《现代信息检索》大作业实验报告

小组成员信息

组长：王猛　　2018E8013261020　计算技术研究所

组员：梁付槐　2018Z8013261003　计算技术研究所

组员：王丽颖　20182801322 9034　计算技术研究所

组员：马聪　　2018E8013261047　计算技术研究所

组员：郑好　　2018E8013261020　计算技术研究所

Email：wangmeng182@mails.ucas.ac.cn

手机号：18311016966

目录

一、实验目的

本课程大作业要求：在 TREC Precision Medicine (PM)2017 数据上进行检索竞赛。TREC 的 PM 评测任务就是为解决临床中的现实需求、促进医疗文献文本检索技术的发展与交流而设立。PM 评测任务致力于解决病人信息匹配，相关文档检索问题，主要有两个子任务：科学文献子任务和临床试验子任务。Scientific Abstract 是医疗文献的摘要部分，目标是为医生提供学术研究上相关的治疗信息。Clinical trials 是病人病历数据库，目标是为医生提供与此病是病人病历数据库，目标是为医生提供与此病人相关的电子病历。大作业要求完成第二个任务，即 Clinical trials。

二、实验环境

开发环境： Ubuntu 16.0.4、Java JRE1.8.0、Python 3.0

开源工具： Terrier 5.0

三、实验过程

1. Terrier 安装

在进行具体的实验操作之前，本小组调研了多个信息检索的开源工具，包括 Lucene 和 Elasticsearch。相比之下，Terrier 更适合用于完成本次任务，其专门为 TREC 比赛定制了接口，而且 Lucene 的评分函数比较简单，适合于工程开发，而 Terrier 对语言模型做了很多优化，更适合科研类任务。根据以往经验，用 terrier 实现要比用 Lucene 实现效果普遍更好。

（1） Terrier 下载：本实验中使用的是 Terrier 5.0 的版本，需要 Java JRE1.8.0 或更高版本。可以使用 echo $JAVA_HOME 命令，查看已安装的 Java 版本。如果不满足要求，则需要重新安装，具体的安装过程不在实验过程中说明。可以从官网

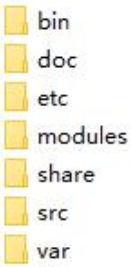
上或者 Github 下载 terrier-project-5.0-bin.tar.gz。

（2） Terrier 安装：下载的版本已经编译过，可以直接解压，如下图所示。

```
mkdir terrier

tar -zxvf terrier-project-5.0-bin.tar.gz
```

（3） 文件说明：



bin: 存放运行脚本，在 linux 系统下用.sh，在 windows 下用相应的.bat

doc：帮助说明文档集

etc: 存放建立索引和检索时的配置文件，如 collection.spec, terrier.properties

var: 检索结果的存放位置，存放 index， result，可以通过修改 etc 文件夹下的 terrier.properties 配置文件，定位到其他文件夹

share：存放 stopword-list.txt 等文件

src: 搜索引擎源代码

2. 数据预处理

在利用 Terrier 进行检索实验之前，首先对给定数据集进行预处理，既是为了满足 Terrier 的数据格式要求，也是为了提升查询检索的效果。

原始数据集是从 Trec 官网上下载的 xml 类型的数据，每一个数据文件中的标签都很多，包含很多不必要的信息，噪声较大，其存在无法提升检索效果。可以只提取部分信息，既可以压缩存储空间，又可以提升检索效果。下图是处理之后的数据文件，<DOC>和<DOCNO>

