

An Information Retrieval System for Improving Efficiency in Scientific Literature Searches

Daniel M. Dunlavy

September 30, 2002

1 Proposal

Conducting scientific research most often involves a search through existing literature in order to avoid repeating research efforts, review methods already developed for solving a problem, gain a better understanding of a problem, etc. Typically, this search is performed using the Internet, which is a convenient portal to various databases of books, journal articles, technical reports, preprints, etc.

In using this approach for information retrieval (IR), a researcher has the advantage of being able to search through great amounts of reference material. However, along with this great access comes the challenge of efficiently retrieving and processing only relevant material. When using an information retrieval engine to search through electronic resources, simple queries can return too many documents or documents not relevant to the intended search criteria.

For example, if a physicist is researching which methods have been used to solve problems in the area of plasma physics, a search for “*methods plasma physics*” on the World Wide Web using Google (www.google.com) yields more than 27,000 documents (out of approximately 2.5 billion). The physicist could never hope to read through all of those documents. Even searching arXiv (xxx.lanl.gov), a server for preprint articles in physics, for this same information on plasma physics yields more than 350 documents (out of approximately 135,000). Still, this is too much information to process. Just reading the abstracts of those 350 articles would require a great deal of time, and there is no guarantee that this document set would be representative enough to constitute a thorough literature search.

With these issues in mind, the goal of this project is to develop an information retrieval system that

- retrieves documents relevant to a query,
- separates the retrieved documents into clusters by topic, and
- creates a single summary document for each topic cluster.

Ideally, such a system would facilitate more efficient literature searches. By categorizing and summarizing the documents, the information presented to a researcher can be reduced in size and organized in a more accessible format.

A suite of software tools, referred to as QCS (query, cluster, summarize) from this point forward, is proposed to perform these tasks. The specific algorithms to be used for each task are described in Section 2. Section 3 presents the details of the implementation of these algorithms and the QCS input/output software tools. Finally, in Section 4, the method used for validating the effectiveness of QCS will be discussed.

2 Algorithms

A set of documents can be represented using an $m \times n$ term-document matrix A , where m is the number of terms and n is the number of documents in the set. For this project, terms represent the words (white space delimited) in a document with the exception of pre-designated *stop words* (e.g. “a”, “the”, “in”, etc.) The entries of the matrix a_{ij} represent the importance (or weight) of term i in document j . The j^{th} column of A is sometimes called the *feature vector* of document j . All of the algorithms presented in this section make use of a term-document matrix.

2.1 Querying and Retrieving Documents

Queries can be represented as a feature vector $q \in \mathbb{R}^m$ in the same way each document is represented. Using the term-document matrix A , the matrix-vector product

$$q^T A \tag{1}$$

computes a vector of relevance scores s for the query q . The higher the score s_j , the more relevant document j is to the query. Scores above a given threshold can be used to identify those documents most relevant to the query. This approach to information retrieval is called the *vector space method*.

Latent Semantic Indexing (LSI) [4] is a variation of the vector space method that will be used in QCS for information retrieval. Instead of working with the full term-document matrix A , LSI uses a rank- k approximation of A (typically with $k \ll \min\{m, n\}$), computed using a truncated singular value decomposition (SVD). This approximation of A has been shown to better represent the conceptual meaning of the documents than A itself [2]. Thus, in LSI the rank- k approximation is used in place of A in (1) when computing the document relevance scores for a given query.

The feature vectors are typically very sparse (since most documents do not contain all of the terms in the entire document set), and the LSI feature vectors can be computed in parallel. The algorithms used in QCS will take advantage of both of these features of LSI. The details are discussed in Section 3.

