Web Searching Mining Project2 – Lemur(Indri)

student ID:110753201 Class:資科碩一 Name: 曹昱維

Introduction

- This project is asked to use Lemur or Indri toolkits to implement several different retrieval methods.
- Corpus : WT2g. This corpus contains Web documents with a 2GB corpus.
- Install indri 5.2
  - Install script:

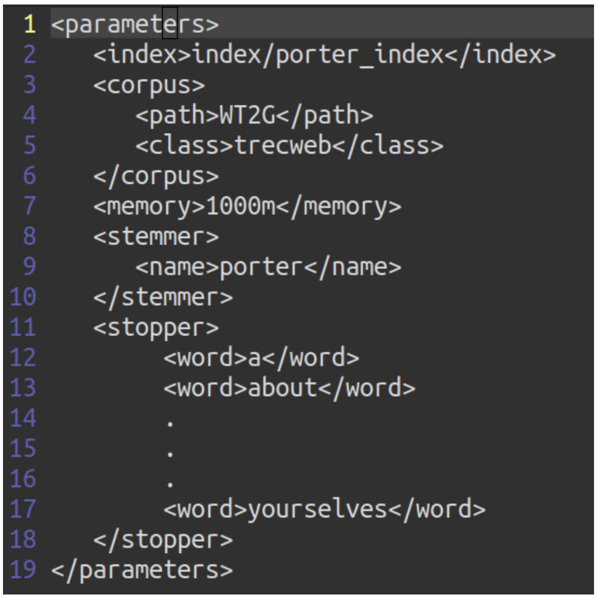
    ```
    echo "deb http://dk.archive.ubuntu.com/ubuntu/ trusty main universe" \
    >> /etc/apt/sources.list \
    apt update && apt upgrade -y && \
    apt install -y build-essential make gcc-4.4 g++-4.4 zlib1g-dev && \
    update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-9 1 && \
    update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.4 2 && \
    update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.4 2 && \
    update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 1 && \
    tar -xzvf /root/indri-5.20.tar.gz -C /root && \
    cd /root/indri-5.20 && ./configure && make && make install
    ```

- The main process consist of three steps:
  1. Buidl Index
     - build two type index of corpus, **index with porter stemming and without stemming**.
  2. Run Query
     - Run query by specified method, for example dirichlet smoothing.
     - We have **four method** to test, Okapi, Language Model with laplace smooth, Language Model with Jelinek-Mercer smoothing, method proposed by student.
  3. evaluation
     - We evaluate 2*4 retrieved result files(two type stemming and 4 methods).

Related work

- The follwing description of How to set up parameters files of `IndriBuildIndex` , `IndriRunQuery` refers to
  - https://sourceforge.net/p/lemur/wiki/Home/ (https://sourceforge.net/p/lemur/wiki/Home/)
  - http://www.cs.cmu.edu/~lemur/1.1/api.html (http://www.cs.cmu.edu/~lemur/1.1/api.html)
  - https://sourceforge.net/projects/lemur/files/lemur/indri-5.20/indri-5.20-doc.zip/download (https://sourceforge.net/projects/lemur/files/lemur/indri-5.20/indri-5.20-doc.zip/download)

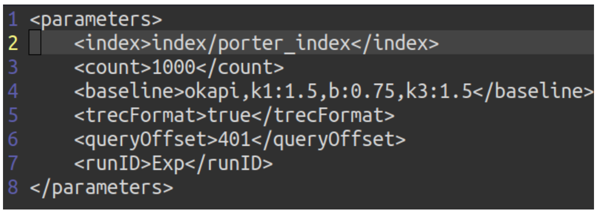
- **`IndriBuildIndex <index_parameter_file>`**
  - Indri 5.2 provide a useful program called `IndriBUildIndex` , which build index file of corpus, you can set several options, such as *stem method*, *stop word* etc. in the `<index_parameter files>
    - ref: https://sourceforge.net/p/lemur/wiki/IndriBuildIndex%20Parameters/ (https://sourceforge.net/p/lemur/wiki/IndriBuildIndex%20Parameters/)
  - An Example of the `IndriBuildIndex` **parameter file** used in Project2 of WSM is shown in Figure 1



```
<parameters>
   <index>index/porter_index</index>
   <corpus>
      <path>WT2G</path>
      <class>trecweb</class>
   </corpus>
   <memory>1000m</memory>
   <stemmer>
      <name>porter</name>
   </stemmer>
   <stopper>
        <word>a</word>
        <word>about</word>
        .
        .
        .
        <word>yourselves</word>
   </stopper>
</parameters>
```

