信息检索与数据挖掘

实验报告-实验一

山东大学 SHANDONG UNIVERSITY

班	级	智能
学	号	201900161140
姓	名	张文浩
时	间	2021 年 9 月 16 日

实验内容

Inverted index and Boolean Retrieval Model

– 使用我们介绍的方法，在 tweets 数据集上构建 inverted index;

– 实现 Boolean Retrieval Model，使用 TREC 2014 test topics 进行测试；

– Boolean Retrieval Model：

• Input：a query (like Ron and Weasley)

• Output: print the qualified tweets.

• 支持 and, or ,not；查询优化可以选做；

实验环境：

win10+spyder（anaconda）

实验步骤

一、数据预处理

读入数据，将“tweetid”作为 docid，并提取出来。将 username 作为 text 的一部分加入待处理的文本。

```
f = open(r"E:\datamining\1.txt")
lines = f.readlines()
```

进行数据预处理，先将所有大写转小写，利用 TextBlob 库对 text 文本进行词干提取（名词变单复数等操作），利用 lemmatize 函数进行词形还原

```
#处理当前行得到tweetidid和text文本
tweetid = Word(line['tweetid'])
text = TextBlob(line['text']+" "+line['username']).words.singularize()
texts = []
for word in text:
    if word in uselessword:
        continue
    temp = Word(word)
    temp = temp.lemmatize("v") #词形还原
```

二、构建 postings

建立一个处理整个文本的函数，来构建 postings 表。

一行一行读入，先将所有字母变为小写，调用上面的函数，得到每一个 document 的 docid 和 terms，构建 postings 表。在处理完所有 document 之后对 postings 表进行排序操作，按照 docid 进行排序，便于优化查询处理

```
    #构建postings表
    #print(texts)
    tempset = set(texts)
    for term in tempset:
        if term in postings.keys():
            postings[term].append(tweetid)
        else:
            postings[term]=[tweetid]
#建立完postings后要进行排序，方便后面优化查询，提高效率
for term in postings.keys():
    postings[term].sort()
```

三、构建查询函数

构建是函数分别处理 and，or，not 操作，三个函数均采用双指针算法进行优化查询，在 O（n）的时间复杂度下解决问题

```python
def myand(a1,a2):
    global postings        #全局变量
    res = []
    if(a1 not in postings) or (a2 not in postings):
        return res
    else:
        #计算a1，a2对应postings数组的长度
        len1 = len(postings[a1])
        len2 = len(postings[a2])
        x=0
        y=0
        #优化查询，双指针算法
        while x<len1 and y<len2:
            if postings[a1][x]==postings[a2][y]:
                res.append(postings[a1][x])
                x += 1
                y += 1
            elif postings[a1][x]<postings[a2][y]:
                x += 1
            else:
                y += 1
        return res
```

```python
def myor(a1,a2):     #或操作
    res = []
    if(a1 not in postings) and (a2 not in postings):
        return res
    elif a2 not in postings:
        res = postings[a1]
    elif a1 not in postings:
        res = postings[a2]
    else:   #a1,a2都存在
        len1 = len(postings[a1])
        len2 = len(postings[a2])
        x=0
        y=0
        #同样采用双指针算法进行优化查询
        while x<len1 and y<len2:
            if postings[a1][x]==postings[a2][y]:
                res.append(postings[a1][x])
                x += 1
                y += 1
            elif postings[a1][x]<postings[a2][y]:
                res.append(postings[a1][x])
                x += 1
            else:
                res.append(postings[a2][y])
                y += 1
        if (x < len1):
            while( x<len1 ):
                res.append(postings[a1][x])
                x += 1
        if (y < len2):
            while( y<len2 ):
                res.append(postings[a2][y])
                y += 1
    return res
```

```python
def mynot(a1,a2):      #非操作
    res = []
    if a1 not in postings:
        return res
    elif a2 not in postings:
        res = postings[a1]
        return res
    else:
        len1 = len(postings[a1])
        len2 = len(postings[a2])
        x=0
        y=0
        #双指针优化查询
        while x<len1 and y<len2:
            if postings[a1][x] == postings[a2][y]:
                x += 1
                y += 1
            elif postings[a1][x]<postings[a2][y]:
                res.append(postings[a1][x])
                x += 1
            else:
                y += 1
        if(x < len1):
            while( x<len1 ):
                res.append(postings[a1][x])
                x += 1
    return res
```

