CLASS 1: (20th dec,2018)

//importing nltk

import nltk

//downloading nltk

nltk.download()

// importing book from nltk

from nltk.book import\*

\*\*\* Introductory Examples for the NLTK Book \*\*\*

Loading text1, ..., text9 and sent1, ..., sent9

Type the name of the text or sentence to view it.

Type: 'texts()' or 'sents()' to list the materials.

text1: Moby Dick by Herman Melville 1851

text2: Sense and Sensibility by Jane Austen 1811

text3: The Book of Genesis

text4: Inaugural Address Corpus

text5: Chat Corpus

text6: Monty Python and the Holy Grail

text7: Wall Street Journal

text8: Personals Corpus

text9: The Man Who Was Thursday by G . K . Chesterton 1908

//brown corpus

from nltk.corpus import brown

brown.categories()

[u'adventure',

u'belles\_lettres',

u'editorial',

u'fiction',

u'government',

u'hobbies',

u'humor',

u'learned',

u'lore',

u'mystery',

u'news',

u'religion',

u'reviews',

u'romance',

u'science\_fiction']

brown.words(categories = "humor")

[u'It', u'was', u'among', u'these', u'that', u'Hinkle', ...]

// inaugural corpus

from nltk.corpus import inaugural

inaugural.words()

[u'Fellow', u'-', u'Citizens', u'of', u'the', ...]

//lexicons

1. Stopwords

import nltk

from nltk.corpus import stopwords

stopwords.words('english')

[u'i',

u'me',

u'my',

u'myself',

u'we',

u'our',

u'ours',

u'ourselves',

u'you',

u"you're",

u"you've",

u"you'll",

u"you'd",

u'your',

u'yours',

u'yourself',

u'yourselves',

u'he',

u'him',

u'his',

u'himself',

u'she',

u"she's",

u'her',

u'hers',

u'herself',

u'it',

u"it's",

u'its',

u'itself',

u'they',

u'them',

u'their',

u'theirs',

u'themselves',

u'what',

u'which',

u'who',

u'whom',

u'this',

u'that',

u"that'll",

u'these',

u'those',

u'am',

u'is',

u'are',

u'was',

u'were',

u'be',

u'been',

u'being',

u'have',

u'has',

u'had',

u'having',

u'do',

u'does',

u'did',

u'doing',

u'a',

u'an',

u'the',

u'and',

u'but',

u'if',

u'or',

u'because',

u'as',

u'until',

u'while',

u'of',

u'at',

u'by',

u'for',

u'with',

u'about',

u'against',

u'between',

u'into',

u'through',

u'during',

u'before',

u'after',

u'above',

u'below',

u'to',

u'from',

u'up',

u'down',

u'in',

u'out',

u'on',

u'off',

u'over',

u'under',

u'again',

u'further',

u'then',

u'once',

u'here',

u'there',

u'when',

u'where',

u'why',

u'how',

u'all',

u'any',

u'both',

u'each',

u'few',

u'more',

u'most',

u'other',

u'some',

u'such',

u'no',

u'nor',

u'not',

u'only',

u'own',

u'same',

u'so',

u'than',

u'too',

u'very',

u's',

u't',

u'can',

u'will',

u'just',

u'don',

u"don't",

u'should',

u"should've",

u'now',

u'd',

u'll',

u'm',

u'o',

u're',

u've',

u'y',

u'ain',

u'aren',

u"aren't",

u'couldn',

u"couldn't",

u'didn',

u"didn't",

u'doesn',

u"doesn't",

u'hadn',

u"hadn't",

u'hasn',

u"hasn't",

u'haven',

u"haven't",

u'isn',

u"isn't",

u'ma',

u'mightn',

u"mightn't",

u'mustn',

u"mustn't",

u'needn',

u"needn't",

u'shan',

u"shan't",

u'shouldn',

u"shouldn't",

u'wasn',

u"wasn't",

u'weren',

u"weren't",

u'won',

u"won't",

u'wouldn',

u"wouldn't"]

import nltk

from nltk.corpus import stopwords

stopwords.words('swedish')