Fig.1

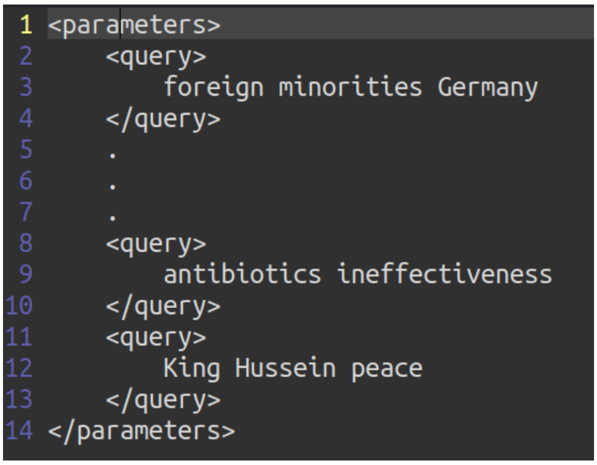
- **`IndriRunQuery [query_parameter_file] [query_data] > result file`**
    - Indri 5.2 provide a useful program called `IndriRunQuery` , which run queries on corpus. The queries should sotred in `[query_data]` . You can set several options such as *retrieve relvance model*, *smoothing mehtod* etc. in the `[query_parameter files]` .
        - ref: https://sourceforge.net/p/lemur/wiki/IndriRunQuery/ (https://sourceforge.net/p/lemur/wiki/IndriRunQuery/)
    - An Example of the `IndriRunQuery` **parameter file** used in Project2 of WSM is shown in Figure 2



```
<parameters>
    <index>index/porter_index</index>
    <count>1000</count>
    <baseline>okapi,k1:1.5,b:0.75,k3:1.5</baseline>
    <trecFormat>true</trecFormat>
    <queryOffset>401</queryOffset>
    <runID>Exp</runID>
</parameters>
```

Fig.2

- An Example of the `IndriRunQuery` **queries data** used in Project2 of WSM is shown in Figure 3

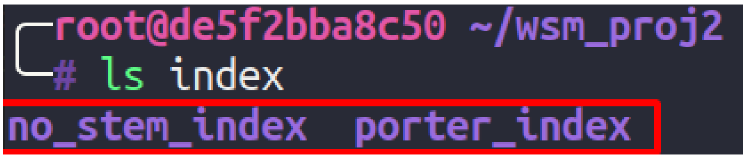


```
<parameters>
    <query>
        foreign minorities Germany
    </query>
    .
    .
    .
    <query>
        antibiotics ineffectiveness
    </query>
    <query>
        King Hussein peace
    </query>
</parameters>
```

Fig.3

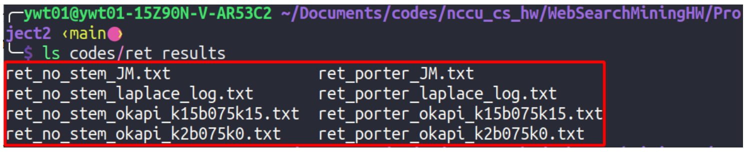
Run Query

- ### Overview of how to run queries
    - We need to test 4 models: *Okapi TF* , *language model with Laplace smoothing*, *Language modeling with Jelinek-Mercer smoothing*, *improve one of the above three IR models*.
    - We have two index files of coupus: *without stemming*. *with stemming*. In this porject, I choose **Porter setmmer**.
    - Therefore, with above description, we will get **2 index files** and **8 retrieved results** as shown in Figure 4 and Figure5 respectively.



```
┌root@de5f2bba8c50 ~/wsm_proj2
└# ls index
no_stem_index  porter_index
```

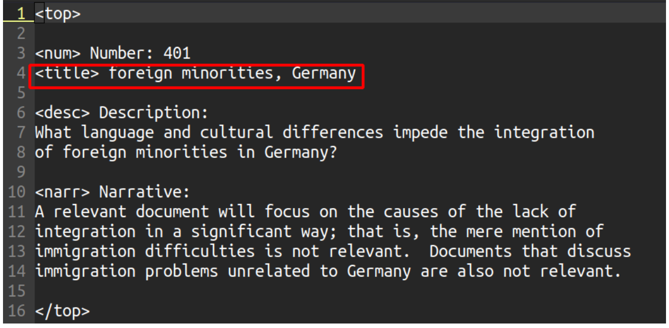
Fig.4



```
┌─ywt01@ywt01-15Z90N-V-AR53C2 ~/Documents/codes/nccu_cs_hw/WebSearchMiningHW/Pro
ject2 ‹main●›
└─$ ls codes/ret results
ret_no_stem_JM.txt                ret_porter_JM.txt
ret_no_stem_laplace_log.txt       ret_porter_laplace_log.txt
ret_no_stem_okapi_k15b075k15.txt  ret_porter_okapi_k15b075k15.txt
ret_no_stem_okapi_k2b075k0.txt    ret_porter_okapi_k2b075k0.txt
```

Fig.5

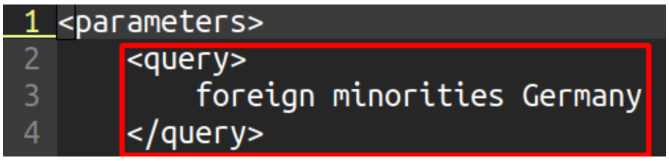
- ### Preprocess query files
    - We used title field of original queries file(`topics.401-450.txt`) as input queries to `IndriRunQuery`
        - An example of original queries file(`topics.401-450.txt`) as shown in Figure 6.



```
<top>

<num> Number: 401
<title> foreign minorities, Germany

<desc> Description:
What language and cultural differences impede the integration
of foreign minorities in Germany?

<narr> Narrative:
A relevant document will focus on the causes of the lack of
integration in a significant way; that is, the mere mention of
immigration difficulties is not relevant.  Documents that discuss
immigration problems unrelated to Germany are also not relevant.

</top>
```

Fig.6

- An example of after-processed query as shown in Figure 7.



```
<parameters>
    <query>
        foreign minorities Germany
    </query>
```

