

# **LEGAL DOCUMENT SEARCH ENGINE**

**MUHAMMAD WAJEEH UZ ZAMAN**

**03071913002**

**DEPARTMENT OF SCIENCE  
QUAID E AZAM UNIVERSITY ISLAMABAD  
PAKISTAN  
2020**



## CONTENTS

	Page
<b>CHAPTER 1 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem statement	1
1.2 Motivation	1
1.3 Impact of legal documents search engine on society	2
<b>CHAPTER 2 METHODOLOGY</b>	<b>3</b>
2.1 Downloading corpus	3
2.2 Categorization of corpus (benchmarking of corpus)	4
2.3 Content extraction	5
2.4 Document indexing	5
2.5 Abstract extraction	6
2.6 Word extraction	7
2.7 Stemming/Lemmatization	7
2.8 Duplicate removal	7
2.9 Word indexing	7
2.10 Word TFIDF and BM25 calculation	8
2.11 Top 10 TF IDF and BM 25 term extraction	10
2.12 Query generation (Query benchmark)	10
2.13 Template generation	11
2.14 Linking template with Back end	11
2.15 Precision recall F measure and mean average precision calculation	12
<b>CHAPTER 3 RESULTS AND DISCUSSIONS</b>	<b>13</b>
<b>CHAPTER 4 CONCLUSION</b>	<b>15</b>



## **CHAPTER 1**

### **INTRODUCTION**

We are living in such a society where seeking justice is not that easy. One of the biggest issues with our justice system is that magistrates and advocates do not have background knowledge with the judgments. Due to this reason, magistrates do not make correct decisions and advocates do not defend their cases properly. This leads to taking an innocent life without a reason. To address this issue, we decided to must make a search engine that allows magistrates and advocates and every one that is associated with the court system to search previous judgments. This will help advocates to find the relevant judgments regarding to their case so they can fight and defend their cases in the court easily. With this, they will be able to defend their current case on the basis of previous decisions done on similar cases. We have created a search engine that takes queries from the users and return relevant judgments to their query. We have used two mechanisms to retrieve legal documents. We used TF IDF and BM 25 algorithms to retrieve over judgments and present to the user. We also created word cloud relative to the judgments to give pictorial view of the judgments retrieved. At the end we evaluated both TF IDF and BM25 algorithms with precision and recall measures. We came to know that BM 25 gives better retrieved legal documents against our user query.

#### **1.1 PROBLEM STATEMENT**

To present advocates and lawyers a search engine that help in searching and retrieval of relevant legal documents against their query.

#### **1.2 MOTIVATION**

We heard several decisions made by magistrates in the court on television. We felt sad, as honorable magistrates release the culprits even having strong evidences. The reason behind this is that our over lawyers and advocates are so poor in

defending cases and providing evidences. Our lawyers and advocates have least knowledge about the decisions of past similar cases. Our goal is to link the lawyers with past judgments and decisions so that they can defend their cases in court, saving innocent lives and get a better position in the eyes of society.

### **1.3 IMPACT OF LEGAL DOCUMENTS SEARCH ENGINE ON SOCIETY**

It will increase the knowledge of lawyers about cases and the decisions made by judges. It will help the lawyers to get connected with all the past judgments and decisions. The lawyers can make the base of previous judgments to win new cases and persuade magistrates towards the correct decision. This will help culprits to their last destination and innocent to peace. This will improve the respect of decisions made by courts in eyes of society. Society tend to believe on courts and have faith on both lawyers and magistrates that they provide them their rights and full justice.