[u'och',

u'det',

u'att',

u'i',

u'en',

u'jag',

u'hon',

u'som',

u'han',

u'p\xe5',

u'den',

u'med',

u'var',

u'sig',

u'f\xf6r',

u's\xe5',

u'till',

u'\xe4r',

u'men',

u'ett',

u'om',

u'hade',

u'de',

u'av',

u'icke',

u'mig',

u'du',

u'henne',

u'd\xe5',

u'sin',

u'nu',

u'har',

u'inte',

u'hans',

u'honom',

u'skulle',

u'hennes',

u'd\xe4r',

u'min',

u'man',

u'ej',

u'vid',

u'kunde',

u'n\xe5got',

u'fr\xe5n',

u'ut',

u'n\xe4r',

u'efter',

u'upp',

u'vi',

u'dem',

u'vara',

u'vad',

u'\xf6ver',

u'\xe4n',

u'dig',

u'kan',

u'sina',

u'h\xe4r',

u'ha',

u'mot',

u'alla',

u'under',

u'n\xe5gon',

u'eller',

u'allt',

u'mycket',

u'sedan',

u'ju',

u'denna',

u'sj\xe4lv',

u'detta',

u'\xe5t',

u'utan',

u'varit',

u'hur',

u'ingen',

u'mitt',

u'ni',

u'bli',

u'blev',

u'oss',

u'din',

u'dessa',

u'n\xe5gra',

u'deras',

u'blir',

u'mina',

u'samma',

u'vilken',

u'er',

u's\xe5dan',

u'v\xe5r',

u'blivit',

u'dess',

u'inom',

u'mellan',

u's\xe5dant',

u'varf\xf6r',

u'varje',

u'vilka',

u'ditt',

u'vem',

u'vilket',

u'sitta',

u's\xe5dana',

u'vart',

u'dina',

u'vars',

u'v\xe5rt',

u'v\xe5ra',

u'ert',

u'era',

u'vilkas']

1. Cmudict:

entries = nltk.corpus.cmudict.entries()

len(entries)

133737

for entry in entries[10000:10025]:

print(entry)

(u'belford', [u'B', u'EH1', u'L', u'F', u'ER0', u'D'])

(u'belfry', [u'B', u'EH1', u'L', u'F', u'R', u'IY0'])

(u'belgacom', [u'B', u'EH1', u'L', u'G', u'AH0', u'K', u'AA0', u'M'])

(u'belgacom', [u'B', u'EH1', u'L', u'JH', u'AH0', u'K', u'AA0', u'M'])

(u'belgard', [u'B', u'EH0', u'L', u'G', u'AA1', u'R', u'D'])

(u'belgarde', [u'B', u'EH0', u'L', u'G', u'AA1', u'R', u'D', u'IY0'])

(u'belge', [u'B', u'EH1', u'L', u'JH', u'IY0'])

(u'belger', [u'B', u'EH1', u'L', u'G', u'ER0'])

(u'belgian', [u'B', u'EH1', u'L', u'JH', u'AH0', u'N'])

(u'belgians', [u'B', u'EH1', u'L', u'JH', u'AH0', u'N', u'Z'])

(u'belgique', [u'B', u'EH0', u'L', u'ZH', u'IY1', u'K'])

(u"belgique's", [u'B', u'EH0', u'L', u'JH', u'IY1', u'K', u'S'])

(u'belgium', [u'B', u'EH1', u'L', u'JH', u'AH0', u'M'])

(u"belgium's", [u'B', u'EH1', u'L', u'JH', u'AH0', u'M', u'Z'])

(u'belgo', [u'B', u'EH1', u'L', u'G', u'OW2'])

(u'belgrade', [u'B', u'EH1', u'L', u'G', u'R', u'EY0', u'D'])

(u'belgrade', [u'B', u'EH1', u'L', u'G', u'R', u'AA2', u'D'])

(u"belgrade's", [u'B', u'EH1', u'L', u'G', u'R', u'EY0', u'D', u'Z'])

(u"belgrade's", [u'B', u'EH1', u'L', u'G', u'R', u'AA2', u'D', u'Z'])

(u'belgrave', [u'B', u'EH1', u'L', u'G', u'R', u'EY2', u'V'])

(u'beli', [u'B', u'EH1', u'L', u'IY0'])

(u'belich', [u'B', u'EH1', u'L', u'IH0', u'K'])

(u'belie', [u'B', u'IH0', u'L', u'AY1'])

(u'belied', [u'B', u'IH0', u'L', u'AY1', u'D'])

(u'belief', [u'B', u'IH0', u'L', u'IY1', u'F'])

1. Wordnet:

from nltk.corpus import wordnet as wn

wn.synsets('motorcar') #synsets= synonyms sets

[Synset('car.n.01')]

wn.synset('car.n.01').lemma\_names()