Fig.7

- Preprocessing linux command:

```
cat topics.401-450.txt|grep "^<title>" \
                      |sed "s/[[:punct:]]//g" \
                      |sed "s/title/<query>\n/g" \
                      |sed "/^<query>/!s/$/\n<\/query>/g" \
                      |sed "/^</!s/^//g" \
                      |sed "s/^//g">query_title.txt ; \
sed -i "1i<parameters>" query_title.txt ; \
echo "</parameters>">>query_title.txt
```

- **OKAPI TF-IDF**
  - OKAPI model for long query formula:

$$\sum_{t\in q} \log \frac{N}{df_t} \cdot \frac{(k_1+1)tf_{td}}{k_1((1-b)+b\times(L_d/L_{ave}))+tf_{td}} \cdot \frac{(k_3+1)tf_{tq}}{k_3+tf_{tq}}$$

  - In order to transform OKAPI model to OKAPI TF-IDF, we have to set $k_3 = 0$
  - Therefore, to use OKAPI TF-IDF on `IndexRunQuery` , we have to add `<baseline>okapi,k1:2.0,b:0.75,k3:0.0</baseline>` in the parameter file

- **Language Model with Laplace smoothing**
  - Laplace smoothing formula:

$$\hat{P}(w_i\,|\,c) = \frac{count(w_i,c)+1}{\sum_{w\in V}(count(w,c)+1)} = \frac{count(w_i,c)+1}{\left(\sum_{w\in V} count(w,c)\right) + |V|}$$

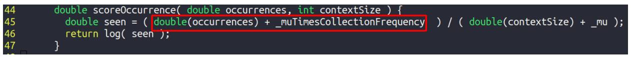
  - Dirichlet smoothing formula:

$$p(w|d) = \frac{c(w,d)+\mu p(w|C)}{|d|+\mu}$$

  - Comparing the above two formulas, the Laplace smoothing can be regarded as a special case of the Dirichlet smoothing
    - reference to : http://www.cs.cmu.edu/~lemur/1.1/api.html (http://www.cs.cmu.edu/~lemur/1.1/api.html)

- Therefore, to use Language Model with Laplace smoothing in `IndexRunQuery` , we have to :

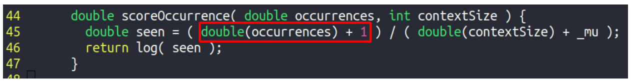
  1. Add `<rule>method:dirichlet,mu: unique terms</rule>` in the parameter file
     - assign `unique terms` to μ make it equals to number of unique terms in corpus
  2. Revise `indri-5.10/include/indri/DirichletTermScoreFunction.hpp`
     - Before revised as shown in Figure 8 :



```
    double scoreOccurrence( double occurrences, int contextSize ) {
      double seen = ( double(occurrences) + _muTimesCollectionFrequency  ) / ( double(contextSize) + _mu );
      return log( seen );
    }
```

Fig.8

     - After revised as shown in Figure 9 :



```
    double scoreOccurrence( double occurrences, int contextSize ) {
      double seen = ( double(occurrences) + 1 ) / ( double(contextSize) + _mu );
      return log( seen );
    }
```

Fig.9

- **Language Model with Jelinek-Mercer smoothing**
  - In order to Language Model with Jelinek-Mercer smoothing in `IndexRunQuery` , we have to add `<rule>method:jm,collectionLambda:0.8</rule>` in the parameter file

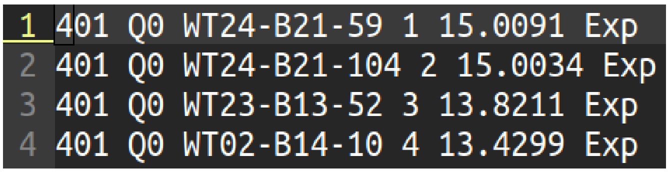
- **Improve one of the above three IR models**
  - I choose OKAPI model as base model, and adjust the parameters k_1, b, k_3
  - In the text book-**Introduction to Information Retrieval** ch11, it mentioned:

    > In the absence of such optimization, experiments have shown reasonable values are to set **k1 and k3 to a value between 1.2 and 2 and b = 0.75.**

  - Therefore, I add `<baseline>okapi,k1:1.5,b:0.75,k3:1.5</baseline>` to the parameters file

Evaluation and Results

- **Overview of how to evaluate**
  - How to evaulate retrieved results : `Trec_eval –q qrels.401-450 <retrieved result> > eval_result`
    - An example of retrieved results from `IndriRunQuery` as shown in Figure 10.

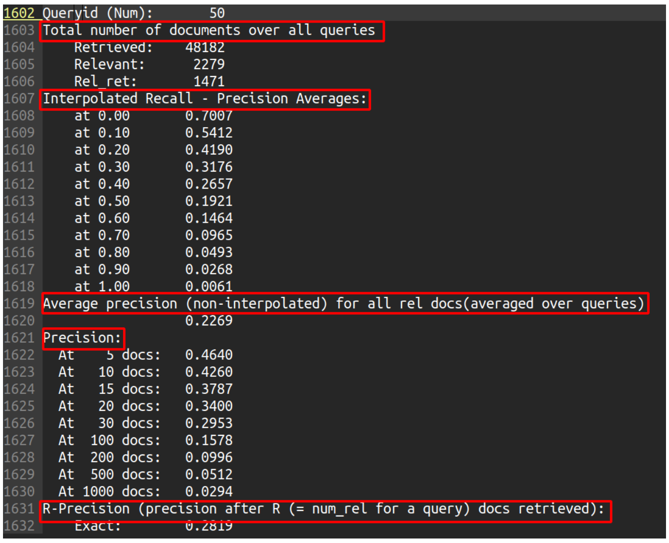


```
401 Q0 WT24-B21-59 1 15.0091 Exp
401 Q0 WT24-B21-104 2 15.0034 Exp
401 Q0 WT23-B13-52 3 13.8211 Exp
401 Q0 WT02-B14-10 4 13.4299 Exp
```

