山东大学 计算机科学与技术 学院

信息检索 课程实验报告

学号：201705111123	姓名： 孙维纬	班级： 17 数据

实验题目：Inverted index and Boolean Retrieval Model

实验步骤

使用 python 实现倒排索引和布尔查询

1，建立词到文档的索引表：遍历数据集，对每个 text 以空格分词，用 set 建词典，用 dict 建词到 text id 的映射。同时建立字母到 text 的索引。

2，建立词片段到词表的索引：遍历词表，用{作为词的开始结束符，用二维数组建[a][b]->(word)abc 的映射

3，布尔原子操作：实现 and，or，not，xor，输入 seq1，seq2(or max_len)，输出操作后的 seq，复杂度为 O(seq1+seq2)

4，表达式解析：根据顺序运算的原则遍历表达式，依次通过布尔原子操作完成运算，对于(...)使用递归处理。函数同时传入表达式和数据源，对于表达式中的符号从数据源中获得索引的序列

5，表达式重整：对于用户输入的表达式，重整为用一个空格分开的布尔运算符和查询索引的重整表达式，对连续的布尔运算基于运算符优先级进行重排以优化查询效率

6，对表达式中的查询索引通过此片段到词，词到文档的两步查询获得对 text 查询结果，以 dict 形式存入数据源中

7，输出结果：可以通过<...>运算符定义输出次序，对于查询结果使用.count(...)排序，<>可以写在表达式中或者在表达式后定义

实验结果，

1，建立词到文档的索引表

```
C:\Users\SUNWW\Anaconda3\envs\tf2\python.exe
data loaded
27799 53113
```

建立了 53113 个词到 27799 个 text 的索引，实验中去除了所有非英文字母的符号，忽略了重复文本。

2，建立词片段到词表的索引

建立二维数组 27*27 表示从[a][b]->(word)abc 的映射，用{作为词的开始结束符因为{的 ascii 编码在 z 后面便于编程

3，布尔原子操作

输入 -> 输出

```
print(_and([1, 3, 4], [3, 4, 5, 6, 7, 8]))      [3, 4]
print(_or([1, 3, 4, 8], [3, 4, 5, 6, 7, 8]))    [1, 3, 4, 5, 6, 7, 8]
print(_not([1, 3, 4, 8], 10))                   [0, 2, 5, 6, 7, 9]
print(_xor([1, 3, 4, 8], [3, 4, 5, 6, 7, 8]))   [1, 5, 6, 7]
```

完成高效的布尔运算

4，表达式解析

输入的表达式和数据源

```
print(cp('a && ( b || ( ! c ) )', {'a': [1, 2, 3], 'b': [3, 4, 5], 'c': [2, 3, 4]}))
```

输出的结果

```
[1, 3]
```

算法可以适应各种组合和长度的合法的表达式，从左向右依次运算，(...)递归处理

5，表达式重整

```
$ a&& b ||(! c)
[optimize] a && b || ( ( ! c ) )
```

重整为用一个空格分开的布尔运算符和查询索引的重整表达式，对！添加()

```
$ a||b&&(c||!(d&&e)&&f)
[optimize] a && ( c && f || ( ! ( d && e ) ) ) || b
```

对连续的布尔运算基于运算符优先级进行重排以优化查询效率，&&具有高于||的优先级

```
$ is{||((!ab ||askm) && < lm>) &&(s{ab && cc)
[optimize] is{ && ( s{ab && cc ) || ( ( ( ! ab ) || askm ) && lm )
```

一个更复杂的表达式，重整后去除了<...>

如果表达式中两个查询词间无运算符默认为 and

6，从词片段到 text，得到数据源

```
$ hello world
[optimize] hello world
[source]
hello : [1045, 4015, 5185, 5550, 6031, 7766, 10
world : [10, 49, 89, 114, 116, 120, 124, 287, 4

$ ron && weasley && birthday
[optimize] ron && weasley && birthday
[source]
ron : [13, 19, 85, 93, 103, 118, 168, 212, 266, 283, 332, 4
weasley : [10058, 10849, 14844, 16063, 16192, 16201, 16225,
birthday : [4988, 5531, 6392, 8404, 8983, 9785, 10058, 1072
```

