**Web Mining (CSE3024)**

**Lab Assignment 3**

Name: **Kritika Mishra**

Registration Number: **16BCI0041**

Slot: L15+L16

Date: 28th August 2014

Question: **Write a program that collects all the words from a set of documents. Build an index from the words. Know what indexing is and Represent a document using the inverted index using python. Also implement a search for (multiple) terms from that index.**

**Code 1:**

from collections import defaultdict

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

def create\_index (data):

index = defaultdict(list)

for i, tokens in enumerate(data):

for token in tokens:

index[token].append(i)

print(index)

stop\_words = ['.',',','a','they','the','his','so','and','were','from','that','of','in','only','with','to']

with open('sample.txt', 'r') as myfile:

text=myfile.read().replace('\n', '')

myfile.close()

f\_text = []

tokens = word\_tokenize(text)

for i in tokens:

if i in stop\_words:

continue

else:

f\_text.append(i)

answer = ''

for i in f\_text:

answer+=i + ' '

#print(answer)

li = list(answer.split(" "))

print(li)

print("BREAK BREAK BREAK \n\n\n")

with open('sample1.txt', 'r') as myfile:

text1=myfile.read().replace('\n', '')

#print(text1)

f\_text1 = []

tokens1 = word\_tokenize(text1)

for i in tokens1:

if i in stop\_words:

continue

else:

f\_text1.append(i)

answer1 = ''

for i in f\_text1:

answer1+=i + ' '

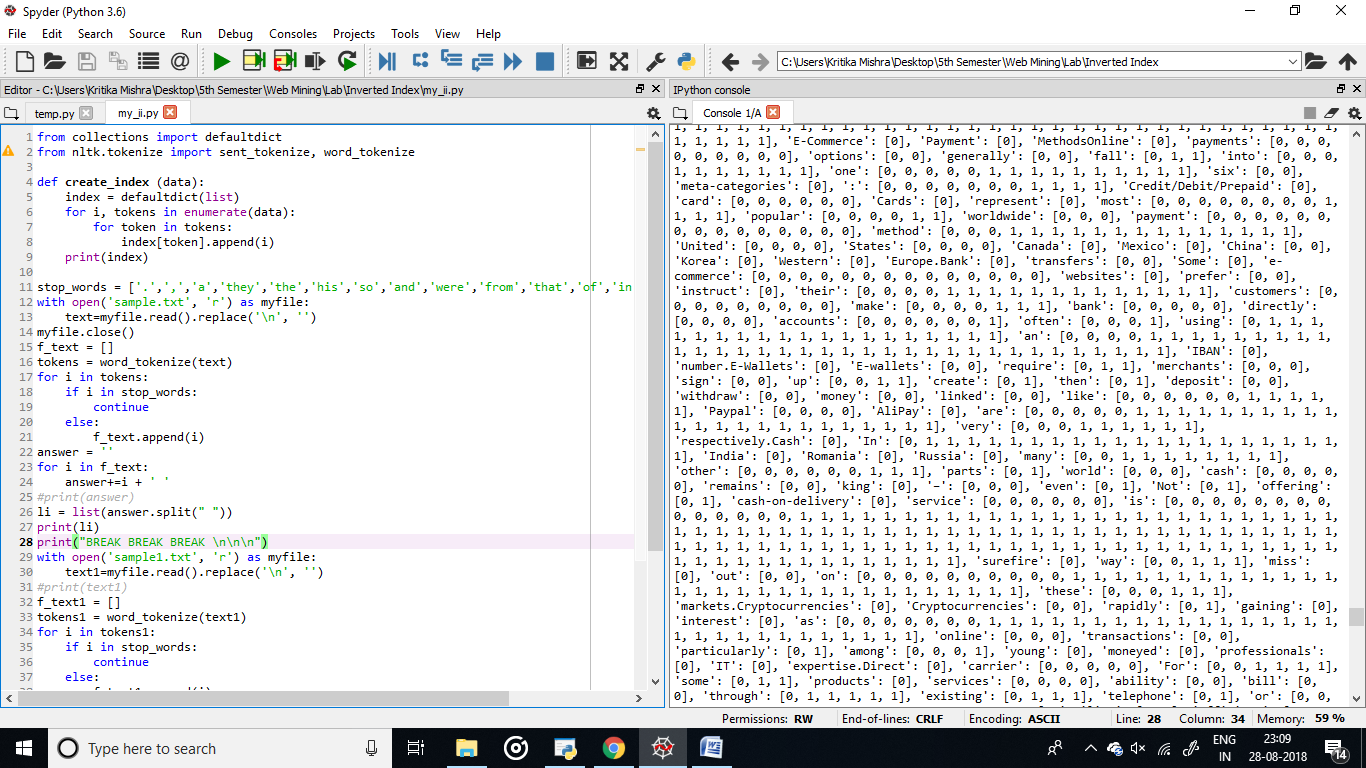
#print(answer)

