README - 590R Assignment 3

Submitted By :- Ankita Mehta

November 10, 2017

This document will take you through the implementation of structured query operators and belief functions for shakespeare dataset written in Python language. This part has been build over the previous submission. To run the project, change the current working directory to Assignment3/src. Paths mentioned in the commands written below are according to the above current working directory.

NOTE: python RetrievalAPI.py -h commands will give you the information about the command line parameters of the wrapper.

1 Code Dependencies

Python 2.7.12 :: Anaconda has been used for this project.
Pip version 9.0.1
Installing trecrun: pip install trectools : Refer this github link for more details: trecrun
Installing argparse: pip install argparse

2 Build and Run the Code

Code structure :
The code structure has been divided into three main folders: Assignment3/data, Assignment3/src, Assignment3/results.

*Assignment3/data:

1. shakespeare.json: It is the input data file for creating compressed/uncompressed index
2. query.txt: It is the query file having 10 queries
3. unc_manifest: : It is the manifest file having the names of files and indexes created for uncompressed file.
4. comp_manifest: It is the manifest file having the names of files and indexes created for compressed file.

*Assignment3/src:
This folder has some codes from the previous assignment :

1. main.py: It is the entry point for finding the indexes, generating query files with unique random words and dice coefficient.
2. indices_creation.py: It is the class having functions for finding the indexes, generating query files with unique random words and dice coefficient.
3. Encoding_decoding.py: It is the class having functions for performing delta and Vbyte encoding-decoding.
4. API_extract_statistics.py : It is the class to extract vocabulary, Collection Term frequency and document frequency for the query word.
5. prob_scores.py : It has the implementation of dirichlet smoothing for $\mu = 2000$.

Codes which have been amended/written specifically for this assignment are :

1. RetrievalAPI.py: : It is the entry point for API Retrieval.

2. unordered_window.py : This code is for finding the scores of the document present in the unordered window of the given query. (Window size is taken as the parameter and assigned value as query length). This also returns the new posting list for that query.
3. ordered_window.py : This code is for finding the scores of the document present in the ordered window of the given query. (Window size is taken as the parameter and assigned value 1 for this assignment). This also returns the new posting list for that query.
4. term.py : It will return the document scores and the posting list if the structured query operator is just the term.
5. boolean_and.py : It will return the document scores and the posting list if the structured query operator is Boolean_and. Its implementation is just the same as unordered window with the window parameter taken as the document length.
6. filters.py : It has implementation for filter require and filter reject structured query operators.
7. belief_operators.py : It has implementation for various belief operators.

Note: Filter require and filter reject operators have been implemented in the Modular fashion. 1st argument is any of the proximity expression and 2nd argument is the combination of any belief operator or proximity expression. For evaluating second argument, multiple dispatch has been used. Also, the format of second argument should be in a specific structure/ class. Then, a recursive function has been written which will handle all the cases for its evaluation.

*Assignment3/results: It has the some previous results (6 for each) for compressed and uncompressed index to be used for this assignment.

1. xxx_docNo_playId: It is the mapping from doc_No to playId
2. xxx_docNo_sceneId: It is the mapping from doc_No to scene_Id
3. xxx_lookup_table: It is the lookup table having mappings from term to doc_No , count , Collection_term_frequency , document_frequency
4. xxx_sceneId_docNo: It is the mapping from scene_Id to doc_No
5. xxx_Inverted_list: It is the binary file having stored inverted lists.
6. xxx_docNo_length: It is the mapping from doc_No to its length.

Note: Here xxx is either 'unc' or 'comp' for uncompressed and compressed index respectively.

Other results relevant to this assignment are:

1. od1.trecrun , uw.trecrun are the files written in TREC format for each model. One more file is present in results folder : judgements.txt which has the judgements for 6-10 queries.

The other files present in the Assignment3 folder are : README, report.pdf

2.1 RetrievalAPI.py

One wrapper have been written for this project i.e. - **RetrievalAPI.py** having location: Assignment3/src

There are 2 variants to run this code using different command line parameters :

-q specifies the oneword query path ;
-m specifies the manifest file path - the file having all paths written
-pe specifies the proximity expression used - It can be : term / ordered_window / unordered_window / boolean_and. Here it is unordered_window

2.1.1 Unordered Window

Command that will be used to test the uncompressed indexer on one-word query file.

```
python RetrievalAPI.py -q ../data/query.txt -m ../data/unc_manifest -pe unordered_window
```

2.1.2 Ordered Window

Command that will be used to test the uncompressed indexer on one-word query file

```
python RetrievalAPI.py -q ../data/query.txt -m ../data/unc_manifest -pe ordered_window
```