系級:資科碩一　姓名:葉冠宏　學號:108753208

WSM Project 2

Introduction

In this project, the main purpose is to compare the ranking accuracy of different evaluation methods. Here, we focus on Okapi TFIDF, MLE with Laplace smoothing, Jelinek-Mercer smoothing, and Two-stage Smoothing. I choose Lemur and Indri API to implement my experiment.

Method

1. Okapi TFIDF

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avgdl}}\right)},$$

```
<count>1000</count>
<baseline>tfidf,k1:2.0,b:0.75</baseline>
```

We set k1 as 2.0, b as 0.75 for the parameter. The function performs retrieval via tf.idf scoring as implemented in lemur::retrieval::TFIDFRetMethod using BM25TF term weighting.

2. MLE with Laplace smoothing

$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$
$$= \frac{c(x)+1}{N+|X|}$$

```
<count>1000</count>
<rule>method:dirichlet,mu:1</rule>
```

We set mu as 1 for the parameter to make the setting become Laplace (Add one) Smoothing.

3. Jelinek-Mercer smoothing

$$\rho_i = \lambda P + (1-\lambda)Q$$

```
<count>1000</count>
<rule>method:jm,collectionLambda:0.8</rule>
```

Λ is the weight attached to the background probability. Here we set λ as 0.8.

P is the estimated probability from document (max likelihood = m_i/n) and Q is the estimated probability from corpus (background probability = cf / terms in the corpus).

4. Two-stage Smoothing

```
<count>1000</count>
<rule>method:twostage,mu=1, lambda=0.8</rule>
```

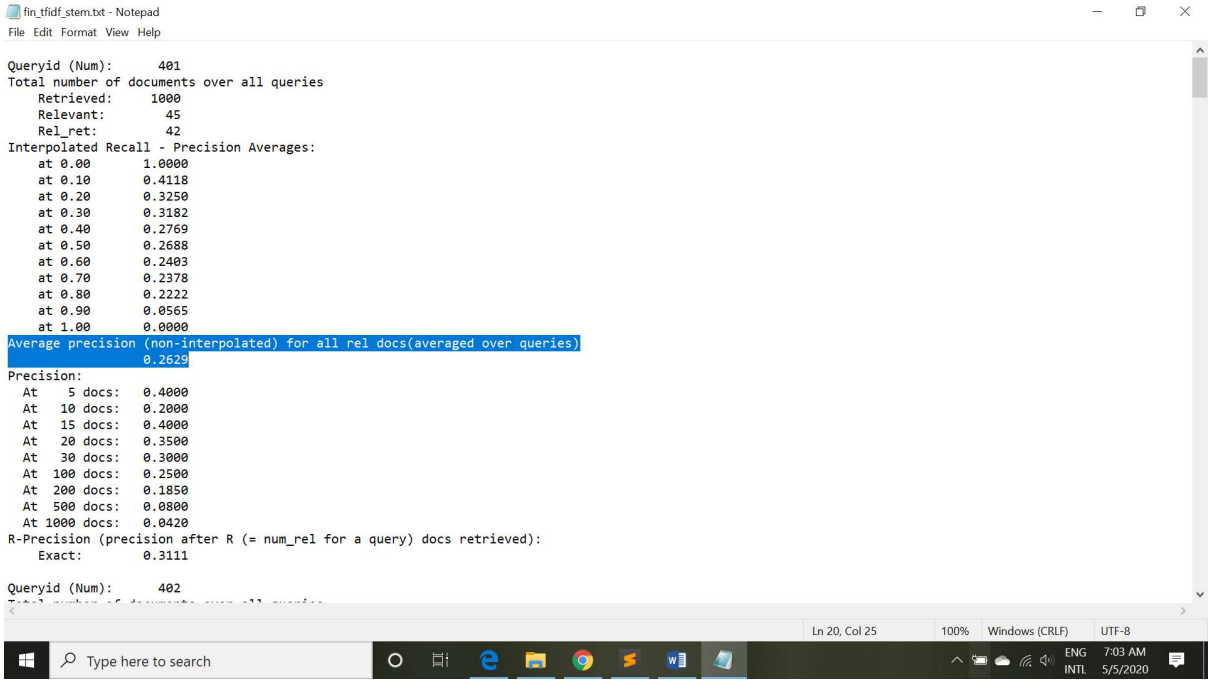
The optimal settings of retrieval parameters often depend on both the document collection and the query, and are usually found through empirical tuning. At the first stage, a document language model is smoothed using Dirichlet prior as the reference model. A the second stage, it is further smoothed using Jelinek-Mercer smoothing with a query background.

Experiment and Result

For the corpus, we construct two indexes, (a) with stemming, and (b) without stemming. Both indexes contain stopwords. For stemming, we use the porter stemmer algorithm to pre-process our datasets.

