SANDIA REPORT

SAND2006-5000
Unlimited Release
Printed October 2006

QCS: A System for Querying, Clustering and Summarizing Documents

Daniel M. Dunlavy, Sandia National Laboratories
Dianne P. O'Leary, University of Maryland, College Park
John M. Conroy, Center for Computing Sciences
Judith D. Schlesinger, Center for Computing Sciences

Sandia National Laboratories

SAND2006-5000
Unlimited Release
Printed October 2006

QCS: A System for Querying, Clustering and Summarizing Documents

Daniel M. Dunlavy
Optimization and Uncertainty Estimation Department
Sandia National Laboratories
P.O. Box 5800, M/S 1318
Albuquerque, NM 87185-1318
dmdunla@sandia.gov

Dianne P. O'Leary
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
oleary@cs.umd.edu

John M. Conroy
Center for Computing Sciences
17100 Science Drive
Bowie, MD 20715-4300
conroy@super.org

Judith D. Schlesinger
Center for Computing Sciences
17100 Science Drive
Bowie, MD 20715-4300
judith@super.org

Abstract

Information retrieval systems consist of many complicated components. Research and development of such systems is often hampered by the difficulty in evaluating how each particular component would behave across multiple systems. We present a novel hybrid information retrieval system—the Query, Cluster, Summarize (QCS) system—which is portable, modular, and permits experimentation with different instantiations of each of the constituent text

analysis components. Most importantly, the combination of the three types of components in the QCS design improves retrievals by providing users more focused information organized by topic.

We demonstrate the improved performance by a series of experiments using standard test sets from the Document Understanding Conferences (DUC) along with the best known automatic metric for summarization system evaluation, `ROUGE`. Although the DUC data and evaluations were originally designed to test multidocument summarization, we developed a framework to extend it to the task of evaluation for each of the three components: query, clustering, and summarization. Under this framework, we then demonstrate that the QCS system (end-to-end) achieves performance as good as or better than the best summarization engines.

Given a query, QCS retrieves relevant documents, separates the retrieved documents into topic clusters, and creates a single summary for each cluster. In the current implementation, Latent Semantic Indexing is used for retrieval, generalized spherical k-means is used for the document clustering, and a method coupling sentence "trimming," and a hidden Markov model, followed by a pivoted QR decomposition, is used to create a single extract summary for each cluster. The user interface is designed to provide access to detailed information in a compact and useful format.

Our system demonstrates the feasibility of assembling an effective IR system from existing software libraries, the usefulness of the modularity of the design, and the value of this particular combination of modules.

Acknowledgment

We thank the authors of `LT TTT`, `GTP`, and `GMEANS` for the use of their code, Timothy O'Leary for his assistance with the MEDLINE data set used in QCS, and Tamara Kolda for her helpful suggestions during the preparation of this manuscript. Daniel Dunlavy was supported in part by the Applied Mathematics Research program of the Office of Advanced Scientific Computing Research of DOEs Office of Science and by Sandia National Laboratories, a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000. Dianne O'Leary was partially supported by NSF Grant CCR-0204084 and CCF-0514213.

Contents

Figures

Tables

1 Introduction

Information retrieval (IR) systems provide users with a vast amount of reference material. Along with this tremendous access comes the challenge of effectively presenting a user with relevant information in response to a query. When using an IR engine to search through electronic resources, simple queries often return too many documents and many are not relevant to the intended search. For instance, there are several million documents on the World Wide Web pertaining to "Michael Jordan." Most of these concern the basketball star, so it is difficult to find information about the television personality, the jazz musician, the mathematician, or the many others who share that name. It would be useful to have a system that could overcome this limitation.

One approach is to *cluster* the documents after retrieval and present a synopsis of each cluster so that a user can choose clusters of interest. This is the motivation for our Query, Cluster, Summarize (QCS) system, which performs the following tasks in response to a query:

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

Our implementation of the QCS system partitions the code into portable modules, making it easy to experiment with different methods for handling the three main tasks listed above. In our current implementation of the QCS system, we use existing software libraries for each task. Throughout this paper, we discuss our choices for each of the modules used, but note that it is possible to exchange individual modules with other existing methods.