Fig.10

The format of retrieved results: `query-number>query-number Q0 document-id rank score Exp`

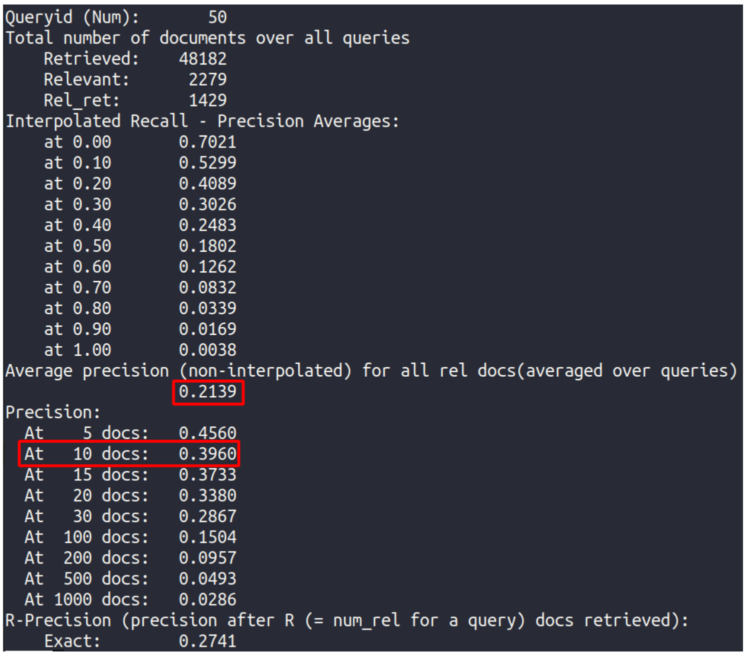
    - An example of evaluation of retrieved results from `Trec_eval` as shown in Figure 11.



```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:     48182
    Relevant:       2279
    Rel_ret:        1471
Interpolated Recall - Precision Averages:
    at 0.00        0.7007
    at 0.10        0.5412
    at 0.20        0.4190
    at 0.30        0.3176
    at 0.40        0.2657
    at 0.50        0.1921
    at 0.60        0.1464
    at 0.70        0.0965
    at 0.80        0.0493
    at 0.90        0.0268
    at 1.00        0.0061
Average precision (non-interpolated) for all rel docs(averaged over queries)
                   0.2269
Precision:
  At    5 docs:    0.4640
  At   10 docs:    0.4260
  At   15 docs:    0.3787
  At   20 docs:    0.3400
  At   30 docs:    0.2953
  At  100 docs:    0.1578
  At  200 docs:    0.0996
  At  500 docs:    0.0512
  At 1000 docs:    0.0294
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:         0.2819
```

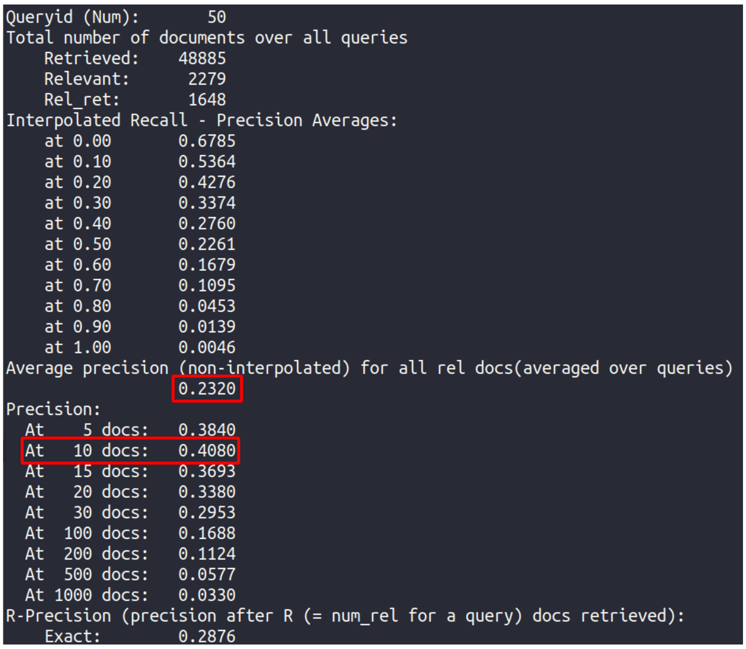
Fig.11

- **Evaluation results of OKAPI TF-IDF**



```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:    48182
    Relevant:      2279
    Rel_ret:       1429
Interpolated Recall - Precision Averages:
    at 0.00       0.7021
    at 0.10       0.5299
    at 0.20       0.4089
    at 0.30       0.3026
    at 0.40       0.2483
    at 0.50       0.1802
    at 0.60       0.1262
    at 0.70       0.0832
    at 0.80       0.0339
    at 0.90       0.0169
    at 1.00       0.0038
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.2139
Precision:
  At    5 docs:   0.4560
  At   10 docs:   0.3960
  At   15 docs:   0.3733
  At   20 docs:   0.3380
  At   30 docs:   0.2867
  At  100 docs:   0.1504
  At  200 docs:   0.0957
  At  500 docs:   0.0493
  At 1000 docs:   0.0286
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.2741
```