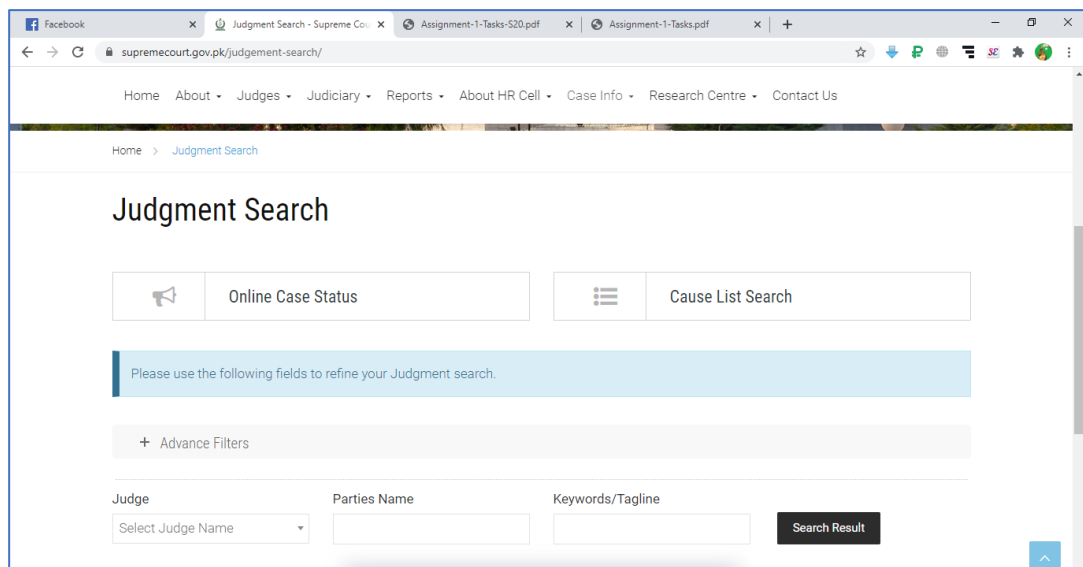
## CHAPTER 2

### METHODOLOGY

This section summarizes how we go through steps to get our work done. It will conclude all steps from downloading corpus to evaluating results. At the end, we evaluated both searching mechanisms and came to know that BM25 mechanism is a better search judgement retrieval mechanism than TF-IDF. We have used 12 categories for case categories with 146 legal documents in those categories in total. We have used python version 3.6.8 x64 bit with flask web technology to get our work done.

#### 2.1 DOWNLOADING CORPUS













First of all, we decided to download all the past judgments through Supreme Court and other court websites of Pakistan. After downloading those judgments, we placed Those files into the correct categories.



The screenshot shows a web browser window with multiple tabs. The active tab is 'Judgment Search - Supreme Court of Pakistan'. The address bar shows 'supremecourt.gov.pk/judgement-search/'. The website has a navigation menu with links: Home, About, Judges, Judiciary, Reports, About HR Cell, Case Info, Research Centre, and Contact Us. Below the navigation menu, there is a 'Judgment Search' section. It contains two buttons: 'Online Case Status' and 'Cause List Search'. Below these buttons, there is a light blue box with the text 'Please use the following fields to refine your Judgment search.' Below this box, there is a button labeled '+ Advance Filters'. At the bottom, there are three input fields: 'Judge' (with a dropdown menu showing 'Select Judge Name'), 'Parties Name', and 'Keywords/Tagline'. To the right of these fields is a black button labeled 'Search Result'.

## 2.2 CATEGORIZATION OF CORPUS (BENCHMARKING OF CORPUS)

We used artificial and manual methods to place files in the relevant categories. We made 12 categories. The categories are; Civil appeal, Civil miscellaneous appeal, civil petition, Constitution petition, criminal appeal, criminal miscellaneous appeal, criminal original petition, criminal petition, human right, original jurisdiction, Suo Moto case respectively. We got 41, 4, 34, 9, 13, 3, 3, 2, 9, 2, 16, 10 judgments in those categories respectively. We got 146 judgments in total after removing outliers and Urdu judgments. We preferred to work on only English judgments. There are 12 categories in total.