对于每个查询词通过索引表获得索引结果

```
$ {hello{ wor{ld
[optimize] {hello{ wor{ld
[source]
{hello{ : [1045, 5185, 5550, 6031, 7766, 12156
wor{ld : []
```

其中对于每个词的检索通过词中字母所有字母组合 and 得到

如 wor{ld = wo && or && r{ && {l && ld

后续表达式和数据源一起进入布尔运算函数

7，输出结果

```
 $ {hello{ {world{
 [optimize] {hello{ {world{
 Find out about 1 results
   14639 Hello World, Cornell scientists 3D print ears w

$ hello world
[optimize] hello world
Find out about 2 results
  12156 Bye Bye #USAirways - hello #AmericanAirline
  14639 Hello World, Cornell scientists 3D print ea
```

通过上述步骤获得查询结果，显示 text 和 id

```
 $ <love>
 [optimize] love
 Find out about 370 results
   12013 \u266a I, I love you like John loves Sherlock! I, I love you
   9671 @kylieminogue #loveisloveislove!!  Equal marriage UK Yes Vote
   15225 I LOVE BEN AFFLECK AND I LOVE ARGO AND I LOVE THE OSCARS. AN
   4640 Harbor Fish Cafe on #Yelp: Love going here when the SUN is ou
```

<...>中的字符串出现的次数作为排序依据，对检索结果按降序排列

```
 $ weasley && birthday <ron>
 [optimize] weasley && birthday
 Find out about 58 results
   16237 It's Ron Weasley's birthday! The ginger
   16263 #HappyBirthdayRonWeasley mumpung lgi ult
   16296 Almost forgot that today is Ron Weasley'
```

<...>写在表达式外面

8，交互

```
 $ Hubble oldest star
 [optimize] hubble oldest star
 Find out about 5 results
   19231 Hubble telescope dates oldest star, 'Methuselah', at 14.
   19258 Hubble Finds Birth Certificate of Oldest Known Star http
   19388 NASA`s Hubble Telescope finds birth certificate of oldes
   19411 Hubble Finds Birth Certificate of Oldest Known Star http
   19715 ''Hubble Finds Birth Certificate of Oldest Known Star''
 Find out about 5 results

 $
```

在$输入表达式查询直到查询为空退出

实验结论

通过实验学习了文档倒排索引和布尔查询，使用 python 实现了一个支持复杂表达式布尔检索和词片段查询的信息检索系统，经测试对各种查询要求具有较高的查询效率，对用户输入有较好的鲁棒性，并进行了运算次序优化。

```
$ is{||((!ab ||askm) && < lm>) &&(s{ab && cc)
[optimize] is{ && ( s{ab && cc ) || ( ( ( ! ab ) || askm ) && lm )
Find out about 758 results
  21393 Download Sherlock Holmes: The Classic BBC Series Starring Dougla
  111 \u2018The King\u2019s Speech\u2019 is top film at producer awards:
  21729 Guys. Guys. GUYS. Sherlock began filming. I repeat: SHERLOCK BEG
  22800 @stephenfry love seeing you in sherlock Holmes game of shadows b
  24060 RT @brownjenjen: \u2192 http://t.co/v347LHa2fp #MrHolmes  Ian Mc
  24062 RT @brownjenjen: \u00bb http://t.co/v347LHa2fp Mr Holmes  #MrHol
```

