

Text Information Systems

Project

Content:

1.	Name and netid:.....	2
2.	Theme of the Project.....	2
3.	Project Summary:	2
4.	Project Status:	2
5.	Software Setup & Usage:	2
6.	Software implementation:.....	4
7.	Ways to improve performance of the search engine:	6

1. Name and netid:

Name: Nita Agarwal

NETID: nitaa2

TeamName: TeamNA (individual participant)

2. Theme of the Project

The Theme of my project is Free Topic

3. Project Summary:

Develop an application to link the MP descriptions to relevant TIS course lectures provided on Coursera.

Task List:

- **Task A: Collecting data (5 hours):** Gather MP assignment and lecture transcript from Coursera for the complete course duration
- **Task B: Preprocess data (8 hours):** Data preprocessing may include removing stopwords and words with less than min length, stemming etc. This step will help in creating the corpus and query data set.
- **Task C: Build Ranking logic (5 hours):** Build the logic for indexing the corpus data set and determine the appropriate metapy ranker for the application
- **Task D: Evaluation and parameter fine tuning (5 hours):** Evaluate the ranking and fine tune the parameter settings based on the human evaluation of the ranked list.

Stretch Goals:

- **Task E:** Build User Interface for the project. This will help the user select the MP and see the corresponding lecture video links on a UI instead of command prompt.
- **Task F:** Build Web based API driven interface for the project. This will help in accessing the application over web

4. Project Status:

I have successfully completed all the tasks in the list(Task A,B,C & D). I have also completed the stretch goals(Task E & F).

5. Software Setup & Usage:

- **Step 1: Setup the raw data** folder as per the location mentioned in adminConfig.toml [rawLectureData]

```

1 prefix = "."
2 rawLectureData = "rawCorpusData"
3 rawMPData = ""
4 processedLectureData = "courseraData"

```

Copy the lectures txt files to this folder. The folder content should look like below screenshot

08_12-8-summary-for-exam-2.en.txt	10/31/2022 7:03 PM	Text Document	15 KB
07_12-7-contextual-text-mining-mining-causal-topics-with-time-series-supervision.en.txt	10/31/2022 7:03 PM	Text Document	16 KB
06_12-6-contextual-text-mining-mining-topics-with-social-network-context.en.txt	10/31/2022 7:03 PM	Text Document	11 KB
05_12-5-contextual-text-mining-contextual-probabilistic-latent-semantic-analysis.en.txt	10/31/2022 7:03 PM	Text Document	14 KB
04_12-4-contextual-text-mining-motivation.en.txt	10/31/2022 7:03 PM	Text Document	6 KB
03_12-3-text-based-prediction.en.txt	10/31/2022 7:03 PM	Text Document	10 KB
02_12-2-opinion-mining-and-sentiment-analysis-latent-aspect-rating-analysis-part-2.en.txt	10/31/2022 7:03 PM	Text Document	13 KB
01_12-1-opinion-mining-and-sentiment-analysis-latent-aspect-rating-analysis-part-1.en.txt	10/31/2022 7:03 PM	Text Document	12 KB
06_11-6-opinion-mining-and-sentiment-analysis-sentiment-classification.en.txt	10/31/2022 7:03 PM	Text Document	10 KB
07_11-7-opinion-mining-and-sentiment-analysis-ordinal-logistic-regression-optional.en.txt	10/31/2022 7:03 PM	Text Document	10 KB
05_11-5-opinion-mining-and-sentiment-analysis-motivation.en.txt	10/31/2022 7:03 PM	Text Document	14 KB

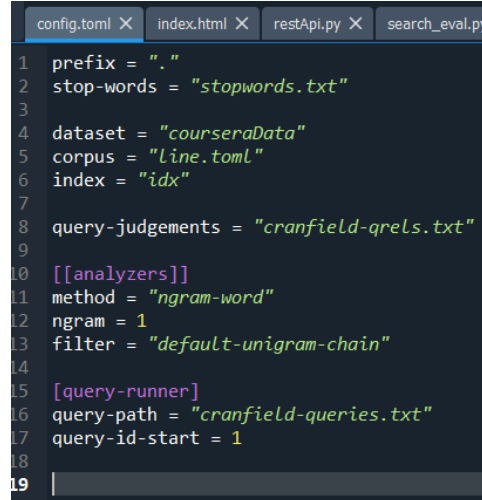
Create the query file: I have created the query file **cranfield-queries.txt**(uploaded) by copying the given MP description in a single line. As we have 6 MPs(1,2.1,2.2,2.3,2.4,3) so we have 6 lines of text in the query file.

