GROUP ASSIGNMENT

BSCS 7th SEMESTER

STUDENT DETAILS

Student Name Muhammad Faizan **Student ID Number** 04071813020

Student Name Syed Ahtsham **Student ID Number** 04071813015

ASSIGNMENT DETAILS

Subject Title: Natural Language Processing

Submission Date: December 17, 2021

Submit To: Dr. Akmal Saeed Khattak

Table of Contents

Chapter 1

Introduction, Motivation, Impact

System: Legal document Search engine

Most of the times, lawyers need to search from the previous cases in order to understand and make their evidence strong. Also, Case law is based on precedents i.e., the judicial decisions from the previous cases. Therefore, Lawyers often need to search the cases for this purpose. They need a modern search engine which can search the particular case, or specified type of decisions by the courts from the past. A lawyer types in some key terms and the system displays the top relevant cases to the given query.

Motivation:

In Pakistan, courts are crowded with new cases daily. The lawyers need a way to present their cases in a short time period and let the courts give the decisions. At times, the lawyers need to search the past cases to get the references and understand the nature of the new case in light of already fought case.

Moreover, it has been seen that there are many cases that are yet pending for a long period of time and the courts have not yet given their decision. The system that we're building will help the courts to decide by looking at the past cases. Newbie lawyers don't have that experience to understand the nature of the cases. And they need to look for the similar cases and their decisions. Our system is helpful to them.

Impact of Legal Document Search Engine on Society:

When the lawyers will use the Legal Document Search Engine in their cases, this will help the court to give decisions in the proper and shorter period of time. Hence, the culprits will be punished on time. And people will get lesson and deterrent of the offence. Thereby, reducing the guilt and society evils from the society.

Impact of Legal Document Search Engine on Lawyers:

New lawyers need to fight many cases to get the experience and ask to the other lawyers about the new cases they receive which takes a lot of time. Therefore, through our system, they'll be able to learn about particular case by giving just specific terms. And our system will present them the relevant past cases. This will help them understand the cases by their own and make references in the future cases.

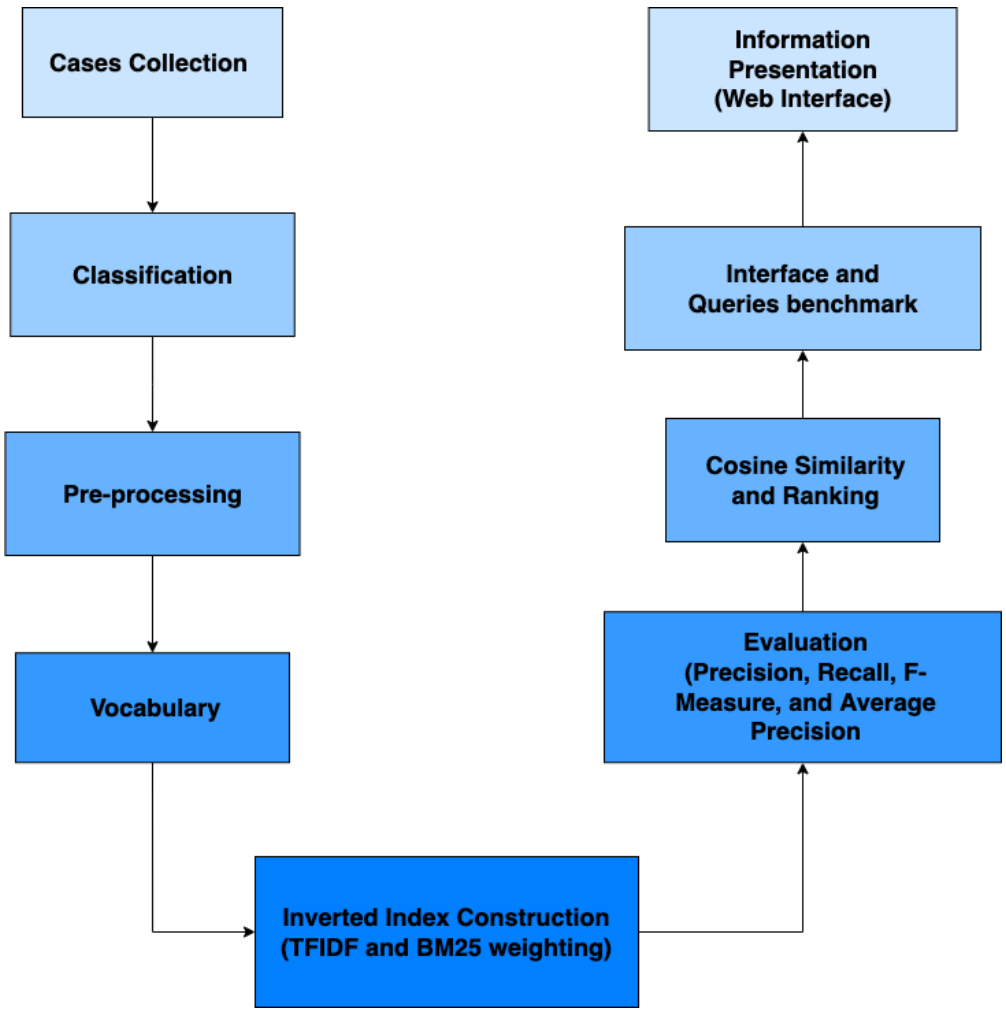
Chapter 2 (Methodology)