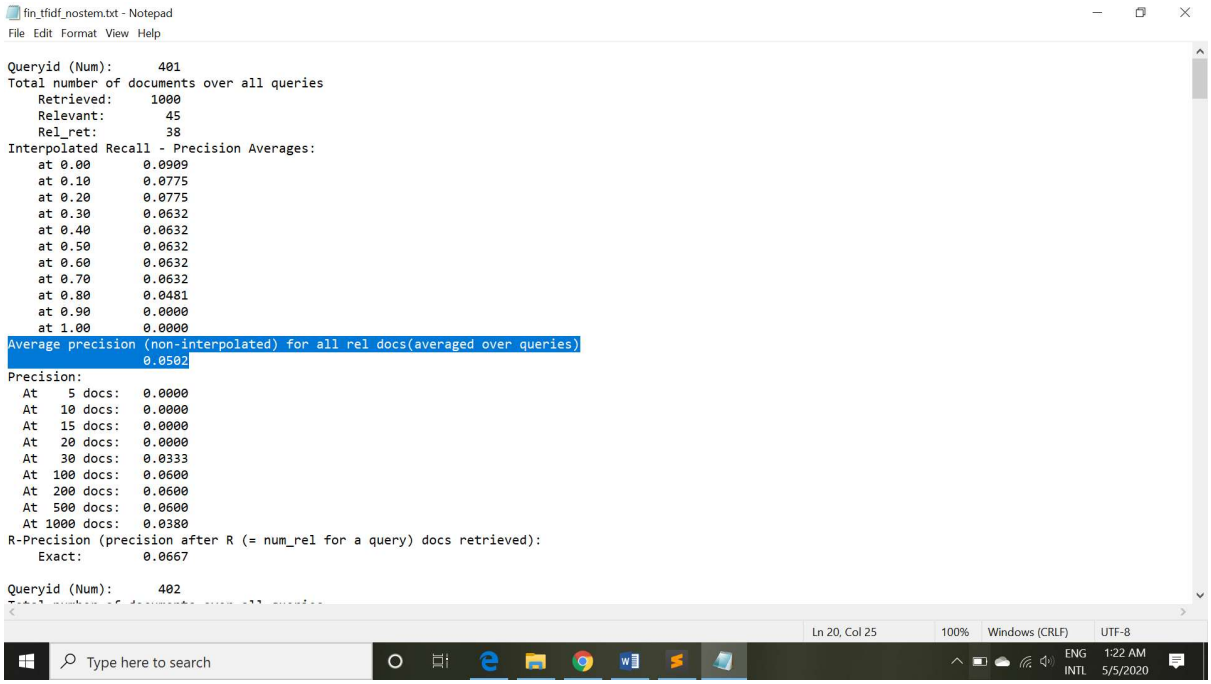
We run 50 queries and return at top 1,000 documents for each query. Here, we take query 401 as our example for illustration.

1. Okapi TFIDF(with stemming)



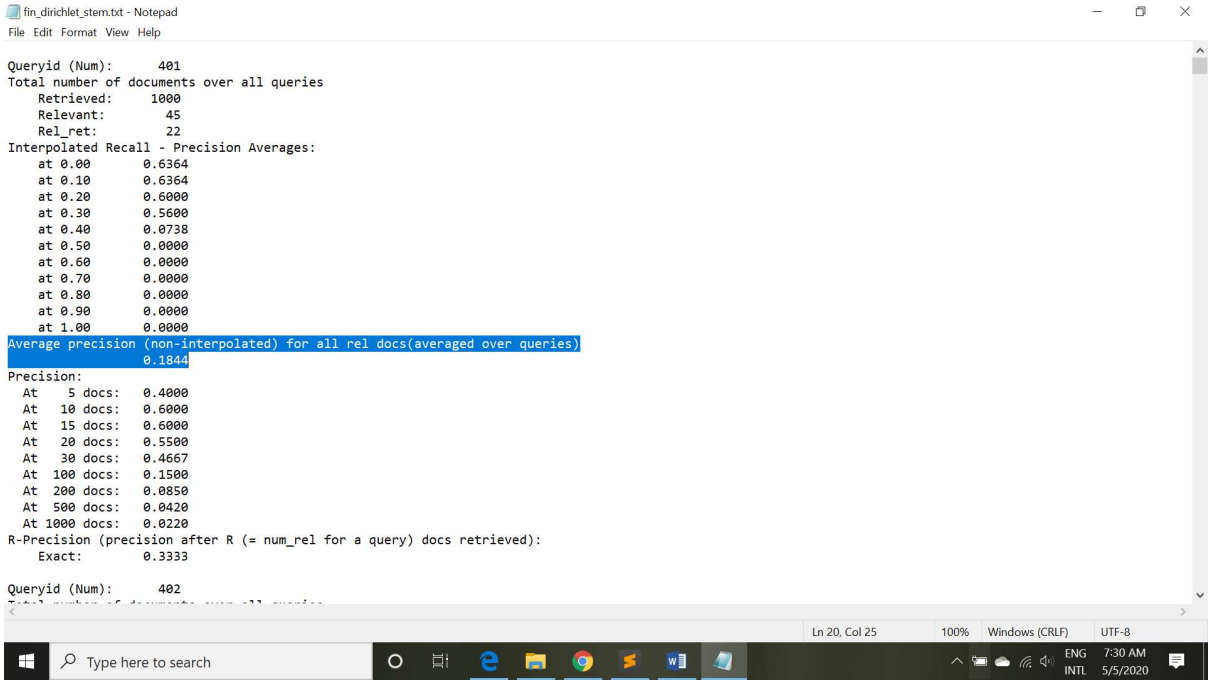
Average precision (non-interpolated) for all relevant docs(averaged over queries) is 0.2629.

