Andrei-Mihai Nicolae
2147392n

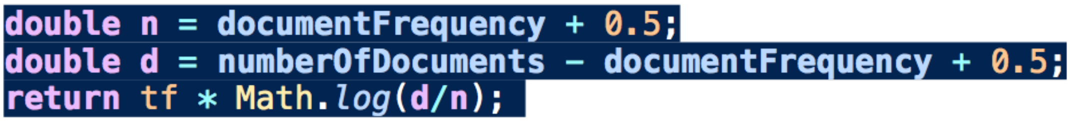
Information Retrieval
Assessed Exercise - Part II

Question 1

Number of Indexed Documents	Size of Vocabulary	Number of Tokens	Number of Pointers
807775	2043788	572916194	177737957

Question 2

The formula used in my Java code is the one included in the lecture slides as well:



```
double n = documentFrequency + 0.5;
double d = numberOfDocuments - documentFrequency + 0.5;
return tf * Math.log(d/n);
```

This is the corresponding mathematical formula:

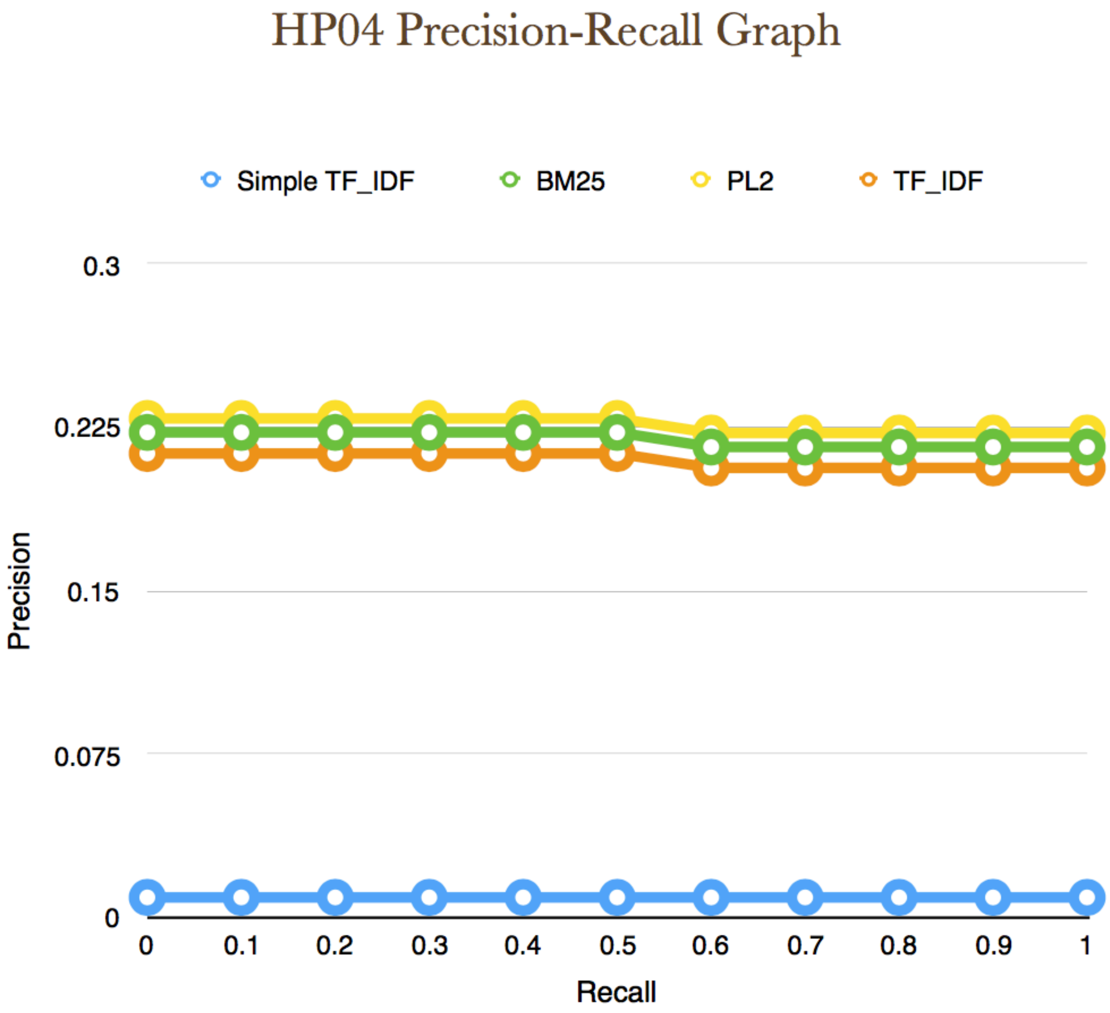
$$w_{kd} = f_{kd}\left(\log\frac{(N-D_k)+0.5}{D_k+0.5}\right)$$

