Relevance Feedback and Query Expansion
October 17, 2012

1. **Module name**: Relevance Feedback and Query Expansion
2. **Scope**: This module introduces the methods to improve the recall of information retrieval systems, mainly focuses on relevance feedback and query expansion.
3. **Learning objectives**
   Students are expected to be able to:
   a. Explain the idea of relevance feedback and the basic procedure of the implementation of relevance feedback.
   b. Design the scheme of query expansion and apply it in real IR systems, such as certain web search engines.
   c. Analyze the documents stored on LucidWorks, compute the similarity score between the documents and figure out the query log using specific CURL command.
4. **5S characteristics of the module**
   a. Streams: In most collections, the same concept may be referred to using synonyms, so the text stream in documents and/or queries can be expanded.
   b. Structures: Indexed documents are stored and retrieved in various formats (e.g., XML and JSON). The input query also can be in a specific format.
   c. Spaces: Documents and queries are represented in a vector space as a result of indexing. The documents are located on the server running LucidWorks software.
   d. Scenarios: Scenarios includes user interaction, similar page function provided by some web search engines, query assist, etc. Relevance feedback and query expansion can improve the recall of information retrieval systems.
   e. Society: Students studying the relevance feedback and query expansion of information retrieval. Users of the system that provides these services.
5. **Level of effort required**
   This module should take at least 4 hours to complete.
   a. **Out-of-class:** students are expected to spend at least 4 hours to complete the module and exercises. Time should be spent reading the material from the textbook and Lucene chapters, as well as revisiting the resources [12a, 12b].
   b. **In-class:** students will have the opportunity to ask and discuss exercises with their teammates.
6. **Relationships with other modules** (flow between modules)
   a. Overview of LucidWorks big data software module

This module introduces the basic concepts and the overview of LucidWorks Big Data software that is specifically designed for searching, discovery, and analysis of massive content sets. We need LucidWorks Big Data software to perform the exercises.

b. LucidWorks: Searching with cURL module

This module introduces how to utilize CURL and the Query admin to search documents. From it, we can get familiar with basic CURL commands and the json format.

c. Information Retrieval System Evaluation
The module introduces evaluation in information retrieval. It focuses on the standard measurement of system effectiveness using relevance judgments. We can learn from it details of evaluation methods for information retrieval systems.

7. Prerequisite knowledge/skills required (what the students need to know prior to beginning the module; completion optional; complete only if prerequisite knowledge/skills are *not* included in other modules)

a. Overview of LucidWorks Big Data Software

b. Chapter 6: Scoring, term weighting and the vector space model

c. Chapter 8: Evaluation in information retrieval

d. Chapter 9: Relevance feedback and query expansion

e. Lucene chapter 8: Tools and extensions

8. Introductory remedial instruction

The following relevance benchmarks are widely used as the most standard test collections.

a. LucidWorks (workflows, discovery, analysis)

9. Body of knowledge

a. Relevance feedback

The idea of relevance feedback is to involve the user in the retrieval process so as to improve the final result set. The basic procedure is:

i. The user issues a (short, simple) query.

ii. The system returns an initial set of retrieval results.

iii. The user marks some returned documents as relevant or nonrelevant.

iv. The system computes a better representation of the information need based on the user feedback.

b. The Rocchio algorithm for relevance feedback

Rocchio seeks the query q_{opt} that maximizes:

$$\vec{q}_{opt} = \arg\max_{\vec{q}} [\cos(\vec{q}, \vec{\mu}(C_r)) - \cos(\vec{q}, \vec{\mu}(C_{nr}))]$$