2. Okapi TFIDF(without stemming)



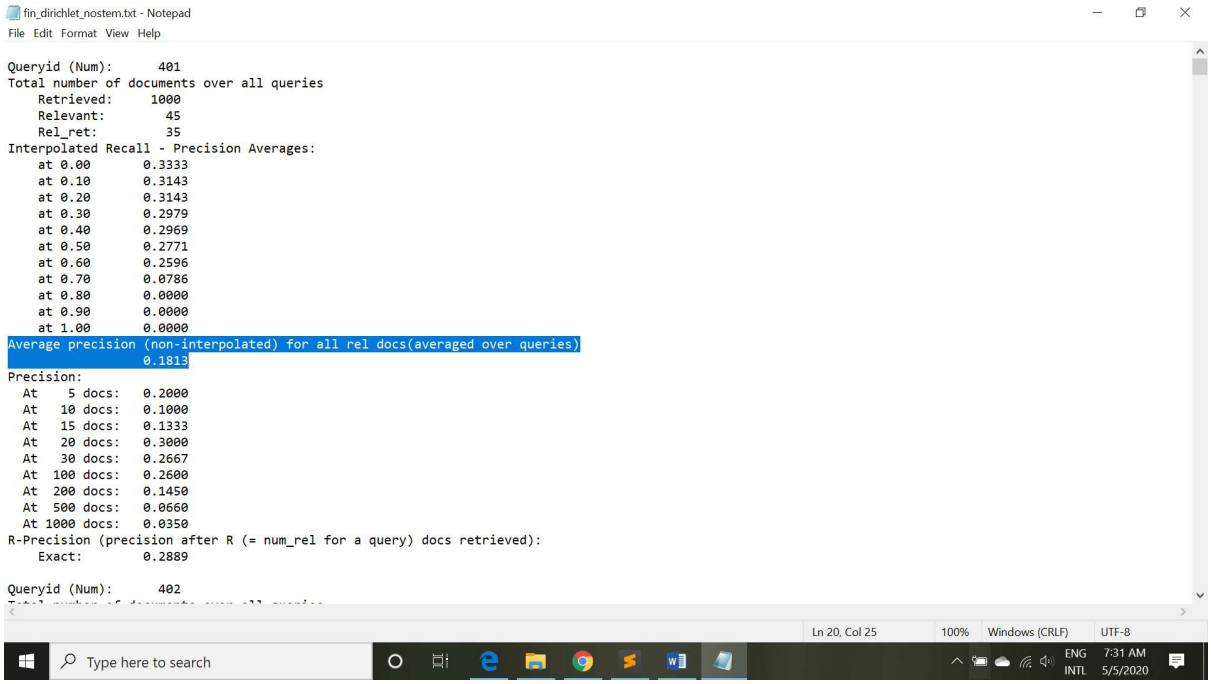
Average precision (non-interpolated) for all relevant docs(averaged over queries) is 0.0502.

3. MLE with Laplace smoothing(with stemming)



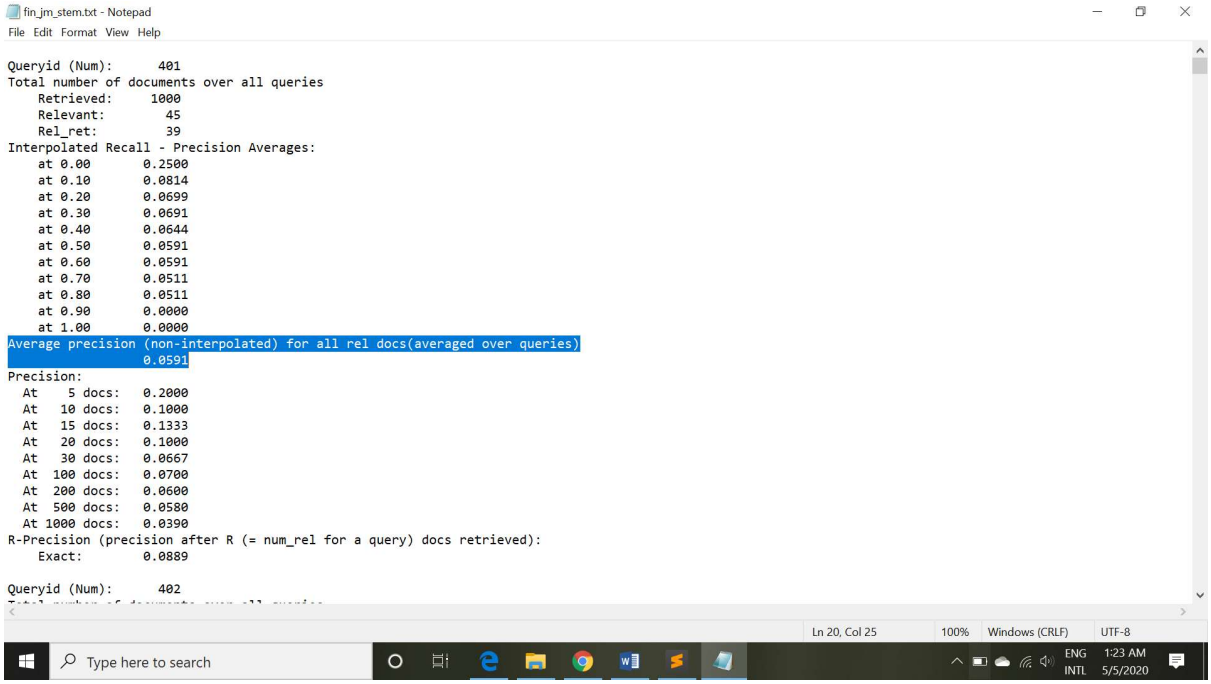
Average precision (non-interpolated) for all relevant docs(averaged over queries) is 0.1844.

