**1.Test the performance of retrieval algorithm "RawTF" with two types of text data (i.e., raw text data and text data by stemming and removing stopwords)**

1.1-

After evaluating **RawTF** algorithm for two types of queries, I found following results:

**RawTF with No stemming and with stopwords-**

Total number of documents over all queries

Retrieved: 3000

Relevant: 442

Rel\_ret: **108**

Average precision:  **0.0449**

R-Precision: **0.0712**

**RawTF with Stemming and No stopwords-**

Total number of documents over all queries

Retrieved: 3000

Relevant: 442

Rel\_ret: **196**

Average precision: **0.1174**

R-Precision: **0.1404**

It is observed that RawTF algorithm with stemming and removing stopwords from the corpus improved the number of relevant retrieval documents than the RawTF performed on corpus with stopwords and without stemming. The precision is also better for stemmed and no stopwords corpus as found in above results.

Hence, the RawTF with stemming and no stopwords is better than RawTF performed on corpus with stopwords and without stemming. Stemming reduces the vocabulary size because of which the no of relevant document in retrieval get increased.

We used Porter stemmer for stemming the corpus. I tried couple of different porter and check the performance difference.

1. Paice porter-

When I used Paice porter for stemming the corpus and evaluate the retrieval, I found that the number of relevant documents were less than the porter stemmer. Also, the Precision found lesser.