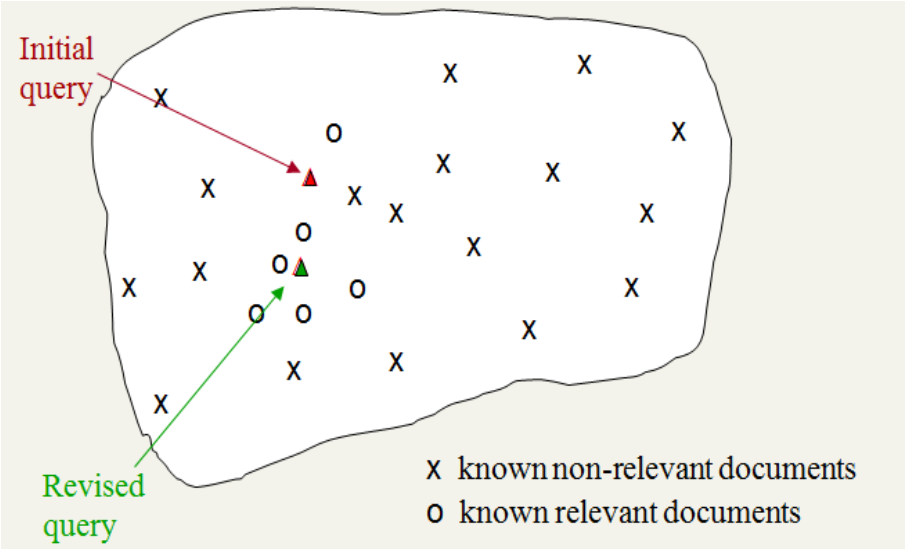
in which, C_r is the set of relevant documents and C_{nr} is the set of nonrelevant documents. Then, try to separate docs marked relevant and nonrelevant

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \notin C_r} \vec{d}_j$$

The algorithm proposes using the modified query $\vec{q}_m$

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

where q_0 is the original query vector, and D_r and D_{nr} are the set of known relevant and nonrelevant documents respectively. Further α, β and γ are weights attached to each term. Starting fromq_0, the new query moves some distance toward the centroid of the relevant documents and some distance away from the centroid of the nonrelevant documents. This new query can be used for retrieval in the standard vector space model (Chapter 6, Section 6.3). The following picture shows an application of Rocchio's algorithm.



c. Evaluation of relevance feedback strategies.

Use two collections each with their own relevance assessments: q_0 and user feedback from first collection, q_m run on second collection and measured

d. Other relevance feedback methods

i. Pseudo relevance feedback: Assume that the top k documents are relevant.

ii. Indirect relevance feedback: Clicked on links are assumed likely to be relevant.

e. Query expansion

   i. Manual thesaurus

      For each term, t, in a query, expand the query with synonyms and related words of t from the manually developed and maintained thesaurus.

   ii. Automatic Thesaurus Generation

      Two ideas: Two words are similar if they co-occur with similar words; Two words are similar if they occur in a given grammatical relation with the same words.

   iii. Refinements based on query log mining

f. Documents analysis on LucidWorks

   i. Workflows

      Workflows describe processes that should be performed on data. Starting a workflow starts a specific Job. The Workflows API is a way to update the definitions for each workflow and start the workflow job. For example:

      cURL -u administrator:foo -X GET

      http://fetcher.dlib.vt.edu:8341/sda/v1/client/workflows

      The above command will list all the workflows.

      In this module, we will only use two workflows: _log_analysis and _etl. To start a workflow, we could use an API call like following:

      http://fetcher.dlib.vt.edu:8341/sda/v1/client/workflows/_log_analysis

      To get the details of a specific workflow, say, _log_analysis, we could use the following command:

      cURL -u administrator:foo -X GET

      http://fetcher.dlib.vt.edu:8341/sda/v1/client/workflows/_log_analysis

   ii. Documents similarity

      Document similarity scores documents with a cosine similarity algorithm (on a scale of 0-1) to measure the level of similarity between two documents.

      To calculate similarities while documents are being loaded, the following attributes can be defined when starting the _etl workflow with the Workflows API if the defaults are not appropriate for your situation.

doSimdoc: instructs the Workflow to include a step to calculate the similarity of incoming documents with the other documents in the collection. The default is false. Set totrue to calculate similarity.

simdoc_threshold: defines the minimum threshold for similarity calculations on a scale of 0 to 1. The default is 0. Setting it to 0.4, for example, would discard any calculations below 0.4.