4. MLE with Laplace smoothing(without stemming)



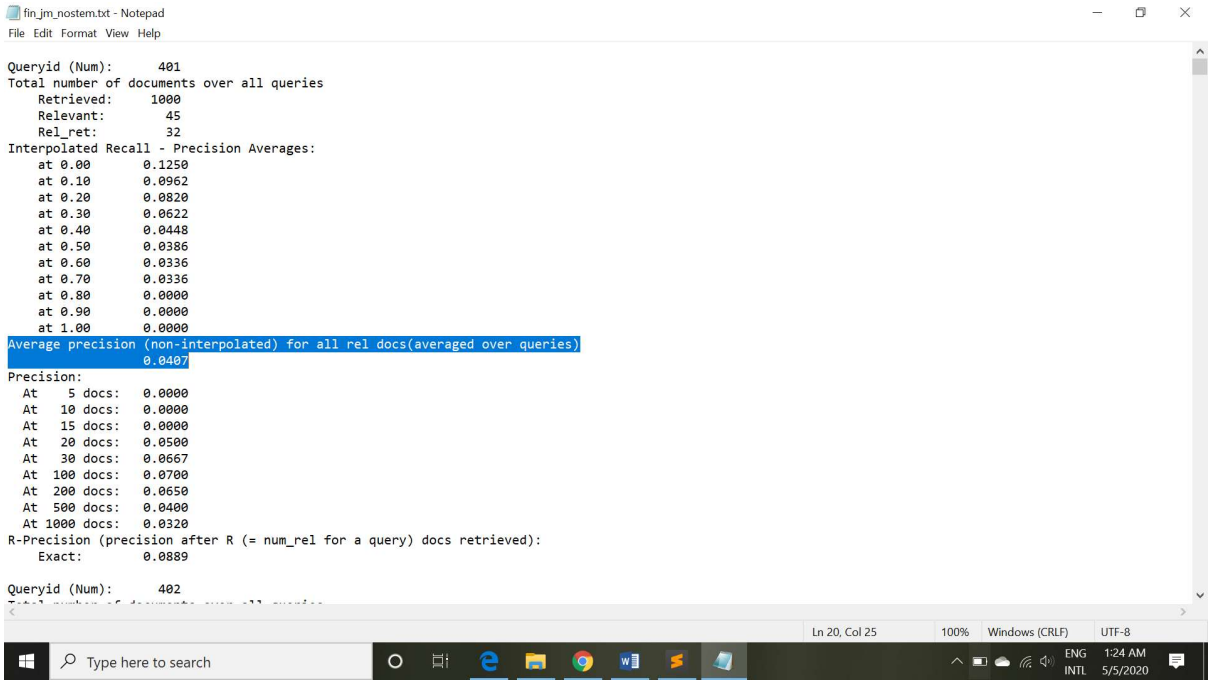
Average precision (non-interpolated) for all relevant docs(averaged over queries) is 0.1813.

5. Jelinek-Mercer smoothing(with stemming)



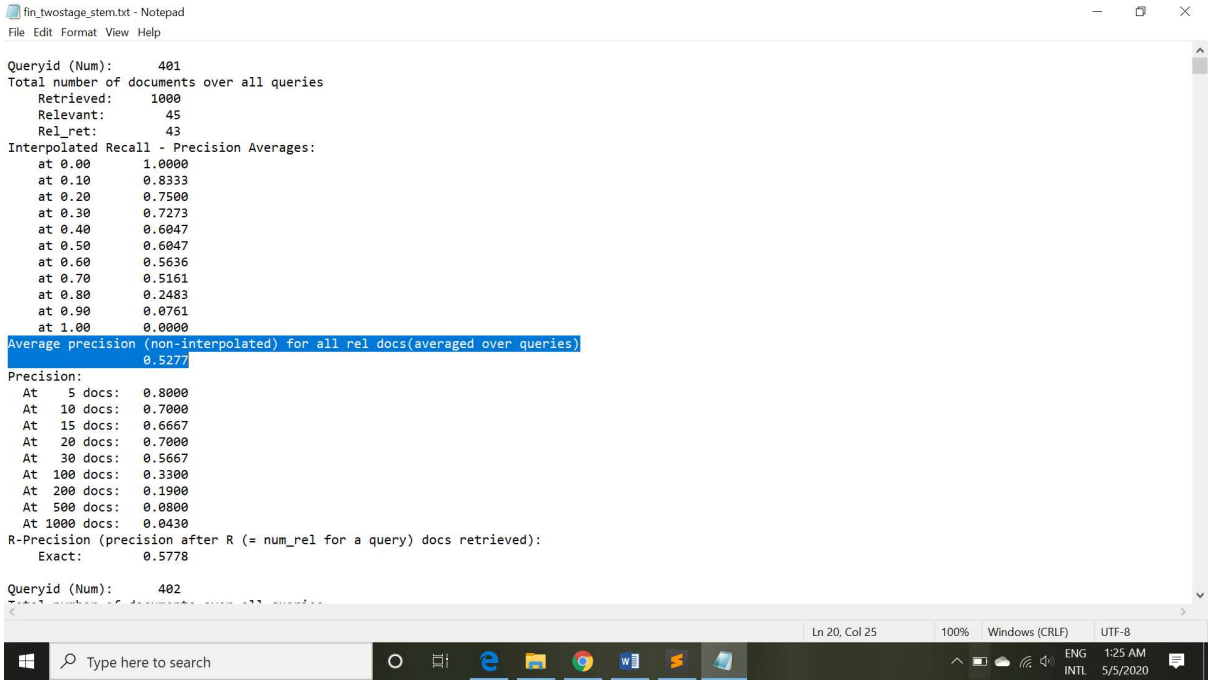
Average precision (non-interpolated) for all relevant docs(averaged over queries) is 0.0591.