Create a db instance. I have done it on SQLite and checked that sqlite3 package is installed. Run the create statements for 2 database tables given in the database_tables.sql. Ensure that the database tables are accessible from the python code. We can check if the database tables are accessible from the python code. I ensured this by copying TISProject.db file in the root folder.

Name	Type	Schema
label_mapping	CREATE TABLE "label_mapping" ("document_id" INTEGER, "label" TEXT, PRIMARY KEY("document_id"))	
search_results	CREATE TABLE "search_results" ("mp_id" TEXT, "document_id" INTEGER, "score" NUMERIC)	

- **Step 2: Run the adminModule.py** to read the raw text from the rawCorpusData folder. Each text file is read at a time and the complete file text is converted into one line. This line is then written to courseraData.dat. Corresponding labels are stored in courseraData.labels. Copy these courseraData.dat & courseraData.labels to the

courseraData folder in the root. As the dataset location mentioned in config.toml is courseraData folder. The label file is then pre-processed so that it can be loaded in the database. The file is label_documentid_mapping.csv (uploaded)

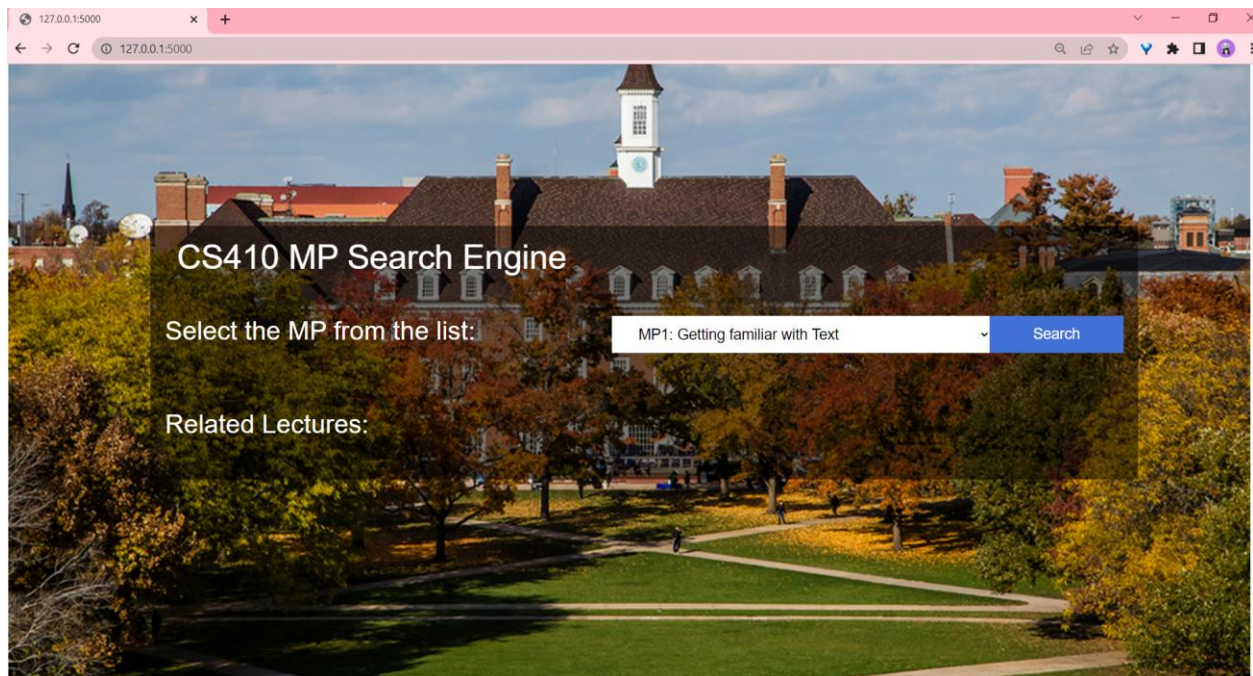


```
1 prefix = "."
2 stop-words = "stopwords.txt"
3
4 dataset = "courseraData"
5 corpus = "line.toml"
6 index = "idx"
7
8 query-judgements = "cranfield-qrels.txt"
9
10 [[analyzers]]
11 method = "ngram-word"
12 ngram = 1
13 filter = "default-unigram-chain"
14
15 [query-runner]
16 query-path = "cranfield-queries.txt"
17 query-id-start = 1
18
19 |
```

- **Step 3: Run search_eval.py**
This will search the course dataset & find the relevant lecture documents as per the MP description written in the cranfield-queries.txt. It stores this result in the database in the table search_results.
- **Step 4: Run restApi.py**
Run the restApi python code. Access the url <http://127.0.0.1:5000/>. It will display the index.html page. Select MP from the pre-populated dropdown list and click 'search' button. The result is displayed on the page with the relevant lecture video labels

