Kevin Nguyen
EID: kdn433
CS371r - Project 2

Project 2 Implementation

Project 2 is about evaluating query results and applying pseudo-feedback. Pseudo-feedback is defined as revising a query without user input. As a result, we make assumptions that the top initial retrievals are relevant. There are a total of 3 files that have been modified for an additional feature called pseudo-feedback and they're InvertedIndex.java, Experiment.java, and Feedback.java. The pseudo-feedback algorithm mimics the feedback operation from InvertedIndex.java; however, user input was removed and relevancy was computed immediately. Pseudo-feedback is performed by marking the initial top retrieved documents as good (relevant) documents. The original features of the program has not changed for this implementation, but some constructors may have been altered to account for additional arguments. This report will detail each change in the class files, how to run the program, and discussion analysis at the end.

The InvertedIndex.java still handles the indexing and basic retrieval. The main method has been extended to account for more arguments such as, new methods were implemented, and global variables have been declared that will help pseudo-feedback. The function, checkForNumeric, will consider if an argument has any numeric strings in them and if they do, then parse and get the literal value. Important values can be the values for top relevant documents to consider, alpha, beta, and gamma. Likewise, the function, getFeedbackParamValues, will set the appropriate argument values to alpha, beta, and gamma. The function, initiatePseudoFeedback, is where the pseudo-feedback begins. This function will get the relevant documents by calling a function in Feedback class and then the retrievals are revised and returned. To sum it up, initial ranked retrievals are taken for pseudo-feedback, top relevant document are considered good documents, retrieval is updated after new query has been received, and now the updated retrieval is returned.

The Feedback.java will manage most of the document relevancy and weighting. Minor additions include a global variable declaration to represent the maximum relevant documents allowed. The new function that was added is called getPseudoFeedback. This function will perform a loop on the current retrieval set and consider the first few documents up to the max relevant documents allowed as relevant. It's important to remember that once the new relevant documents have been considered a new query (a new vector) must be obtained to reflect these new changes. However, that part for the new query is already implemented in the Feedback class and called by the InvertedIndex class. Feedback.java does not have any more additional features.

The Experiment.java will handle the program entry point, call the InvertedIndex class to get retrievals, and get the graph for the rp curve. This class also has the same functions called checkForNumeric and getFeedbackParamValues which are identical to the ones in InvertedIndex.java. The reason for the duplicate methods is to allow the program to run pseudo-feedback in either InvertedIndex.java or Experiment.java as the program entry point.

There have been no drastic additions to this class other than global declarations and expanded argument considerations within the main method and constructors respectively. Most of the rp graphing functionality has been already implemented and it wasn't altered in any way. It's important to note that the goal is to get closer to the upper right hand corner of the resulting graph because that is where recall and precision is optimized.

To use the program, ensure that the extracted files are in the designated folder. In the command line, "javac *.java" to compile all java files. Then, type in, "java ir.eval.Experiment -pseudofeedback m /u/mooney/ir-code/corpora/cf /u/mooney/ir-code/queries/cf/queries/ outfile," where m is the value of the maximum number of relevant documents and outfile is the name of the output file. The output file is a gplot file and it can be converted to a pdf file for viewing by typing into the command line as follows: "gnuplot outfile.gplot > result.ps" then "ps2pdf result.ps," where result is the name of the pdf file and ps file. The result pdf will show the varying plots of data with different inputs; as a result, the graph will not show identical plots. It's important to notice that the algorithm is not properly optimized. As a result, the system could have produced much better results on a more efficient implementation.

The data shown in some of resulting graphs indicates that pseudofeedback improves retrieval performance. It's likely because of the initial set of retrievals having high precision and marking the top documents as relevant improves the recall over long documents. However, precision may have decreased, but the overall curve moves closer to the optimal results of high precision and recall in the output graph. The best value for pseudo-feedback retrieval seems to be between 5 and 10 because the graph shows better results between those values. The initial retrieval results are already sorted when pseudo-feedback was applied; as a result, the top handful of documents are relevant. When we mark the top documents as relevant and they are correct documents, it will increase the recall and precision curve. In fact, it's more likely to indicate better recall curves. than the original that was a slope downwards. Higher values for pseudo-feedback appears to decrease the precision even further and it could be the result of marking too many documents as relevant. As a result, the system could have developed a partial bias of the top retrieved documents as always relevant. The goal is to improve the system retrieval with high precision and recall without forming any concern or bias. This reason explains why the remaining documents aren't marked as bad. Overall, it seems to appear that pseudo-feedback generally improves the recall aspect more than the precision. The differing beta values also affected some of the plotted points on the graph. The beta values are used to normalize the good document length. As a result, the lower beta values will show only a slight improvement to non-pseudo-feedback. Likewise, higher beta values will further optimize the results with better recall (mostly). We don't change the values of alpha and gamma because they're trivial. Alpha values normalizes the query is not necessary to change. Gamma normalizes the bad document length; however, we do not mark any documents as bad due to forming a bias retrieval. We can make some improvements to the pseudo-feedback so it can produce better results. The system is dependent on how the initial retrieval is well executed. It could be possible to use the initial retrieval precisions to set the value for pseudo-feedback. The precisions show the accuracy of relevant documents out of all documents. The value, precision, can be used to mark documents as relevant up to the precision value. However, it's only a possibility for improvement since no retrieval system is entirely perfect.