

CSI 550 Term Project Suggestions Fall 2009

Students will develop variants of information retrieval systems and test them on an official dataset before the end of the semester.

1. *Programming projects – Lucene extensions*

All projects below require writing a working program that solves a specific problem and writing a brief description and documentation. Programs need to be demonstrated to the instructor at the end of the semester (e.g., as part of the final workshop presentation). During Fall 2005 and 2006 semesters, CSI 550 students have adapted Lucene IR system (open source search engine) to work on TREC data format. This version of Lucene is now a common resource for this class and must be used in any of the projects below. All projects below will add new capabilities to the baseline Lucene system in order to improve its performance. All resulting expanded systems will be evaluated on TREC data and compared to the baseline performance of Lucene against the same data. This will involve running the system on a prepared set of queries against a prepared database. All programming must be implemented in Java using methods compatible with the main system.

- 1. Term weighting strategies:** Additional variants of $tf*idf$ term weighting scheme into Lucene system, including the pivoted normalization. Baseline Lucene system uses basic $tf-idf$ term scoring. Other terms scoring techniques can be added to improve the performance. For example, TREC experiments have shown that pivoted normalization scoring improves precision by 20% on average.
- 2. Passage-based scoring:** Passage level scoring and document ranking based on passage scores; option to return top N passages. Normally documents are scored on the combination of query terms occurring in them. You need to modify this approach by dividing documents into smaller passages and implementing a 2 (or more) tiered scoring system where larger units are scored by combining scores of smaller units contained in them. The advantage of this approach is localization of scores which may produce more search precision. You need to experiment with various options of subdividing documents and combining the scores to maximize the overall performance gain.
- 3. Passage retrieval:** Modify the system so that it returns relevant passages rather than entire documents. You must decide how to select and score passages within a document. You may consider indexing the data collection by passages or dynamically extracting passages from documents at retrieval time.
- 4. Pseudo Feedback:** Pseudo feedback based on top N relevant passages. Unlike user-in-the-loop relevance feedback, this approach assumes that the top N retrieved items are relevant and uses this assumption to modify the query and do another search. You should consider ways of maximizing the performance gain by exploiting document vs. passage based feedback, selecting an optimal N, varying N for different queries, making one or more iterations, sampling for non-relevant documents, etc.
- 5. Linguistically based indexing:** Implement an extension where certain linguistic objects such as noun phrases, verb phrases, proper names, etc. are also used for

indexing and retrieval. You will require some linguistic processing software such as a part-of-speech tagger, phrase chunker, or an English parser. These are available on line and may also be available at the lab.

6. **Query expansion techniques:** Use the large lexical database known as Wordnet (available on line) to automatically expand search queries with synonyms, hypernyms, hyponyms, etc. in order to obtain better topic coverage and increase document recall while keeping precision steady. You can use any additional on-line resources, including the Internet itself (i.e., web search, sampling, etc.).
7. **Generating effective queries:** The starting point is a narrative specification of information need as used in full TREC queries and their various sections. The goal is to convert this narrative into the most effective search query that would maximize recall and/or precision of the results. A good measure here would be recall and precision at N retrieved documents where N varies from 10 to 200, for example.
8. **Document classifier.** In some situations it may be more natural to consider a routing problem in which documents from the database are assigned to one or sometimes several queries. In other words, the retrieval system considers all queries at once, not one at a time like in ad-hoc retrieval. This approach may provide an additional advantage of selecting the best fitting query, but it is usually limited to a fixed number of standing queries or profiles.
9. **Passage based query expansion.** In pseudo-feedback queries are expanded using terms selected from returned documents or passages. In this approach, queries are expanded using entire text passages rather than only certain terms. Because of possibility of introducing significant noise to the expanded query, selection of expansion passages is a very sensitive task. You will need to experiment with techniques to select the optimal passage(s) and their lengths to maximize the overall performance.
10. **Fusing results from multiple searches.** One way to improve retrieval results is to fuse (or merge) several retrieval results for the same query in order to obtain a single combined ranking. The goal is to have the merged ranking with a better precision and recall than any of the individual rankings. This fusion technique can be applied to outputs of different search systems (e.g., Lucene, InQuery, Smart) or to significantly different runs produced by the same system. Specifically, runs produced by any of the projects described above can be used as inputs to the fusion program. You need to consider various ways of computing final document scores in order to maximize the overall performance.
11. **Pseudo feedback.** Implement an extension to Lucene which would allow the system to perform retrieval feedback operations without human intervention. An effective method of doing pseudo feedback is to use highly scored passages or even sentences from the retrieved documents and use them to revise the original queries. The revised queries are then resubmitted to search and their results are returned. This feedback loop can be performed more than once to see if results can be further improved and how.

- 12. Cross-lingual retrieval:** Implement an extension to Lucene that would allow it to search foreign language data with English language queries. This project requires further specification of the scope and must be discussed with the instructor ahead of selection.

2. Programming projects – web search

- 13. Web crawler and indexer:** Write a simple single-threaded web crawler that would find new pages on the web and pass text found in them as documents to be indexed by Lucene. Starting from a single input URL the crawler should download a page, then wait at least 5 seconds before downloading the next page. Your program should find other pages to crawl by parsing link tags found in previously crawled documents.

3. Other possible programming projects

These projects are performed as components and add-ons to a retrieval system. For example, term associations can be used to augment query expansion. Students selecting any of the projects below **MUST** partner with a Lucene project (1-10) in order to produce their results.

- 14. A self-learning concept spotter.** Information extraction is a technique to locate references to certain types of entities in text, such as people or organizations or locations, etc. Given a simple rule for extracting one kind of entity, and a large text corpus, implement an automatic method for expanding this rule to extract all entities of this kind. Discovered concepts can be used to augment the collection of linguistic objects used in linguistically motivated indexing.
- 15. Automatic summarizer.** Write a program that would automatically produce a summary of a text document. The summary must be of a specified length and must be readable and capture the most important aspects of the document. The summary can be either generic (most important content) or topic-biased (what is most important on a given topic). The output of a summarization program can be used in passage-based query expansion.
- 16. Computing term associations from text.** Given a large text corpus in a certain domain, implement a system that would compute correlations between words and other terms in this corpus. For example, you may discover that “account” is related to “money” and that “president” is related to “chief” etc. The goal is to make the program produce as many associations as possible while making sure they are all acceptable within the domain. Term associations produced by this project can be used as an additional lexical resource to augment Wordnet relations with domain-specific associations.

3. Evaluation projects

Additional projects not involving programming may be undertaken by students who do not have strong programming preparation. These projects must be negotiated individually with the instructor. Some possible ideas are listed below:

- 17. Relevance feedback & manual query reformulation experiments.** Perform various experiments with user-in-the-loop relevance feedback

18. Interactive information retrieval. Perform user studies with task-oriented searches involving series of queries.

19. Collaborative information retrieval. Perform studies of users working in groups to complete a joint information task.

4. Other projects

Other project selections can be negotiated with the instructor. Students may propose well thought out project ideas for instructor approval.