实验代码

ex1

-agent.py （主程序）

-load_data.py

-boolop.py

-op_compile.py

load_data.py

```
import re
from collections import defaultdict
import pickle

def get_data(file='tweets2.txt'):
    text = []
    set_text = set()
    raw_text = []
    letter0 = defaultdict(set)
    with open(file, 'r') as f:
        for i, line in enumerate(f):
            ss = re.sub(' +', ' ', re.sub('[^a-zA-Z]', ' ',
line.split('"')[9])).lower().strip()
            if len(ss) < 1:
                pass
            if ss in set_text:
                pass
            set_text.add(ss)
            text.append(ss)
            raw_text.append(line.split('"')[9])
            for item in ss:
                letter0[item].add(i)

    letter = {}
    for key in letter0:
```

```
        letter[key] = sorted(list(letter0[key]))
    vocab = set()
    vocab_index = defaultdict(set)

    for i, line in enumerate(text):
        for item in line.split(' '):
            if i not in vocab_index[item]:
                vocab_index[item].add(i)
            if item not in vocab:
                vocab.add(item)
    vocab = list(vocab)
    for key in vocab_index:
        vocab_index[key] = sorted(list(vocab_index[key]))
    index0 = [[set() for _ in range(27)] for _ in range(27)]

    for i, item in enumerate(vocab):
        s = '{' + item + '{'
        for sp in range(len(s) - 1):
            if i not in index0[ord(s[sp]) - 97][ord(s[sp + 1]) - 97]:
                index0[ord(s[sp]) - 97][ord(s[sp + 1]) - 97].add(i)
    index = [[[] for _ in range(27)] for _ in range(27)]
    for i in range(27):
        for j in range(27):
            index[i][j] = sorted(list(index0[i][j]))
    return text, raw_text, vocab, vocab_index, index, letter

if __name__ == '__main__':
    text, raw_text, vocab, vocab_index, index, letter = get_data()
    print('data loaded')
    print(len(text), len(vocab))
```

boolop.py

```
def _and(seq1, seq2):
    flag1 = 0
    flag2 = 0
    seq = []
    while flag1 < len(seq1) and flag2 < len(seq2):
        if seq1[flag1] == seq2[flag2]:
            seq.append(seq1[flag1])
            flag1 += 1
            flag2 += 1
        elif seq1[flag1] < seq2[flag2]:
            flag1 += 1
```

```
        else:
            flag2 += 1
    return seq


def _or(seq1, seq2):
    flag1 = 0
    flag2 = 0
    seq = []
    while flag1 < len(seq1) and flag2 < len(seq2):
        if seq1[flag1] == seq2[flag2]:
            flag1 += 1
        elif seq1[flag1] < seq2[flag2]:
            seq.append(seq1[flag1])
            flag1 += 1
        else:
            seq.append(seq2[flag2])
            flag2 += 1
    if flag1 == len(seq1):
        seq += seq2[flag2:]
    else:
        seq += seq1[flag1:]
    return seq


def _not(seq1, max_len):
    seq = []
    seq1.append(max_len + 1)
    flag = 0
    for i in range(max_len):
        if seq1[flag] == i:
            flag += 1
            continue
        seq.append(i)
    return seq


def _xor(seq1, seq2):
    flag1 = 0
    flag2 = 0
    seq = []
    while flag1 < len(seq1) and flag2 < len(seq2):
        if seq1[flag1] == seq2[flag2]:
            flag1 += 1
```

```
            flag2 += 1
        elif seq1[flag1] < seq2[flag2]:
            seq.append(seq1[flag1])
            flag1 += 1
        else:
            seq.append(seq2[flag2])
            flag2 += 1
    if flag1 == len(seq1):
        seq += seq2[flag2:]
    else:
        seq += seq1[flag1:]
    return seq


if __name__ == '__main__':
    print(_and([1, 3, 4], [3, 4, 5, 6, 7, 8]))
    print(_or([1, 3, 4, 8], [3, 4, 5, 6, 7, 8]))
    print(_not([1, 3, 4, 8], 10))
    print(_xor([1, 3, 4, 8], [3, 4, 5, 6, 7, 8]))
```

op_compile.py

```
import re
from ex1.boolop import _and, _not, _or, _xor


def ana(seq1, seq2, op, max_len=27799):
    if op == '&&':
        return _and(seq1, seq2)
    if op == '||':
        return _or(seq1, seq2)
    if op == '!':
        return _not(seq2, max_len)
    if op == '^':
        return _xor(seq1, seq2)
    return seq1


def cp(exp, source, max_len=27799):
    if not isinstance(exp, list):
        exp = reg(exp)
    i = 0
    cur = [s for s in range(max_len)]
    op = '&&'
```

```
    while i < len(exp):
        if exp[i] == '(':
            j = i + 1
            m = 0
            while m >= 0:
                if exp[j] == '(':
                    m += 1
                if exp[j] == ')':
                    m -= 1
                j += 1
            cur = ana(cur, cp(exp[i + 1:j - 1], source), op, max_len)
            i = j - 1
        if exp[i] in source:
            cur = ana(cur, source[exp[i]], op, max_len)
        else:
            op = exp[i]
        i += 1
    return cur


def reg(seq):
    seg = []
    for item in seq.split(' '):
        if item != '':
            seg.append(item)
    return seg


if __name__ == '__main__':
    print(cp('a && ( b || ( ! c ) )', {'a': [1, 2, 3], 'b': [3, 4, 5], 'c':
[2, 3, 4]}))
```