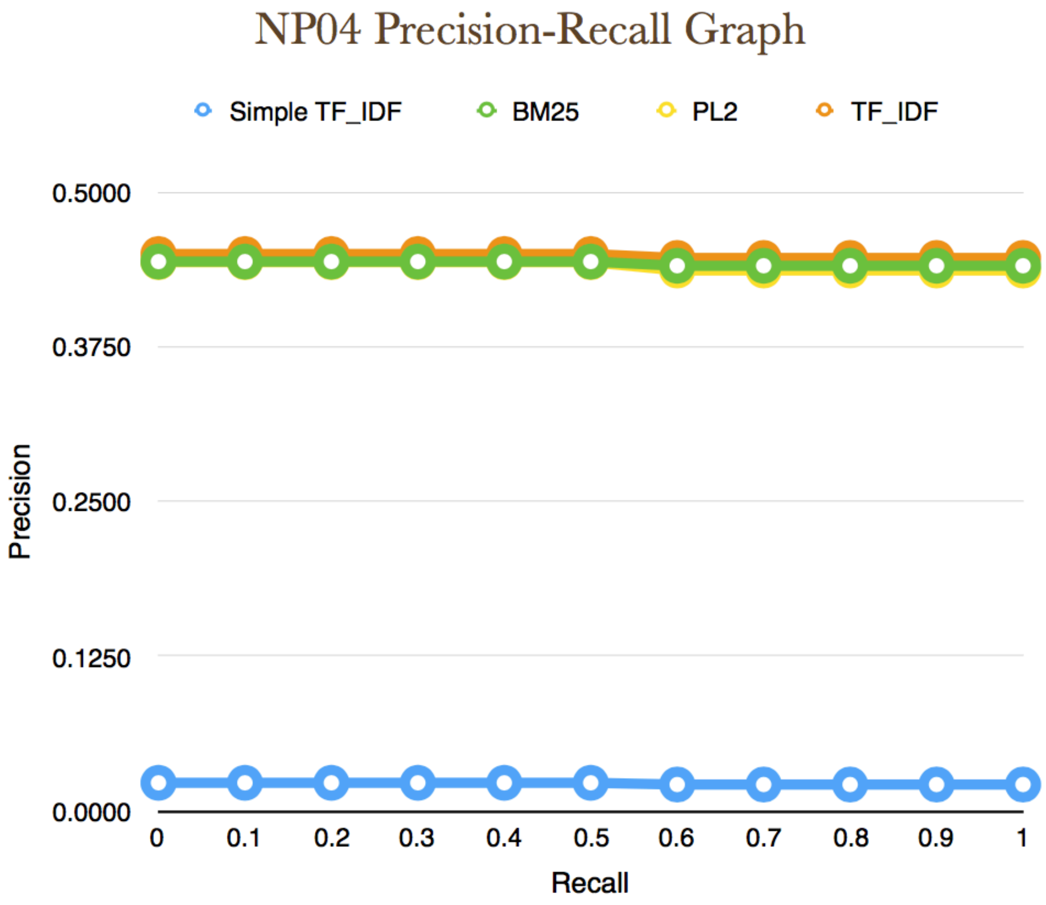
In the model shown, we are computing the weight of the kth term in the document (that is wkd). In order to achieve that, we are using its frequency in the document (fkd), the number of documents where the term appears (Dk) as well as the number of documents (N). This model also avoids the case where Dk could be 0, thus we add 0.5 to it.

