

实验四 Pivoted Length Normalization VSM and BM25

本次实验是在实验三布尔检索的基础上进行改进，对文档集中每个出现检索词的文档计算相似度评分，并将 doc_id 按其相似度从大到小的顺序输出。两种相似度评分公式如下：

State of the Art VSM Ranking Functions

- Pivoted Length Normalization VSM [Singhal et al 96]
$$f(q, d) = \sum_{w \in q \cap d} c(w, q) \frac{\ln[1 + \ln[1 + c(w, d)]]}{1 - b + b \frac{|d|}{avdl}} \log \frac{M + 1}{df(w)}$$
- BM25/Okapi [Robertson & Walker 94]
$$f(q, d) = \sum_{w \in q \cap d} c(w, q) \frac{(k + 1)c(w, d)}{c(w, d) + k(1 - b + b \frac{|d|}{avdl})} \log \frac{M + 1}{df(w)}$$

$b \in [0, 1]$
 $k_1, k_3 \in [0, +\infty)$

1. 建立 inverted index

读取 tweets 数据集、截取正文、分词等预处理等步骤与实验三一致，在建立 inverted index 时，由于在计算 query 与文档相似度时需要用到 TF 与 DF，因此建立倒排索引时仍然使用实验三中将字典的 value 设为集合的方法不再合适，这里字典中每个 word 的 value 设为一个列表，它的元素为包含此 word 的每个 doc_id 及其 c(w,d) 组成的列表。

```
#建立 inverted index
dict1 = {}
label = 0
for word_list in textword:
    for word in word_list:
        if word not in dict1:
            dict1[word] = []
            dict1[word].append([label, word_list.count(word)])
        elif label != dict1[word][-1][0]:
            dict1[word].append([label, word_list.count(word)])
        label = label + 1
```

2. 检索、打分

采用与实验三类似的思路，以两种方法的缩写“PLNVSM”，“BM25”作为输入的标识符，采取相应的公式进行计算。对 query 进行预处理，得到包含每个检索 word 的列表，对字典中 word 对应的每个文档进行打分（参数 $b = 0.5$ ， $k = 2$ ），将列表

在搜索 Ron Weasley Birthday 完成后，查询排名靠前的 doc 的内容，发现内容简单一致：

```
enter a doc_id to browse:
16745
Happy Birthday Ron Weasley ;;)
enter a doc_id to browse:
16787
HAPPY BIRTHDAY RON WEASLEY !!!
enter a doc_id to browse:
17011
Happy birthday ron weasley
enter a doc_id to browse:
```

这也解释了为何会有许多打分高且相等的 doc 的现象。