6. Jelinek-Mercer smoothing(without stemming)



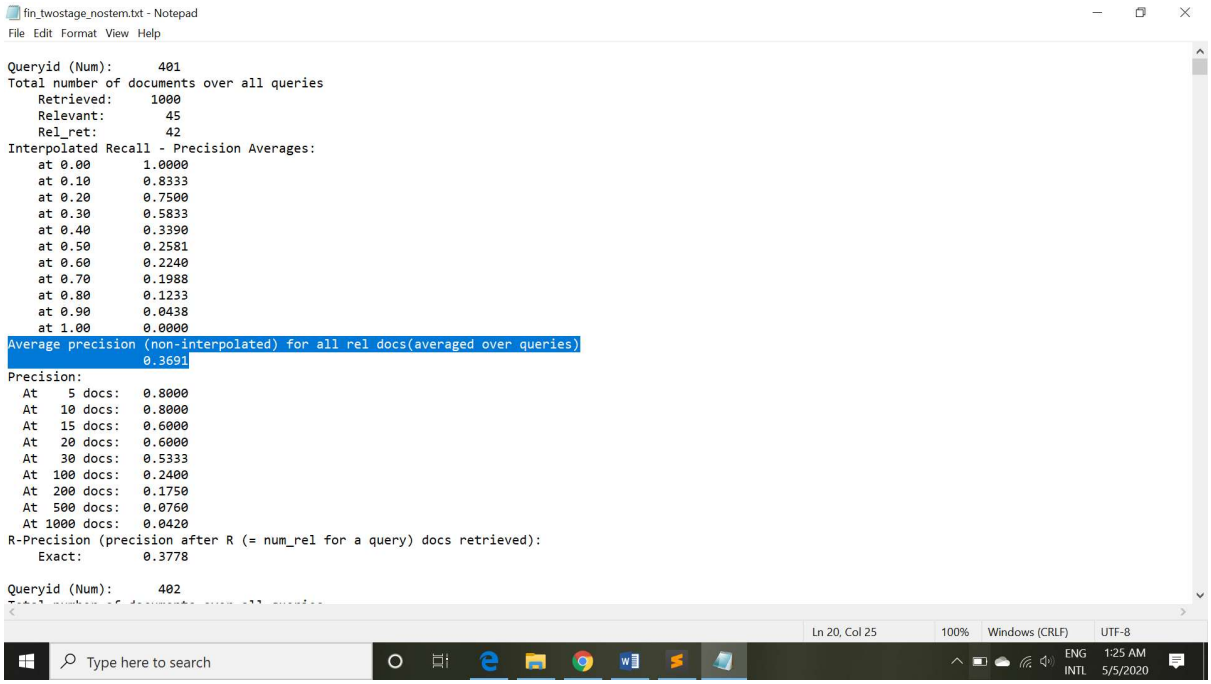
Average precision (non-interpolated) for all relevant docs(averaged over queries) is 0.0407.

7. Two-stage Smoothing(with stemming)



Average precision (non-interpolated) for all relevant docs(averaged over queries) is 0.5277.

8. Two-stage Smoothing(without stemming)



Average precision (non-interpolated) for all relevant docs(averaged over queries) is 0.3691.

