[Project2 report] **Jeeho Shin**

In this project, I had to add the pseudo relevance feedback functionality to the code. As mentioned in the project document, I had to modify four java codes; **Experiment.java**, **Feedback.java**, **InvertedIndex.java**, and **ExperimentRated.java**. For **InvertedIndex.java** and **ExperimentRated.java**, I added some code in their main functions to take care of 'pseudofeedback' flag and 'feedbackparams' flag. When ExperimentRated constructor is called in the main function, the Experiment constructor is called instead because it is the super class of ExperimentRated. In the Experiment constructor, InvertedIndex object is created. Then, makeRpCurve function is called and this is where processQueries function is called. In the processQueries function, each query in '/u/mooney/ir-code/queries/cf/queries' are processed in processQuery function. The **processQuery** function in ExperimentRated.java is where I made a substantial modification. Originally, the query is used to retrieve relevant documents and nothing else is done. However, with pseudo relevance feedback, top m relevant documents should be used to reformulate the query. So, using the query, I made a **Feedback** class object called 'pseudo'. In the constructor of Feedback, **getPseudoFeedback** function is used to save top m documents relevant to the query in 'goodDocRefs'. Then, using the **newQuery_pseudo** function that I made, query reformulation is processed by using ide_regular algorithm. In the newQuery function, vector of negative documents is subtracted from the query vector. However, in the newQuery_pseudo function, I omitted subtraction part because this functionality was not mentioned in the project instruction. After reformulating the query, document retrieval takes place one more time and the retrieved documents are used to generate recall-precision curve and NDCG graph. Following the experiment settings that project instruction specifies, I plotted the recall-precision curves and NDCG graphs.

In the **amount_results_rp**, I noticed that recall-precision curves with pseudo relevance feedback were closer to upper right corner than the curve without pseudo relevance feedback (m=0). As I learned in the lecture, this implies that pseudo relevance feedback improved the performance of VSR. Also, in the **amount_results_ndcg**, I noticed that NDCG graphs with pseudo relevance feedback were above the graph without pseudo relevance feedback (m=0). From the formula of NDCG I learned in the lecture, I can infer that the performance of VSR is better when the NDCG graph is higher because it means that DCG is getting closer to IDCG. So, considering both findings, I concluded that pseudo relevance feedback improved the performance of VSR.

With the pseudo relevance feedback, VSR could know which documents are relevant to the query and this information is incorporated to the query vector. I think this is the main reason why those curves with pseudo relevance feedback showed better performance. However, I also noticed that the performance was not improved when the amount of pseudo relevance feedback passed a certain number. In the **amount_results_rp**, curves seem to move toward lower left corner when the

value of m passes 10. In the **amount_results_ndcg**, graphs seem to reach the peak when the value of m is 15 and move downward when it passes 15. Therefore, I think VSR shows the best performance for this corpus when the amount is around 10 to 15. My inference is that when the amount of pseudo relevance feedback was more than 15, many irrelevant documents to the query were included in the goodDocRefs, which deteriorated the performance of VSR.

In the **beta_results_rp**, I noticed that recall-precision curves with pseudo relevance feedback were closer to upper right corner than the curve without pseudo relevance feedback (m=0). As the beta increases, the corresponding graph is closer to the upper right. So, I can infer that the performance improves as the beta increases. Also, in **beta_results_ndcg**, most NDCG graphs with pseudo relevance feedback were above the graph without pseudo relevance feedback. It seems that the graph with beta value 0.5 is the uppermost graph among them. So, I concluded that the higher beta value usually benefits the performance of VSR and its optimal value is between 0.5 and 1.0.

The value of alpha and gamma didn't matter for the beta_results setting. They didn't matter because in the ide_regular algorithm, alpha is applied to the initial query itself and gamma is applied to irrelevant document vectors, which are not considered in our setting. I thought about some improvements I could make for the pseudo relevance feedback algorithm in this project. The first thing that comes to my mind is to incorporate the effect of irrelevant documents in the algorithm just like I learned in the lecture. By adding this functionality, VSR could give more plausible retrieval. Also, I would like to add another parameter which is going to be used to set the number of irrelevant documents. This option can be used with additional 'negfeedbackparams' flag. When user specifies a number after this flag, VSR uses this number to check pseudo irrelevant documents and incorporate their vector to reformulating query. I think this method would be useful because using all other documents except pseudo relevant documents can be inefficient when there are too many documents. By setting the number of irrelevant documents, VSR will be able to reformulate query faster.

I included the commands that I used to reproduce my results in '**JeehoShin_proj2_command list'** file, which is in '**ir**' directory. To reproduce my result, you must replace my directory name to yours. I also included four .gplot files that are used to combine graphs in one plot. To plot the graphs, you must include those four files in the 'ir' directory.