Report

This is a script which is used to retrieve queries in documents, and find the similarities between them.

To run this script, type command: python retrieval.py (options) documents.txt queries.txt.
The options contain:

-s stop_list	[whether to use stoplist to filter document]
-I filename	[write results in this file]
-n number	[retrieving for a specific query, number is docid of query]
-I	[whether to use NLTK library to process document]
-A	[retrieving for the full query set]

To get the results of full query set, type command:

python retrieval.py -s stop_list.txt -I -I results.txt -A documents.txt queries.txt

documents.txt and queries.txt will be read by ReadDocuments() in read_documents.py. Then the words are filtered by stoplist and NLTK stemmer. Finally, the frequency of each word will be calculated and stored in a new dictionary with word (word is key, frequency is value).

```
40 #Filter the document
41 def filter_document(filename):
42     documents = ReadDocuments(filename)
43     wordRE = re.compile(r'[A-Za-z]+')
44     stemmer = PorterStemmer()
45     filtereddoc = []
46     for doc in documents:
47         doclist = []
48         for line in doc.lines:
49             line.replace("\n","")
50             for word in wordRE.findall(line.lower()):
51                 if word not in stops:
52                     if '-I' in opts:
53                         word = stemmer.stem(word)
54                     doclist.append(word)
55         c = Counter(doclist)
56         filtereddoc.append(c)
57     return filtereddoc
58
```

After filtering documents.txt and queries.txt, calculating the tf-idf for each word.

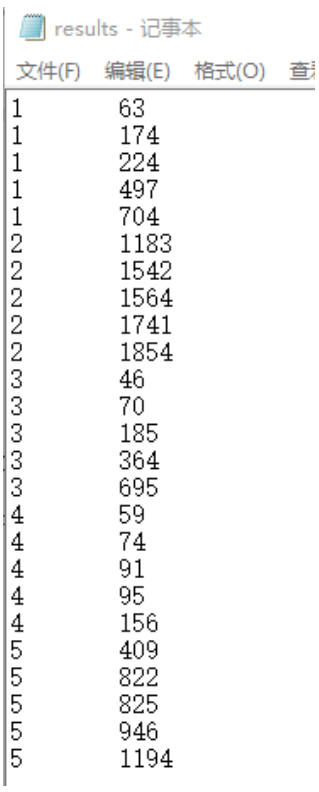
```
#calculate idf
def tf(word, count):
    return count[word] / sum(count.values())

def n_containing(word, count_list):
    return sum(1 for count in count_list if word in count)

def idf(word, count_list):
    return math.log(len(count_list) / (1 + n_containing(word, count_list)))

def tfidf(word, count, count_list):
    return tf(word, count) * idf(word, count_list)

#filter documents and calculate tf-idf for each document
countlist = filter_document(filename)
doctfidf = []
for i, count in enumerate(countlist):
    scores = {word: tfidf(word, count, countlist) for word in count}
    doctfidf.append(scores)

#filter queries and calculate tf-idf for each query
querylist = filter_document(queryname)
quetfidf = []
for i, count in enumerate(querylist):
    scores = {word: tfidf(word, count, querylist) for word in count}
    quetfidf.append(scores)
```

Loop the document in queries and documents. For each pair of them, find the same word and calculate cos similarity.

For each document in queries, sort the similarity with document in documents. Then print 5 pairs with the highest similarity into results.txt

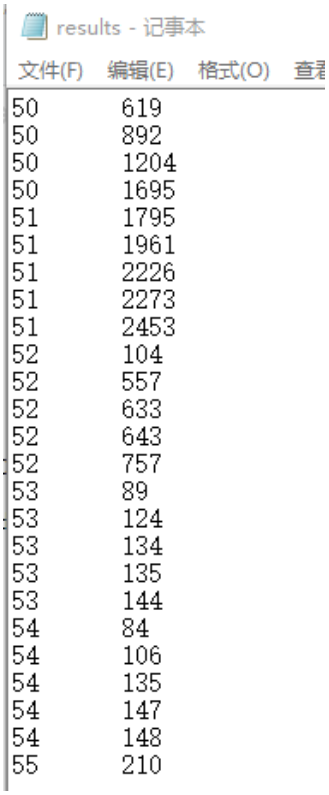
```
#retrieving all queries
if '-A' in opts:
    for i,query in enumerate(quetfidf):
        sims = {}
        for j,doc in enumerate(doctfidf):
            si={}
            for word in quetfidf[i]:
                if word in doctfidf[j]:
                    si[word] = 1
            if len(si) == 0:
                sims[j] = 0
                continue
            pSum = sum([quetfidf[i][word]*doctfidf[j][word] for word in si])
            sum1Sq = sum([pow(quetfidf[i][word],2) for word in si])
            sum2Sq = sum([pow(doctfidf[j][word],2) for word in si])
            den = math.sqrt(sum1Sq*sum2Sq)
            if den == 0:
                sims[j] = 0
                continue
            r = float(pSum)/den
            sims[j] = r
        sorted_sims = sorted(sims.items(),key=lambda x: x[1],reverse=True)
        for k,v in sorted_sims[:5]:
                f.write(str(i+1)+'\t')
                f.write(str(k+1)+'\n')
```

