华东师范大学数据学院上机实践报告

课程名称：信息检索　　年级：2018　　上机实践成绩:
指导教师：张蓉　　姓名：孙秋实
上机实践名称：支持统配查询处理的检索系统　　学号：10185501402　　上机实践日期：2021/11/15

Part 1
实验目的

(1) 为文档构建倒排索引

(2) 实现支持统配查询处理的检索系统

Part 2
实验任务

(1) 在单机伪分布式下使用 Spark 为文档数据集构建倒排索引

(2) 建立轮排索引

(3) 构建支持统配查询处理的检索系统

(4) 对结果进行分析和展示

Part 3
使用环境

(1) Google Colab

(2) Python 3.7

(3) Spark 3.2.0

(4) Scala 2.12

(5) Idea 2020.02.03

Part 4
实验过程

Section 1
轮排索引与通配查询

在本次实验中涉及轮排索引概念，使用 $ 是一个特殊字符，来标记词的结尾，随后为单词建立轮排表。如图 1 所示，为单词 hello 建立轮排。

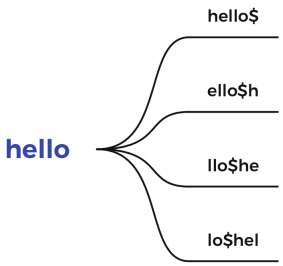


图 1: Permuterm indexes

以下代码片段是轮排的简单实现

```
# simple demo of permuterm
word = 'hellworld'
word += '$'
for i in range(len(word)):
    print('permute:',i,word)
    word = word[-1]+word[:-1]
```

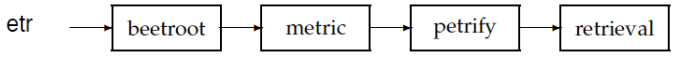
输出为

```
permute: 0 hellworld$
permute: 1 $hellworld
permute: 2 d$hellworl
permute: 3 ld$hellwor
permute: 4 rld$hellwo
permute: 5 orld$hellw
...
```

在之前的实验中实现了布尔查询，本次实现的通配查询在输入的形式上更加多样，可以支持以下种类的查询：

- *aaa，如 *ent: 代表所有包含以 ent 结尾的词项的文档
- bbb*，如 mon*: 代表所有包含以 mon 开头的词项的文档
- *ccc*，如 *hel*: 代表所有包含中间部分有 hel 的词项的文档
- dd*ee，如 map*duce: 代表所有包含以 map 开头，duce 结尾的词项的文档

$*$ 是一个特殊字符，代表着查询时从这里“切分”



► **Figure 3.4** Example of a postings list in a 3-gram index. Here the 3-gram etr is illustrated. Matching vocabulary terms are lexicographically ordered in the postings.

图 2: Index-Vocab Matching

如图 2 所示，我们需要对切分好的片段进行匹配。

Section 2
倒排索引的 Spark 实现

本次实验使用与 Project2 中一样的方式倒排索引，同样是生成 dict.index.txt，因为之前的实验已经涉及，所以此处简要略过，具体内容可见 Project2 的倒排索引实现。

**Spark3.2.0 + Scala2.12*

```
object ReadGetIndex {
//  val writer = new PrintWriter(new File("spark_index_test.txt" ))
  def main(args: Array[String]): Unit = {
    //get Spark Session
    val spark = SparkSession.builder().appName("Information-Retrieval-Project2")
      .master("local")
      .getOrCreate()
    //Get Dir
    val document_source = spark.sparkContext.wholeTextFiles("src/data-2500")
    //Get File Name
    val res = document_source.flatMap{ x =>
      val doc = x._1
        .split(s"/")
        .last // get file name.txt
        .split("\\.")
        .head // get file name
      x._2.split("\r\n") //seperate -> line -> blank
        .flatMap(_.split(" ").map { y => (y, doc)})}
        .groupByKey.map{case(x,y)=>(x,y.toSet.mkString(","))} // mkstring即用指定字符串分隔
    // pattern matching
    // 此处需要注意partition!
    res.coalesce(1).saveAsTextFile("spark_index_test.txt")
    res.foreach(println)
  }
}
```

Section 3
B-Tree

本次实验使用 Python 的 BTrees 库

```
from BTrees.OOBTree import OOBTree
BTree_index = OOBTree() # 初始化一个BTree对象
```

随后将轮排结果插入 BTree，获得轮排的方式与前面的 Demo 一致

```
def Permuterm_Insert(BTree_index,key,word):
    word += '$'
    for i in range(len(word)):
        BTree_index.update({word:key})
```

```
        # 插入Key-Value Pair
        word = word[-1]+word[:-1]
```

