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:201800130036	姓名： 戎思宇	班级：dashuj
实验题目：Inverted index and Boolean Retrieval Model		
实验学时：2	实验日期： 2020.10.10	
实验目的：实现 Boolean Retrieval Model		
软件环境：		
实验内容与设计： 1. 实验内容（题目内容，输入要求，输出要求） 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; 2. 算法描述 （整体思路描述，所需要的数据结构与算法） 预处理过程使用 python 将数据集以 tweet 为单位进行读取，并对字符串切片，完成对属性分割，建立倒排索引，布尔查询 3. 测试结果（测试输入，测试输出，结果分析） 4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）		
分析与体会： 学会了初步处理数据来获得有效信息，了解了布尔查询和倒排索引如何处理查询 and or not 的操作以及查询优化		

附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）

```
import sys
from textblob import TextBlob
from textblob import Word
from collections import defaultdict

uselessTerm = ["username","text","tweetid"]
postings = defaultdict(dict)
```

```
def merge2_and(term1,term2):
    global postings
    answer = []
    if (term1 not in postings) or (term2 not in postings):
        return answer
    else:
        i = len(postings[term1])
        j = len(postings[term2])
        x=0
        y=0
        while x<i and y<j:
            if postings[term1][x]==postings[term2][y]:
                answer.append(postings[term1][x])
                x+=1
                y+=1
            elif postings[term1][x] < postings[term2][y]:
                x+=1
            else:
                y+=1
        return answer

def merge2_or(term1,term2):
    answer=[]
    if (term1 not in postings)and(term2 not in postings):
        answer = []
    elif term2 not in postings:
        answer = postings[term1]
    elif term1 not in postings:
         answer = postings[term2]
    else:
        answer = postings[term1]
        for item in postings[term2]:
            if item not in answer:
                answer.append(item)
    return answer

def merge2_not(term1,term2):
    answer=[]
    if term1 not in postings:
        return answer
    elif term2 not in postings:
        answer = postings[term1]
        return answer
```

```
        else:
            answer = postings[term1]
            ANS = []
            for ter in answer:
                if ter not in postings[term2]:
                    ANS.append(ter)
            return ANS


def do_rankSearch(terms):
    Answer = defaultdict(dict)
    for item in terms:
        if item in postings:
            for tweetid in postings[item]:
                if tweetid in Answer:
                    Answer[tweetid]+=1
                else:
                    Answer[tweetid] = 1
    Answer = sorted(Answer.items(),key = lambda asd:asd[1],reverse=True)
    return Answer


def token(doc):
    doc = doc.lower()
    terms=TextBlob(doc).words.singularize()

    result=[]
    for word in terms:
        expected_str = Word(word)
        expected_str = expected_str.lemmatize("v")
        result.append(expected_str)
    return result

def tokenize_tweet(document):

    document=document.lower()
    #提取用户名、tweet 内容和 tweetid 三部分主要信息
    a = document.index("username")
    b = document.index("clusterno")
    c = document.rindex("tweetid")-1
    d = document.rindex("errorcode")
    e = document.index("text")
    f = document.index("timestr")-3
    document = document[c:d]+document[a:b]+document[e:f]
```

```
    #print(document)
    terms=TextBlob(document).words.singularize()

    result=[]
    for word in terms:
        expected_str = Word(word)
        expected_str = expected_str.lemmatize("v")
        if expected_str not in uselessTerm:
            result.append(expected_str)
    return result


def get_postings():

    global postings
    f = open(r"D:/777/tweets.txt")
    lines = f.readlines()#读取全部内容

    for line in lines:
        line = tokenize_tweet(line)
        #print(line)
        tweetid = line[0]
        line.pop(0)
        unique_terms = set(line)
        for te in unique_terms:
            if te in postings.keys():
                postings[te].append(tweetid)
            else:
                postings[te] = [tweetid]
    #按字典序对 postings 进行升序排序,但返回的是列表，失去了键值的信息


def do_search():
    terms = token(input("Search query >> "))
    if terms == []:
        sys.exit()
    #搜索的结果答案

    if len(terms)==3:
        #A and B
        if terms[1]=="and":
            answer = merge2_and(terms[0],terms[2])
            print(answer)
        #A or B
```

```
            elif terms[1]=="or":
                answer = merge2_or(terms[0],terms[2])
                print(answer)
            #A not B
            elif terms[1]=="not":
                answer = merge2_not(terms[0],terms[2])
                print(answer)
            else:
                print("input wrong!")

        #进行自然语言的排序查询，返回按相似度排序的最靠前的若干个结果
        else:
            leng = len(terms)
            answer = do_rankSearch(terms)
            print ("[Rank_Score: Tweetid]")
            for (tweetid,score) in answer:
                print (str(score/leng)+": "+tweetid)

def main():
    get_postings()
    while True:
        do_search()


if __name__ == "__main__":
    main()
```