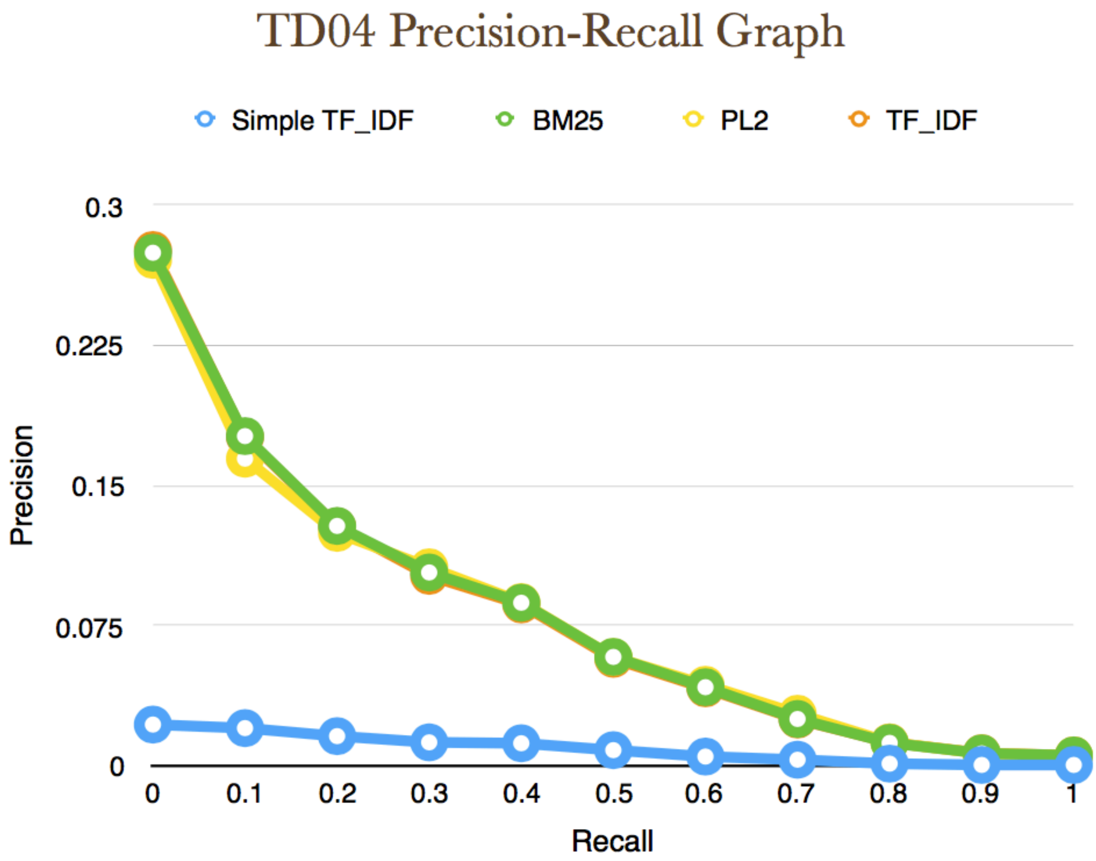
Question 3

The following table and graphs show performance across various weighting models (on the three topics provided):

The Mean Average Precision over the 3 topics for different models				
	HP04	NP04	TD04	AM
Simple TF-IDF	0.0093	0.0223	0.0078	0.0131
TF-IDF	0.2089	0.4477	0.0698	0.2421
BM25	0.2186	0.4416	0.0703	0.2435
PL2	0.2251	0.4392	0.0695	0.2446




NP04 Precision-Recall Graph


TD04 Precision-Recall Graph

As we can see in the graphs below, the simple TF_IDF that was implemented for Question 2 is far less performant than the other three competitors provided by Terrier.

The differences between BM25, TF_IDF and PL2 are very small, however the minor discrepancies that we can notice are:

- In topic distillation, it is extremely hard to distinguish which performed better; however, the winner seems to be BM25
- In name page findings, TF_IDF looks like the most performant model
- In homepage findings, PL2 was above the other two

Even though the differences seem extremely small, after careful calculations, it appears that PL2 is the most performant weighting model overall, having an MAP of 0.244, followed by BM25 and TF_IDF (0.243 and 0.242).

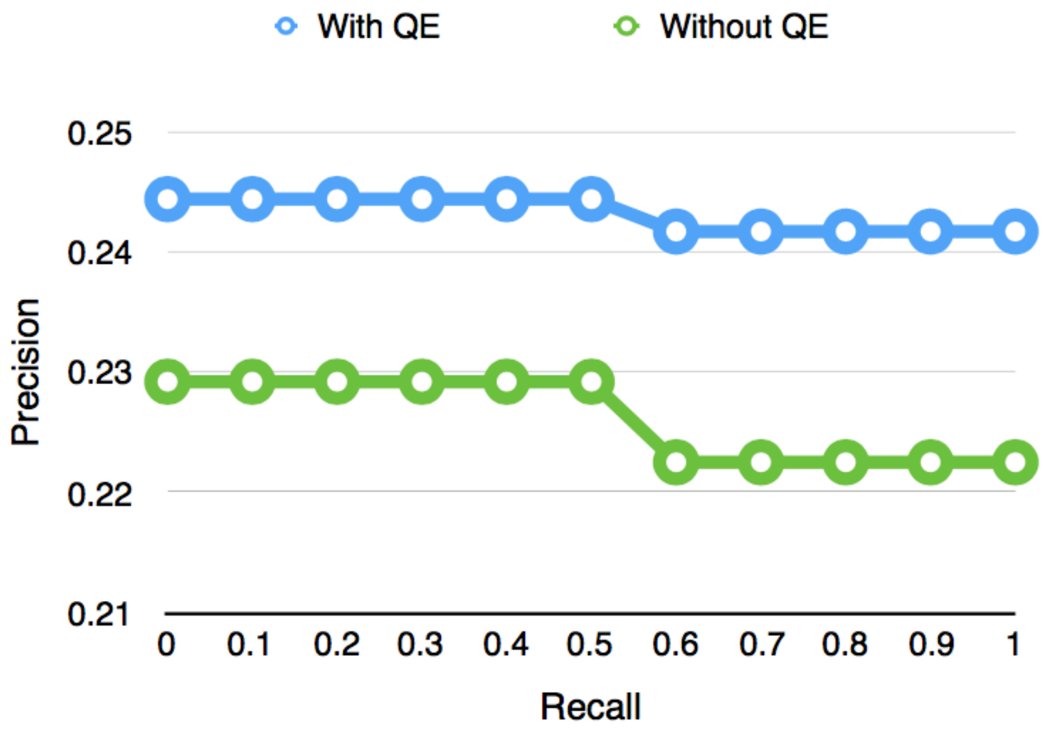
The conclusion that needs to be drawn here is, however, that a simple model will never be more performant than the more complex, well-built models such as the ones provided by the Terrier platform.