[u'car', u'auto', u'automobile', u'machine', u'motorcar']

//pipelining

texts = ["""VIT offers academic programs in Engineering, Technology, Applied Sciences, and Management. It offers 20 undergraduate programs, 34 postgraduate, four integrated MS courses, and four doctoral programs. VIT consolidated its disciplines into 10 Schools of Study with the addition of the VIT Law School at its Chennai campus.Research centers are part of the schools to encourage collaboration between the research and coursework areas and provide an opportunity for coursework students to participate in research projects. VIT organizes industrial workshops like Automotive Engineering, organized by ParaMek Technologies in GraVitas Fest."""]

for text in texts:

sentences = nltk.sent\_tokenize(text)

for sentence in sentences:

words = nltk.word\_tokenize(sentence)

tagged\_words = nltk.pos\_tag(words)

print(tagged\_words)

[('VIT', 'NNP'), ('offers', 'VBZ'), ('academic', 'JJ'), ('programs', 'NNS'), ('in', 'IN'), ('Engineering', 'NNP'), (',', ','), ('Technology', 'NNP'), (',', ','), ('Applied', 'NNP'), ('Sciences', 'NNPS'), (',', ','), ('and', 'CC'), ('Management', 'NNP'), ('.', '.')]

[('It', 'PRP'), ('offers', 'VBZ'), ('20', 'CD'), ('undergraduate', 'NN'), ('programs', 'NNS'), (',', ','), ('34', 'CD'), ('postgraduate', 'NN'), (',', ','), ('four', 'CD'), ('integrated', 'VBN'), ('MS', 'NNP'), ('courses', 'NNS'), (',', ','), ('and', 'CC'), ('four', 'CD'), ('doctoral', 'JJ'), ('programs', 'NNS'), ('.', '.')]

[('VIT', 'NNP'), ('consolidated', 'VBD'), ('its', 'PRP$'), ('disciplines', 'NNS'), ('into', 'IN'), ('10', 'CD'), ('Schools', 'NNP'), ('of', 'IN'), ('Study', 'NNP'), ('with', 'IN'), ('the', 'DT'), ('addition', 'NN'), ('of', 'IN'), ('the', 'DT'), ('VIT', 'NNP'), ('Law', 'NNP'), ('School', 'NNP'), ('at', 'IN'), ('its', 'PRP$'), ('Chennai', 'NNP'), ('campus.Research', 'NN'), ('centers', 'NNS'), ('are', 'VBP'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('schools', 'NNS'), ('to', 'TO'), ('encourage', 'VB'), ('collaboration', 'NN'), ('between', 'IN'), ('the', 'DT'), ('research', 'NN'), ('and', 'CC'), ('coursework', 'NN'), ('areas', 'NNS'), ('and', 'CC'), ('provide', 'VB'), ('an', 'DT'), ('opportunity', 'NN'), ('for', 'IN'), ('coursework', 'NN'), ('students', 'NNS'), ('to', 'TO'), ('participate', 'VB'), ('in', 'IN'), ('research', 'NN'), ('projects', 'NNS'), ('.', '.')]

[('VIT', 'NNP'), ('organizes', 'VBZ'), ('industrial', 'JJ'), ('workshops', 'NNS'), ('like', 'IN'), ('Automotive', 'NNP'), ('Engineering', 'NNP'), (',', ','), ('organized', 'VBN'), ('by', 'IN'), ('ParaMek', 'NNP'), ('Technologies', 'NNPS'), ('in', 'IN'), ('GraVitas', 'NNP'), ('Fest', 'NNP'), ('.', '.')]

10th January,2019

Conclusions that are drawn from the various stemmers that we have used.

Conclusion 1:

In both the stemmer(LANCASTER & PORTER) the words that are ending with e are not correctly predicted.

For Ex:

stemmerporter.stem('troubling'): troubl

stemmerlancaster.stem('Troubling'): troubl

Conclusion 2:

Also, both the stemmers dont take into consideration the prefixes and correctly predict the words. but the extend is different in both.

For Ex:

stemmerporter.stem('disadvantage'): disadvantag

stemmerlancaster.stem('disadvantage'): disadv

Reason: Due to the change in the meaning of the words.