The workflow request might look like:

cURL -u administrator:foo -X POST -H 'Content-type: application/json'

-d
'{"doSimdoc":"true","simdoc_threshold":"0.2","extractType":"arc","inputDir":"/path/to/arc/files","inputType":"application/html","collection":"documentation"}'

http://fetcher.dlib.vt.edu:8341/sda/v1/client/workflows/_etl

Once the workflow is finished, documents can be requested using the Document Retrieval API. For example, the following command will find duplicates of a document with the ID of hdfs://localhost:57013/input/reuters/

reut2-021.sgm-18.txt

cURL -u administrator:foo -X POST -H 'Content-type: application/json'

-d'{"DUPLICATES":{"id":"hdfs://localhost:57013/input/reuters/reut2-021.sgm-18.txt"}}'

http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/documents/retrieval | python -mjson.tool

iii. Query log

In addition to the raw output of data, calculations can be performed on the aggregate data. Calculations are defined as aggregates in the API request, and can include sum, minimum value, maximum value, mode of values, median, count, mean, and variant values. Multiple aggregates can be requested at once. We can use these calculations to do the log mining.

For example, let's request the total queries per day for a one-month time period, and get the median daily query value. First we need to have run the _log_analysis workflow. It can be started with an API call like this:

cURL -u administrator:foo -X POST -d '{"date":"2006-03","collection":"collection1"}'

-H 'Content-type: application/json'

http://fetcher.dlib.vt.edu:8341/sda/v1/client/workflows/_log_analysis

The “date” field indicates the dates for which we would like to process log data. We also need to specify the collection, which tells the workflow where to find the log data. Once the logs have been processed, we would construct a request like this:

```
cURL -u administrator:foo -v -X POST -H "Content-Type: application/json" -d
'{"metrics":
  {"total-queries":
    {"startDate":"2006-03-01",
     "endDate":"2006-03-31",
     "aggregates":["median"],
     "includeSeries":"true"}
   }
}'
http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/analysis | python -mjson.tool
```

With this request, we have defined the metric (total-queries), the date range (startDate and endDate), the calculations to perform (aggregates), and said we'd like the raw data output as well (includeSeries).

10. Exercises / Learning activities

a. **Textbook Exercise 9.5, pg. 188**

Suppose that a user’s initial query is ‘cheap CDs cheap DVDs extremely cheap CDs’. The user examines two documents, d_1 and d_2. She judges d_1, with the content ‘CDs cheap software cheap CDs relevant’ and d_2 with content ‘cheap thrills DVDs nonrelevant’. Assume that we are using direct term frequency. There is no need to length-normalize the vectors. Using Rocchio relevance feedback, what would the revised query vector be after relevance feedback? Assume $\alpha = 1, \beta = 0.75, \gamma = 0.25$.

b. **Document similarity.**

Assume you find a document that is interesting to you, and you know the document ID is “hdfs://128.173.49.66:50001/input/reuters/reut2-021.sgm-141.txt”. It is reasonable that you want to find relevant documents that are similar to it. Give the command to fetch the top 5 similar documents.

c. **Query log**

   Give the command to find the average number of queries submitted from 2012-10-01 to 2012-10-31.

11. Evaluation of learning objective achievement

a. Give the right revised query vector.

b. Find document similarity using cURL commands.

c. Obtain the average number of queries for the time range using cURL commands.

12. Resources

a. http://lucidworks.lucidimagination.com/display/bigdata/Discovery

b. http://lucidworks.lucidimagination.com/display/bigdata/Analysis

c. http://fetcher.dlib.vt.edu:8888/solr/#/test_collection_vt/

d. CS5604F20120925Ch9.mp4 lecture located in CS5604/Resources/MP4Fall2012 on Scholar

e. Manning, C., Raghavan, P., and Schutze, H. (2008). Chapter 9: Relevance feedback and query expansion. In *Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

f. McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). Tools and Extensions. In Lucene in Action (2nd Ed.). Stamford: Manning Publications Co.

13. Glossary

14. Additional useful links

15. Contributors

Authors: Sichao Wu (sichao@vt.edu), Yao Zhang (yaozhang@vt.edu)
Reviewers: Dr. Edward A. Fox, Kiran Chitturi, Tarek Kanan
Class: CS 5604: Information Retrieval and Storage. Virginia Polytechnic Institute and State University