Question 4

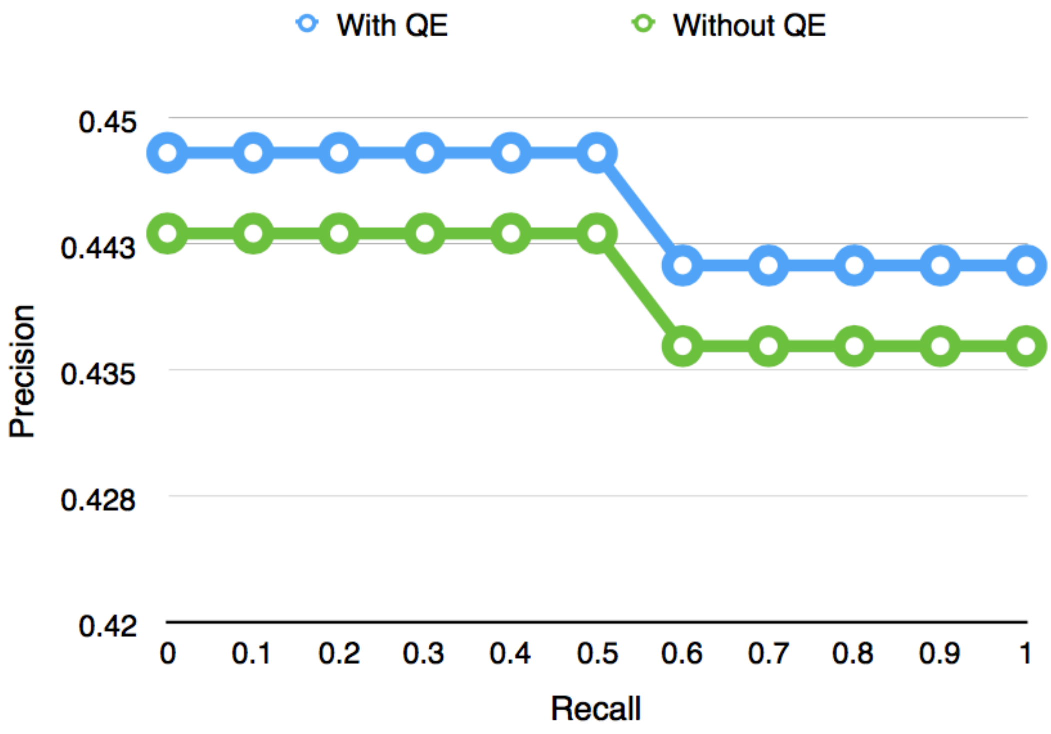
After concluding that PL2 was the most performant model overall, we will analyse below how it performs with/without the Query Expansion.

	HP04	NP04	TD04	Average (Overall)
Without QE	0.2251	0.4392	0.0695	0.2444
With QE	0.2423	0.4442	0.0671	0.2512

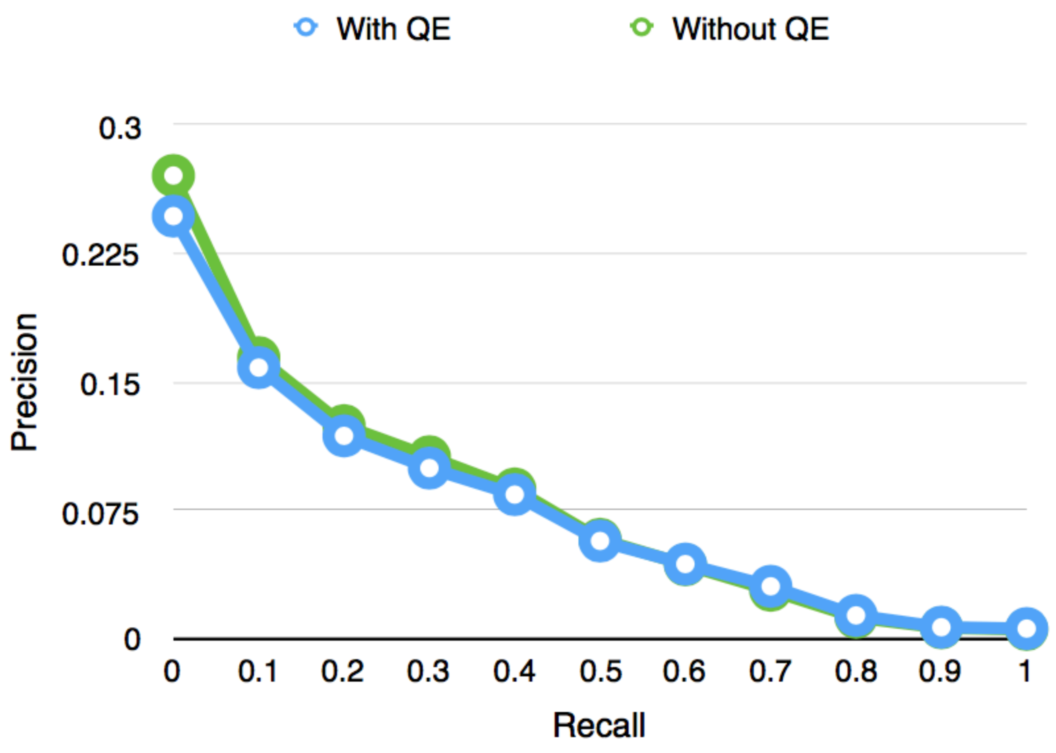
PL2 Precision-Recall HP4 (with/without QE)



