CS 4300/INFO 4300 Information Retrieval Due Dec. 4 (Wednesday)

Project 3 - Programming Questions

Instructor: Claire Cardie *Total Points: 100*

Course Policy: Assignments are due Dec. 4 (Wednesday), 11:59pm via CMS and in hard copy at the beginning of class on Dec. 5 (Thursday).
Write your NetIDs on the first page of the submitted hard copy.
Late assignments lose 5 points for each late day. Hard copies should be submitted directly to the TAs. Assignments submitted after solutions are made available will not be accepted.
Groups of two or three are required.
All sources of material must be cited. Assignment solutions will be made available along with the graded homework solutions. The University Academic Code of Conduct will be strictly enforced, including running cheating detection software.

Overview In this exercise, you'll be experimenting with various query refinement techniques and their effects on the overall performance of the search engine. As you do this assignment, remember the following:

- To get started, download *Lucene Core* from http://lucene.apache.org/core/. (No installation is necessary, but you will have to add the 3 jar files listed in the next section to the Java build path. Also, you may use any version of Lucene, meaning you may use the current installation.) Then, download *proj3.tgz* from CMS. You should be able to compile and run *EvaluateQueries.java*, after setting 2 static variables(*VERSION* and *DATA_DIR*) in *EvaluateQueries.java*.
- Use *StandardTokenizer* with *LowerCaseFilter* and *StopFilter* (using *STOP_WORDS_SET* in *StandardAnalyzer*) when you process the query and the documents. Use *PorterStemmer*(in org.tartarus.snowball.ext package) for stemming.
- **You may ASSUME anything when you see underspecified instructions, as long as you explicitly state your assumptions in the report.** We hope everything's clear, though!

Problem 1: Query Refinement [100 points]

1. **[20 pts]** Identify all stem classes (a set of words that share the same stem) of the words in CACM collection provided to you in the project package. Group stem classes by the number of elements in the stem class, and report the number of stem classes in each group.[1]

[1] In other words, for $\forall i \in \{|S| : \text{S is a stem class of words in CACM collection}\}$, report the number of stem classes consisting of i words.

2. **[10 pts]** Manually split stem classes that have more than 10 elements into smaller clusters based on their semantics. (e.g. Usually "police" and "policy" are in the same stem class, even though they are semantically different. In such cases, split them into different clusters.) Report the resulting clusters, and for each cluster, briefly explain the characteristics of the words that comprise the cluster.

3. **[30 pts]** Write a *main()* method in *WordCluster.java* to "output" results for this problem. The Computation should be done in other existing/new methods. (See the bottom of this assignment specification for what needs to be submitted.)

   (a) Complete the following table for the 10 words in the collection most strongly associated with the word "computer". (MI = Mutual information, EMI = Expected Mutual Information, χ^2 = Chi-square, Dice = Dice's coefficient)

MI		EMI		χ^2		Dice	
Word	Score	Word	Score	Word	Score	Word	Score

   (b) Split all stem classes found in part 1 by doing the following: Given a stem class, (1) Construct an edgeless graph where vertices represent words in the stem class (2) Connect 2 vertices with an edge if and only if the Dice's coefficient score of the respective words is over a threshold T (Test a few values, and report the result using what you think gives the best clustering. Explicitly state the T you use for the reported results.). (3) Set each connected component[2] as a cluster. (Vertices with no edges attached form its own cluster.) Report the resulting clusters for stem classes that originally had more than 10 elements. Compare the results to that of part 2. [Use unlimited window size (i.e. each document is a window) when computing association scores.]

4. **[40 pts]** Report the performance(P@5) of the search engine by running EvaluateQueries.java under the following settings: [Be sure to only modify the queries. Other components of the search engine should be untouched.]

   (a) Baseline (Run the code as given to you.) Briefly explain the preprocessing done by the search engine on the query side and on the indexing side.

   (b) Replace each word in the query with its stem. Does this outperform the baseline? Why or why not?

[2] You may want to review elementary graph theory.

(c) Expand each query with the stem class of each word (i.e. for every word in the query, append all words in its stem class to the query). Compare the results to the baseline.

(d) Repeat Part c, but this time, make use of the clusters found in Question 3 Part b instead of stem classes. Report the results using all 4 association measures mentioned in Part 3, not just Dice's coefficient. [You are expected to test and report results using various thresholds, but you may keep the window size constant.] Analyze the results. [The analysis should mention specific queries in the dataset as examples. The whole analysis should fit in a page.]

Checklist for submission.

- A report that answers the above questions. You should have written answers whenever the question explicitly says to "report", "explain", "compare", or "complete".
- A source folder, which contains all the code. Compile and running *EvaluateQueries.java* should output the best setting & results for problem 4 (Explain in the header comment what needs to be done to get results for other settings, e.g. "comment out line X and uncomment line Y", if you can't get the code to output results for all settings.). *WordCluster.java* should output the results for problem 3 part a and b.