Homework 4 Pivoted Length Normalization VSM and BM25 实验报告

实验内容：

任务：

– 实现Pivoted Length Normalization VSM;

– 实现BM25;

注意

- 改进Postings：(docID, Freq)，不仅记录单词所在的文档ID，也记录其在文档中的Frequency;
- 构建inverted index时，记录文档的长度，以及计算average document length (avdl)

实验步骤：

1. 读入查询和 tweet

与上个实验的不同的地方是在读入的时候也要记录查询的 id 和推特的 id。修改了上个实验的代码。

```
def get_tweets(path):
    # 读取json格式的tweet并提取其内容并返回分词后结果
    import json
    from nltk.tokenize import word_tokenize
    tweets = []
    f = open(path,'r',encoding='utf-8',errors='ignore')
    for line in f.readlines():
        tweets.append((json.loads(line)['tweetId'], word_tokenize(json.loads(line)['text'])))
    return tweets
```

```
def get_querys(path):
    from nltk.tokenize import word_tokenize
    f = open(path,'r',encoding='utf-8',errors='ignore')
    arr1 = []
    arr2 = []
    for line in f.readlines():
        if line[0:5] == '<num>':
            arr1.append(line[16:-8])
        if line[0:7] == '<query>':
            arr2.append(word_tokenize(line[7:-9]))
    arr = []
    for i in range(len(arr1)):
        arr.append((arr1[i],arr2[i]))
    return arr
```

2. 在 tweets 数据集上构建 inverted index

在上个实验的代码的基础上进行修改。不同的是需要统计更多的东西，比如单词在文档中出现的次数，文档的长度即平均长度，和单词在多少文档中出现(df)。

统计文档平均长度 avdl:

```
#calculate avdl
avdl = 0
for tweet_id, tweet in tweets:
    avdl += len(tweet)
avdl = 1.0 * avdl / num
```

统计 df：

```
#calculate df
df = {}
for tweet_id, tweet in tweets:
    for word in tweet:
        if word not in df.keys():
            df[word] = 0
        df[word] = df[word] + 1
```

统计 inverted index:

```
#calculate inverted_index
inverted_index = {}
for tweet_id, tweet in tweets:
    for word in tweet:
        if word not in inverted_index.keys():
            inverted_index[word] = {}
        if tweet_id not in inverted_index[word].keys():
            inverted_index[word][tweet_id] = 0
        inverted_index[word][tweet_id] += 1
```

统计各个文档长度：

```
#calculate doc_len
doc_len = {}
for tweet_id, tweet in tweets:
    doc_len[tweet_id] = len(tweet)
return (num,avdl,df,inverted_index,doc_len)
```

3. 用 BM25 进行查询

根据公式

$$f(q,d)=\sum_{w\in q\cap d} c(w,q)\frac{(k+1)c(w,d)}{c(w,d)+k(1-b+b\frac{|d|}{avdl})}\log\frac{M+1}{df(w)}$$

每个查询对每个文档进行打分，并将结果按评分排序：

```
for query_id,query in querys:
    #calculate counts of words in the query
    cnt = {}
    for word in query:
        if word not in cnt.keys():
            cnt[word] = 0
        cnt[word] = cnt[word] + 1

    score = {}
    #calculate scores
    import math
    for word in cnt.keys():
        if word not in inverted_index.keys():
            continue
        for tweet_id,num in inverted_index['to'].items():
            if tweet_id not in score.keys():
                score[tweet_id] = 0
            score[tweet_id] += cnt[word] * (k + 1) * num / (num + k * (1 - b + b
    # print(score)
    res = []
    for tweet_id,value in score.items():
        res.append((value,tweet_id))
    res.sort(reverse = True)
    # print(res)
    for term in res:
        ans.append((query_id,term[1]))
return ans
```

里面套公式的那一句没截全，是这样的：

score[tweet_id] += cnt[word] * (k + 1) * num / (num + k * (1 - b + b * doc_len[tweet_id] / avdl)) * math.log(1.0 * (M + 1) / df[word])

4. 查询优化

调用学长的代码进行 evaluation，得到的 MAP 值和 NDCG 值都是零点几，比较低。

实验总结：

在上个实验的代码的基础上实现了 BM25