Section 4
通配搜索系统的构建

在搜索时候，需要将输入的查询分为两种

- *aaa，如 *ent: 代表所有包含以 ent 结尾的词项的文档
- bbb*，如 mon*: 代表所有包含以 mon 开头的词项的文档
- *ccc*，如 *hel*: 代表所有包含中间部分有 hel 的词项的文档

它们在去除匹配符后只有一个 token

- dd*ee，如 map*duce: 代表所有包含以 map 开头，duce 结尾的词项的文档

这种情况下，去除匹配符后会有两个 tokens
分类讨论进行查询，代码如下

```
def Permuterm_Search(BTree_index,raw_word):
    ans = []
    # insert$
    raw_word += '$'
    word_sep = raw_word.split('*')
    if len(word_sep) == 1:
        ans_idx = Word_Search(BTree_index,word_sep[0])
    else:
        target = word_sep[1] + word_sep[0]
        ans_idx = Word_Search(BTree_index, target)

    ans_idx = sorted(ans_idx,key = lambda x: x[1]) # 排序
    ans = []
    # get top 2 data
    for i in ans_idx[:2]:
        ans = SearchOR(ans, dic[i[0]])
    ansWord = [x[1] for x in ans_idx]
    return ans,ansWord
```

最后使用一个 While 循环启动查询（配备适当的退出条件），根据输入完成不同类型的查询需求
根据实验手册的要求，输出

- 按照字典序输出的所有词项
- 以字典序前 2 个词项作输入，查询后获得的 DocID 并集

```
while (True):
    query = input('输入通配查询').split()
    print('Query',query)
    Doc_id_ans, Word_ans = Permuterm_Search(BTree_index, query[0])
```

```
if len(query) > 1 and query[0] != 'exit':
    print(Word_ans[:int(query[1])])
elif query[0] == 'exit':
    print('The end')
    break
else:
    print(Word_ans)
print('DocID:',Doc_id_ans)
print('==================')
```

Section 5
查询演示

在此部分做查询的演示

- 实现单个关键词查询

```
输入通配查询 *sment
Query ['*sment']
['assessment', 'embarrassment', 'harassment', 'office_of_technology_assessment', 'radiological-assessment', 'reassessment', 'self-ass
essment']
DocID: [1077, 1141, 1144, 1166, 1246, 1294, 1314, 1477, 1498, 1593, 1629, 1637, 1702, 1767, 1770, 1792, 1871, 1920, 1940, 2102, 2124,
2218, 2255, 2263, 2303, 2359, 4414, 8477, 10221, 10869, 13966, 17498, 17901, 18425, 18983, 19759, 20193, 20276, 21062, 21576, 22351,
22703, 22850, 23516, 23538, 23610, 23922, 24395, 24837, 24999, 25003, 25614, 25671, 25683, 25705, 26369, 26519, 27429, 27430, 27549,
27648, 28591, 28931, 31024, 31031, 32373, 32513, 32668, 33161, 34353, 36063, 36064, 36126, 36429, 37897, 37909, 38251, 39051, 39703,
40003, 40077, 40168, 40275, 40310, 40517, 41804, 42304, 42378, 42604, 43799, 46769, 47509, 50354, 52230, 53403, 53742, 54079, 55261,
55715, 55847, 56409, 57319, 57328, 57333, 58347, 60504, 61548, 61587, 61651, 61661, 61669, 61672, 61824, 64341, 64342, 65233, 65316,
65468, 65670, 65718, 65839, 65842, 67912, 67942, 68196, 68203, 68261, 68262, 68268, 68359, 68360, 68451, 68470, 68513, 68516, 68648,
68732, 68869, 71625, 71700, 74213, 74249, 74544, 74723, 74835, 75628, 76113, 77715, 79512, 80272, 81248, 81292, 81471, 81700, 82027,
82049, 82643, 82775, 82959, 83049, 83432, 84164, 86264, 88372, 88815, 90889, 91753, 92406, 92603, 92610, 92921, 92927, 93875, 93987,
93988, 94008, 95664, 96143, 96499, 96993, 97982]
==================
```

