实验报告

计**53** 吴昊哲 **2015011297**

BM25算法的实现

BM25算法的实现较为简单，但是此处修改了simpleScorer的构造函数，传入了indexReader，以便获得doc中abstractString的长度，具体实现见如下代码:

```
assert doc != -1;
         try
         {
              Document currentDoc = reader.document(doc);
              String absStr = currentDoc.get("abstract");
              int docLen = absStr.length();
              float denominator = K1 * (1 - b + b * (float)docLen / avgLength) + termDocs.freq();
              float numerator = (K1 + 1) * termDocs.freq();
              float tf = numerator / denominator;
              return tf * idf;
         }
         catch (IOException e)
         {
              System.out.println(e);
              return idf * this.termDocs.freq();
         }
```

VSM模型与BM25的对比

VSM模型则直接使用了lucene自带的term query

对爆笑这个词进行了搜索，VSM与BM25对比的结果如下:

vsm:

爆笑 查询

结果显示如下：

1. 爆笑趣图 爆笑真人 百变小胖

2. 爆笑趣图 爆笑真人 百变小胖

3. 爆笑趣图 爆笑真人 百变小胖

4. 爆笑趣图 爆笑真人 百变小胖

bm25:

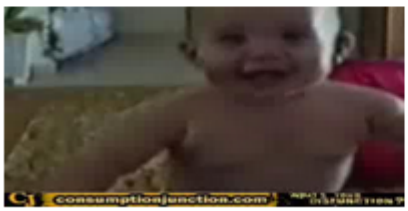
爆笑

结果显示如下：

1. 爆笑趣图 爆笑真人 爆笑宝宝

2. 爆笑趣图 爆笑真人 爆笑宝宝

3. 爆笑趣图 爆笑真人 爆笑宝宝



可见VSM中爆笑出现的频率更高，但是BM25中爆笑出现的频率并不一定是最高的，这与它们各自的计算方式相关。

用**html**扩充语料库

首先用python写了一个脚本将各个html中的title抽取出来，并处理了编码问题，随后生成了一个新的xml文件供ImageIndexer生成新的index，下面对比了是否扩充语料库的搜索结果:

美女(扩充语料库)

美女 查询

结果显示如下:

1. 酷车靓影 车展模特 国外车模

2. 美女时尚 美女自拍 自拍艺术

3. 性感女星 网络美女 小天女

美女(未扩充语料库)

美女 查询

结果显示如下：

1. 美女时尚 美女自拍 美女自拍

2. 美女时尚 美女自拍 美女自拍

3. 美女时尚 美女自拍 美女自拍

可见扩充语料库后，搜索结果比扩充前更加丰富。

查询词分词

对查询词同样适用了IKAnalyzer进行分词，随后每个term都生成一个termQuery，并将termQuery add到BooleanQuery中,add的方式为occur.SHOULD。这样即使当搜索词与索引不是完美匹配时，也可以利用分词有一个较好的搜索结果。下面给出一个对比:

分词前

商务部 查询

结果显示如下：

no such result

1 2 3 4 5 6 下一页

分词后

商务部 查询

结果显示如下：

1. 网页素材 精美佳图 商务金融

2. 网页素材 资料图片 商务金融

3. 网页素材 精美佳图 商务金融

可见一些分词前无法搜到的图片在分词后可以搜到一些折中的结果。