Conclusion 3:

In Lancaster stemmer the case of the letters doesn’t matter, whereas in porter stemmer the case of the letters in a word helps in correctly identifying the stem of the given word

For Ex:

stemmerporter.stem('Dying'): dy

stemmerporter.stem('dying'): die

stemmerlancaster.stem('Dying'): dying

stemmerlancaster.stem('dying'): dying

After working with the various types of stemmers, the above stated conclusions were drawn. The conclusions can vary according to people while working with the stemmers.

Class 2: (7th Feb,2019)

from nltk.corpus import names

def gender\_features(word):

return{'last\_letter': word[-1]}

gender\_features('sherlock')

{'last\_letter': 'k'}

labeled\_names = ([(name, 'male') for name in names.words('male.txt')]+[(name, 'female') for name in names.words('female.txt')])

labeled\_names

[(u'Aamir', 'male'),

(u'Aaron', 'male'),

(u'Abbey', 'male'),

(u'Abbie', 'male'),

(u'Abbot', 'male'),

(u'Abbott', 'male'),

(u'Abby', 'male'),

(u'Abdel', 'male'),

(u'Abdul', 'male'),

(u'Abdulkarim', 'male'),

(u'Abdullah', 'male'),

(u'Abe', 'male'),

(u'Abel', 'male'),

(u'Abelard', 'male'),

(u'Abner', 'male'),

(u'Abraham', 'male'),

(u'Abram', 'male'),

(u'Ace', 'male'),

(u'Adair', 'male'),

(u'Adam', 'male'),

(u'Adams', 'male'),

...]

import random

random.shuffle(labeled\_names)

labeled\_names

[(u'Shana', 'female'),

(u'Tommy', 'female'),

(u'Zebulon', 'male'),

(u'Kalman', 'male'),

(u'Sibel', 'female'),

(u'Terrianne', 'female'),

(u'Arie', 'male'),

(u'Duffy', 'male'),

(u'Bessy', 'female'),

(u'Stefa', 'female'),

(u'Hallam', 'male'),

(u'Glynda', 'female'),

(u'Ann', 'female'),

(u'Astrid', 'female'),

(u'Rosy', 'female'),

(u'Otho', 'male'),

(u'Thibaut', 'male'),

(u'Hephzibah', 'female'),

...]

featuresets = [(gender\_features(n),gender) for (n,gender) in labeled\_names]

featuresets

[({'last\_letter': u'a'}, 'female'),

({'last\_letter': u'y'}, 'female'),

({'last\_letter': u'n'}, 'male'),

({'last\_letter': u'n'}, 'male'),

({'last\_letter': u'l'}, 'female'),

({'last\_letter': u'e'}, 'female'),

({'last\_letter': u'e'}, 'male'),

({'last\_letter': u'y'}, 'male'),

({'last\_letter': u'y'}, 'female'),

({'last\_letter': u'a'}, 'female'),

({'last\_letter': u'm'}, 'male'),

({'last\_letter': u'a'}, 'female'),

({'last\_letter': u'n'}, 'female'),

({'last\_letter': u'd'}, 'female'),

({'last\_letter': u'y'}, 'female'),

({'last\_letter': u'o'}, 'male'),

({'last\_letter': u't'}, 'male'),

...]

train\_set, test\_set = featuresets[500:], featuresets[:500]

classifier = nltk.NaiveBayesClassifier.train(train\_set)

classifier.classify(gender\_features('David'))

'male'

print(nltk.classify.accuracy(classifier, test\_set))

0.788

Stemmers

Types of stemmers:

Porter

Snowball

Lanacaster

Class 3: (26th Feb,2019)

import nltk

from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

example = "The cat was chasing a mouse"

example = [stemmer.stem(token) for token in example.split(" ")]

print(example)

['the', 'cat', u'wa', u'chase', 'a', u'mous']

print(" ".join(example))

the cat wa chase a mous

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

example = "The cat was chasing a mouse"

example = [lemmatizer.lemmatize(token) for token in example.split(" ")]

print(example)

['The', 'cat', u'wa', 'chasing', 'a', 'mouse']

print(" ".join(example))

The cat wa chasing a mouse

example = (lemmatizer.lemmatize('better', pos = 'a'))

print(example)

good

from sklearn.feature\_extraction.text import CountVectorizer

print(example)

good

from sklearn.feature\_extraction.text import CountVectorizer

vect = CountVectorizer(binary = True)

corpus = ["Tesseract is good optical character recognition engine", " optical character recognition is significant"]

vect.fit(corpus)

print(vect.transform(["Today is good optical"]).toarray())

[[0 0 1 1 0 0 0]]

Class 4: (28th Feb,2019)

SIMILARITY SEARCH TASK:

Paragraph 3

HOW DEEP IS YOUR LOVE?