是为了满足 Terrier 的数据格式要求，这里也可以不做预处理，但是需要修改配置文件，这会在第四步中说明。提取的标签主要有索引要求中涉及的，例如 brief_title、official_title、brief_summary、detailed_description、MeSH Terms 和 criteria 等，以及查询文件中涉及的标签，例如 minimum_age、maximum_age、gender。

```
<DOC>
<DOCNO>NCT00000102</DOCNO>
  <verification_date>January 2004</verification_date>
  <minimum_age>14 Years</minimum_age>
  <maximum_age>35 Years</maximum_age>
  <gender>All</gender>
  <mesh_term>Hyperplasia</mesh_term>
  <mesh_term>Adrenal Hyperplasia, Congenital</mesh_term>
  <mesh_term>Adrenogenital Syndrome</mesh_term>
  <mesh_term>Adrenocortical Hyperfunction</mesh_term>
  <mesh_term>Nifedipine</mesh_term>
  <brief_title>Congenital. . . .Targets</brief_title>
  <brief_summary>
      This study will. . . . . .
    </brief_summary>
  <description>
      This protocol is . . . . . .
    </description>
  <criteria>
       Inclusion Criteria:. . . . . .
      </criteria>
</DOC>
```

评测的查询输入是半结构化文本，描述了患者的癌症类型、相关的基因变异、年龄性别以及其他可能相关的因素，文件格式如下。需要说明的是，原始查询文件的根标签是 topic 和 number，改成 TOP 和 NUM 之后，便于之后处理。

```
<TOP>
    <NUM>1</NUM>
    <disease>Liposarcoma</disease>
    <gene>CDK4 Amplification</gene>
    <demographic>38-year-old male</demographic>
    <other>GERD</other>
</TOP>
```

3. 收集数据文档

收集须要建索引的文档，设置 data 的路径，将会在 etc 目录下创建一个 collection.spec 文件，包含了刚刚选定的语料库目录下的文件列表。执行命令如下图。

```
#收集文档
echo 收集文档
./bin/trec_setup.sh ../clinicaltrials_xml/
```

4. 修改配置文件

配置文件中的部分参数需要调整，配置文件如下图所示。由于在预处理过程中已经处理了标签问题，而且，不做特殊说明的话，系统默认对文件中的所有域建立索引，所以此处不修改也可以。如果之前没有修改标签，此处需要改成第二张图的格式。

```
TrecDocTags.doctag=DOC
TrecDocTags.idtag=DOCNO
TrecDocTags.skip=DOCHDR
#set to true if the tags can be of various case
TrecDocTags.casesensitive=false

#query tags specification
TrecQueryTags.doctag=TOP
TrecQueryTags.idtag=NUM
TrecQueryTags.process=TOP,NUM,TITLE
TrecQueryTags.skip=DESC,NARR
```

```
xml.doctag=clinical_study
xml.idtag=nct_id
xml.terms=textblock,description,

#set to true if the tags can be of various case
TrecDocTags.casesensitive=false

#query tags specification
TrecQueryTags.doctag=topic
TrecQueryTags.idtag=number
TrecQueryTags.process=topic,disease,gene,demographic
#TrecQueryTags.skip=DESC,NARR
```

5. 建立索引

以此命令*./bin/trec_terrier.sh -i* 建立索引。如果不想保留直接索引结构，可以加上参数*-j*，用更快的单通道索引，但会导致无法进行查询扩展。

6. 查询检索

检索命令含有多个参数，可以在 *etc/terrier.properties* 文件里预先设好，也可以一个

一个在命令行里指定。–Dtrec.topics 指定查询的位置，也就是 topic 文件路径，-Dtrec.model 指定要使用的加权模型，-r 参数指示 Terrier 做检索，-c 指定 Terrier 加权模型的参数。下图以 BM25 为例，正常情况下将在/var/results 的位置产生.res 文件。

```
#检索 BM25模型
./bin/trec_terrier.sh -r -Dtrec.model=BM25 -c 0.75 -Dtrec.topics=../topics2017.xml
```

res 文件格式如下图：

```
1   1 Q0 NCT01209598 0 42.88400158550261 BM25
2   1 Q0 NCT02187783 1 40.82808803504023 BM25
3   1 Q0 NCT02571829 2 35.83861538814905 BM25
4   1 Q0 NCT03096912 3 34.20675190002517 BM25
5   1 Q0 NCT03024489 4 29.176565838904374 BM25
```

