) for X in doc.ents])

print(\*l1, sep = "\n")