College of Computing and Information Science
Northeastern University
Spring 2019

CS6200/IS4200
Information Retrieval
April 2, 2019

Final Project Requirements

Overview

You have been introduced to the core information retrieval concepts and processes throughout the course of this semester. In this project, you will get to put these into practice by building and using your very own retrieval systems!

Goal

Design and build your own information retrieval systems and evaluate and compare their performance levels in terms of retrieval effectiveness.

Dataset

CACM test-collection which is comprised of the following:

1- Textual corpus: cacm.tar.gz (3204 raw documents)
2- Queries (64 unprocessed in cacm.query)
3- Relevance judgments (cacm.rel)
4- Stoplist: common_words.txt

Note: For Task 3 Part b), use the stemmed version of the corpus in cacm_stem.txt, and queries from cacm_stem.query.

Teams

Teams of **2 or 3 members** are to be formed. Declare team members on the designated Piazza post by Thursday, April 4th, 5:00pm. Once formed, teams cannot be altered.

Milestones

April 2nd: Release of the online description for the project
April 4th by 5:00PM: Team declaration due
April 10th by 11:59PM: Design document detailing scope and design choices
April 20th by 11:59PM: Project & report (implementation & documentation) submission due.

Assessment

The project will be graded out of 100 points and then scaled to 20% of your overall grade (see syllabus for course grade details)
Implementation: 75 points (detailed point breakdown in project task descriptions)
Documentation: 25 points. Project submissions lacking documentation (report) will NOT be accepted and hence will NOT be graded at all.
Extra credit: 20 points: All or nothing. Awarded credit applies to project & home work.

Academic honesty: If you get help from others, you must write their names down on your submission and explain how they helped you. If you use external resources, you must mention them explicitly. You may use third party libraries, but you need to cite them.

Implementation – Phase 1 :: Indexing and Retrieval

Task 1 (15 points): Build your own retrieval systems:

a) From scratch! (You may re-use your indexer and searchers from HWs 3 and 5)
b) Using Lucene: an open source library that provides indexing and searching functionalities (you may re-use your code from HW4)

Task 1 Output: *Four baseline runs*.
Setup: Implement any two of the following retrieval models combined with your word unigram indexer.

- *Tf/idf*
- *Binary Independence Model*
- *Query Likelihood Model (JM smoothed)*
- *Query Likelihood Model (Dirichlet Smoothed)*

The third run uses your *BM25* model from HW5. The fourth run uses Lucene's default retrieval model. The top 100 (at most) retrieved ranked lists (one list per run/retrieval system) are to be reported.

Task 2 (15 points): Pick *one*[1] of the four runs above and perform *query enhancement* using *two distinct* approaches. You may use the any of the suggested approaches below or adopt ones that are not listed here. Make sure to properly cite related literature and resources. Justify your design decisions, technical choices, and parameter setting and back them up with demonstrated evidence from literature and/or experiments whenever applicable.

a) query time stemming
b) pseudo relevance feedback (you may re-use or improve upon your work from HW4)
c) semantic query expansion using Thesauri/ontologies

Task 2 Output: *Two* runs using one of the base search engines with each of the two distinct query expansion techniques.

Task 3 (10 points): Perform the following on *two* baseline runs of your choice:

a) Stopping (using common_words.txt) with no stemming.
b) Index the stemmed version of the corpus (cacm_stem.txt). Retrieve results for the queries in cacm_stem.query. Examine and analyze the results for three queries that you find interesting.

Note: When parsing the documents, you may ignore the digits that commonly appear in the end of the documents' content.

Task 3 Output: *Four runs:* using *two* baseline search engines *and* two variations (with stopping, with the stemmed corpus and stemmed query subset).

---

[1] In practice, it is advised to perform Tasks 2 and 3 using all four base search engines from Task 1 to obtain a broad view of indexing strategies and retrieval models combinations. This, however, is not mandatory for this project

Implementation – Phase 2 :: Displaying Results (15 points)

Implement a *snippet generation* technique and *query term highlighting* within results, in one of the baseline runs. It is for you to figure out the techniques. However, you are required to back up your choices with the algorithm(s)/ technique(s) details and cite the respective literature.

Implementation – Phase 3 :: Evaluation (20 points)

By now, you should have *eight* distinct runs with results for all 64 queries. Namely, 4 baseline runs, 2 query enhancement runs, and 2 stopping runs (we're not counting the stemming runs here).

Produce one more (ninth) run that does one of the following:

a) Combines a query expansion technique with stopping; or
b) Uses a different base search engine than the one you chose earlier, and adopts either a query expansion technique and/or stopping.

It is now time to assess the performance of your retrieval systems (runs) in terms of their effectiveness. Implement and perform the following (do NOT use TREC-Eval):

1- MAP
2- MRR
3- P@K, K = 5 and 20
4- Precision & Recall (provide full tables for all queries and all runs)

Note: Queries that don't have any entries in the relevance judgment should be excluded from evaluation.

Documentation (25 points):

a) ReadMe.txt: Explain in detail how to setup, compile, and run your project.
b) Report NOT to exceed 2000 words[2] in PDF format, named as follows:

firstNameInitialLastName1_firstNameInitialLastName2[_firstNameInitialLastName3].pdf

Please follow this structure:

i. First page: Project members' names, course name and semester, instructor name.
ii. Introduction: Short description of your project, detailed description of each member's contribution to the project and its documentation

[2] This document's word count is about 1000 words

iii. Literature and resources: overview of the techniques used (chosen query refinement approach, snippet generation approach) citing the scholarly work and research articles you used to back up your technique and algorithm choices.
iv. Implementation and discussion: More thorough description of your project and design choices. Include query-by-query analysis in this section.
v. Results: tables reporting all results obtained for all runs and queries for all required metrics. For query level results, please provide tables/spreadsheets, too.
vi. Conclusions and outlook: state your findings, observations and analyses of the results. Which system do you think works best? Why? For "outlook": write a few sentences stating what you would envision doing to improve your project, what other features you would choose to incorporate.
vii. Bibliography: citations and links to resources and references
viii. Don't forget to submit your code! Make sure to include intermediate result files (e.g. estimated language models)

Extra credit (20 points):

This part is optional and is all or nothing. Awarded extra credit points apply to project and homework.

1. Build a search engine based on the *Relevance Model* using pseudo-relevance feedback and *KL-Divergence* for scoring. Provide the top 100 results (Phase 1, Task 1) and results from the query enhancement (Phase 1, Task 2). In addition, repeat the evaluation step (Phase 3). Compare the results with those obtained from the 4 base runs.

OR

2. Design and implement a query interface that is tolerant of spelling errors in the query terms. Given a query term that is not in the index, your program should provide a ranked list of up to 6 possible corrections for the user to choose from.

   You can make your own design choices. Here are some suggestions you may consider:

   - Use of noisy channel model for the spelling corrector
   - Use of unigram or bigram language model
   - Error model where all errors within the same edit distance have equal probability
   - Consider only terms within an edit distance of 1 or 2 from the potentially misspelt query term.