Total number of documents over all queries

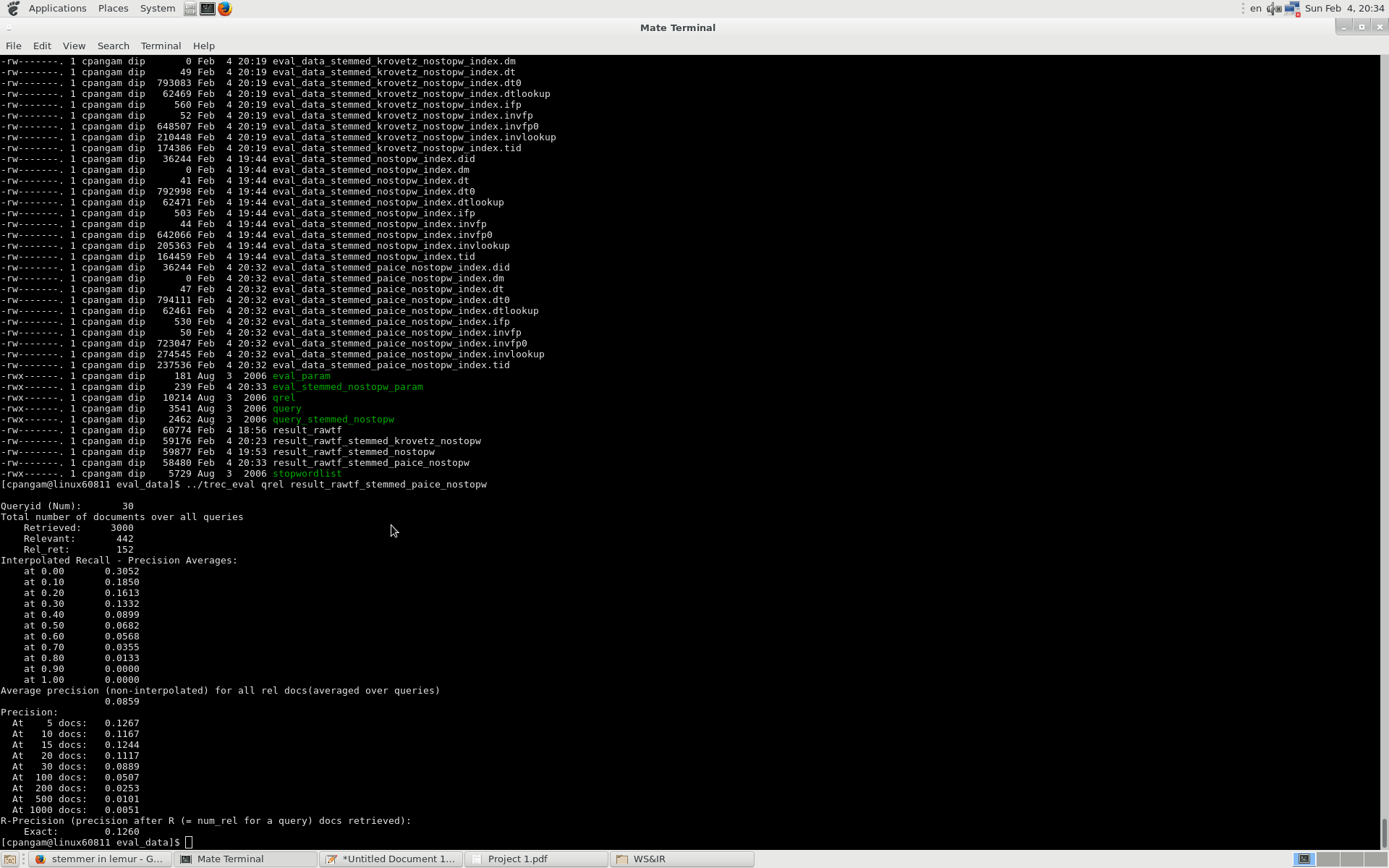
Retrieved: 3000

Relevant: 442

Rel\_ret: **152**

Average precision: **0.0859**

R-Precision: **0.1260**



1. Krovetz porter-

Similar to Paice stemmer, I tried another stemmer called Krovetz. I found that it also retrieved less number of documents and has less precision than Porter stemmer but it is better than Paice stemmer.