This chapter includes all the methods applied to each module of the system. In each module, we used different methodologies relevant to them to make them working.

To implement all modules we used **Python Programming Language**

Methodologies for different modules are given below:

Block Diagram:



Module 1:

1) For corpus of **1000** documents, we scraped website https://www.supremecourt.gov.pk/ and stored all the computer storage in **8** different categories
2) For abstract extraction, we developed an algorithm to read all the cases of each category and extract particular abstract area using **regular expressions**

Module 2:

1) For vocabulary extraction, we developed an algorithm to read all the cases of each category. After reading, we made valid token of all the text after preprocessing (tokenization, removing punctuation marks and less than three character words, normalization, stemming, lemmatization)
2) For document indexing, we developed an algorithm which assigns a unique Id number to each document in sequence manner

Module 3:

1) For raw term frequency, we developed an algorithm to count occurrence of each term in vocabulary for each document

2) Using raw term frequency in first part, we calculated log frequency weighting of each term in vocabulary for each document using formula:

$$w_{t,d} = 1 + \log10\ tf_{t,d} \quad \text{if} \quad tf_{t,d} > 0$$
$$w_{t,d} = 0 \quad \text{if} \quad tf_{t,d} = 0$$

3) For inverse document frequency weighting, we calculated document frequency of each term and the calculated idf weighting of each term in vocabulary for each document using formula:

$$idf = \log_{10} (N / df)$$

N is total number of documents in collection

4) Using log frequency and inverse document frequency, we calculated tf-idf weighting of each term in vocabulary for each document using formula:

$$tf\text{-}idf = w_{t,d} * idf$$

5) For BM25 weighting, we used raw term frequency and document frequency of vocabulary and calculated bm25 weighting of each term in vocabulary for each document using formula:

$$BM25 = c(w,q) * (((k+1) * c(w,d)) / c(w,d) + k) * \log((M+1)/df(w))$$

Module 4:

1) For top10 TFIDF terms, we sum up add values of tf-idf for each term and then extracted 10 top weighted terms

2) For top10 Bm25 terms, we sum up add values of bm25 for each term and then extracted 10 top weighted terms

3) By using top 10 terms of both schemes, we made different 5 random queries having Two/Three/Four Words

4) For interface, we used **flask module** of python and developed a simple web page like Google having a search bar and button for entering query.

Module 5:

1) For cosine similarity score using TF-IDF scheme, we used tf-idf values calculated in module **3**, for terms in both query and document and calculated cosine similarity score
2) For cosine similarity score using BM25 scheme, we used bm25 values calculated in module **3**, for terms in both query and document and calculated cosine similarity score
3) To rank them, documents, scores list should be sorted in descending order and top **10** documents will be displayed

Module 6:

1) For all the queries, ranked in module **6,** we evaluated our search engine using precision, recall, f-measure, average precision and mean average precision using their formulas

Formulas:

Precision = Relevant Retrieved / Total Retrieved

Recall = Relevant Retrieved / Total Relevant

F-Measure = (2 * Precision * Recall) / (Precision + Recall)

Average Precision = Sum of precision / Total Relevant

Mean Average Precision = Sum of Average Precision / Total Queries

Module 7:

1) By using interface developed in module 4 using **flask**, we integrated that interface with other modules to perform ranking operations. We let user to enter desired query and show him top 10 relevant documents on page 1 and next 10 relevant on other pages.
2) Word cloud is displayed on each page using terns in relevant documents. For this we used modules of python like **matplotlib, wordcloud etc**

Chapter 3 (Dataset, Experimental Setup, Results)

Dataset:

To make this assignment, we needed a dataset containing judgment cases of different categories scraped from https://www.supremecourt.gov.pk.