agent. py

```
import ex1.load_data as data
import ex1.op_compile as op
from ex1.boolop import _and, _or

OPERATOR = {'(', ')', '&&', '||', '!', '^', '<', '>'}
RANK = {'&&': 1, '||': 2, '!': 0, '^': 1, '_': -1}


def org(seq):
    cur = [i for i in range(len(vocab))]
```

```
    if len(seq) == 1:
        return letter.get(seq, [])
    for sp in range(len(seq) - 1):
        cur = _and(cur, index[ord(seq[sp]) - 97][ord(seq[sp + 1]) - 97])
    text_list = []
    for item in cur:
        text_list = _or(text_list, vocab_index[vocab[item]])
    return text_list


def optimize(exp):
    i = 0
    cur = []
    operator = '_'
    while i < len(exp):
        if exp[i] == '(':
            j = i + 1
            m = 0
            while m >= 0:
                if exp[j] == '(':
                    m += 1
                if exp[j] == ')':
                    m -= 1
                if j == len(exp):
                    break
                j += 1
            cur.append((operator, '( ' + optimize(exp[i + 1:j - 1]) + ')'))
            i = j - 1
        if exp[i] not in OPERATOR:
            cur.append((operator, exp[i]))
        if exp[i] == '!':
            j = i
            m = 0
            f = True
            while m > 0 or f:
                j += 1
                f = True
                if j == len(exp):
                    break
                if exp[j] == '(':
                    m += 1
                if exp[j] == ')':
                    m -= 1
                if exp[j] in OPERATOR and exp[j] != ')':
```

```
                f = False
            cur.append((operator, '( ! ' + optimize(exp[i + 1:j]) + ')'))
            i = j - 1
        if exp[i] in OPERATOR:
            operator = exp[i]
        i += 1
    cur.sort(key=lambda x: RANK[x[0]])
    res = ''
    for seq in cur:
        if seq[0] == '_':
            res += seq[1] + ' '
        else:
            res += seq[0] + ' ' + seq[1] + ' '
    return res

def add_space(seq):
    for item in OPERATOR:
        seq = seq.replace(item, ' '+item+' ')
    seq = seq.lower().strip()
    return seq

def match(exp):
    regular = None
    exp = add_space(exp)
    exp = op.reg(exp)
    if '<' in exp:
        seg = exp.index('<')
        regular = exp[seg + 1]
        if seg == 0 or exp[seg - 1] in OPERATOR:
            exp.pop(seg)
            exp.pop(seg + 1)
        else:
            exp = exp[:seg]
    exp = optimize(exp)
    print('[optimize]', exp)
    exp = op.reg(exp)
    source = {}
    for item in exp:
        if item not in OPERATOR:
            source[item] = org(item)
    res = op.cp(exp, source, max_len=len(text))
    if regular is not None:
        res.sort(key=lambda x: text[x].count(regular), reverse=True)
    return res
```

```
if __name__ == '__main__':
    text, raw_text, vocab, vocab_index, index, letter =
data.get_data(file='tweets2.txt')

    m1 = 'ron && weasley && birthday'
    m0 = 'is{||((!ab ||askm) && < lm>) &&(s{ab && cc)'
    while True:
        s = input('\n$ ')
        if s == '':
            break
        s = match(s)
        print('Find out about {} results'.format(len(s)))
        for tex_id in s:
            print(' ', tex_id, raw_text[tex_id])
        print('Find out about {} results'.format(len(s)))
```