If there is a phrase I would prefer to retire from online bios, personal and professional, it is, “I love travel.” Or some proximation of that sentiment. To clarify, I’m not against travelers and those who proudly flaunt their passion for travel. On the contrary, editing a travel magazine has now made me oddly protective of travelers and their ilk. My submission is that “love to travel,” suggesting so casually, just doesn’t feel adequate to the depth of emotion it sparks in true devotees.

Paragraph 2

Summer is a charming flirt. Easy-going and casual. Summer doesn.t huff and puff to win our affections. It has us at "Hello". Winter broods like the tortured protagonist of big fat Russian novel. It is dauting and dramatic, burning with a slow intensity. The season's reputation precedes itself, and often, not in a good way. It has a way of whittling down everything to its bare bones. Even relationship not attuned to its ebbs and flows can fray. At a dinner conversation I once attended. I listened in bemusement as a recent divorcee made the case that it was the Scandinavian frost that had cooled his ex-wife's ardor. How original.

Paragraph 3

One of the finer books I read this year was John Kaag’s Hiking With Nietzsche, in which Kaag, a professor of philosophy, rekindles his passion for the German thinker while tracing picturesque hiking trails in the mountains of Switzerland. It’s a near-precise rendering of the travelogue as a self-help book. A young Kaag was an avowed Nietzsche acolyte but given the ravages of responsibilities and adulthood, the writer put his affinity to test by undertaking physically enduring hikes through the Alps, revisiting haunts that the philosopher escaped to, in search of solitude and salve. The journey’s demands, coupled with his own inner turmoil, are catnip for anybody feeling at cross purposes with their own life.

Code:

example1="""If there is a phrase I would prefer to retire from online bios, personal and professional, it is, “I love travel.” Or some proximation of that sentiment. To clarify, I’m not against travelers and those who proudly flaunt their passion for travel. On the contrary, editing a travel magazine has now made me oddly protective of travelers and their ilk. My submission is that “love to travel,” suggesting so casually, just doesn’t feel adequate to the depth of emotion it sparks in true devotees"""

example2="""Summer is a charming flirt. Easy-going and casual. Summer doesn.t huff and puff to win our affections. It has us at "Hello". Winter broods like the tortured protagonist of big fat Russian novel. It is dauting and dramatic, burning with a slow intensity.The season's reputation precedes itself, and often, not in a good way. It has a way of whittling down everything to its bare bones. Even relationship not attuned to its ebbs and flows can fray. At a dinner conversation I once attended. I listened in bemusement as a recent divorcee made the case that it was the Scandinavian frost that had cooled his ex-wife's ardor. How original."""

example3="""One of the finer books I read this year was John Kaag’s Hiking With Nietzsche, in which Kaag, a professor of philosophy, rekindles his passion for the German thinker while tracing picturesque hiking trails in the mountains of Switzerland. It’s a near-precise rendering of the travelogue as a self-help book. A young Kaag was an avowed Nietzsche acolyte but given the ravages of responsibilities and adulthood, the writer put his affinity to test by undertaking physically enduring hikes through the Alps, revisiting haunts that the philosopher escaped to, in search of solitude and salve. The journey’s demands, coupled with his own inner turmoil, are catnip for anybody feeling at cross purposes with their own life."""

from sklearn.feature\_extraction.text import TfidfVectorizer

vect = TfidfVectorizer(binary = True)

corpus = [example1,example3]

vect.fit(corpus)

TfidfVectorizer(analyzer=u'word', binary=True, decode\_error=u'strict',

dtype=<type 'numpy.int64'>, encoding=u'utf-8', input=u'content',

lowercase=True, max\_df=1.0, max\_features=None, min\_df=1,

ngram\_range=(1, 1), norm=u'l2', preprocessor=None, smooth\_idf=True,

stop\_words=None, strip\_accents=None, sublinear\_tf=False,

token\_pattern=u'(?u)\\b\\w\\w+\\b', tokenizer=None, use\_idf=True,

vocabulary=None)

from sklearn.metrics.pairwise import cosine\_similarity

similarity = cosine\_similarity(vect.transform([example1]).toarray(), vect.transform([example2]).toarray())

print(similarity)

[[0.31005704]]