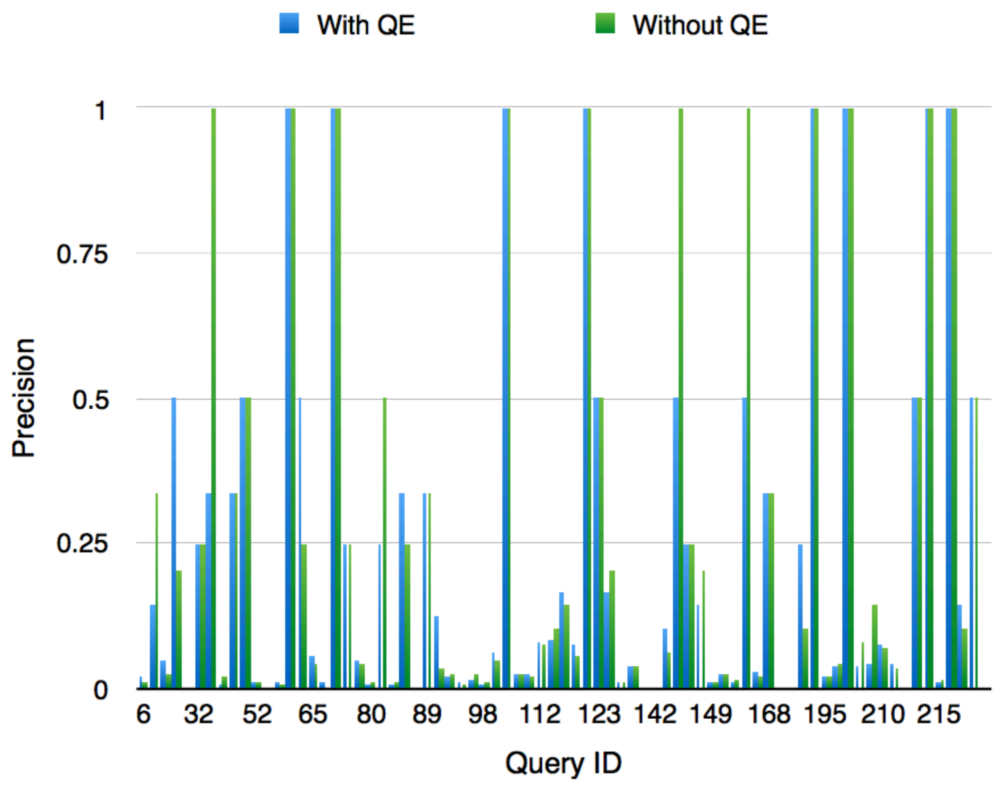
PL2 Precision-Recall NP4 (with/without QE)



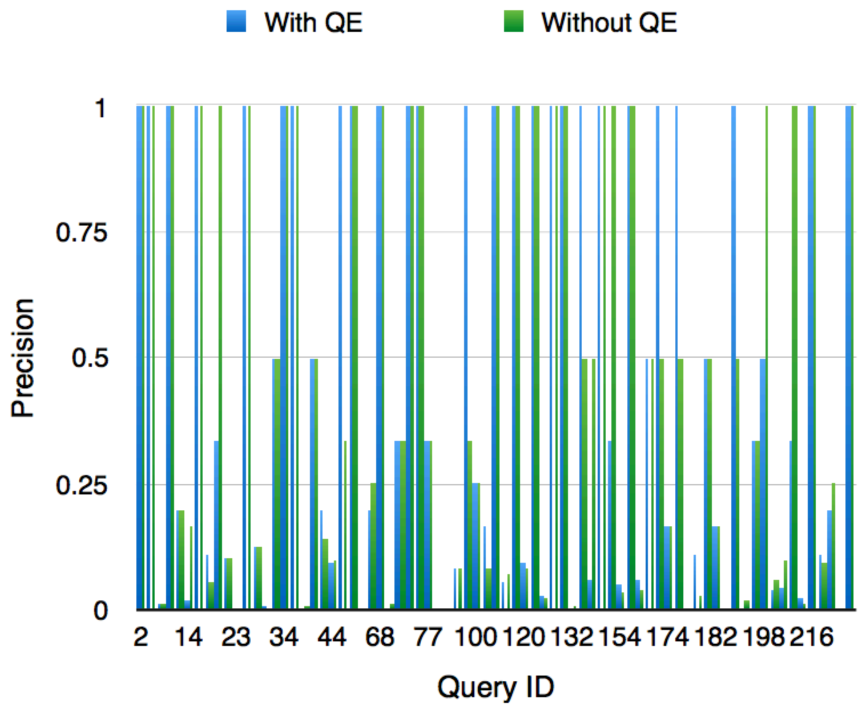
PL2 Precision-Recall TD4 (with/without QE)