Total number of documents over all queries

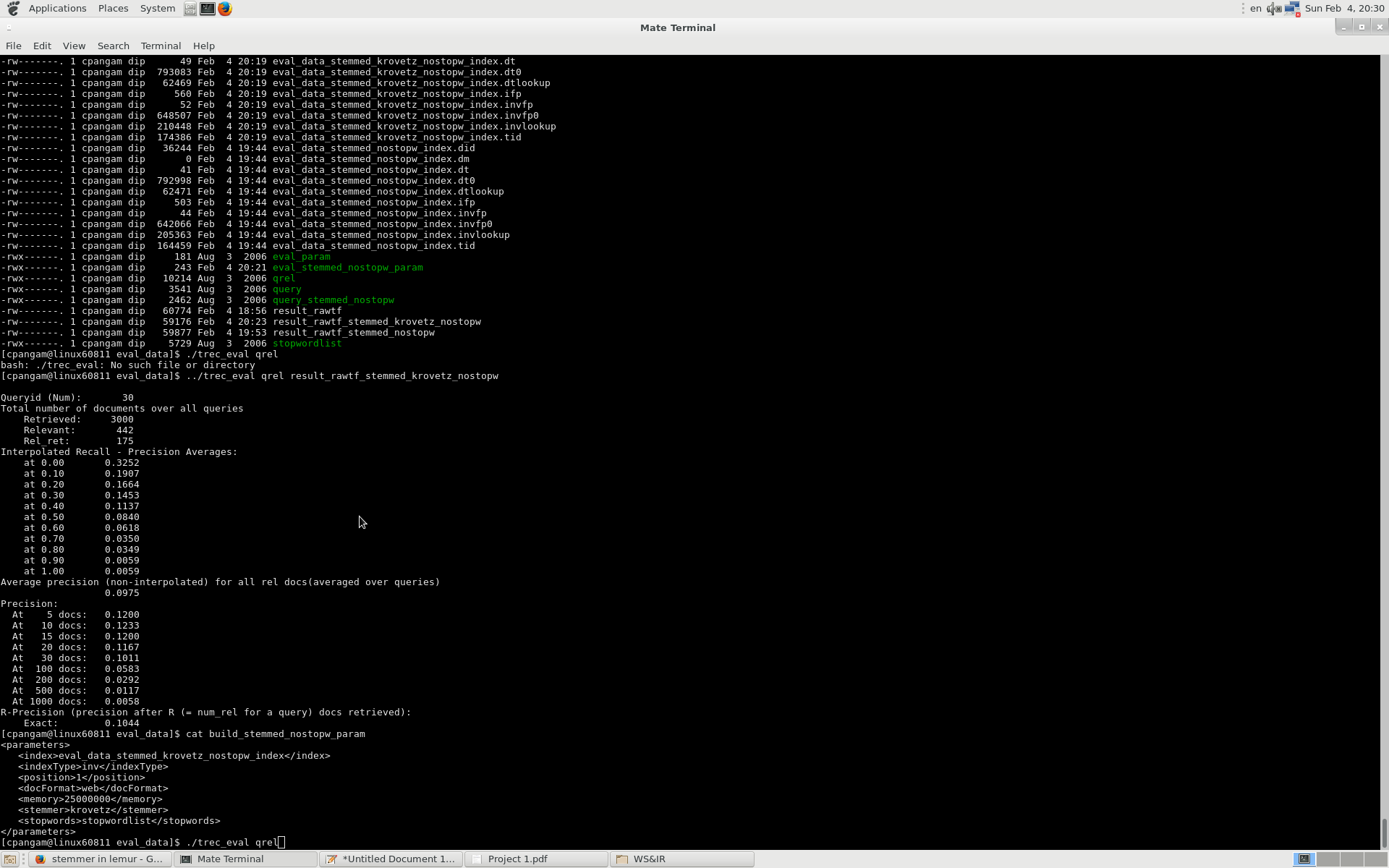
Retrieved: 3000

Relevant: 442

Rel\_ret: **175**

Average precision: **0.0975**

R-Precision: **0.1044**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr No | Stemmer | Relevant\_Retrieval | Avg Precision | R-Precision |
| 1 | Porter | 196 | 0.1174 | 0.1404 |
| 2 | Krovetz | 175 | 0.0975 | 0.1044 |
| 3 | Paice | 152 | 0.0859 | 0.1260 |

1.2- To remove stopword, we change the build parameter and add <Stopword> tag with stopwords file name. I evaluate the RawTF after removing stopwords and found following observations.