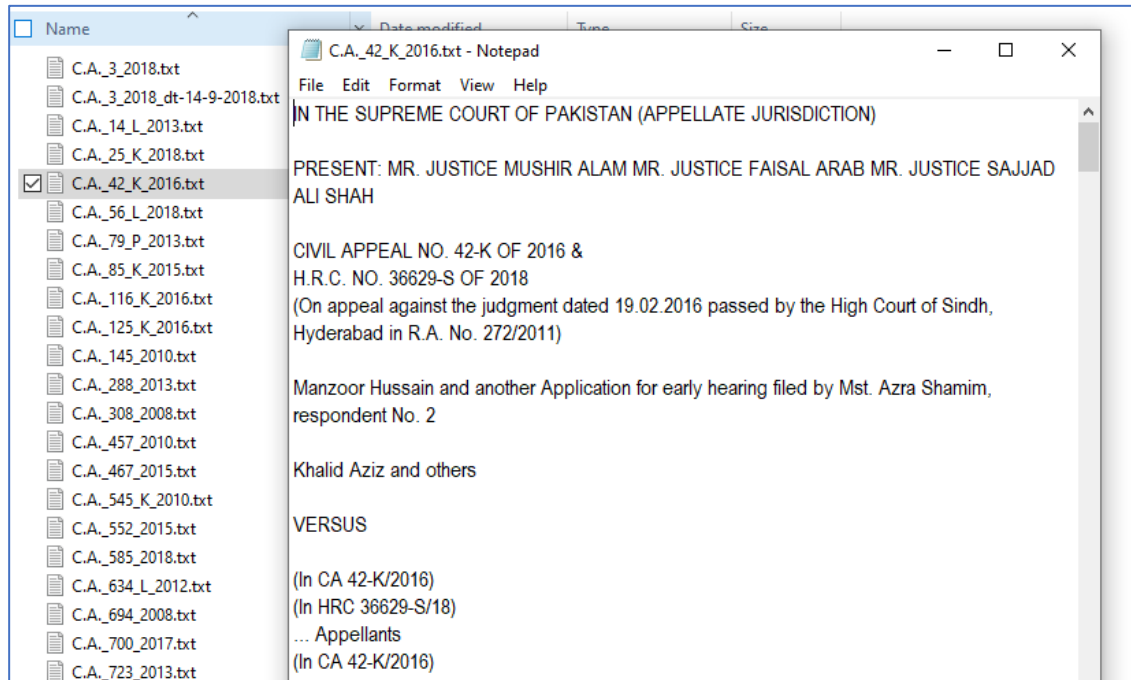
 civil Appeal	19/06/2020 10:03 PM	File folder
 Civil Misc Application	19/06/2020 10:02 PM	File folder
<input type="checkbox"/>  Civil Petition	19/06/2020 10:00 PM	File folder
 Constitution Petition	19/06/2020 10:00 PM	File folder
 Criminal Appeal	19/06/2020 10:01 PM	File folder
 Criminal Mislineous Application	19/06/2020 9:59 PM	File folder
 Criminal Orignal Pition	19/06/2020 9:59 PM	File folder
 Criminal Petition	19/06/2020 9:59 PM	File folder
 Human Right	19/06/2020 10:02 PM	File folder
 Jail Petition	19/06/2020 9:58 PM	File folder
 Original Jurisdiction	19/06/2020 10:01 PM	File folder
 Suo Moto Case	19/06/2020 9:58 PM	File folder

Category / Class	Name	Number of Judgments
civil Appeal		41
Civil Misc Application		4
Civil Petition		34
Constitution Petition		9
Criminal Appeal		13
Criminal Mislineous Application		3
Criminal Orignal Pition		3
Criminal Petition		2
Human Right		9
Jail Petition		2
Original Jurisdiction		16
Suo Moto Case		10



## 2.3 CONTENT EXTRACTION

The judgments are portable document (PDF) files. after downloading and categorization of judgments, the content of those judgments' extraction is necessary for future steps. So, we accepted text from all PDF judgments and save in TXT files. We use direct PDF reading and optical character recognition (OCR) for text extraction from judgments.