7. 相关反馈

加入伪相关反馈，其命令如下图。相较于上一步的检索命令，增加了 -q 和 -Dqe.feedback.filename 参数，用于指定用哪一个模型的检索结果作为反馈的输入。下图中是以 In_expB2 模型的结果作为反馈输入。

```
#BM25 + In_expB2模型
./bin/trec_terrier.sh -r -Dtrec.model=BM25 -c 0.5 -Dtrec.topics=../topics2017.xml
-q -Dqe.feedback.filename=./var/results/In_expB2_2.res
```

本小组利用 Terrier 开源工具中已经实现的 Bo1、Bo2、KL 等伪相关反馈模型进行实验。可以在配置文件 terrier.properties 中添加相关的配置项。trec.qe.model 表示使用的伪相关反馈模型，expansion.terms 表示从前 k 篇文档中提取的重要词项的个数，即前文中的 n 值，expansion.documents 表示使用返回结果中的前几个文档来提供伪相关反馈信息，qe.feedback.filename 是前一次查询返回的结果文件所在路径。另一种方法是在没一行命令下添加上述参数。

8. 结果评估

利用给定数据 qrels-final-trials.txt，对检索结果进行评估，命令如下图所示。

```
#检验结果
echo query结果评估
./bin/trec_eval.sh ../qrels-final-trials.txt ./var/results/BM25_0.res  >> ../eval_res/BM25.eval
./bin/trec_eval.sh ../qrels-final-trials.txt ./var/results/BB2_1.res  >> ../eval_res/BB2.eval
```

该命令执行后会生成一个模型对应的 eval 文件。该文件中包含了 MAP、Bpref、Recall、P_5、P_10 等度量指标值，下图是 BB2.eval 文件内容。

```
Setting TERRIER_HOME to /usr/local/terrier-project-5.0
runid                   all BB2
num_q                   all 29
num_ret                 all 29000
num_rel                 all 1171
num_rel_ret             all 900
map                     all 0.2429
gm_map                  all 0.1524
Rprec                   all 0.2901
bpref                   all 0.2798
recip_rank              all 0.5807
iprec_at_recall_0.00    all 0.6357
iprec_at_recall_0.10    all 0.5343
iprec_at_recall_0.20    all 0.4652
iprec_at_recall_0.30    all 0.3484
iprec_at_recall_0.40    all 0.2840
iprec_at_recall_0.50    all 0.2317
iprec_at_recall_0.60    all 0.1713
iprec_at_recall_0.70    all 0.1322
iprec_at_recall_0.80    all 0.0765
iprec_at_recall_0.90    all 0.0499
iprec_at_recall_1.00    all 0.0164
P_5                     all 0.4276
P_10                    all 0.3483
P_15                    all 0.3034
P_20                    all 0.2793
P_30                    all 0.2506
P_100                   all 0.1528
P_200                   all 0.1045
P_500                   all 0.0546
P_1000                  all 0.0310
```

9. 界面交互

执行./bin/http_terrier.sh 命令，启动 HttpServer 服务，这样就可以基于 web 的查询界面与检索系统交互。

```
root@iZm5e296kyz53dtw98j42jZ:~/terrier/terrier-project-5.0# ./bin/http_terrier.sh
Setting TERRIER_HOME to /root/terrier/terrier-project-5.0
14:17:06.395 [main] INFO  org.eclipse.jetty.util.log - Logging initialized @578ms
14:17:06.566 [main] INFO  org.eclipse.jetty.server.Server - jetty-9.2.z-SNAPSHOT
14:17:06.723 [main] INFO  o.e.j.server.handler.ContextHandler - Started o.e.j.w.WebAppContext@6121c9d6{/,file:/root/terrier/terrie
BLE}
14:17:06.725 [main] INFO  o.e.j.server.handler.ContextHandler - Started o.e.j.s.h.ContextHandler@87f383f{/images,null,AVAILABLE}
14:17:06.739 [main] INFO  o.e.jetty.server.ServerConnector - Started ServerConnector@78e118d2{HTTP/1.1}{0.0.0.0:8080}
14:17:06.739 [main] INFO  org.eclipse.jetty.server.Server - Started @925ms
```