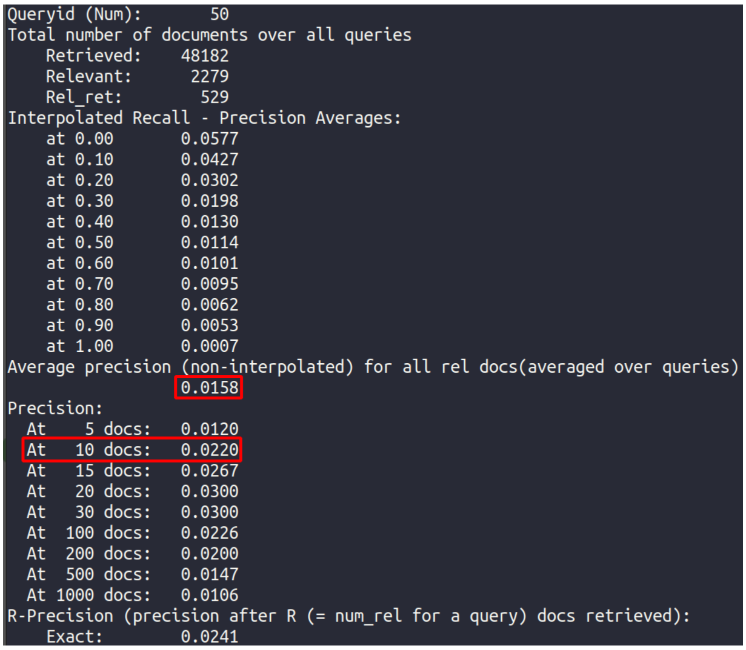
OKAPI TF-IDF with no stemming



```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:    48885
    Relevant:      2279
    Rel_ret:       1648
Interpolated Recall - Precision Averages:
    at 0.00       0.6785
    at 0.10       0.5364
    at 0.20       0.4276
    at 0.30       0.3374
    at 0.40       0.2760
    at 0.50       0.2261
    at 0.60       0.1679
    at 0.70       0.1095
    at 0.80       0.0453
    at 0.90       0.0139
    at 1.00       0.0046
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.2320
Precision:
  At    5 docs:   0.3840
  At   10 docs:   0.4080
  At   15 docs:   0.3693
  At   20 docs:   0.3380
  At   30 docs:   0.2953
  At  100 docs:   0.1688
  At  200 docs:   0.1124
  At  500 docs:   0.0577
  At 1000 docs:   0.0330
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.2876
```

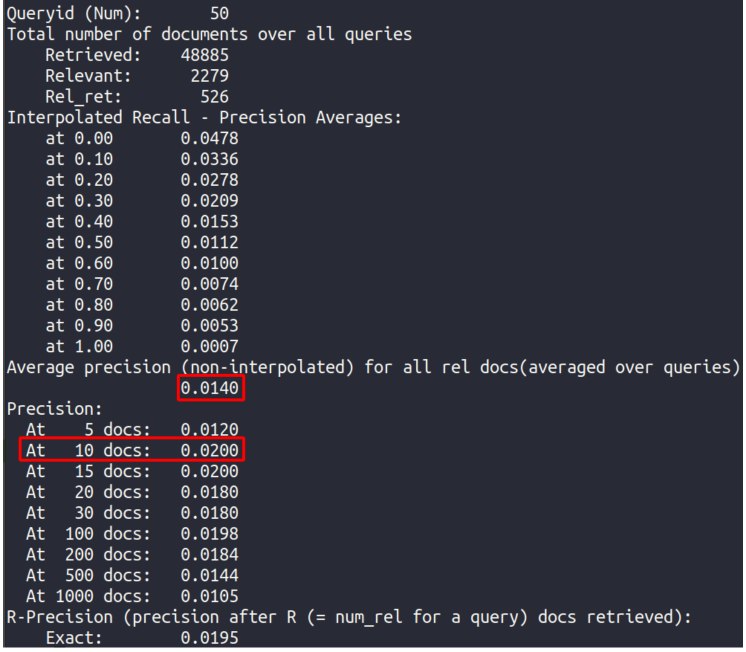
OKAPI TF-IDF with porter stemming

- **Evaluation results of Language Model with Laplace smoothing**



```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:    48182
    Relevant:      2279
    Rel_ret:        529
Interpolated Recall - Precision Averages:
    at 0.00       0.0577
    at 0.10       0.0427
    at 0.20       0.0302
    at 0.30       0.0198
    at 0.40       0.0130
    at 0.50       0.0114
    at 0.60       0.0101
    at 0.70       0.0095
    at 0.80       0.0062
    at 0.90       0.0053
    at 1.00       0.0007
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.0158
Precision:
  At    5 docs:   0.0120
  At   10 docs:   0.0220
  At   15 docs:   0.0267
  At   20 docs:   0.0300
  At   30 docs:   0.0300
  At  100 docs:   0.0226
  At  200 docs:   0.0200
  At  500 docs:   0.0147
  At 1000 docs:   0.0106
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.0241
```