For this purpose we used:

- 1000 cases
- 8 Categories
- Categories includes:
    - Civil Appeals
    - Civil Petetion
    - Criminal Cases
    - Family Cases
    - Human Rights
    - Murder Cases
    - Property Cases
    - Suo Moto

Experimental Setup:

Experimental setup includes different feature of python programming and perform different NLP steps on dataset. These steps include:

- Categrization
- Pre-Processing
- Vocabulary Extraction
- Calculating Raw Term Frequency
- Calculating IDF
- Calculating TF-IDF
- Calculating BM25
- Calculating cosine similarity score
- Web Interface

System Evaluation:

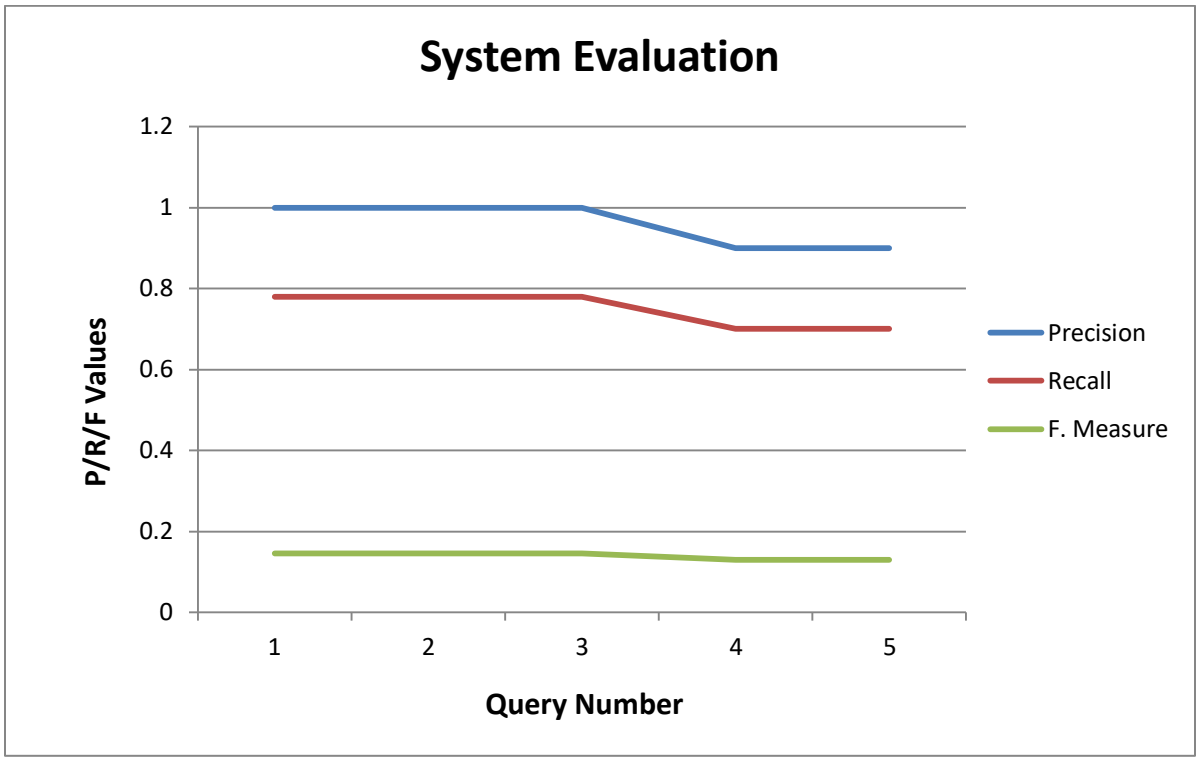
System Evaluation consists of readings of different measurements like Precision, Recall and F-Measure. It also contains Average Precision and Mean Average Precision calculations.