在浏览器输入 http://47.10.152.193:8080，就可以进入查询界面。

四、实验结果

1. 模型选择

Terrier 开源工具中集成了许多信息检索模型，可以直接进行接口调用。实验过程中，本小组选择了 10 中模型进行测试，下表展示了不同模型的测试指标值，MAP、P_5 和 P_10 均是通过 5 折交叉验证并在测试集上求平均得到。在没有添加相关反馈的情况下，MAP 的最优结果为 0.2620，由 In_expB2 模型得到；P@5 的最优结果为 0.4276，由 BB2 模型得到；P@10 的最优结果为 0.3724，由 BM25 模型得到。经过实验分析，发现各个模型参数的变动对最终结果几乎没有影响，所以在实验中使用了默认参数。

Model	MAP	P@5	P@10
In_expB2	**0.2620**	0.4000	0.3621
BB2	0.2429	**0.4276**	0.3483
BM25	0.2582	0.3862	**0.3724**
IFB2	0.2610	0.3724	0.3552
TF-IDF	0.2130	0.3310	0.3276
DPH	0.1715	0.3172	0.3034
PL2	0.1992	0.3586	0.3241
DFIC	0.1505	0.3241	0.2621
DLH13	0.1529	0.3310	0.2828
DLH	0.1526	0.2966	0.2586

2. 相关反馈

通过 5 折交叉验证，选出 4 个结果较好的模型添加相关反馈进行下一步测试，分别是 In_expB2、BB2、BM25 和 IFB2，也使用了一些初始结果较差的模型进行相关反馈实验。由于组合较多，下表只展示部分结果。

Model	MAP	P@5	P@10
BM_25+In_expB2	0.2865	0.3862	0.3828
IFB2+BM25	0.2859	0.4000	0.3621
In_expB2+BM25	0.2634	0.3655	0.3414
BB2+BM25	0.2703	0.4207	0.3414
DLH13+In_expB2	0.2166	0.3448	0.2966

在使用伪相关反馈的情况下，MAP 的最优结果为 0.2865，由 BM25+In_expB2 模型得到，相较于未添加相关反馈的结果提升了近 10%；P@5 的最优结果为 0.4207，由 BB2+ BM25 模型得到，相较于未添加相关反馈的结果没有提升；P@10 的最优结果为 0.3828，由 BM25+In_expB2 模型得到，相较于未添加相关反馈的结果提升了近 6%；。经过实验分析，发现利用 BM25 的查询结果作为反馈，再利用 In_expB2 模型进行查询检索，得出的效果最优。

五、实验基本原理

1. 评价指标

（1） 召回率(Recall): RR/(RR + NR)，返回的相关结果数占实际相关结果总数的比率，也称为查全率，R∈ [0,1]。

（2） 正确率(Precision): RR/(RR + RN)，返回的结果中真正相关结果的比率，也称为查准率， P∈ [0,1]。

（3） Bpref：基本的思想：在相关性判断(Relevance Judgment) 不完全的情况下，计

算在进行了相关性判断的文档集合中，在判断到相关文档前，需要判断的不相关文档的篇数。

（4） NDCG：每个文档不仅仅只有相关和不相关两种情况，而是有相关度级别，比如0,1,2,3。我们可以假设，对于返回结果，相关度级别越高的结果越多越好，相关度级别越高的结果越靠前越好。

