Information Retrieval **Chang Liu – cliu70**
Homework 2 **chang.liu@jhu.edu**

Part 3 extensions:

9. Implemented the interactive query so that user could enter their own query in the console. My program will then search the whole doc set and return the document list ordered by similarity. Each time user enter a query, I will dynamically convert the inputted raw query into both tokenized term set (use own implemented tokenizer) and stemmed term set (use nstemmer), this could be used in conjunction with another extra extension I made so that my interactive query function will work under both stemmed and unstemmed initialization which chose by the user at the program startup.

4. Augment the term set to include all bigrams in the document/query that do not contain stopwords. Each time a user chose to use bigrams set at the program initialization, my program will scan through both the stemmed and tokenized term set, add the consecutive words pairs in the set that are both not in the stopword list (common_word.stemmed and common_word) into an array. Then store the pair set into a new .bigrams file alone with the original set and use this new file as the term set. ***make_hist.prl*** will then calculate the frequency based on this set and store them in a new file named .hist.bigrams and use it for the further calculation.

Extra extensions:

Implemented *tokenize* in perl: located in ***interactive.prl***. This function is will convert the raw query into the tokenized format, which did exactly the same job as the provided tokenize and token1. After conversion, it will store all the results in the file. I used this function to dynamically convert my interactive query into tokens.

Implemented all the parameters permutation: At the program startup, user could specify any special parameter permutations they want to use. The program will then run using the specified parameters.

gen_bigrams.prl: This script is called when user chose to use term set that includes bigrams. It will generate a new term set file that contains both the original term set and bigram set.

Print out the full precision / recall table in part 2: My program can print out all the parameter permutations and their precision/recall at one place, which will be convenient when we need to compare performance on each permutation.

Show relevant doc: Implemented the SHOW_RELVNT function so that my program will print out all the starred (relevant) documents for a given query.