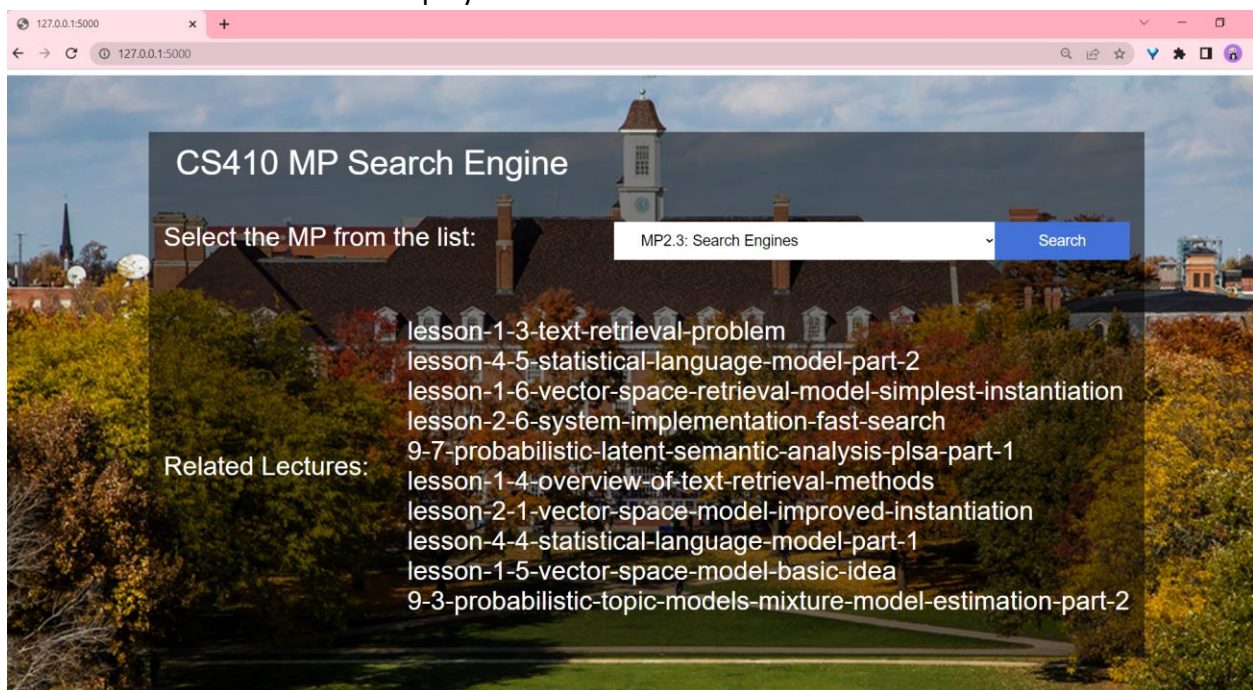
6. Software implementation:

- **UserInterface Layer:**
 - **Index.html +Main2.css**



For the UI I have created index.html. It refers to the main2.css for the styling of the page. (I got the initial idea of layout of the search page was taken from webpage colorlib.com/etc/searchf/colorlib-search-1. I made major changes in the design to suit it to the project requirement and for showing the search results for our MPs).

It has a prepopulated dropdown with the list of MPs in the course. We need to select one MP from this drop-down list and click search. The list of related lectures is displayed on the screen as shown below



- **restApi.py**

I have created Rest API using Flask. I have defined the route '/result' with GET & POST methods. When the user clicks the 'Search' button on the index.html page, the request goes to the restAPI with the MP id. The restAPI.py then connects with the database. It fetches the relevant lecture names using the below query

```
"SELECT l.label FROM search_results s, label_mapping l where s.mp_id=? and  
s.document_id=l.document_id ORDER by s.score desc "
```

The result is then returned and shown in the webpage to the user.

- **Backend Layer:**

- **adminModule.py**

This python file contains the logic to convert the lecture video text to the .dat file and its corresponding label. It refers to the adminConfig.toml for its configuration parameters

- **search_eval.py**

This python file contains the logic to search the course dataset as per the MP text (in queries file) provided. It uses OkapiBM25 to search for the relevant course lectures. It refers to the config.toml for its configuration parameters

- **StorageLayer (SQLite database):**

- **Table search_results**

This table stores the search result (mpid, relevant documentid, score)

- **Table label_mapping**

This table stores the mapping of the label and the corresponding course lecture document id

7. Ways to improve performance of the search engine:

- **Effective Query text:** Due to the limited time for implementation, currently I have added the text provided as the descriptions of the MPs to the cranfield-queries.txt. We can add more relevant keywords to this file corresponding to each MP
- **Parameter modification for OkapiBM25:** The current parameters mentioned for the algorithm can be optimized further and made to fit the course dataset better