（5） MAP：平均正确率(Average Precision, AP)：对不同召回率点上的正确率进行平均。

（6） P@K：计算前 K 个位置的平均正确率。

2. 模型介绍

（1） PL2（DFR）：随机性的泊松估计，第一归一化的拉普拉斯连续，以及术语频率归一化的归一化。

（2） BM25：BM25 模型是一种经典的概率检索模型，由于它的高效性而被广泛应用在信息检索领域。给定一篇文档 d 和一个查询 Q，排序函数：

$$\text{score}(d,Q) = \sum_{t \in Q} \log_2 \frac{N - df_t + 0.5}{df_t + 0.5} \cdot \frac{(k_1 + 1)tf}{k_1\left((1-b) + b \cdot \frac{l}{avg\_l}\right) + tf} \cdot \frac{(k_3 + 1)qtf}{k_3 + qtf}$$

其中 t 表示查询中的词项；qtf 为词项 t 在查询 Q 中的词频；tf 为词项 t 在文档 d 中的词频；df_t 表示词项 t 的文档频率；l 和 avg_l 分别表示文档 d 的长度和语料库中文档的平均长度；N 表示语料库中的文档总数目；k1，k3 和 b 是可调参数，此处我们使用其默认值 k1=1.2，k3=1000，b=0.75。

（3） BB2（DFR）：随机性的 Bose-Einstein 模型，第一次归一化的两个伯努利过程的比率，以及术语频率归一化的归一化。

（4） DPH（DFR）：使用 Popper 归一化（无参数）的不同超几何 DFR 模型。

（5） LGD（DFR）：对数逻辑 DFR 模型。

（6） IFB2（DFR）：随机性的反向项频率模型，第一次归一化的两个伯努利过程的比率，以及术语频率归一化的归一化。

（7） DLH13（DFR）：DLH 的改进版本（无参数）。

（8） InL2（DFR）：随机性的逆文档频率模型，第一次归一化的拉普拉斯序列，以及术语频率归一化的归一化 2。

六、程序运行

1. 运行方法

实验过程中，本小组开发了“一键执行”的脚本 runTerrier.sh，可以直接从原始文件得到结果文件，用于结果评估检测。解压源代码压缩包，进入解压目录，执行如下命令。

```
bash ./runTerrier.sh
```

2. 文件说明

解压的文件目录如下图所示，clinicaltrials_xml 是经过预处理的数据文件，qrels-final-trials.txt 是用于结果评估的文件，topics2017.xml 也是经过预处理的查询文件，terrier-project-5.0 是开源工具的安装目录，finally.res 是最终的检索结果，可以用于结果评估检查，finally.eval 是最终评估结果。

```
drwxr-xr-x  2 root root 8265728 Dec 26 18:32 clinicaltrials_xml/
-rw-r--r--  1 root root    4012 Dec 30 15:21 dataprocess.py
-rw-r--r--  1 root root    1075 Dec 30 15:05 finally.eval
-rw-r--r--  1 root root 1615500 Dec 30 15:13 finally.res
-rw-r--r--  1 root root  256740 Dec 27 15:02 qrels-final-trials.txt
-rw-r--r--  1 root root    5261 Dec 29 12:59 runTerrier.sh
drwxr-xr-x  9 root root    4096 Dec 28 01:45 terrier-project-5.0/
-rw-r--r--  1 root root    5096 Dec 27 15:04 topics2017.xml
```

