# README - 590R Assignment 5

Submitted By :- Ankita Mehta

December 07 , 2017

This document will take you through the implementation of one of the Query Independent Features i.e Prior on shakespeare dataset . It has been written in Python language. This part has been build over the previous submission of Inference Network. To run the project, change the current working directory to Assignment5/src. Paths mentioned in the commands written below are according to the above current working directory.

**NOTE: python test_belief_with_prior.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 argparse: pip install argparse

## 2 Build and Run the Code

**Code structure** :
The code structure has been divided into three main folders: Assignment5/data, Assignment5/src, Assignment5/results.

*Assignment5/data:

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

*Assignment5/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 ? = 2000.
6. 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.

7. 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.

8. term.py : It will return the document scores and the posting list if the structured query operator is just the term.

9. 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.

10. filters.py : It has implementation for filter require and filter reject structured query operators.

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

1. dump_prior.py : It dumps the uniform and random prior probabilities which are required by the belief node implementation.

2. belief_operators.py : This code has been taken from the previous assignment and functionality for "prior", a document dependent feature, has been added into it.

3. test_belief_with_prior.py : This is a wrapper for implementing this assignment, in which a structure has been created manually to test the belief_and operator with prior probabilities.

*Assignment5/results: It has 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. random.prior : It has the randomly assigned probabilities to all the documents present in our index.

2. uniform.prior : It has the uniformly assigned probabilities to all the documents present in our index. i.e 1/total_documents.

3. random.trecrun : It has the results generated by assigning random probabilities to all the documents.

4. uniform.trecrun : It has the results generated by assigning uniform probabilities to all the documents.

The other files present in the Assignment5 folder are : README, report.pdf

## 2.1 test_belief_with_prior.py

One wrapper have been written for this project i.e. - **test_belief_with_prior.py** having location: Assignment5/src

python test_belief_with_prior.py -p uniform

python test_belief_with_prior.py -p random

**-p** specifies the prior feature : either random or uniform