Web Mining (CSE3024)

Lab Assignment 3

Name: **Kritika Mishra**

Registration Number: **16BCI0041**

Slot: L15+L16

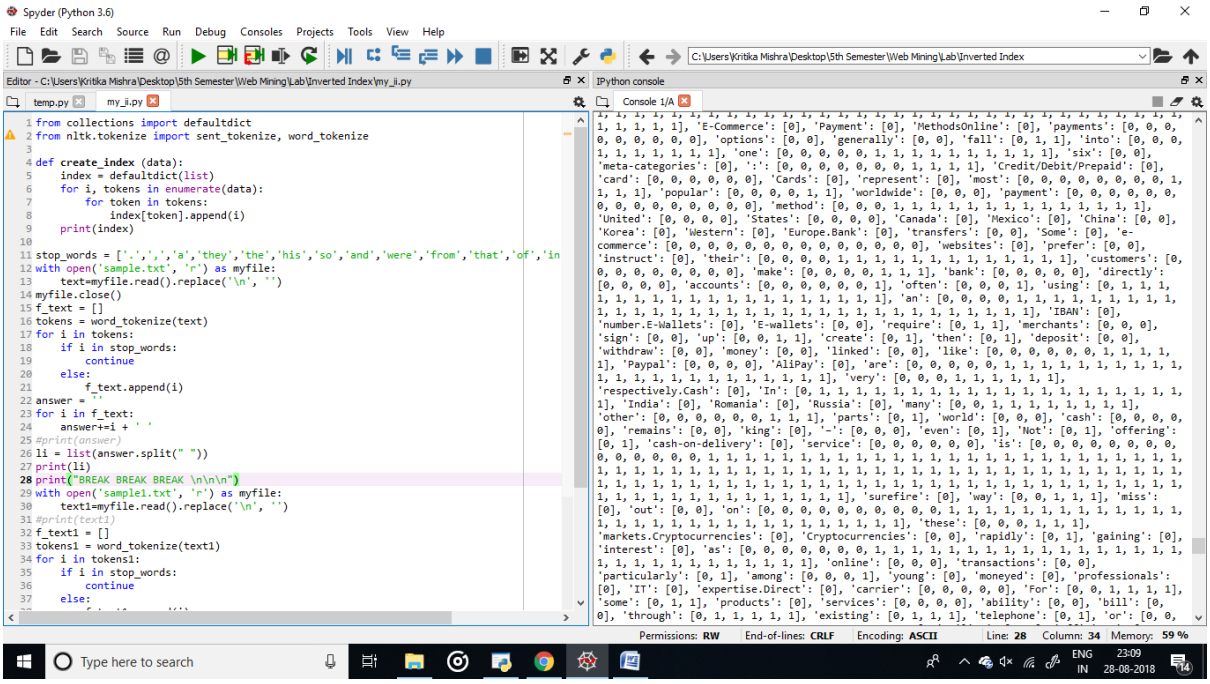
Date: 28th August 2014

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)
```



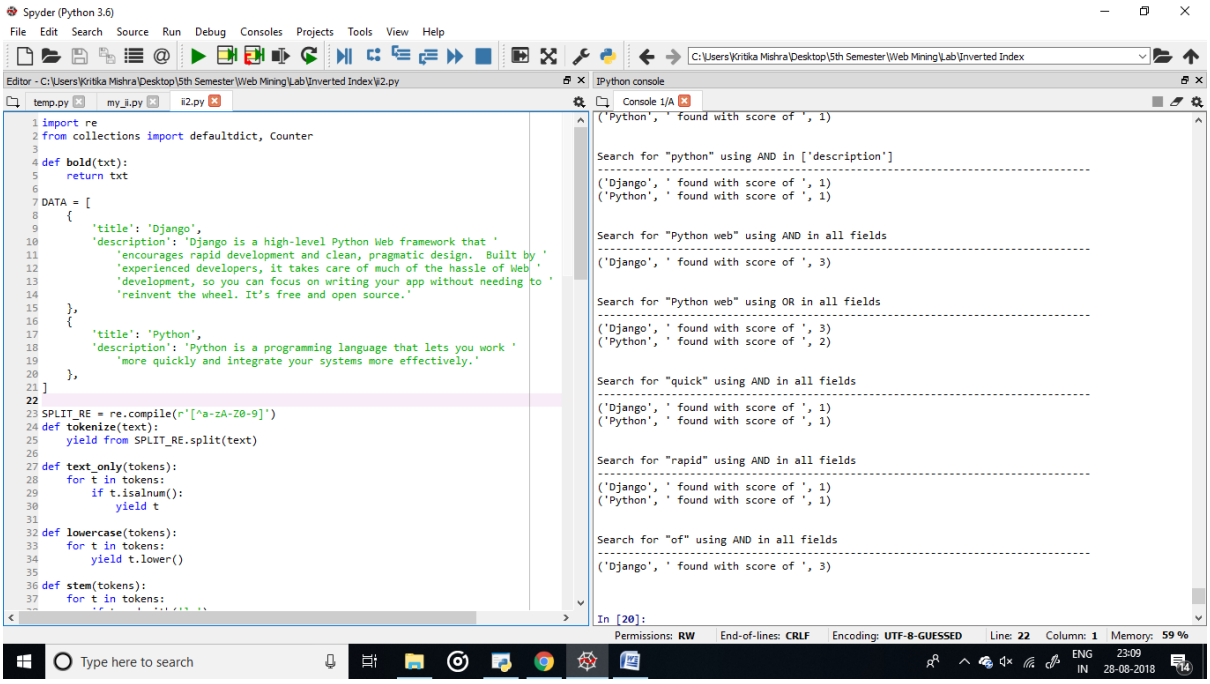
Code 2:

```
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')
```


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Kritika Mishra\Desktop\5th Semester\Web Mining\Lab\Inverted Index
Editor - C:\Users\Kritika Mishra\Desktop\5th Semester\Web Mining\Lab\Inverted Index\ii2.py
temp.py
my_ii.py
ii2.py
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:
IPython console
Console 1/A
('Python', ' found with score of ', 1)
Search for "python" using AND in ['description']
('Django', ' found with score of ', 1)
('Python', ' found with score of ', 1)
Search for "Python web" using AND in all fields
('Django', ' found with score of ', 3)
Search for "Python web" using OR in all fields
('Django', ' found with score of ', 3)
('Python', ' found with score of ', 2)
Search for "quick" using AND in all fields
('Django', ' found with score of ', 1)
('Python', ' found with score of ', 1)
Search for "rapid" using AND in all fields
('Django', ' found with score of ', 1)
('Python', ' found with score of ', 1)
Search for "of" using AND in all fields
('Django', ' found with score of ', 3)
In [20]:
Permissions: RW End-of-lines: CRLF Encoding: UTF-8-GUESSED Line: 22 Column: 1 Memory: 59 %
Type here to search
ENG
IN
23:09
28-08-2018