## 2.4 DOCUMENT INDEXING

we have used Python programming language to index documents it is very important step in information retrieval for creating TF IDF and BM 25 ranking mechanisms. We assigned ID with every particular document name. While indexing we have also extracted abstracts from those documents. The abstracts are the first 20 lines of each document. We save those abstracts separately in separate file.

casedocuments.txt - Notepad	
File	Edit Format View Help
1	C.A.401_2011.txt
2	C.A._1042_2018.txt
3	C.A._1083_2017.txt
4	C.A._116_K_2016.txt
5	C.A._1171_2017.txt
6	C.A._1178_2008.txt
7	C.A._1189_2017.txt
8	C.A._1203_2014.txt
9	C.A._125_K_2016.txt
10	C.A._1338_2007.txt
11	C.A._1340_2018.txt
12	C.A._1459_2018.txt
13	C.A._145_2010.txt
14	C.A._14_L_2013.txt
15	C.A._1509_2016.txt

2.5 ABSTRACT EXTRACTION

While extracting and indexing documents, we also extracted first 20 lines from each file as abstracts. These abstracts will help in showing and ranking files in front end search engine.

abstracts.txt - Notepad		
File	Edit Format View Help	
Category / Class	Name S.No.	Abstract of Judgments
civil Appeal	1	MR. JUSTICE MUNIB AKHTAR
CIVIL APPEAL NO. 401 OF 2011.		
(On appeal from judgment dated 2.11.2010		
Lahore High Court,		
passed by		
in		
Bahawalpur		
Bahawalpur		
C.R.No.24 of 1993.)		
Muhammad Sadiq and others		
Bench,		
the		
VERSUS		
...Appellant (s)		
...Respondent (s)		

## **2.6 WORD EXTRACTION**

After indexing of documents, we extracted words from those documents. We read every file, read every line of file, separate the words with the help of spaces, added those words into an array.

## **2.7 STEMMING/LEMMATIZATION**

Stemming and lemmatization are the important steps of information retrieval system. We have applied stemming and lemmatization on an array that we have retrieved from word extraction. Stemming and lemmatization convert all words into their root form. We can say that we have applied a word normalization. We saved those stemmed and lemmatized words in a new array.

## **2.8 DUPLICATE REMOVAL**

As the words are in their root form, there is need to remove the duplicated word. So, we removed the duplicated words from stemmed and lemmatized array.

## **2.9 WORD INDEXING**

Now it is necessity to uniquely identify the words and saving in into some text file. So, we saved that stemmed and lemmatized array into a file.



## 2.10 WORD TFIDF AND BM25 CALCULATION

After saving stemmed and lemmatized words into an array and file, there is need to calculate the TFIDF and BM25 words of those words. So, we calculated separately both TF IDF and BM 25 weights of words.