9.Comparison:

Un-interpolated mean average precision numbers for all 8 runs:

0.5277(Two-stage Smoothing(with stemming)) >

0.3691(Two-stage Smoothing(without stemming)) >

0.2629(Okapi TFIDF(with stemming)) >

0.1844(MLE with Laplace smoothing(with stemming)) >

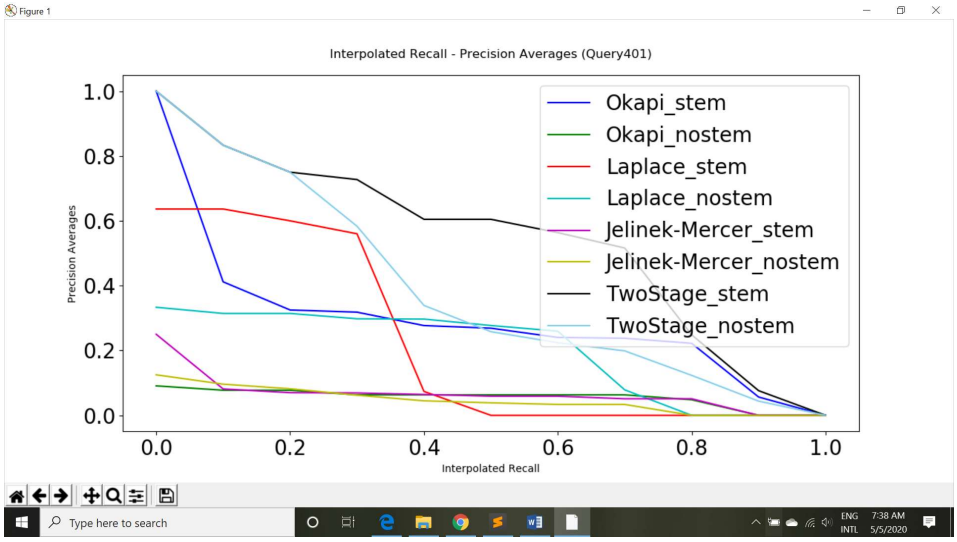
0.1813(MLE with Laplace smoothing(without stemming)) >

0.0591(Jelinek-Mercer smoothing(with stemming)) >

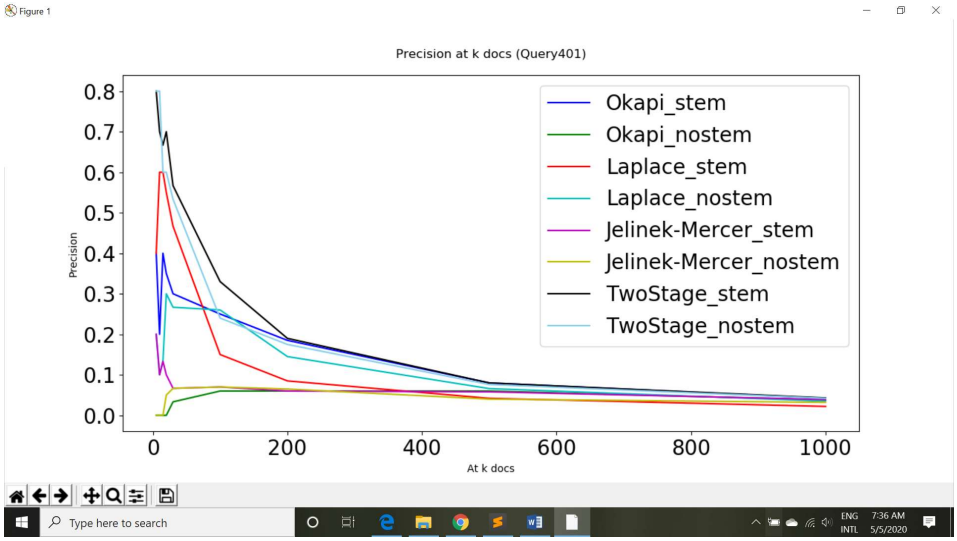
0.0502(Okapi TFIDF(without stemming)) >

0.0407(Jelinek-Mercer smoothing(without stemming))

Interpolated Recall - Precision Averages(Precision at rank 10 documents for all 8 runs):



Precision at k docs:



Analysis

Based on the above result, we can observe that the Two Stage method is the best for information retrieval ranking. The reason is that we use both Dirichlet smoothing and Jelinek-Mercer for the model. Dirichlet smoothing is generally sensitive to the collection. With the help of Jelinek-Mercer, the model becomes less sensitive to the type or length of the queries.

We can also observe that Okapi TFIDF (with stemming) outperforms Laplace smoothing(with stemming) for most of the interpolated recall because the tuning effect of IDF in Okapi and the normalization of the document may help improve our performance.

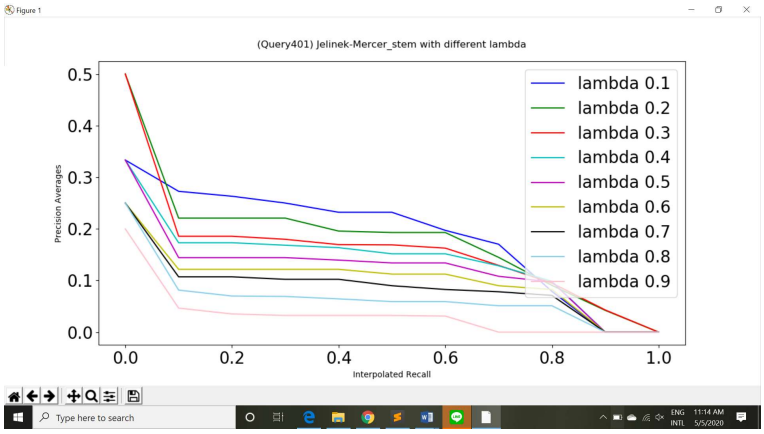
Jelinek-Mercer performs the worst regardless of stemming process or not. Yet, with the help of Dirichlet smoothing before the process, which is the Two Stage smoothing method, the performance increases dramatically.