2.2 Clustering Documents by Topic

Suppose we have a set of n points $x_1, \dots, x_n \in \mathbb{R}^m$. The problem of clustering this data into k clusters can be defined as finding k points $\{c_j\}_{j=1}^k$ (called centroids), such that

$$\frac{1}{n} \left(\sum_{i=1}^n \min_j [d(x_i, c_j)]^2 \right)$$

is minimized, where $d(x_i, c_j)$ is the distance between x_i and c_j using some distance function (typically Euclidean distance). This problem has been shown to be NP-complete, but the *k-means* algorithm gives an approximate solution to it. The *k-means* algorithm is an iterative algorithm that (1) assigns each point x_i to the nearest centroid, and (2) computes new centroids for each new group of points at each iteration.

If x_i is the feature vector of a document retrieved by LSI, we can cluster the documents using the *k-means* algorithm. Since a feature vector represents the content of a document, clustering

documents using feature vectors can be viewed as clustering documents by topic. This approach has been used as the basis for many clustering algorithms in information retrieval.

The specific algorithm used in QCS will be the *spherical k-means* (spk-means) algorithm [5]. One advantage of using the spk-means algorithm is that it employs an efficient strategy for computing the feature vector similarities (distances), which is the computational bottleneck of the classical k-means algorithm.

2.3 Multiple Document Summarization

An extract summary is a set of sentences extracted from a set of documents which summarizes the set of documents. Typically, the number of sentences in an extract summary is far fewer than the number of sentences in the original set of documents.

An extract summary for each topic cluster will be produced. This process is inherently parallelizable, and the task of producing a summary for a topic cluster can be assigned to a single processor in a parallel computing environment. If the number of topic clusters is greater than the number of processors being used, then the work will be divided among the processors so that each processor has roughly the same amount of topic clusters to process. This is most likely not the optimal assignment of work in the context of load balancing, but it will be the simplest to implement and act as a good starting point for a parallel implementation of the algorithm described below.

For each document, the sentences will be parsed using a neural network of part-of-speech probabilities and heuristic rules for recognizing sentence boundaries [1]. Next a single-document extract summary will be produced using a hidden Markov model (HMM) [3]. Finally, for each topic cluster, the set of single-document summaries will be concatenated into a single file. It is assumed that the single-document summaries will contain an overlap in content since the documents are clustered by topic. In order to extract a representative sentence from each group of sentences that overlap in content to a high degree, a term-document matrix is created using the summary sentences and a pivoted QR matrix decomposition is used to prevent overlap. The details of this approach to multiple-document summarization are presented in [6].

3 Implementation

Implementation of QCS involves the development and integration of software tools designed to pre-process the documents, provide a graphical user interface (GUI) for the system, execute the algorithms described in Section 2, and format and display the summary documents produced by the system. The computer languages that will be used to develop these tools are listed in the table below in the order reflecting the flow of data through the system.

<i>Language</i>	<i>Task</i>
Perl	Document Pre-processing
Java	User Interface
C/C++	Document retrieval Document clustering Multiple-document summarization
Java	Summary document formatting Creating dynamic HTML

3.1 Software

The documents used in QCS will be ASCII-formatted files which will have to be converted to a standard format (e.g., SGML¹). Thus, the major task will be processing text, and so the choice of Perl as the programming language is obvious, since the task of processing text is the most notable feature of Perl.

The remainder of QCS will be implemented as a client-server application. The user interface (client) should be platform independent in order for QCS to be accessible by the largest audience possible. Thus, Java was the language chosen for the GUI, as it is a language designed with cross-platform GUI applications in mind.

The computational server, handling the querying, clustering, and summarization of the documents, will be implemented using several existing libraries and code written in C/C++ and Fortran 77.

- **Query**

Parallel General Text Parser (PGTP) will be used to implement LSI. It uses MPI for parallel communication and SVDPACK for the SVD of the term-document matrix. The package is written in C++ and released under the GNU Public License (GPL).

- **Cluster**

Generalized Spherical K-Means (gmeans) will be used for the clustering of documents. The package is written in C++ and released under GPL.

