Information Retrieval

Project 2021-22

Specification

The aim of the project is to complete your Python Information Retrieval system, index a set of documents with Gold Standard answers, and evaluate it using Precision, Recall and F-Measure.

1. Outline

1. Tokenised Chinese and looked at the frequency distribution of words (Lab 1)

2. Looked at the format of the gold standard documents. (Lab 2)

3. Read in and tokenised a gold standard document (Lab 2)

4. Computed basic TF and IDF and created an inverted index (Lab 2)

5. Computed TF and IDF according to standard equations (Lab 3)

6. Matched a document with a query (Lab 3)

7. Experimented with search of a disk based index using Hashing (Lab 4)

Now we come to the project. What we need to do is:

1. Use a larger document collection

2. Write code to save the index on disk (not using hashing but an easier way)

3. Normalise the index so the maximum match is 1

4. Index the document collection

5. Go through all the queries and retrieve the results. Evaluate them using Precision, Recall, F Measure.

2. Document Collection

So far we had 20 queries on one topic, Taobao shopping. Now we have ten topics:

Cats
Dogs
Cigarettes
Nikola Tesla
Huawei
Taobao Shopping
Black Holes
Hayao Miyazaki (Japanese Anime)
Trains
Guzheng (Traditional Chinese Instrument)

You can download them from Moodle. They are all in an identical format to Taobao and they all come from the Baidu Baike online encyclopedia. Each has 20 questions, complete with correct answers.

3. Save Index to Disk

We tried indexing direct to disk using a hash table but that was just an experiment. Instead, we can use an easier way: Pickle. Example:

```
import pickle

a = {'hello': 'world'}

with open('filename.pickle', 'wb') as handle:
   pickle.dump(a, handle, protocol=pickle.HIGHEST_PROTOCOL)

with open('filename.pickle', 'rb') as handle:
   b = pickle.load(handle)

print( a == b )
```

(Taken from https://stackoverflow.com/questions/11218477/how-can-i-use-pickle-to-save-a-dict)

(See pickle_demo on Moodle.)

You can use this to save your index. To save multiple things (e.g. several different dictionaries) is possible, but the easiest way is to use a separate file for each or to group them together (e.g. a list of dicitionaries) and save the list.

Required functions in your program: save_index(), load_index()

Make sure you include all required functions in your program. This is to simplify marking of the projects. You can include any other functions that you need.

4. Normalise the Index

At present there is no guarantee that match results are within a certain range. The answer is to normalise the index. See lecture notes, Slide 108 ‘Vector Interpretation of the VSM’.

Recall that the Forward Index looks like this:

```
index = { '315消费电子投诉网_2017115146.htm': { 'cat': 4, 'dog': 6 },\\
             '7天无理由退换货_2017115146.htm': { 'cat': 3, 'dog': 8 }, ....}
```

So, for document `'315消费电子投诉网_2017115146.htm'` the ‘TF*IDF’ value for ‘cat’ is 4 and for ‘dog’ it is 6. These are not proper TF*IDF values of course.

To normalise, we need to make sure that the sum of the squares of k*4 and k*6 add up to one, for some value of k. To compute k:

k = sqrt(1 / (4^2 + 6^2) = 0.139 (approx)

So we get (approx):

```
index = { '315消费电子投诉网_2017115146.htm': { 'cat': 0.555, 'dog': 0.832
},\\ .... }
```

Note: These are the values for `'315消费电子投诉网_2017115146.htm'`. For other files, e.g. `'7天无理由退换货_2017115146.htm'` the values will be different. That is why you use the forward index.

After computing the forward index, you need to invert it to produce the inverted index (recall Lab 2).

When you match a query with a document, you need to normalise the query vector first, using the same procedure. For the query vector, you can assume the TF*IDF score for each query term is 1.

Required functions in your program: normalise_index()

5. Index the Document Collection

Take all the document collections and place them in project_docs directory under ir. Each .zip/.rar already has project_docs under ir. So you need to combine all the project_docs into one.

Write code to index all the documents, compute IDF scores, create the forward index with TF*IDF scores, normalise, create the inverted index and safe it in a Pickle file. You already have this mostly done from the previous labs.

Required functions in your program: index_collection()

6. Run Queries and Evaluate the Results

Any .txt file in project_docs contains one query and one or more results. There are 200 such .txt files.

e.g. 2017115146_01.txt is like this:

淘宝的诞生
马[淘宝商城租金单位]_2017115146.htm
双十二[双十二购物狂欢节]_2017115146.htm
双十一购物狂欢节_2017115146.htm

Query is 淘宝的诞生

There are three docs which contain the answers马[淘宝商城租金单位]_2017115146.htm plus the two others.

Note: These three results are **unordered**. The order in this file does not signify anything.

To evaluate, we need to compute Precision, Recall and F-Measure. See lecture slides 180 onwards.

1. We can compute P, R, F for each query and print it out.

2. At the end we can compute the average P, R, F, for the whole document collection.

Required functions in your program: evaluate()

evaluate(), when run, should produce an output log file **eval.txt** which says for each query, the query text, the first 5 matching results returned by the system, and the P, R, F values, like this:

```
Query: 淘宝的诞生
file1.htm
file2.htm
file3.htm
file4.htm
file5.htm
P = 0.200 R=0.124 F=0.153

Query: 淘宝交易
file6.htm
file7.htm
file8.htm
file9.htm
file10.htm
P = 0.100 R=0.097 F=0.098
```

...etc etc, then at the end:

```
All Queries:
P = 0.201 R=0.104 F=0.137
```

Note 1: Real files are not called file1.htm etc. They are things like 双十一购物狂欢节_2017115146.htm or 淘宝旺旺_2017115146.htm

Note 2: Write all values with three digits after the '.', exactly as shown above. You can look back at the Python lecture notes from last semester to remember how to do this!

Required file produced by your program: eval.txt

7. Write Report

Compose a short report by filling in the report_template.doc on Moodle and hence creating the file report.doc. It is very short, maximum 8 sentences!

Required file: report.doc

8. Files and Functions

Call your program ir.py

As described above it **must** contain these functions:

save_index(), load_index()
normalise_index()
index_collection()
evaluate()

Your ir.py can also contain any other functions you like, e.g. from labs etc.

tcf_tokenise_chinese_file.py (as previously)

Evaluation output .txt file eval.txt

Report report.pdf

9. Upload your Files to Moodle

On Moodle, go to Week 10, i.e. 1 November - 7 November.
Deadline: 11h59 CST , Sunday 7 November, Week 10.

Please check you have:

- Uploaded correct files
- Comment at the start of ir.py (marks for this) with your name and number
- Code for the functions specified (use other functions / classes in addition as you wish)
- Output in eval.txt is in the correct format
- File report.pdf starts with your own name and number