Total number of documents over all queries

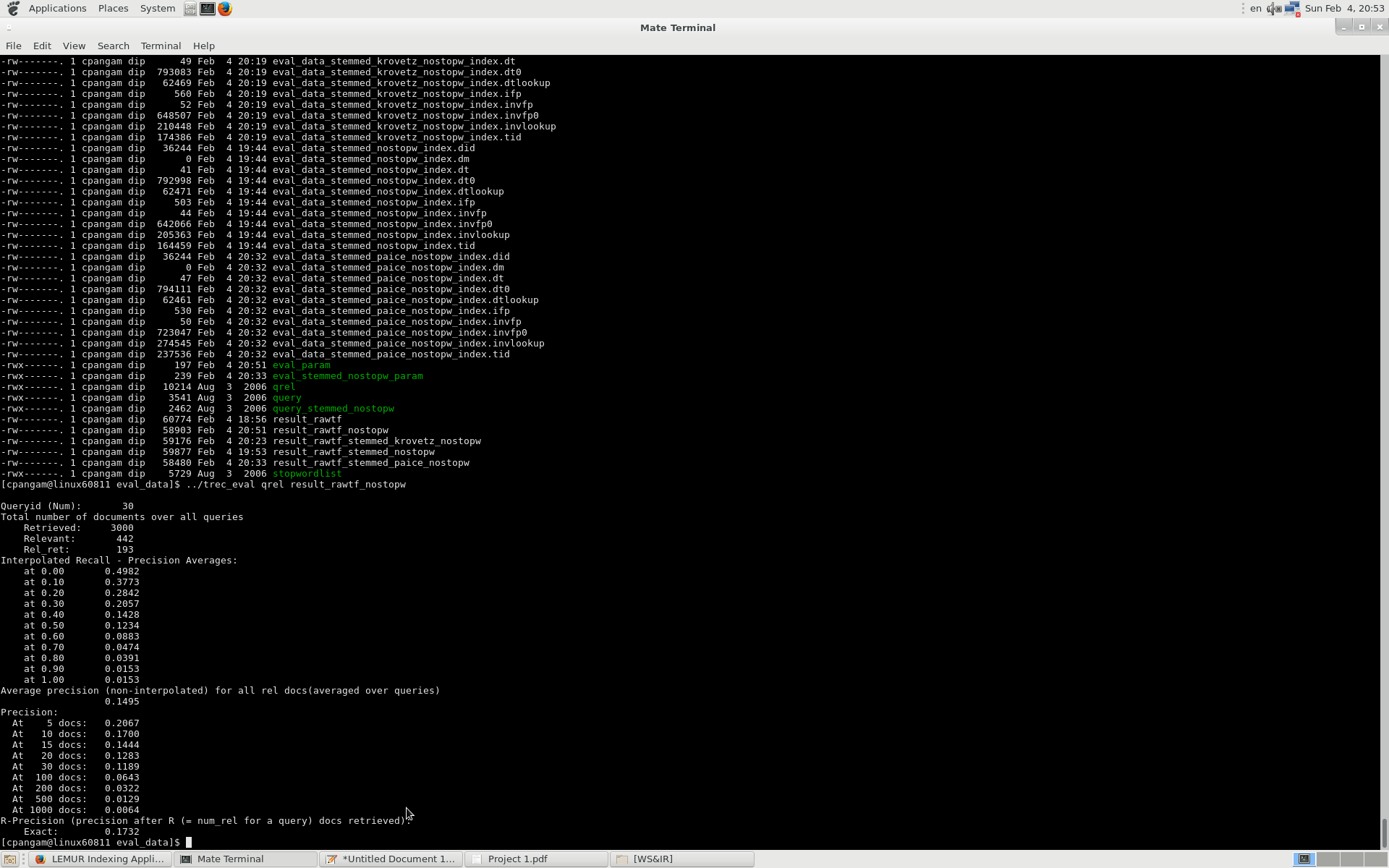
Retrieved: 3000

Relevant: 442

Rel\_ret: **193**

Average precision: **0.1495**

R-Precision: **0.1732**



It is observed that removing stopword improved the results for rawTF than with stopwords.

The number of relevant retrieval documents are more as well as the precision is also improved.

**2. Implement three different retrieval algorithms and evaluate their performance.**

I have implement other three algorithms. To check evaluation of each algorithm with different input combination, I modified building parameters as well as evaluation parameters accordingly.

Code change in- RetrievalEval.cpp

For No stemming-

Query file used – Query

Parameter files modified- Build\_Param , Eval\_Param

For Stemming-

Query file used – query\_stemmed\_nostopw

Parameter files modified- build\_stemmed\_nostopw\_param, eval\_stemmed\_nostopw\_param

After observing every evaluation, I found that Okapi algorithm works better than other three i.e. RawTF, TFIDF, and LogTFID. We can see that the precision for Okapi algorithm with stemmed corpus and no stopword is better than that of others.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr No | Algorithm | No Stemming  No Stopwords | No Stemming  With Stopwords | Stemming  No Stopwords | Stemming  With Stopwords |
| 1 | **RawTF**  Avg Precision  Recall  R-Precision | 0.1495  0.4366  0.1732 | 0.0449  0.2443  0.0712 | 0.1174  0.4434  0.1404 | 0.1183  0.4502  0.1404 |
| 2 | **RawTFIDF**  Avg Precision  Recall  R-Precision | 0.2170  0.5  0.2361 | 0.1861  0.4457  0.2147 | 0.2137  0.5542  0.2403 | 0.2137  0.5542  0.2451 |
| 3 | **LogTFIDF**  Avg Precision  Recall  R-Precision | 0.3119  0.5271  0.352 | 0.2687  0.466  0.2857 | 0.3241  0.5769  0.3744 | 0.3236  0.5769  0.3744 |
| 4 | **Okapi**  Avg Precision  Recall  R-Precision | 0.3126  0.5769  0.3314 | 0.3004  0.5588  0.3136 | 0.3584  0.6471  0.3813 | 0.3522  0.6584  0.3641 |