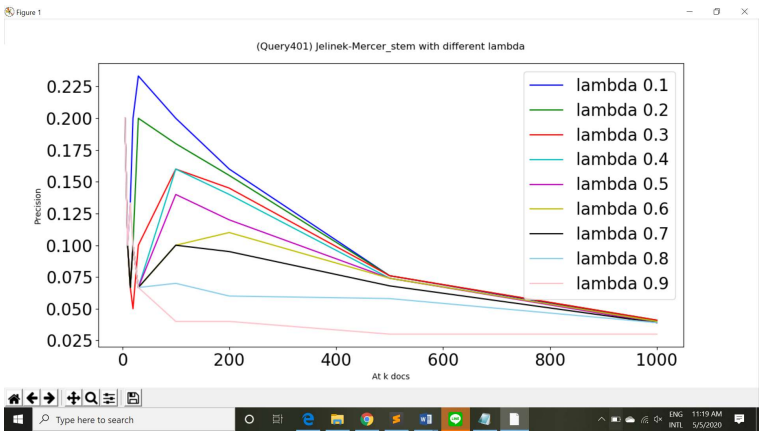
Overall, we can observe that the corpus with stemming pre-processing is better for all the models since the process help us define similar context of the words and classify them as the same group, which makes ranking more efficiently and accurately. However, if the algorithm doesn't design well, it may lead to misclassification of the words.

Besides, we also see that the performance of the ranking precision average decays to zero at 0.8 interpolated recall except Two Stage smoothing method. Jelinek-Mercer smoothing decays even at an earlier interpolated recall. The reason for the phenomena is that you usually get worse result when you concern more documents retrieved since the relevant documents may be "diluted" by the amount of irrelevant document. As you retrieve more documents, you also get more irrelevant documents, which lowers your precision averages.

More Interesting comparison..

1. Comparisons of Jelinek-Mercer methods with different lambda





$$\rho_i = \lambda P + (1 - \lambda)Q$$

P is the estimated probability from document (max likelihood = m i/n) and Q is the estimated probability from corpus (background probability = cf / terms in the corpus). Lambda is the weight attached to the background probability.

From the above figure, we observe that Jelinek-Mercer methods with less lambda values perform better. As you can see, less lambda value means that we put more emphasis on the estimated probability from corpus. Hence, the query likelihood depends less on the estimated probability from document, which may be biased sometimes. As a result, less lambda values could generate better result.

2.Comparisons between Okapi scoring and tf.idf scoring using BM25TF term weighting

Implementation:

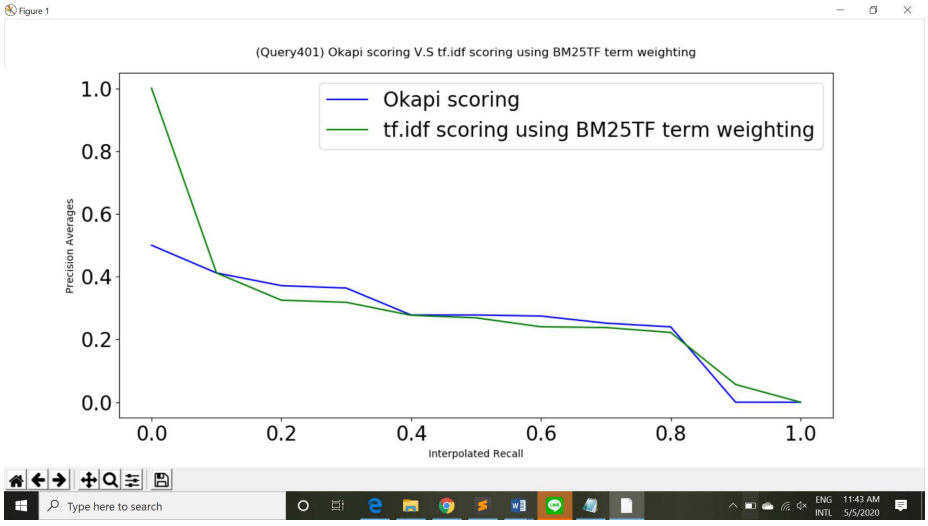
- tf.idf scoring using BM25TF term weighting