First, we calculate raw term frequency for each term each document. And saved it into a separate file. After that we have calculated log term frequency and inverse document frequency and save those into separate file for each term and each document. After that we have calculated TF IDF by multiplying LTF with IDF from files and saved TF IDF weights for every term for each document in separate file. Non zero entries were ignored.

```
tfidf.txt - Notepad
File Edit Format View Help
1 145:3.196
2 44:2.164
3 38:2.164
4 29:2.181 37:1.146 40:1.146 56:1.693 74:1.693 77:1.693 113:1.146 118:1.146 133:1.146 136:1.947
5 136:3.467
6 35:2.164
7 96:2.298 132:1.556 137:1.556 145:1.556
8 96:2.483 137:1.681 145:1.681
9 145:2.164
10 122:3.196
11 122:2.164
12 122:2.164
13 30:1.556 35:1.556 61:2.024 84:2.024
14 23:1.114 24:1.114 25:1.114 39:1.114 41:1.114 85:1.114 95:1.114 105:1.114 133:1.114 139:1.114 140:1.114
15 18:2.164
16 76:2.815
17 45:2.164
18 125:1.863 130:1.863
19 122:2.164
20 1:0 2:0 3:0 5:0 6:0 7:0 8:0 9:0 10:0 11:0 12:0 13:0 14:0 15:0 16:0 17:0 18:0 19:0 21:0 22:0 23:0 24:0 25:0 26:0 27:0 28:0 29:0
51:0 52:0 53:0 54:0 55:0 56:0 57:0 58:0 59:0 60:0 61:0 62:0 63:0 65:0 66:0 67:0 68:0 69:0 70:0 72:0 73:0 74:0 75:0 76:0 77:0 78:0
99:0 100:0 101:0 102:0 103:0 104:0 105:0 106:0 107:0 108:0 109:0 110:0 111:0 112:0 113:0 114:0 115:0 116:0 117:0 118:0 119:0
139:0 140:0 141:0 142:0 144:0 145:0 146:0
21 3:1.556 28:1.556 62:1.556 118:3.58
22 59:2.815
```

Secondly, we calculated BM 25, we used 1.75 the value of K and 0.65 the value of b. And also used LTF and IDF weights for each term for each document and save the values of BM 25 for each term and for each document in a separate file.

```
bm25.txt - Notepad
File Edit Format View Help
1 145:4.403
2 44:2.935
3 38:2.935
4 29:5.218 37:2.935 40:2.935 56:4.403 74:4.403 77:4.403 113:2.935 118:2.935 133:2.935
136:4.892
5 136:4.696
6 35:2.935
7 96:4.403 132:2.935 137:2.935 145:2.935
8 96:4.403 137:2.935 145:2.935
9 145:2.935
10 122:4.403
11 122:2.935
12 122:2.935
13 30:2.935 35:2.935 61:3.914 84:3.914
14 23:2.935 24:2.935 25:2.935 39:2.935 41:2.935 85:2.935 95:2.935 105:2.935 133:2.935
139:2.935 140:2.935
```

## 2.11 TOP 10 TF IDF AND BM 25 TERM EXTRACTION

After calculating the TF IDF BM 25 weights of each term, we summed up all TF IDF weights and BM 25 weights for each term in separate file. Then We arranged both TF IDF and BM 25 weights in descending order and save those in separate file.

TFIDF	
con	cons, constitution, constitutional, constable
sta	stable, stadium, stack, stain, stamp, stand, standalone
pro	procure, procastination. process, proclaim proceedings
court	high court, supreme court
men	mention, mental, menial, menace
ear	earth, earn, early
pre	preacher precaution preceedings, precise, prediction, preferable, preface, prejudic
thi	thicket thick thief thi thir think
per	permit personal premission prejure premissable permit periop perform percolation perfect persistence
judg	judge judging judged judgement
BM25	
resent	resented resentment
present	presentation presenting presented presents presented
hear	hearing heard hearer heard
justic	justice justiceable justicability
date	dated datewise dating dated
sent	sent sentiment sentence sentencing
jurisdict	jurisdiction jurisdictional jurisdictions
juri	jury jurors juror juristic jurisprudence jurisdiction
pass	pass passages passenger passcode passed passedout passcode passes passion passenger passengers
suprem	suprema supremaci supreme

## 2.12 QUERY GENERATION (QUERY BENCHMARK)

After summing up TF IDF BM 25 weights and arranging them into descending order, we extracted top 10 TF IDF and BM 25 weights terms. We combined those terms to create our queries.