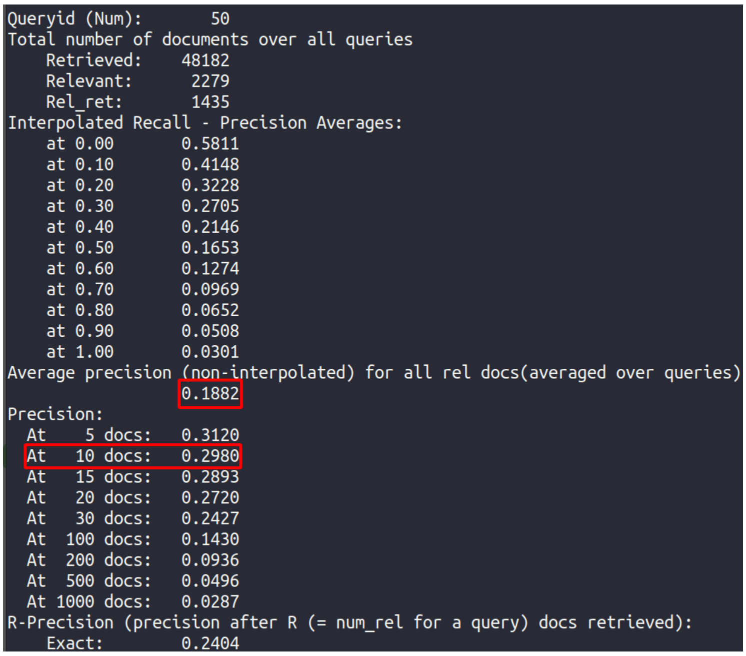
Laplace with no stemming



```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:    48885
    Relevant:      2279
    Rel_ret:        526
Interpolated Recall - Precision Averages:
    at 0.00       0.0478
    at 0.10       0.0336
    at 0.20       0.0278
    at 0.30       0.0209
    at 0.40       0.0153
    at 0.50       0.0112
    at 0.60       0.0100
    at 0.70       0.0074
    at 0.80       0.0062
    at 0.90       0.0053
    at 1.00       0.0007
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.0140
Precision:
  At    5 docs:   0.0120
  At   10 docs:   0.0200
  At   15 docs:   0.0200
  At   20 docs:   0.0180
  At   30 docs:   0.0180
  At  100 docs:   0.0198
  At  200 docs:   0.0184
  At  500 docs:   0.0144
  At 1000 docs:   0.0105
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.0195
```

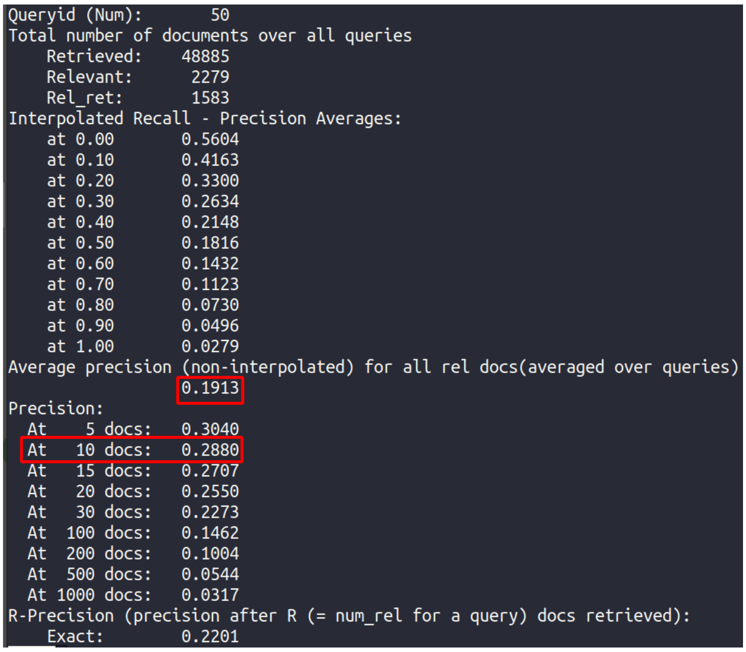
Laplace with porter stemming

- **Evaluation results of Language Model with Jelinek-Mercer smoothing**



```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:    48182
    Relevant:      2279
    Rel_ret:       1435
Interpolated Recall - Precision Averages:
    at 0.00       0.5811
    at 0.10       0.4148
    at 0.20       0.3228
    at 0.30       0.2705
    at 0.40       0.2146
    at 0.50       0.1653
    at 0.60       0.1274
    at 0.70       0.0969
    at 0.80       0.0652
    at 0.90       0.0508
    at 1.00       0.0301
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.1882
Precision:
  At    5 docs:   0.3120
  At   10 docs:   0.2980
  At   15 docs:   0.2893
  At   20 docs:   0.2720
  At   30 docs:   0.2427
  At  100 docs:   0.1430
  At  200 docs:   0.0936
  At  500 docs:   0.0496
  At 1000 docs:   0.0287
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.2404
```