data_process.py 用于对数据进行预处理，由于处理时间较长，在一键运行脚本中没有添加该命令，每次运行都是使用的已经预处理过的文件。runTerrier.sh 文件内容如下图所示。

```
#!/bin/bash

#删除之前生成的文件
echo 删除之前生成的文件
rm -rf ./terrier-project-5.0/var/results ./terrier-project-5.0/var/index/data* ./finally*
cd ./terrier-project-5.0/

#收集文档
echo 收集文档
./bin/trec_setup.sh ../clinicaltrials_xml/

#建立索引
echo 建立索引
./bin/trec_terrier.sh -i

#检索
echo 搜索query
#In_expB2模型
./bin/trec_terrier.sh -r -Dtrec.model=In_expB2 -Dtrec.topics=../topics2017.xml

#检验结果
#echo query结果评估
#./bin/trec_eval.sh ../qrels-final-trials.txt ./var/results/In_expB2_0.res  >> ../eval_res/In_expB2.eval

#相关反馈
echo 相关反馈
#BM25 + In_expB2模型
./bin/trec_terrier.sh -r -Dtrec.model=BM25 -c 0.5 -Dtrec.topics=../topics2017.xml -q -Dqe.feedback.filename=./var/results/In_expB2_0.res

#检验结果
echo 相关反馈结果评估
./bin/trec_eval.sh ../qrels-final-trials.txt ./var/results/BM25_d_3_t_10_1.res  >> ../finally.eval

cp ./var/results/BM25_d_3_t_10_1.res ./finally.res
```

finally.eval 就是最终的评估结果文件，文件内容如下图所示。

```
Setting TERRIER_HOME to /root/terrier/terrier-project-5.0
runid                   all     BM25_d_3_t_10
num_q                   all     29
num_ret                 all     29000
num_rel                 all     1171
num_rel_ret             all     999
map                     all     0.2865
gm_map                  all     0.1585
Rprec                   all     0.3138
bpref                   all     0.3038
recip_rank              all     0.6368
iprec_at_recall_0.00    all     0.6763
iprec_at_recall_0.10    all     0.5412
iprec_at_recall_0.20    all     0.4975
iprec_at_recall_0.30    all     0.3801
iprec_at_recall_0.40    all     0.3396
iprec_at_recall_0.50    all     0.2941
iprec_at_recall_0.60    all     0.2377
iprec_at_recall_0.70    all     0.1916
iprec_at_recall_0.80    all     0.1326
iprec_at_recall_0.90    all     0.0874
iprec_at_recall_1.00    all     0.0269
P_5                     all     0.3862
P_10                    all     0.3828
P_15                    all     0.3379
P_20                    all     0.3138
P_30                    all     0.2816
P_100                   all     0.1828
P_200                   all     0.1228
P_500                   all     0.0634
P_1000                  all     0.0344
```

七、实验总结

为了高效地完成本次任务，小组成员调研了多个信息检索的开源工具，最终决定使用 Terrier。另外，原始的数据文件过于冗杂，含有大量的不必要信息，占用空间大，处理过程非常消耗时间，因此我们进行了数据预处理工作。数据预处理之后，我们建立索引，选择模型。在选择检索模型的过程中，我们基于交叉验证技术比较了多种加权模型，依据实验要

求选择了十个不同的加权检索模型进行实验的检索评估，并在其中四个得到的检索结果较好的模型基础上进行相关反馈，优化检索结果。

当然，针对上述的实验方案进行改进时遇到了许多问题，比如伪相关反馈技术，MAP 和 P@10 都有比较明显得提高，然而 P@5 结果并没有像预期一样得到较高的提升，反而降低。我们分析认为，可能是不同模型的 P@5 均已经取得较好的结果，再进行相关反馈，可能带来负面影响。如果查询反馈轮数过多，还会引起主题漂移，导致结果更差。

未来还有许多地方值得改进。首先，对 Terrier 工具的理解程度不够。另外，我们没有使用词嵌入技术对语料进行处理，在进行伪相关反馈时，只利用了前 k 个文档中 TF-IDF 权重比较大的词语进行查询扩展，却无法使用前 k 篇文档语义层面的信息。比如，深度学习方面有著名的 Word2Vec 模型，HMM 等模型，可以对文档进行更深层次的主题定义，在数据量较大的时候，往往具有很好的效果，CNN，RNN 对文档特征的抽取效果很好。