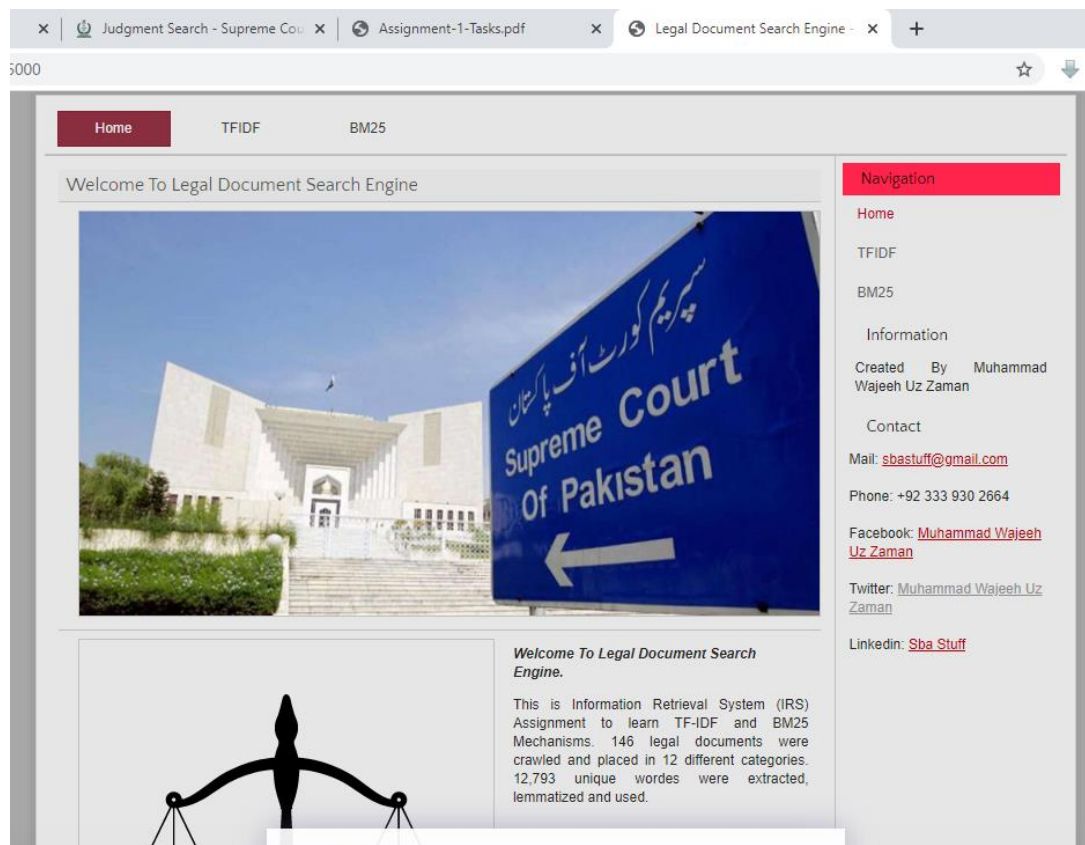
Query 1: judge heard earth cases  
Query 2: judges as jurist  
Query 3: supreme court hearers  
Query 4: represent court  
Query 5: supreme court mental cases

## 2.13 TEMPLATE GENERATION

We use HTML and CSS to create front end for users to make queries. Right now, this template is dummy it contains the search box and a button. But typing anything in the search box and hitting search a button did nothing.

## 2.14 LINKING TEMPLATE WITH BACK END

After template generation we worked on backend. We make algorithms of cosine similarity to such query words against documents and return relevant results. We did this for both TF IDF and BM 25 terms. Top 100 documents were returned for both TF IDF and BM 25 weights. We have saved the ranked documents for each query for both TF IDF and BM 25 terms of query into a separate file for later calculation of precision recall f measure and mean average precision. We also showed a word cloud on the left form abstracts of documents retrieved.