图 3: 单侧匹配查询（左侧）

可以使用连续查询

```
输入通配查询 *sment
Query ['*sment']
['assessment', 'embarrassment', 'harassment', 'office_of_technology_assessment', 'radiological-assessment', 'reassessment', 'self-ass
essment']
DocID: [1077, 1141, 1144, 1166, 1246, 1294, 1314, 1477, 1498, 1593, 1629, 1637, 1702, 1767, 1770, 1792, 1871, 1920, 1940, 2102, 2124,
2218, 2255, 2263, 2303, 2359, 4414, 8477, 10221, 10869, 13966, 17498, 17901, 18425, 18983, 19759, 20193, 20276, 21062, 21576, 22351,
22703, 22850, 23516, 23538, 23610, 23922, 24395, 24837, 24999, 25003, 25614, 25671, 25683, 25705, 26369, 26519, 27429, 27430, 27549,
27648, 28591, 28931, 31024, 31031, 32373, 32513, 32668, 33161, 34353, 36063, 36064, 36126, 36429, 37897, 37909, 38251, 39051, 39703,
40003, 40077, 40168, 40275, 40310, 40517, 41804, 42304, 42378, 42604, 43799, 46769, 47509, 50354, 52230, 53403, 53742, 54079, 55261,
55715, 55847, 56409, 57319, 57328, 57333, 58347, 60504, 61548, 61587, 61651, 61661, 61669, 61672, 61824, 64341, 64342, 65233, 65316,
65468, 65670, 65718, 65839, 65842, 67912, 67942, 68196, 68203, 68261, 68262, 68268, 68359, 68360, 68451, 68470, 68513, 68516, 68648,
68732, 68869, 71625, 71700, 74213, 74249, 74544, 74723, 74835, 75628, 76113, 77715, 79512, 80272, 81248, 81292, 81471, 81700, 82027,
82049, 82643, 82775, 82959, 83049, 83432, 84164, 86264, 88372, 88815, 90889, 91753, 92406, 92603, 92610, 92921, 92927, 93875, 93987,
93988, 94008, 95664, 96143, 96499, 96993, 97982]
==================
输入通配查询 ende*
Query ['ende*']
['endear', 'endeared', 'endearing', 'endearingly', 'endearment', 'endeavor', 'endeavors', 'ended', 'endemic', 'ender', 'enderle', 'en
ders', 'endesa']
DocID: [15885, 24634, 26089, 35837, 49260, 56858, 56871, 70866, 70879, 88850, 90895, 90903]
==================
```

图 4: 单侧匹配查询（右侧）

```
输入通配查询 p*nn
Query ['p*nn']
['paarmann', 'pahlmann', 'pat_quinn', 'patrick_quinn', 'penn', 'peter_hermann', 'peter_neumann', 'peter_r._kann', 'pierre_closterman
n', 'pohlmann']
DocID: [2432, 2435, 33479, 39664, 69260, 75058]
==================
输入通配查询 bn*d
Query ['bn*d']
['bnd']
DocID: [1324, 1949, 65420, 68608, 94117]
==================
```

图 5: 中间匹配查询

最后是对匹配单词的中间字段进行查询

```
输入通配查询 *sment*
Query ['*sment*']
['assessment', 'assessments', 'embarrassment', 'embarrassments', 'harassment', 'office_of_technology_assessment', 'radiological-asses
sment', 'reassessment', 'self-assessment']
DocID: [1077, 1141, 1144, 1166, 1246, 1294, 1314, 1477, 1498, 1593, 1601, 1629, 1637, 1639, 1702, 1767, 1770, 1792, 1871, 1920, 1940,
2102, 2124, 2218, 2227, 2255, 2263, 2265, 2303, 2359, 4414, 8477, 9159, 9531, 9904, 10764, 10776, 10800, 10812, 11936, 13341, 13903,
13966, 17498, 17901, 18425, 18983, 19759, 20193, 21576, 22703, 22850, 23538, 23922, 24395, 24999, 25003, 25614, 25639, 25664, 25671,
25683, 25705, 26369, 26519, 26644, 27429, 27648, 28591, 28931, 29120, 30994, 31024, 32373, 32668, 34353, 36063, 36064, 36200, 36276,
36429, 36903, 37260, 37897, 37909, 39051, 39656, 39703, 40003, 40077, 40275, 40485, 40517, 41804, 42304, 42378, 42604, 46769, 47509,
48893, 50354, 50633, 51502, 52230, 53403, 54079, 54117, 55261, 56409, 57319, 57325, 57328, 57333, 57432, 57894, 59850, 60504, 61101,
61548, 61552, 61564, 61587, 61651, 61661, 61669, 61672, 61824, 62077, 64341, 64342, 65233, 65316, 65468, 65474, 65477, 65670, 65718,
65740, 65839, 65842, 65846, 65888, 67912, 67942, 68050, 68196, 68203, 68261, 68262, 68268, 68359, 68360, 68451, 68470, 68513, 68516,
68545, 68648, 68732, 68869, 71625, 71700, 74213, 74249, 74544, 74723, 74835, 75628, 76109, 76113, 79512, 80272, 81248, 81292, 81318,
81471, 81700, 82027, 82029, 82049, 82065, 82643, 82775, 82959, 83047, 83049, 84164, 84912, 84917, 84951, 86264, 88815, 90889, 91753,
92603, 92630, 92653, 93692, 94008, 94530, 94536, 95664, 96143, 96298, 96315, 96499, 99379]
==================
```

图 6: 两侧匹配查询

以上为查询系统的四种不同类型查询的结果。