# README - 590R Assignment 1

Submitted By :- Ankita Mehta

October 6, 2017

This document will take you through the implementation of Indexer for shakespeare dataset written in Python language. To run the project, change the current working directory to Assignment/src. Paths mentioned in the commands written below are according to the above current working directory.

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

## 1 Code Dependencies

Python 2.7.12 :: Anaconda has been used for this project.
Pip version 9.0.1
Installing JSON : pip install json
Installing argparse: pip install argparse

## 2 Build and Run the Code

**Code structure** :
The code structure has been divided into three main folders: Assignment1/data, Assignment1/src, Assignment1/results.

*Assignment1/data:

1. shakespeare.json: It is the input data file for creating compressed/uncompressed index
2. oneword_query.txt: It is the query file having 100 rows with each row having 7 words generated after making indexes
3. twoword_query.txt: It is the query file having 100 rows with each row having 14 words generated after making indexes
4. unc_manifest: : It is the manifest file having the names of files and indexes created for uncompressed file.
5. comp_manifest: It is the manifest file having the names of files and indexes created for compressed file.

*Assignment1/src:

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. RetrievalAPI.py: : It is the entry point for API Retrieval.
5. API_extract_statistics.py : It is the class to extract vocabulary, Collection Term frequency and document frequency for the query word.
6. Calculate_doc_score.py : It is the class to calculate doc scores using document-at-a-time evaluation method for compressed and uncompressed indices.

*Assignment1/results: It has the separated results (6 for each) for compressed and uncompressed index.

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.

The other files present in the Assignment1 folder are : README, report.pdf
oneword_query.txt , twoword_query.txt - these are the query files generated for calculating the dice coefficient and API retrieval.

Two wrappers have been written for this project i.e. - **main.py , RetrievalAPI.py** having location: Assignment1/src

## 2.1 main.py

There are 6 variants to run this code using different command line parameters :

### 2.1.1 Creating uncompressed Index with Dice coefficient

Command that will be used when a user wants to create uncompressed index and also wants to compute Dice Coefficient for random 100 queries :

```
python main.py -d ../data/shakespeare-scenes.json -u -dc
```

**-d** specifies the data path ;
**-u** that the user wants to create uncompressed index ;
**-dc** that the user wants to compute dice coefficient as well

Results computed in this case are stored in Assignment1/results/uncompressed/ and query files are stored in Assignment1/

### 2.1.2 Creating uncompressed Index without Dice coefficient

Command that will be used when a user wants to create uncompressed index but doesnt want to compute Dice Coefficient :

```
python main.py -d ../data/shakespeare-scenes.json -u
```

**-d** specifies the data path ;
**-u** that the user wants to create uncompressed index ;
**-dc** will not be specified here

Results computed in this case are stored in Assignment1/results/uncompressed/

### 2.1.3 Creating compressed Index with Dice coefficient

Command that will be used when a user wants to create compressed index and also wants to compute Dice Coefficient for random 100 queries :

```
python main.py -d ../data/shakespeare-scenes.json -c -dc
```

**-d** specifies the data path ;
**-c** that the user wants to create compressed index ;
**-dc** that the user wants to compute dice coefficient as well

Results computed in this case are stored in Assignment1/results/compressed/ and query files are stored in Assignment1/

#### 2.1.4 Creating compressed Index without Dice coefficient

Command that will be used when a user wants to create compressed index but doesn't want to compute Dice Coefficient :

python main.py -d ../data/shakespeare-scenes.json -c

**-d** specifies the data path ;
**-c** that the user wants to create compressed index ;
**-dc** will not be specified here

Results computed in this case are stored in Assignment1/results/compressed/

#### 2.1.5 Ony Dice coefficient

Command that will be used when a user just wants to compute Dice Coefficient for random 100 queries. In this case, he must have indices ready, so the default index taken is compressed

python main.py -d ../data/shakespeare-scenes.json -dc

**-d** specifies the data path ;
**-u** will not be specified
**-dc** that the user wants to compute dice coefficient as well

Results computed in this case are stored in Assignment1/results/compressed/ and query files are stored in Assignment1/

#### 2.1.6 No command line parameter

Another varient to run this code is : passing no command line parameters except data path. In this case, it will just create compressed index ( taken as the default index )

python main.py -d ../data/shakespeare-scenes.json

**-d** specifies the data path

Results computed in this case are stored in Assignment1/results/compressed/

### 2.2 RetrievalAPI.py

There are 4 variants to run this code using different command line parameters :

#### 2.2.1 Uncompressed Index with one word query file

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

python RetrievalAPI.py -q ../oneword_query.txt -m ../unc_manifest

**-q** specifies the oneword query path ;
**-m** specifies the manifest file path - the file having all paths written

#### 2.2.2 compressed Index with one word query file

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

python RetrievalAPI.py -q ../oneword_query.txt -m ../comp_manifest

#### 2.2.3 uncompressed Index with two word query file

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

```
python RetrievalAPI.py -q ../twoword_query.txt -m ../unc_manifest
```

#### 2.2.4 compressed Index with two word query file

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

```
python RetrievalAPI.py -q ../twoword_query.txt -m ../comp_manifest
```