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：201600301304	姓名：贾乘兴	班级：人工智能 16
实验题目：布尔检索模型		

实验内容：

一. 对 tweets 文本进行处理，建立关键词的文档频率与出现文档名的列表

1. json 格式读取文本
2. 文本预处理
3. 建立倒排索引（文档频率与包含的文档名）
4. 代码

```
# 去除停用词
def cutstopwords(str):
    stopwords = {}.fromkeys([line.rstrip() for line in
open('estopwords.txt')])
    segs = str.replace('\n','').lower().split(' ')
    new_str = ''
    for seg in segs:
        if seg not in stopwords:
            new_str = new_str + " " +seg
    return new_str

# 去除标点
def cutsyms(str):
    new_str = re.sub('[,.\'\"\t\n*_+=?/|!@#$%^&*()`~<>:;\-\[\]]',"
",str)
    return new_str

# 词干提取
def stemming(str):
    s = nltk.stem.SnowballStemmer('english')
    segs = str.replace('\n', '').lower().split(' ')
    new_str = ''
    for seg in segs:
        new_str = new_str + " " + s.stem(seg)
    return new_str

# 读取文本并进行处理
path = '/Users/apple/Desktop/ir/hw3/tweets.txt'
file = open(path,'r',encoding='UTF-8',errors='ignore')
tweets = []
twords = {}
```

```
counts = 0
alist = []
for line in file:
    tweets.append(json.loads(line))
    tweets[counts]['text'] = tweets[counts]['text'].lower()
    tweets[counts]['text'] = cutsyms(tweets[counts]['text'])
    tweets[counts]['text'] = cutstopwords(tweets[counts]['text'])
    #tweets[counts]['text'] = stemming(tweets[counts]['text'])
    for seg in tweets[counts]['text'].split(' '):
        if seg in twords:
            length = len(twords[seg])
            if twords[seg][length-1]!=counts:
                twords[seg].append(counts)
                twords[seg][0] = twords[seg][0] + 1
        else:
            twords[seg] = []
            twords[seg].append(1)
            twords[seg].append(counts+1)
    counts = counts + 1
    alist.append(counts)
    print(counts)
file.close()
```

二. 建立基本的查询结构，以 and、or、not 为基本关系

1. and 关系：将两个 list 进行交操作，线性复杂度，代码如下：

```
def fAnd(listA,listB):
    list = []
    la = len(listA)
    lb = len(listB)
    ca = 0
    cb = 0
    while (ca!=la)&(cb!=lb):
        if (listA[ca]==listB[cb]):
            list.append(listA[ca])
            ca = ca + 1
            cb = cb + 1
        else:
            if listA[ca]>listB[cb]:
                cb = cb + 1
            else:
                ca = ca + 1
    return list
```

2. or 关系：将两个 list 进行并操作，线性复杂度，代码如下：

```
def fOr(listA,listB):
    list = []
    la = len(listA)
    lb = len(listB)
    ca = 0
    cb = 0
    while (ca!=la)&(cb!=lb):
        if (listA[ca]==listB[cb]):
            list.append(listA[ca])
            ca = ca + 1
            cb = cb + 1
        else:
            if listA[ca]>listB[cb]:
                list.append(listB[cb])
                cb = cb + 1
            else:
                list.append(listA[ca])
                ca = ca + 1
        if (ca==la)&(cb<lb):
            for i in range(cb, lb):
                list.append(listB[i])
            break
        if (cb==lb)&(ca<la):
            for i in range(ca, la):
                list.append(listA[i])
            break
    return list
```

3. not 关系：取 list 的补集，线性复杂度，代码如下：

```
def fNot(listA):
    la = len(listA)
    list = []
    ca = 0
    global counts
    for i in range(counts):
        if ca<la:
            if listA[ca] > i:
                list.append(i)
            else:
                if listA[ca] < i:
                    ca = ca + 1
    return list
```

三. 基于 and、or、not 关系建立较为复杂的查询关系

1. 确定优先级为先 not 后 and 最后执行 or 的优先级次序
2. 注意对 query 的处理与原文本处理一致
3. 简单的布尔检索，不支持模糊查询
4. 代码：

```
def testA(test):
    # 使用函数实现一定程度的多元关系的检索
    # test = 'NOT home AND house AND NOT sarge OR NOT circle AND
unlikely AND NOT recall'
    test = cutsyms(test)
    tlist = test.split(' ')
    tcount = {}
    i = 0
    while i < len(tlist):
        if (tlist[i] != 'NOT') & (tlist[i] != 'AND') & (tlist[i] !=
'OR'):
            tcount[tlist[i]] = twords[tlist[i]][1:len(twords[tlist[i]])]
        i = i + 1
    i = 0
    while i < len(tlist):
        if tlist[i] == 'NOT':
            tcount[tlist[i + 1]] = fNot(tcount[tlist[i + 1]])
            tlist.remove(tlist[i])
            i = i - 1
        i = i + 1
    print(tlist)
    i = 0
    while i < len(tlist):
        if tlist[i] == 'AND':
            tcount[tlist[i + 1]] = fAnd(tcount[tlist[i - 1]],
tcount[tlist[i + 1]])
            tlist.remove(tlist[i])
            tlist.remove(tlist[i - 1])
            i = i - 1
        i = i + 1
    print(tlist)
    i = 0
    while i < len(tlist):
        if tlist[i] == 'OR':
            tcount[tlist[i + 1]] = fOr(tcount[tlist[i - 1]],
tcount[tlist[i + 1]])
            tlist.remove(tlist[i])
            tlist.remove(tlist[i - 1])
            i = i - 1
        i = i + 1
```

```
print(tlist)
print(tcount[tlist[len(tlist) - 1]])
```

5. 对问题中的 query 进行回答

```
def qTest():
    # query
    path = '/Users/apple/Desktop/ir/hw3/topics.MB171-225.txt'
    file = open(path, 'r', encoding='UTF-8', errors='ignore')
    txt = file.read()
    file.close()
    txt.replace('\n', ' ')
    txtlist = txt.split(' ')
    q = 0
    for i in range(len(txtlist)):
        if txtlist[i] == '</num>\n<query>':
            q = q + 1
            i = i + 1
            klist = alist
            while txtlist[i] != '</query>\n<querytime>':
                txtlist[i] = cutsyms(txtlist[i].lower())
                if txtlist[i] in twords:
                    klist = fAnd(klist, twords[txtlist[i]])
                i = i + 1
            if klist == alist:
                klist = []
            print(q)
            print(klist)
```

四. 结果分析

得到的结果中，我们发现部分 query 无法被回答（返回 list 为空），分析可得对查询到的 list 处理较为简单，没有从语义层面和词间关系上进行分析

结论分析与体会： 本次实验对倒排索引和布尔检索进行了简单的实现，同时分析了一些复杂问题的情形