li1 = list(answer1.split(" "))

print(li1)

f\_list=[li,li1]

create\_index(f\_list)



**Code 2:**

import re

from collections import defaultdict, Counter

def bold(txt):

return txt

DATA = [

{

'title': 'Django',

'description': 'Django is a high-level Python Web framework that '

'encourages rapid development and clean, pragmatic design. Built by '

'experienced developers, it takes care of much of the hassle of Web '

'development, so you can focus on writing your app without needing to '

'reinvent the wheel. It’s free and open source.'

},

{

'title': 'Python',

'description': 'Python is a programming language that lets you work '

'more quickly and integrate your systems more effectively.'

},

]

SPLIT\_RE = re.compile(r'[^a-zA-Z0-9]')

def tokenize(text):

yield from SPLIT\_RE.split(text)

def text\_only(tokens):

for t in tokens:

if t.isalnum():

yield t

def lowercase(tokens):

for t in tokens:

yield t.lower()

def stem(tokens):

for t in tokens:

if t.endswith('ly'):

t = t[:-2]

yield t

SYNONYMS = {

'rapid': 'quick',

}

def synonyms(tokens):

for t in tokens:

yield SYNONYMS.get(t, t)

def analyze(text):

tokens = tokenize(text)

for token\_filter in (text\_only, lowercase, stem, synonyms):

tokens = token\_filter(tokens)

yield from tokens

def index\_docs(docs, \*fields):

index = defaultdict(lambda: defaultdict(Counter))

for id, doc in enumerate(docs):

for field in fields:

for token in analyze(doc[field]):

index[field][token][id] += 1

return index

def combine\_and(\*args):

if not args:

return Counter()

out = args[0].copy()

for c in args[1:]:

for doc\_id in list(out):

if doc\_id not in c:

del out[doc\_id]

else:

out[doc\_id] += c[doc\_id]

return out

def combine\_or(\*args):

if not args:

return Counter()

out = args[0].copy()

for c in args[1:]:

out.update(c)

return out

COMBINE = {

'OR': combine\_or,

'AND': combine\_and,

}

def search\_in\_fields(index, query, fields):

for t in analyze(query):

yield COMBINE['OR'](\*(index[f][t] for f in fields))

def search(index, query, operator='AND', fields=None):

combine = COMBINE[operator]

return combine(\*(search\_in\_fields(index, query, fields or index.keys())))

def query(index, query, operator='AND', fields=None):

print('Search for "%s" using %s in %s' % (bold(query), bold(operator), fields or 'all fields'))

print('-'\*80)

ids = search(index, query, operator, fields)

for doc\_id, score in ids.most\_common():

print((bold(DATA[doc\_id]['title']),' found with score of ', bold(score)))

print('\n')

index = index\_docs(DATA, 'title', 'description')

query(index, 'Python')

query(index, 'Python', fields=['title'])

query(index, 'python', fields=['description'])

query(index, 'Python web')

query(index, 'Python web', 'OR')

query(index, 'quick')

query(index, 'rapid')

query(index, 'of')