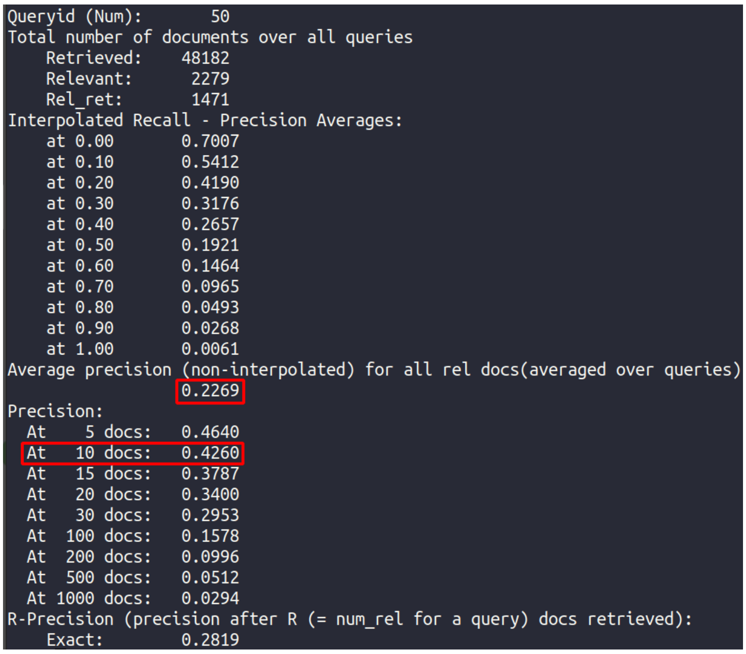
JM with no stemming



```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:    48885
    Relevant:      2279
    Rel_ret:       1583
Interpolated Recall - Precision Averages:
    at 0.00       0.5604
    at 0.10       0.4163
    at 0.20       0.3300
    at 0.30       0.2634
    at 0.40       0.2148
    at 0.50       0.1816
    at 0.60       0.1432
    at 0.70       0.1123
    at 0.80       0.0730
    at 0.90       0.0496
    at 1.00       0.0279
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.1913
Precision:
  At    5 docs:   0.3040
  At   10 docs:   0.2880
  At   15 docs:   0.2707
  At   20 docs:   0.2550
  At   30 docs:   0.2273
  At  100 docs:   0.1462
  At  200 docs:   0.1004
  At  500 docs:   0.0544
  At 1000 docs:   0.0317
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.2201
```

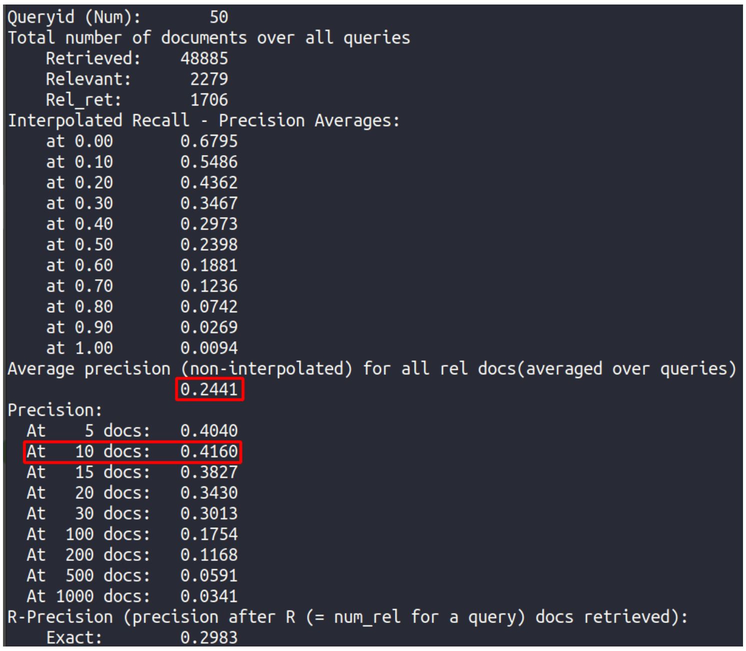
JM with porter stemming

- **Evaluation results of Improved-OKAPI(k_1=1.5,k_3=1.5,b=0.75)**



```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:    48182
    Relevant:      2279
    Rel_ret:       1471
Interpolated Recall - Precision Averages:
    at 0.00       0.7007
    at 0.10       0.5412
    at 0.20       0.4190
    at 0.30       0.3176
    at 0.40       0.2657
    at 0.50       0.1921
    at 0.60       0.1464
    at 0.70       0.0965
    at 0.80       0.0493
    at 0.90       0.0268
    at 1.00       0.0061
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.2269
Precision:
  At    5 docs:   0.4640
  At   10 docs:   0.4260
  At   15 docs:   0.3787
  At   20 docs:   0.3400
  At   30 docs:   0.2953
  At  100 docs:   0.1578
  At  200 docs:   0.0996
  At  500 docs:   0.0512
  At 1000 docs:   0.0294
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.2819
```