优化查询：

仅做了 and 的优化查询操作，输入时可输入多个单词，单词之间用空格分隔开，查询这几个单词同时出现的 docid。

函数实现如下：

```python
def inter(a1,a2):
    res = []
    if(len(a1) == 0) or (len(a2) == 0):
        return res
    else:
        #计算a1，a2对应postings数组的长度
        len1 = len(a1)
        len2 = len(a2)
        x=0
        y=0
        #双指针算法
        while x<len1 and y<len2:
            if a1[x]==a2[y]:
                res.append(a1[x])
                x += 1
                y += 1
            elif a1[x]<a2[y]:
                x += 1
            else:
                y += 1
        return res

def intersect(query):
    global postings
    #按照出现频率排序，查询优化
    query = sorted(query,key=lambda x:len(postings[x]))
    res = postings[query[0]]
    for i in range(len(query)):
        res = inter(res,postings[query[i]])
    return res
```

inter 函数是将两个序列的取交集操作，待 intersect 调用

在 intersect 函数中先对输入的单词按照在 document 中出现的次数从小到大排序，进行优化查询，再进行两两合并。

构建处理 input 的 search 查询函数，在接收输入的单词之后也要进行 singularize 和 lemmatize 数据归一化词形还原的预处理操作。

```python
def myinput(doc):
    #处理手动输入的数据，方便与terms中的存好的term进行匹配
    doc = doc.lower()
    terms = TextBlob(doc).words.singularize()
    res = []
    for term in terms:
        term = Word(term)
        term = term.lemmatize("v") #词形还原
        res.append(term)
    return res
```

```python
def search():       #查询
    query = myinput(input("请输入要查询词,退出请键入exit:  "))
    if (query[0] == "exit") or (query == []):
        sys.exit()
    if len(query) == 3:
        if query[1] == "and":
            answer = myand(query[0],query[2])
            print(answer)
        elif query[1] == "or":
            answer = myor(query[0],query[2])
            print(answer)
        elif query[1] == "not":
            answer = mynot(query[0],query[2])
            print(answer)
        else:
            print("输入错误")
    else:
        length = len(query)
        rankdic = rank(query)
        print("[Rank_Score: Tweetid]")
        for(tweetid,score) in rankdic:
            print(str(score/length)+ ": " + tweetid)
```

实验结果

and：

```
请输入要查询词,退出请键入exit:  i and like
数量为
197
['29173602393784320', '29197406167900160',
'29218495354904576', '29367651801243648',
'29377314487799808', '29385533499121665',
'29391931402358784', '297314564461195264',
'297437302362357760', '297485604000784384',
'297580575596883968', '297682430066835456',
'297878452214353920', '298119522890547 21',
```

or：

```
请输入要查询词,退出请键入exit:  i or like
数量为
2708
['28971749961891840', '28974862038994945',
'28981945648021504', '28995339910389760',
'29002642571132928', '29015605554188289',
'29017472875102209', '29021146103939072',
'29028510521630720', '29035904588845056',
'29038355773657088', '29047015467917312',
```

not

```
请输入要查询词,退出请键入exit:  i not like
数量为
2137
['28971749961891840', '28974862038994945',
'28981945648021504', '28995339910389760',
'29015605554188289', '29017472875102209',
'29021146103939072', '29035904588845056',
'29038355773657088', '29047015467917312',
'29052291231252481', '29101955431276544',
'29138077012205568', '29178579929538561',
```

and 优化查询

```
请输入要查询词,退出请键入exit:  i love you
and优化查询
数量为
40
['29257299906269184', '29331910735958016',
'29687259007553536', '297854975344791553',
'299325120873390080', '299710652874911744',
'300042011300478977', '300396903940624384',
'301797008757383168', '30312146520776704',
'304544579879845888', '307192657879396352',
'307585630601371648', '308963602075885568',
'309048939410239489', '313911144358227969',
'31494358188425218', '31549597696393216',
'316302606484004864', '317066154348797952',
'317438558241705984', '317786983067811 84',
```

实验总结

通过本次实验，我实现在 tweets 数据集上构建 inverted index， 学习了如何利用

TextBlob 库进行文本预处理操作，实践了 and，or，not 的优化查询，对课本上的内容有了更加清晰的认识