In results.txt, the first column is docid of queries.txt. The second column is docid of documents.txt. Each docid of queries.txt will correspond to 5 docids of documents.txt. They are the first 5 similar pairs in this query.


results - 记事本

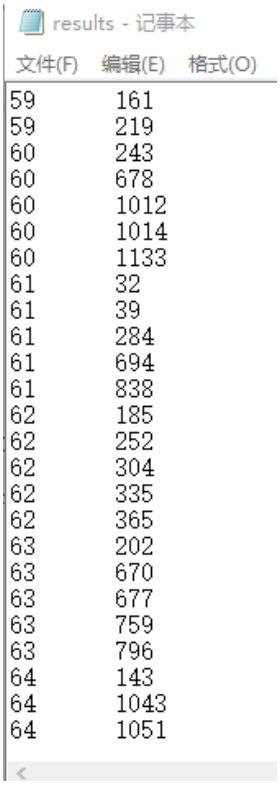
文件(F) 编辑(E) 格式(O) 查

1	63
1	174
1	224
1	497
1	704
2	1183
2	1542
2	1564
2	1741
2	1854
3	46
3	70
3	185
3	364
3	695
4	59
4	74
4	91
4	95
4	156
5	409
5	822
5	825
5	946
5	1194


results - 记事本

文件(F) 编辑(E) 格式(O) 查

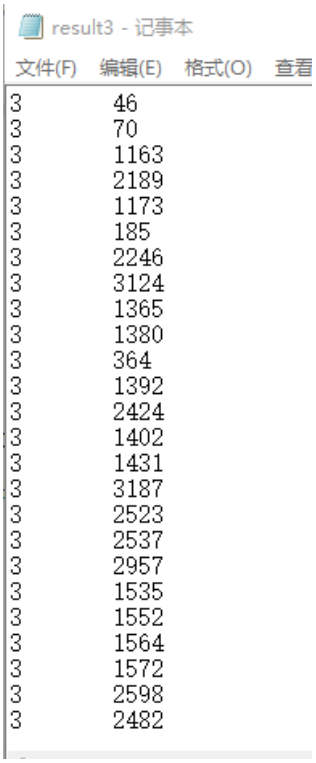
50	619
50	892
50	1204
50	1695
51	1795
51	1961
51	2226
51	2273
51	2453
52	104
52	557
52	633
52	643
52	757
53	89
53	124
53	134
53	135
53	144
54	84
54	106
54	135
54	147
54	148
55	210


results - 记事本

文件(F) 编辑(E) 格式(O)

59	161
59	219
60	243
60	678
60	1012
60	1014
60	1133
61	32
61	39
61	284
61	694
61	838
62	185
62	252
62	304
62	335
62	365
63	202
63	670
63	677
63	759
63	796
64	143
64	1043
64	1051

If type : python retrieval.py -s stop_list.txt -I -I result3.txt -n 3 documents.txt queries.txt
It means only retrieve docid 3 in queries.txt. It will compare with each docid in documents.txt. Then sort all the pairs that similarity not equal to 0. The result3.txt:


result3 - 记事本

文件(F) 编辑(E) 格式(O) 查看

3	46
3	70
3	1163
3	2189
3	1173
3	185
3	2246
3	3124
3	1365
3	1380
3	364
3	1392
3	2424
3	1402
3	1431
3	3187
3	2523
3	2537
3	2957
3	1535
3	1552
3	1564
3	1572
3	2598
3	2482