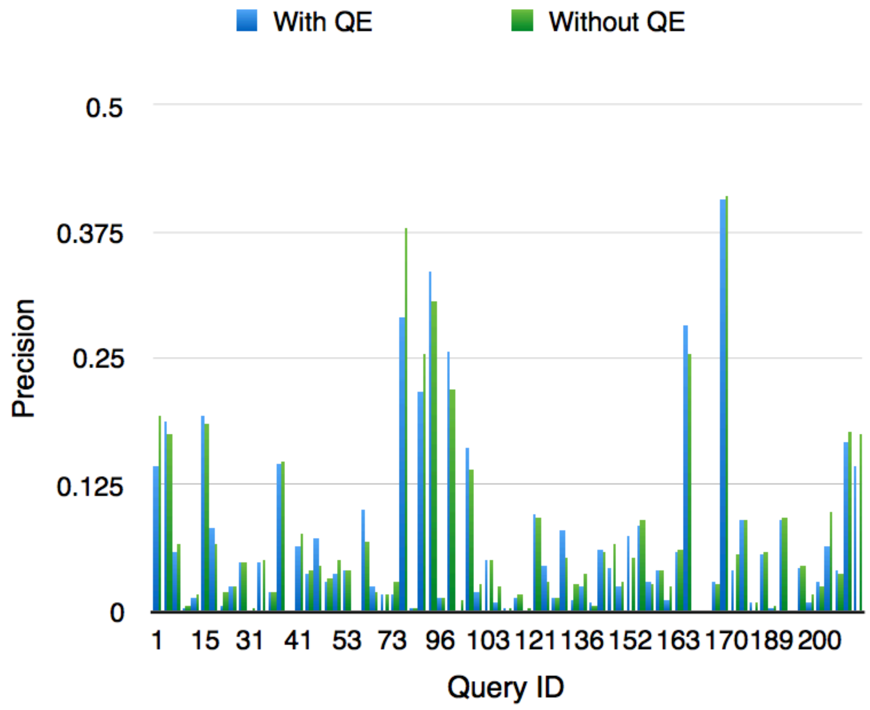
PL2 Avg. Precision Per-Query HP4 (with/without QE)



PL2 Avg. Precision Per-Query NP4 (with/without QE)



PL2 Avg. Precision Per-Query TD4 (with/without QE)