Improved OKAPI with no stemming



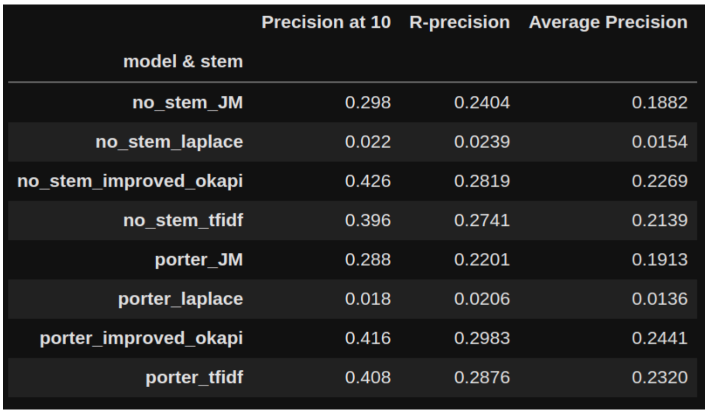
```
Queryid (Num):       50
Total number of documents over all queries
    Retrieved:    48885
    Relevant:      2279
    Rel_ret:       1706
Interpolated Recall - Precision Averages:
    at 0.00       0.6795
    at 0.10       0.5486
    at 0.20       0.4362
    at 0.30       0.3467
    at 0.40       0.2973
    at 0.50       0.2398
    at 0.60       0.1881
    at 0.70       0.1236
    at 0.80       0.0742
    at 0.90       0.0269
    at 1.00       0.0094
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.2441
Precision:
  At    5 docs:   0.4040
  At   10 docs:   0.4160
  At   15 docs:   0.3827
  At   20 docs:   0.3430
  At   30 docs:   0.3013
  At  100 docs:   0.1754
  At  200 docs:   0.1168
  At  500 docs:   0.0591
  At 1000 docs:   0.0341
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.2983
```

Improved OKAPI with porter stemming

Analysis

Preprocess raw evaluation results

- I used a script to summarize the all evaluation results
  - summarize table:



	Precision at 10	R-precision	Average Precision
model & stem			
no_stem_JM	0.298	0.2404	0.1882
no_stem_laplace	0.022	0.0239	0.0154
no_stem_improved_okapi	0.426	0.2819	0.2269
no_stem_tfidf	0.396	0.2741	0.2139
porter_JM	0.288	0.2201	0.1913
porter_laplace	0.018	0.0206	0.0136
porter_improved_okapi	0.416	0.2983	0.2441
porter_tfidf	0.408	0.2876	0.2320

Conclusion

- An overview of the performance of each **model + stemming method** is shown in Figure 12.
- Considering *Average precision*, *R-precision* and *Precision at 10*, **the improved-okapi model has the best performance**.

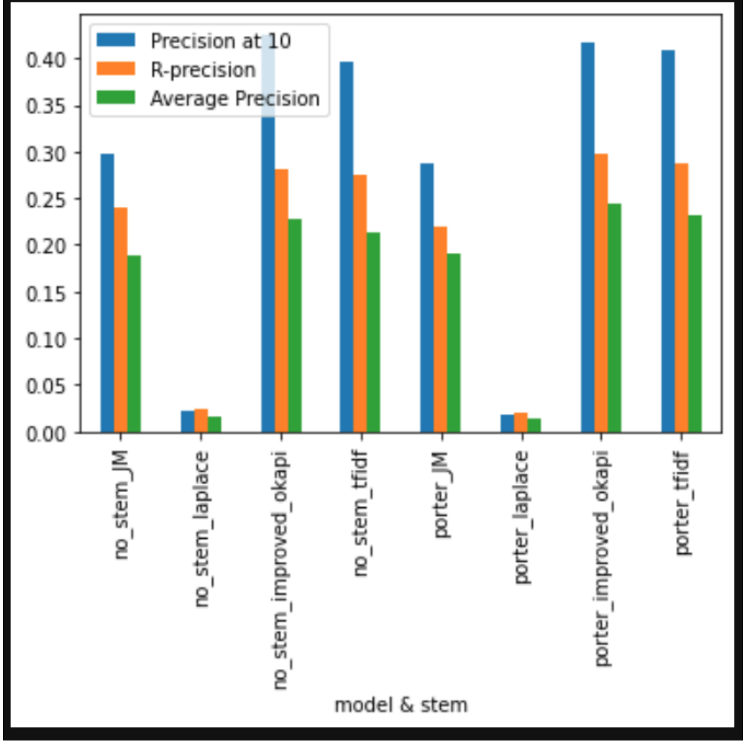


Fig.12

- Considering **average precision**, we group by each model to examine how stemming affects performance, as shown in Figure 13.
- As we can see in Figure 13, in most cases stemming improves performance, but sometimes stemming removes more information, leading to the worst results

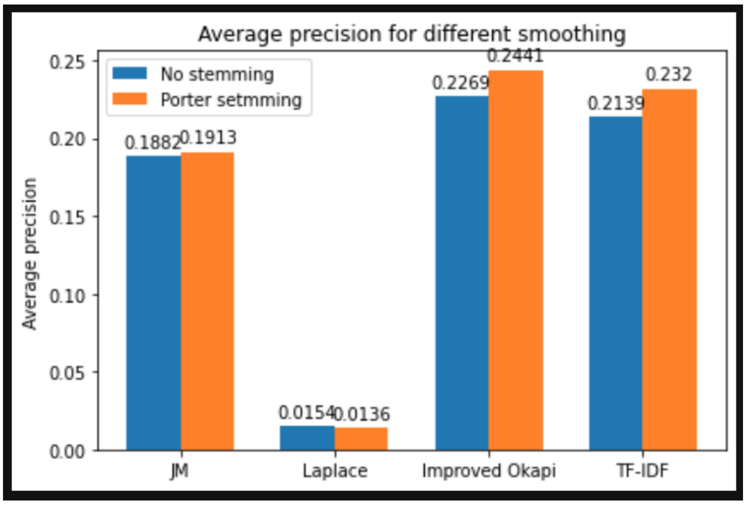


Fig.13