- **Summarize**

GNU Hidden Markov Model Library (ghmm) will be used to implement the HMM. The package is written in C (with a C++ wrapper) and released under GPL.

Linear Algebra Package (LAPACK) will be used for the pivoted QR decomposition. LAPACK is written in Fortran 77 and is freely available.

The code required for integrating the computational server components will be written in C++. A serial prototype of the summarization algorithm has been developed in MATLAB (MathWorks, Inc.) and will be translated into C++ and use MPI for the parallel communication.

The output of QCS will be in the form of HTML documents containing summaries of each topic cluster and hyperlinks to the original documents summarized. The ability of Java to create dynamic HTML documents and its proven success in this task in many Internet applications makes Java an ideal choice for development of the formatting/output tool.

3.2 Hardware

The target platform is a PC Linux Cluster housed at the University of Maryland Institute for Advanced Computer Studies (UMIACS). The specifications for the cluster and each node is given in Table 1. I will be using the MPICH implementation of MPI for all parallel communication.

¹Standard Generalized Markup Language

Cluster Specifications	
<i>Nodes</i>	16
<i>Processors</i>	32
<i>Memory</i>	16 GB
<i>Disk</i>	288 GB Internal

Node Specifications	
<i>Processor</i>	Pentium II
<i>CPU</i>	2 × 400 MHz
<i>Memory</i>	1GB
<i>Disk</i>	18GB
<i>Interconnect Networks</i>	Gigabit Ethernet Fast Ethernet

Table 1: Specifications of the UMIACS PC Linux Cluster

4 Code Validation

A data set for testing multiple-document summarization tools has been developed for the Document Understanding Conferences (DUC). This data set contains 59 sets of documents used for evaluation in the Text Retrieval and Extraction Conferences (TREC). The TREC data is a standard test set for various information retrieval tasks. Thus, it seems appropriate to use the DUC document sets to test and evaluate the effectiveness of the QCS information retrieval system.

The DUC document sets were produced using data from the question-answering track in TREC-9 (2000). Specifically these include newswire articles from the Wall Street Journal, AP newswire, San Jose Mercury News, Financial Times, Los Angeles Times, and Foreign Broadcast Information System (FBIS).

Each set has between 5 and 15 documents, with an average of 10 documents. The documents are at least 10 sentences long, but there is no maximum length. There are 2 multi-document extract summaries for each set (400 words and 200 words). Each such extract consists of some subset of the sentences in the document set. Pieces of sentences are not used in the extract summaries. The general topic of each set is either a single natural disaster (e.g., 1989 California earthquake), a single event reported in at most a seven day window (e.g., launch and activation of the Hubble Space Telescope), multiple distinct events of a single type (e.g., heart attacks), or biographical information about a single individual (e.g., Andrei D. Sakharov).

To test the entire QCS system, queries will be created using the most frequently occurring terms (minus the stop words) in several of the DUC document sets. The output of QCS (several extract summaries each corresponding to one of the topic clusters identified by QCS) will be evaluated using DUC multiple-document summarization evaluation tools. The expected outcome is that an extract summary for one of the topic clusters produced from a query will match the DUC extract summary for the document set from which the query was created.

References

- [1] N. Aluthgedara. Recognizing Sentence Boundaries and Boilerplate. In preparation.
- [2] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Rev.*, 37(4):573–595, 1995.
- [3] J. M. Conroy and D. P. O'Leary. Text Summarization via Hidden Markov Models and Pivoted QR Matrix Decomposition. Technical report, University of Maryland, College Park, Maryland, March, 2001.
- [4] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- [5] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, Jan 2001.
- [6] J. D. Schlesinger, M. E. Okurowski, J. M. Conroy, D. P. O'Leary, A. Taylor, J. Hobbs, and W. H. T. Wilson. Understanding Machine Performance in the Context of Human Performance for Multi-document Summarization. In *Proceedings of the Workshop on Automatic Summarization*, 2002. Volume 2: Draft DUC Papers.