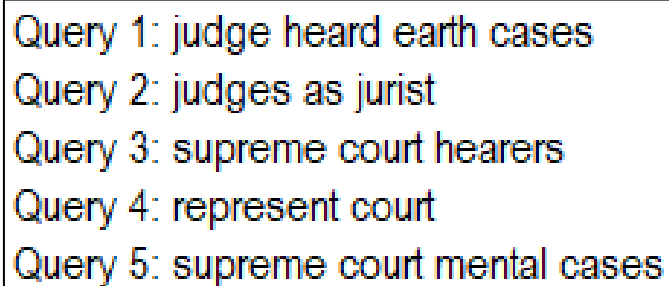




## CHAPTER 3

### RESULTS AND DISCUSSIONS

We are living in such society where seeking for justice is not an easy task. There are many issues to our justice system. One of the biggest issues faced by our courts that magistrates and advocates do not have proper knowledge about past judgements to listen and defend cases. Due to these issues, magistrates' releases culprits and advocates even on the right side lose their cases. We decided to resolve this issue by linking magistrates and advocates to the previous judgements through legal judgement search engine. We used two searching mechanisms to retrieve judgements. The two popular methods are called BM25 and TF-IDF. At the end, we evaluated both searching mechanisms and came to know that BM25 mechanism is a better search judgement retrieval mechanism than TF-IDF. We have used 12 categories for case categories with 146 legal documents in those categories in total. We have used python version 3.6.8 x64 bit with flask web technology to get our work done. We searched query terms online and extracted their precision, recalls, f-measure and mean average precision. These measures were calculated on the basis of top 100 results retrieved. There were 15,9,5,32,8 relevant documents against every query respectively. The query terms we decided to evaluate our system are shown in figure below.



Query 1: judge heard earth cases  
Query 2: judges as jurist  
Query 3: supreme court hearers  
Query 4: represent court  
Query 5: supreme court mental cases

TF-IDF is term scoring method, it multiplies log term frequencies of terms with inverse document frequency. It incorporates a document using cosine similarity. The results for all queries for TFIDF are shown below;

Query Terms	Weighting	P	R	F	AP
1	TFIDF	0.03	0.2	0.05	0.15
2	TFIDF	0.02	0.22	0.04	0.25
3	TFIDF	0.04	0.8	0.08	0.44
4	TFIDF	0.12	0.38	0.18	0.07
5	TFIDF	0.03	0.38	0.06	0.28
Mean Average Precision for 5 queries = 0.24					

Bm25 is more than a term scoring method. It also penalizes document length along with terms against a query to get better results.

Query Terms	Weighting	P	R	F	AP
1	BM25	0.09	0.16	0.12	0.26
2	BM25	0.06	0.6	0.11	0.43
3	BM25	0.05	1.0	0.1	0.77
4	BM25	0.26	0.81	0.39	0.12
5	BM25	0.06	0.75	0.11	0.48
Mean Average Precision for 5 queries = 0.41					

The precision, recall, f-measure, and mean average precision of BM25 for benchmark queries is more than precision, recall, f-measure, and mean average precision of TFIDF. Thus BM25 search results outclassed and better than TFIDF results.

## **CHAPTER 4**

### **CONCLUSION**

Information retrieval is an interesting field. It had developers to create applications for users so that the users may find what they are looking for. The relevant results displayed to users guarantees user satisfaction and happiness. Our developed site engine application helps magistrates and lawyers to study the previous cases to build up their knowledge so that they can defend and make decisions better in order to make their positions better in society and provide justice to both culprits and innocent people. Our searching an application has few flaws that cost to be addressed in future. Like it crashes when the user enters the wrong word. There is need to be providing users with valid keywords so that he may retrieve relevant documents for what he is looking for. Moreover 146 documents are not enough and as such must not be stopped. This mechanism must be applied every month by updating the corpus so that lawyers and magistrates build up with their knowledge on new decisions on every judgment. New algorithms must be tested to get better relevant results against the lawyers are magistrates queries. We can apply the same mechanism to other fields in order to improve them too. The mechanism of judgement scraping and assigning into categories in an automated manner can lower the burden on admins and developers and it keep application up to date without manual involvement. These search engines not only improve public but also private sectors too and help Pakistan grow in their respective fields.