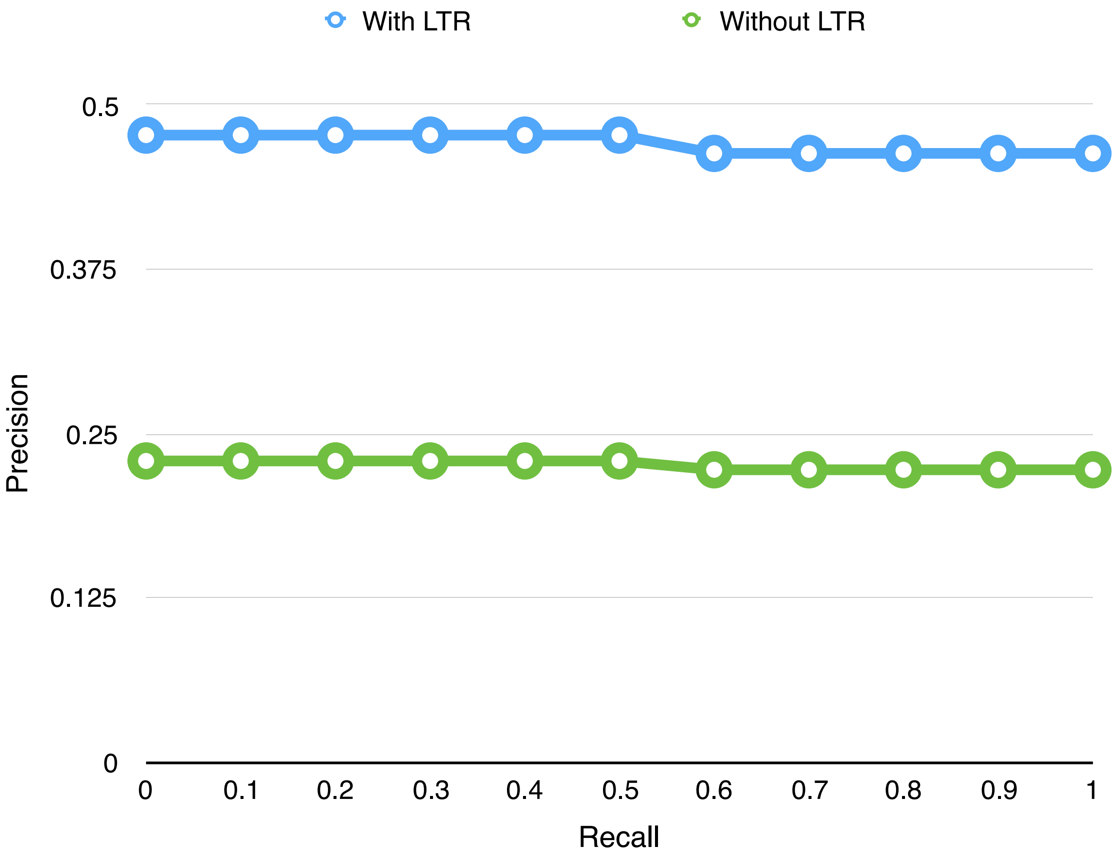


There can be many variations seen and, sometimes surprisingly, some values can be favourable when not using query expansion.

Big differences in the above graphs are noticeable as the PL2 performed quite bad on the TD topic, but much better on NP and HP.

In the end, the algorithm worked better with the query expansion as seen in the graphs than without it.

Question 5



Weighting Model	Mean Average Precision
PL2 with Baseline LTR	0.4690
PL2 without Baseline LTR	0.2251

As we can see from the graph and results table presented above, when the Baseline LTR approach is used, we have drastic improvements in performance. One of the reasons we have a relative 2x precision bump is that the LTR applies features, gets the set with most relevant documents and learns from training data using a re-ranking loss function in order to achieve best outcome. On the other hand, the PL2 generative approach is more primitive in this regard, as it relies on the closeness of terms' co-occurrences (compared to the machine learning techniques used by the LTR to not assume how documents should be retrieved, but only how to optimally classify them).

Question 6

I have chosen the following two functions from "Learning in a pairwise term - term proximity framework for information retrieval" written by Ronan Cummins and Colm O'Riordan to implement my 2 Java DocumentScoreModifier classes:

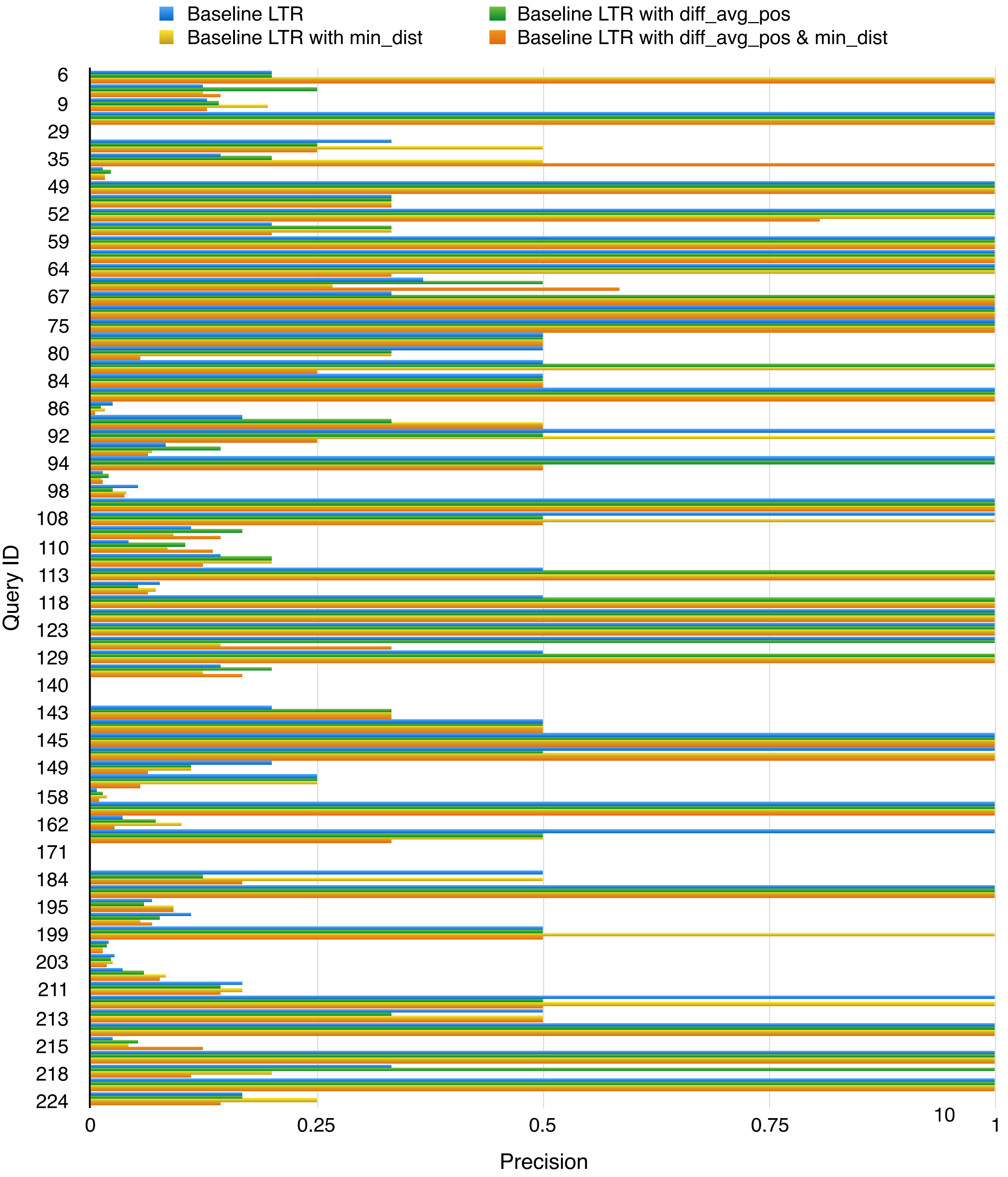
- diff_avg_pos (a, b, D): this first function presented in the paper calculates the distance between the average positions of the query terms in a certain document (i.e. a, b - query terms, D - document). diff_avg_pos performs in a way that it calculates the average position of every term and then retrieves the distances between any pair of positions returned at step 1. As it can be noticed in the implementation, there needs to be extra attention given to negative numbers because, if one did not take only the absolute value of the number, we would return wrong results.

