

山东大学 计算机科学与技术 学院

信息检索与数据挖掘 课程实验报告

学号：201600130053

姓名：王斌

班级：16 人工智能

实验题目：预处理文本数据集，并且得到每个文本的 VSM 表示。

实验内容：

#Homework 1: VSM

#预处理文本数据集，并且得到每个文本的 VSM 表示。

The 20 Newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups.

#20news-18828.tar.gz (<http://qwone.com/~jason/20Newsgroups/20news-18828.tar.gz>) ?- 20 Newsgroups; duplicates removed, only "From" and "Subject" headers (18828 documents)

实验环境：

Spyder+python3.6

Win10

实验过程中遇到和解决的问题：

(记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。)

一、实现前相关准备

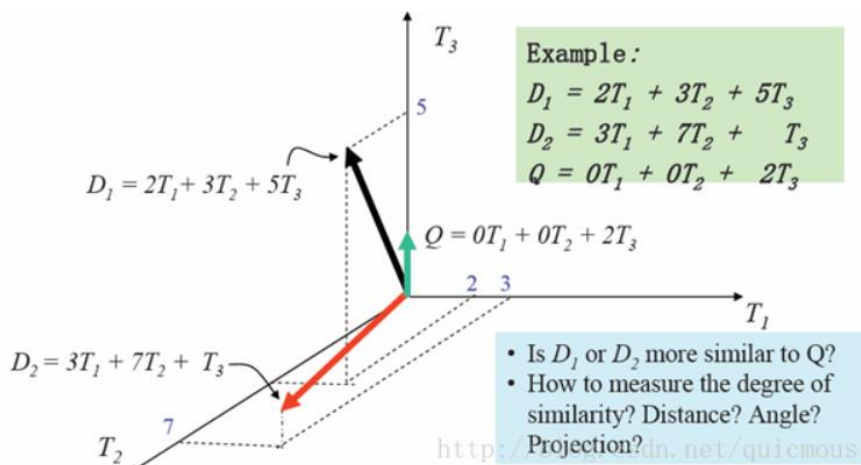
查阅相关 VSM 的资料，在此感谢以下网页所提供的内容：

<http://allmybrain.com/2007/10/19/similarity-of-texts-the-vector-space-model-with-python/>

<http://blog.josephwilk.net/projects/building-a-vector-space-search-engine-in-python.html/>

<https://blog.csdn.net/quicmous/article/details/71263844/>

向量空间模型中将文本表达为一个向量，看作向量空间中的一个点。



下面，就到了最关键的部分。基于tf-idf，我们将得到Weight Matrix（权重矩阵）。

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth ...
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0
MERCY	1.51	0.0	1.90	0.12	5.25	0.88
WORSER	1.37	0.0	0.11	4.15	0.25	1.95
...						

<http://blog.csdn.net/gcyxf>

二、Python 代码实现过程中所遇问题

1. 所有文档读取的实现，探索后使用 os 模块内的函数能高效完成，如下所示：

```
# 测试文档路径
Newspath=(r"C:\Users\93568\Documents\GitHub\DataMining\Homework1VSM\20news-18828")
#小规模测试文档路径
#Newspath=(r"C:\Users\93568\Documents\GitHub\0123")
folders=[f for f in os.listdir(Newspath)]
print(folders)
files=[]
for folderName in folders:
    folderPath=os.path.join(Newspath, folderName)
    files.append([f for f in os.listdir(folderPath)])

document_filenames={}
i=0
for fo in range(len(folders)):
    for fi in files[fo] :
        document_filenames.update({i:os.path.join(Newspath,os.path.join(folders[fo],fi))})
        i+=1
```

2. Gbk 和 unicode 编码转换问题：

```
document = f.read()
```

```
UnicodeDecodeError: 'gbk' codec can't decode byte 0xff in position
11597: illegal multibyte sequence
```


改变编码方式，忽略错误后，修改代码如下完美解决：

```
f = open(Newspath, 'r', encoding='utf-8', errors='ignore')
fp=open(r"**** \GitHub\dictionary.txt", 'w', encoding='utf-8')
```

3. 文档预处理 tokenize 相关问题:

使用 lower()、split() 以及 strip(characters) 后发现效果不理想, 创建得词典太大而且杂乱, 打印到 txt 文档后居然有 3M 多, 于是决定使用正则化的 re.sub(), 去掉除字母外的所有符号字符 (包括数字), 得到纯单词, 测试后得到的词典效果大为好转, 如下:

```
def tokenize(document):
    terms=document.lower()
    terms=re.sub(r'\W|\d|_', "", terms)
    terms=re.sub(r"\s{2,}", " ", terms)
    terms=terms.split()
    return [term for term in terms]
```

 dictionary.txt 2018/9/25 11:12 文本文档 876 KB

<<VAM_DICTIONARY>>

mqthh fluorescent nrizwt government comms stojadinovic presbytery thorough rutkj sheikh undergo sexual amdahl faster
 difficult gook nflkid joad sebinkarahisari groups multiplexes gris lumpkin scottmi ratzlav gev jdi bobl thtjp mcgee pstlb
 includeing meharg irrelevance baha rezin beyound usefully sanction cottage brighten bbbbbb nobleman weeds nite
 defragmenting impudently explanation chaining mgz nbz whitchurch firenza quadrimomials gaffney sen pronouncing
 wqzq barryf procede automata xcalc lilley clift qksuq viacheslav pnml smacking beamers bandages petri wrong oaw nut
 trionfo indefensible mug zay swimmer mpd gotchas awaiting zbe thinly lawton rejects nhara cumbersome nameless
 calculated swindled versus stadium etha accession abstaining klaus clive ndallen bitter ddennis wheat sickens boyadjian
 gunshyjudges listmember terrier sunlib casavant rvo digitalk axon andreadis cwruslip haverstock conceptions deckard
 rtfuhge pythagorean summaries otdy cuesta remembrance seizure potentialtion refreshingly urtm constants shutoff
 bridging trod kse strike dhalden kzum edtg nc render stony crawls contradicton semipermeable ahmad mcauley hofkin

4. 向量权重取值计算, TF/IDF 相关方法代码实现:

$$TF-IDF = tf * idf$$

计算词频(tf):

```
def initialize_terms_and_postings():
    global dictionary, postings
    for id in document_filenames:
        f = open(document_filenames[id], 'r', encoding='utf-8', errors='ignore')
        document = f.read()
        f.close()
        terms = tokenize(document)
        d_tnum=len(terms)#预处理后每篇文档的总词数
        print(d_tnum)
        unique_terms = set(terms)
        dictionary = dictionary.union(unique_terms)#并入总词典
        for term in unique_terms:
            #postings[term][id] = terms.count(term)
            postings[term][id] = (terms.count(term))/d_tnum
            # the value is the frequency of term in the document
```

计算 df 及 idf:

```
def initialize_document_frequencies():

    global document_frequency
    for term in dictionary:
        document_frequency[term] = len(postings[term])
```

```
def inverse_document_frequency(term):
    if term in dictionary:
        return math.log(N/document_frequency[term],2)
    else:
        return 0.0
```

计算权重:

```
return postings[term][id]*inverse_document_frequency(term)
```

三、在已有实现上新增功能应用:

>>>增加查询功能:

根据用户输入的单词或一句话或一段话,对其预处理后在已有的向量空间模型中进行相似度查询(类似于搜索引擎的网页搜索),然后返回最为相似的文档,根据相似值降序排列,相关核心代码如下:

```
def do_search():

    query = tokenize(input("Search query >> "))
    if query == []:
        sys.exit()
    # find document ids containing all query terms. Works by
    # intersecting the posting lists for all query terms.
    relevant_document_ids = intersection(
        [set(postings[term].keys()) for term in query])
    if not relevant_document_ids:
        print ("No documents matched all query terms.")
    else:
        scores = sorted([(id,similarity(query,id))
                        for id in relevant_document_ids],
                        key=lambda x: x[1],
                        reverse=True)
        print ("Score: filename")
        for (id,score) in scores:
            print (str(score)+" "+document_filenames[id])

def intersection(sets):

    return reduce(set.intersection, [s for s in sets])

def similarity(query,id):

    similarity = 0.0
    for term in query:
        if term in dictionary:
            similarity += inverse_document_frequency(term)*imp(term,id)
    similarity = similarity / length[id]
    return similarity
```

小样本数据测试结果如下:

```
Search query >> math school
No documents matched all query terms.

Search query >> get book
Score: filename
0.9000328540276249: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53410
0.4820689396014227: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53527
0.47335223569819695: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53604
0.3892750837864274: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53183
0.3558999662650913: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53521
0.2746217847955606: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53170
0.26625802964410766: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53557
0.26516979986736644: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53362
0.25690668052488785: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53790
0.25597554742990725: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\54187
0.24044789577304335: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53331
0.23343631997464614: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53808
0.21902313365667422: C:\Users\93568\Documents\GitHub\0123\alt.atheism
\53493
```

结论分析与体会：

向量空间模型基本实现，并且能在模型基础上进行一些基本的测试，但由于对于文档的处理完全都是有编写的函数完成，入对单词的单复数、形式等未能处理好，还有对于过于低频和高频的用处不大的单词未处理，都有待优化。