Readings for TF-IDF Scheme:

Query Terms Weighting	P	R	F	AP
QueryTerm-1 TFIDF	1	0.78	0.145	0.007
QueryTerm-2 TFIDF	1	0.78	0.145	0.007
QueryTerm-3 TFIDF	1	0.78	0.145	0.007
QueryTerm-4 TFIDF	0.9	0.70	0.13	0.006
QueryTerm-5 TFIDF	0.9	0.70	0.13	0.006

Mean Average Precision for 5 queries = 0.0066

Analysis:

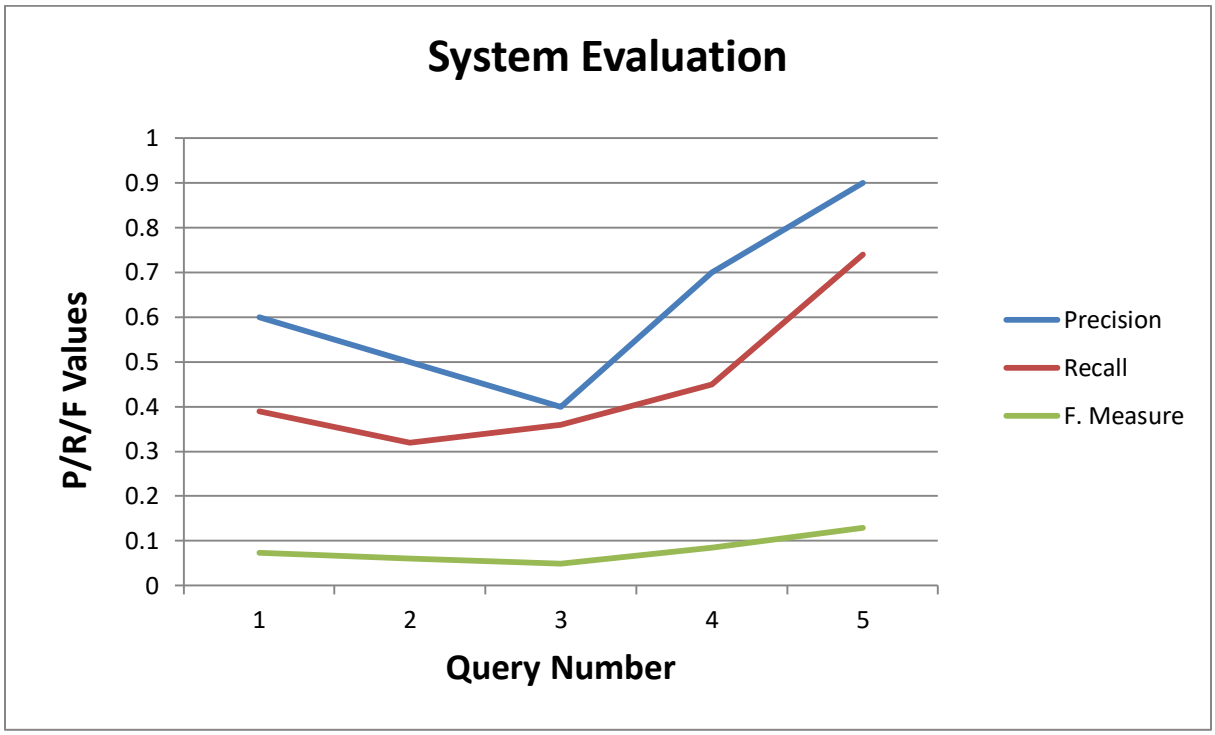


Readings for BM25 Scheme:

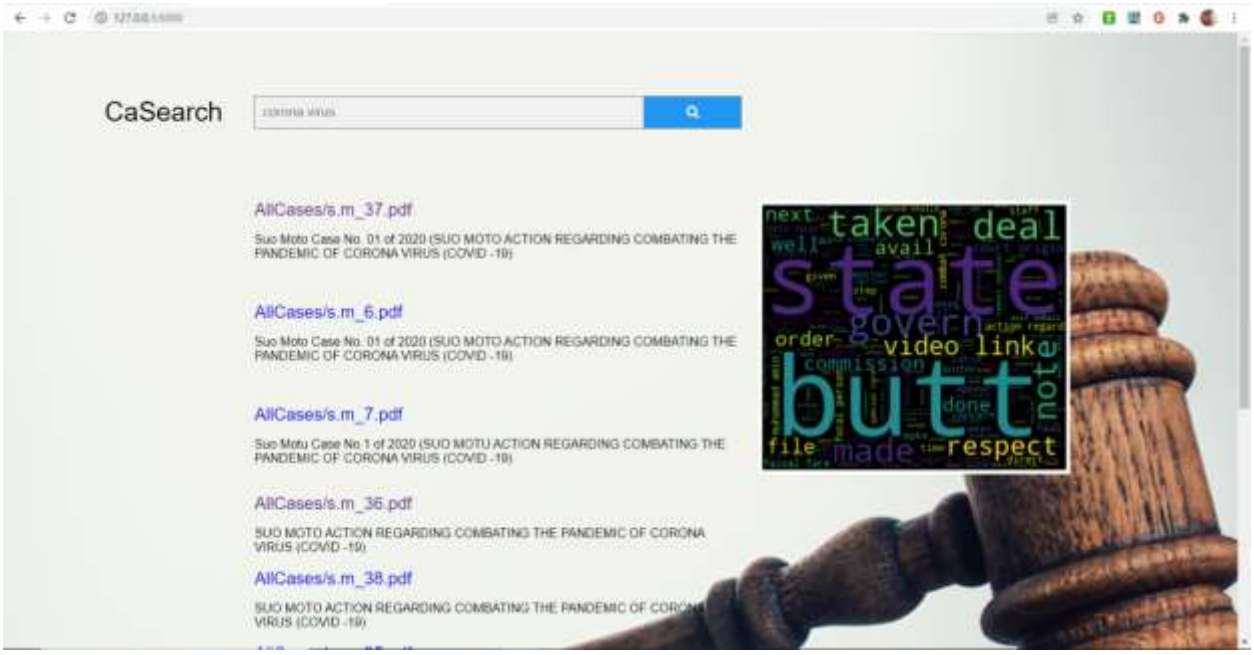
Query Terms	Weighting	P	R	F	AP
QueryTerm-1	BM25	0.6	0.039	0.073	0.003
QueryTerm-2	BM25	0.5	0.032	0.06	0.003
QueryTerm-3	BM25	0.4	0.026	0.049	0.002
QueryTerm-4	BM25	0.7	0.045	0.085	0.004
QueryTerm-5	BM25	0.5	0.074	0.129	0.007

Mean Average Precision for 5 queries = 0.003

Analysis:


System Evaluation
P/R/F Values
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
Query Number
Precision
Recall
F. Measure

System Interface:



Chapter 4 (Conclusion, Future Enhancement)

Conclusion:

We are successfully able to gather a significant number of documents (1000) and classified them into 8 classes on the basis of their nature. And, we have successfully pre-processed each document's text, generated a vocabulary, calculated the tf-idf, precision, recall, and bm-25 weighting of each term, and also created a search engine through which we give a query, and the engine retrieves the relevant/important documents on the basis of the given query and display us in the form a list according to their ranks.

Our CaSearch Engine is working on the basis of bm-25 weighting scheme. Since, this weighting scheme gave us better efficiency in terms of relevancy of the retrieved documents.

Now, lawyers or anyone related to judiciary can search important cases by giving a query to our system. And they will be returned the important documents on the top. This way, they can understand the already made decisions by the courts of Pakistan.

Future Enhancements:

Right now, our system is working only for the Pakistani courts decision. In the future, we can extend and enhance our system to collect the cases

from different countries having similar rules and laws, and we can add those documents into our collection of documents. We can classify those documents into their specific classes. This way, our system not only will search for cases in Pakistan but also will consider the cases from other countries.

Our system's user interface can be enhanced like it's of Google, which shows the links to a given query on the top that were visited most for a given query. This can be done through machine learning. Also, by understanding the nature of the lawyer by looking for the key terms that he/she searched in the past, we can show more relevant documents next time by using machine learning techniques.

Moreover, right now the collection is on our hard disks, we can store the document collection into the cloud database. Thus, can be accessed from anywhere.

The document classification that we did on the first step manually, can be done using the Principle Component Analysis or other clustering techniques and methods like K-Mean Clustering etc. This will save a lot of time and will automate the whole system.

References:

- **https://www.nlp-techniques.org/**
- **https://www.datacamp.com/community/tutorials/stemming-lemmatization-python**
- **Slides for Log Frequency, IDF, TF-IDF, BM25 Formulas**
- **Natural Language Processing with Python by Steven Bird, Ewan Klein, and Edward Loper, Printed in the United States of America, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2009.**