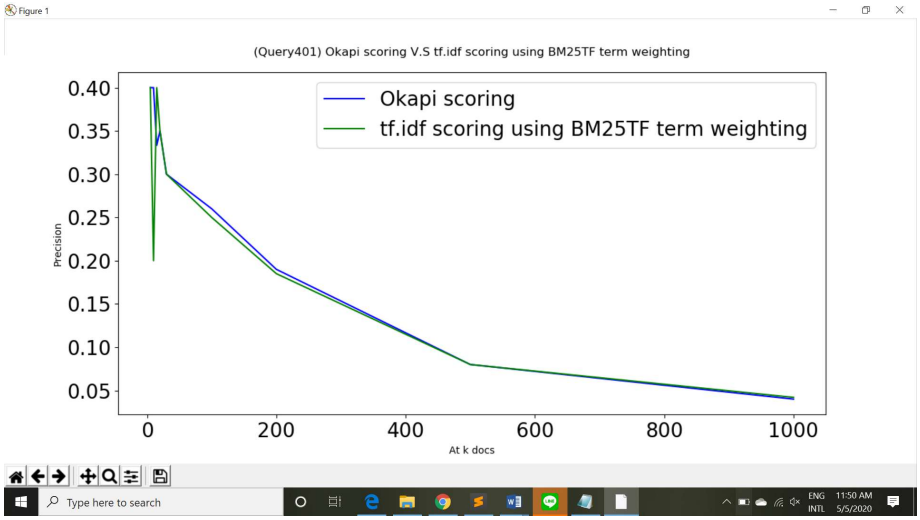
```
<count>1000</count>
<baseline>tfidf,k1:2.0,b:0.75</baseline>
```

-Okapi scoring

```
<baseline>okapi,k1:2.0,b:0.75</baseline>
```

Stemming case:



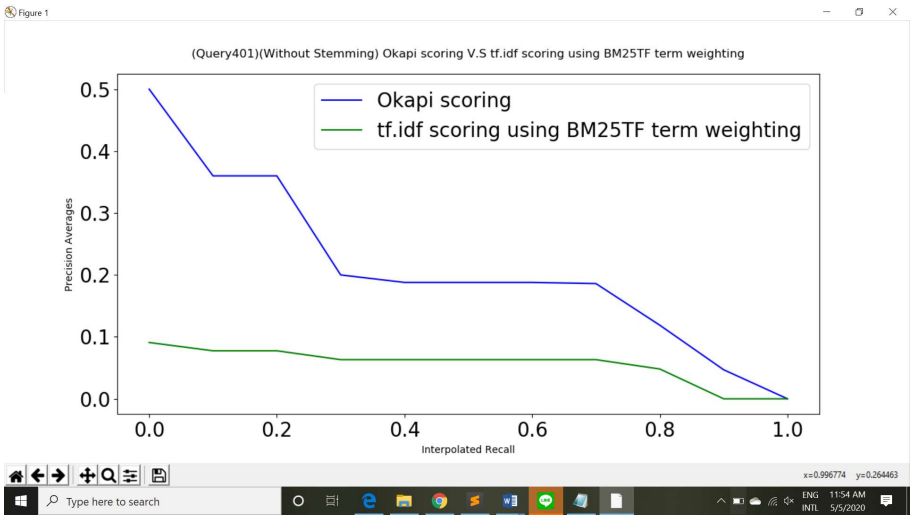


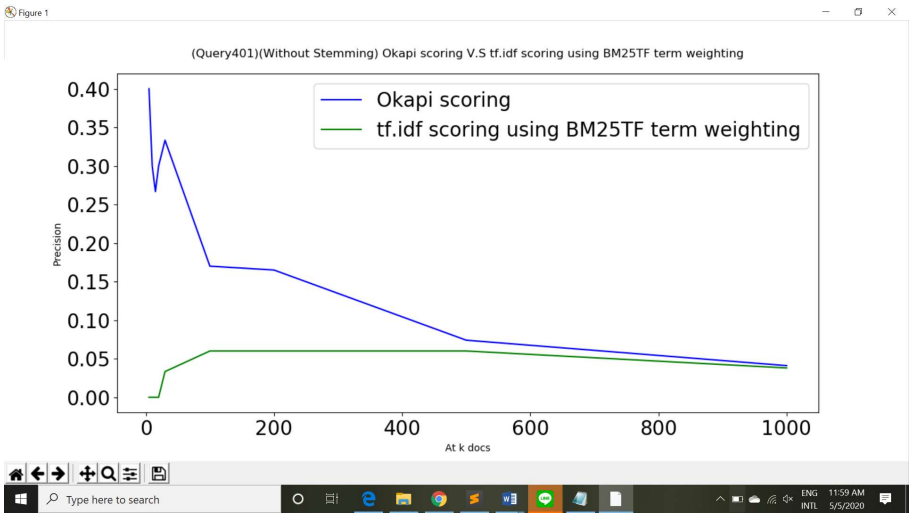
Average precision (non-interpolated) for all relevant docs(averaged over queries):

-tf.idf scoring using BM25TF term weighting: 0.2629

-Okapi scoring: 0.2634

Without Stemming case:





Average precision (non-interpolated) for all relevant docs(averaged over queries):

-tf.idf scoring using BM25TF term weighting: 0.0502

-Okapi scoring: 0.1901

Observation and Explanation:

As you can see from the results above, in the case of pre-processing with stemming method, tf.idf scoring using BM25TF term weighting performs nearly as good as Okapi scoring. Yet, in the case of pre-processing without stemming method. Okapi scoring outperforms the other

method in all ranges. The explanation for the result is that Okapi scoring has one more parameter for query term weight. Hence, the method is more flexible in different query cases.

Reference:

https://en.wikipedia.org/wiki/Okapi_BM25

https://inst.eecs.berkeley.edu/~cs188/sp12/slides/cs188%20lecture%2020%20--%20naive%20bayes%206PP.pdf

http://sifaka.cs.uiuc.edu/czhai/pub/lmir2001-dualrole.pdf