- min_dist (a, b, D): the second function I chose was the min_dist one from O'Riordan and Cummins' paper. This one computes and retrieves the minimum distance between occurrences of query terms in a certain document (i.e. a, b - query terms, D - document). In the end, one can have the minimum distances between any possible pair of query terms retrieved (in both cases, the score of the document - for a specific query - is the mean of the distances returned by the function).

One interesting aspect of my Java implementation is handling the case where the query has only one term. In this specific event, we get no query pairs, therefore we assign the score 0 to the document with respect to the query.

These are the results using various combinations of the 2 approaches:

	MAP
Baseline LTR	0.4690
Baseline LTR with diff_avg_pos	0.5092
Baseline LTR with min_dist	0.4820
Baseline LTR with diff_avg_pos & min_dist	0.4494



As we can see above, it is clear that using Baseline LTR has good precision, but adding the min_dist feature to it gives us the best results. This points out clearly that having the terms next to each other gives the best outcome and the most relevant documents.

However, even adding the diff_avg_pos feature gives better results (in most cases) than basic Baseline LTR. We have a prioritisation on terms that appear in the same area of the document that is searched. On the other hand, if we add both of them in, we get bad precision altogether.

In the end, any of the approaches listed above (apart from inserting both features at once) gives good results based on the specific query - even basic Baseline LTR performs better in certain cases than the other approaches (one such example is the "Maryland Rural Development" query where the document is returned right away). Certainly, the aspect that performance depends most on is how the documents are organised such that the developer knows exactly what approach to use to get best performance.

Question 7

In stage I of the report we were introduced on how to setup and use Terrier, how to implement a weighting model based on TF_IDF, as well as learn about other models that were provided by Terrier (e.g. TF_IDF, PL2 and BM25). We learned how to use experiment with all these models, why the Terrier-provided models outperform the simple one we implemented, as well as why using query expansion or not can improve the performance of our applications (using it makes it better).

We also learned in stage I how to use Recall Precision graphs and histograms to analyse our results and conclude on which approach performs best. I believe Terrier and how it built, as well as the models and the whole features it provides out-of-the-box, gave us a great and easy-to-approach platform where we got our hands dirty with basic concepts of Information Retrieval. The way we had to conduct experiments and have to find out on our own which approaches are best (and why) was a valuable experience and a great add-on especially for people (such as myself) who want some practice apart from pure theory.

In stage II, we learned about LTR (Learning To Rank) and use Terrier's support for it. We again experimented with various approaches and I believe that having Cummins and O'Riordan's paper to read about different proximity features that one can use was a great experience. Also, the fact that we had to implement them in Java (and test!) was really great.

We basically tried using Baseline LTR and, afterwards, combining it with one or two of our features of choice from the paper. Again, we conducted experiments to see on our own which is the best and which specific queries. We learned that document organisation in general affects the performance of any approach, but by utilising Recall Precision graphs and histograms we could see for which query what feature was the best and why.

In conclusion, I believe the assessed coursework for Information Retrieval was a great and beneficial experience that let us see in practice how a state-of-the-art platform is used for conducting and utilising the concepts we learned in class.