Previous work on using a combination of clustering and summarization to improve IR is summarized in [26]. Of existing IR systems employing this combination, QCS most resembles the NewsInEssence system [25] in that both systems can produce multidocument summaries from document sets clustered by topic. However, NewsInEssence is designed for IR from HTML-linked document sets while QCS has been designed for IR from generic document sets.

Another system that leverages clustering and summarization for information organization similarly to QCS is the Columbia Newsblaster system [21]. Newsblaster, like NewsInEssence, is a web-based system which crawls news websites and then clusters and summarizes the news stories, but it does not currently accept queries. Recently, the value of summarization to users in IR has been demonstrated in [20], where a study showed increases in user recall of retrieved information when clustering and summarization were included in the output of the IR system.

We have used QCS for information retrieval in two information domains: newswire documents from the 2002–2004 Document Understanding Conferences (DUC) and biomedical abstracts from the U.S. National Library of Medicine's MEDLINE database.

(See [12] for a description of the use of the MEDLINE documents in QCS.)

In Section 2, we discuss our choices for each of the components of the QCS system. An example of use of the QCS system is presented in Section 3. Section 4 presents results of experiments evaluating some of the components of the implementation. Section 5 focuses on future directions for QCS, and we conclude in Section 6.

2 The QCS System

QCS is a collection of software modules developed in the languages C and C++ and tested under the operating systems SunOS 5.8 (Solaris 8) and Linux (kernel v2.4). Preprocessing tools for all QCS data, including processing of the data passed from one module to another were developed in the Perl language. QCS has been developed as a client-server application, and the implementation took approximately 6 person-months of full-time effort.

In this section we describe the components of our system: document preprocessing, the representation of documents and queries, and the querying, clustering, and summarization of documents. Table 1 presents a summary of the software libraries and packages used in implementing QCS v1.0.

Table 1. Implementation details of QCS v1.0.

Task	*Implementation*	*Language*
Document Preprocessing		
SGML conversion	`HTML-Parser` 3.27 [1]	Perl
POS tagging/sentence detection	`LT TTT` 1.0 [22]	compiled library
stype tagging	`sTag (QCS)`	Perl
Term parsing/indexing/SVD	`GTP` 3.0 [16]	C++
QCS Modules		
Querying	`GTPQUERY` 3.0 [16]	C++
Clustering	`GMEANS` 2.0 [10]	C++
Summarizing	`HMM+QR` [5]	C (from Matlab)
	Sentence Trimmer [6]	Perl
QCS Interface		
Java Server	`TOMCAT` 4.1.12 [2]	C++
QCS Client	`QCS` 1.0	Java

2.1 Document Preprocessing

In preprocessing a document set for use with QCS, we

- convert the documents to a standardized format,
- determine the parts of speech of all words,
- detect and mark the sentence boundaries,
- classify sentences by their content, and
- develop a compact representation for the document information.

This information can be computed once and stored for use by any of the QCS modules.

If not already in SGML format, documents are converted into SGML-encoded documents, with start and end tags around each part of the text. For example, the tags <DOC> and </DOC> are placed at the beginning and end of each document.

Determining the parts of speech for document terms and sentence boundary detection is performed primarily using a probabilistic part-of-speech tagger and sentence splitter based on a combination of hidden Markov and maximum entropy models [22]. The default models, trained on the Brown corpus [15], are used in the current implementation of QCS. This method was chosen due to its ability to handle the two most crucial preprocessing tasks required by the QCS system without modifications and for its proven performance in performing part-of-speech tagging and sentence boundary detection [22].

An important part of preprocessing the data for use in the summarization module of QCS is assessing the value of the content of each sentence based on the role of that sentence in the document. Thus we tag each sentence as a candidate for extract summaries (*stype* = 1), not a candidate but possibly containing useful key terms or phrases (*stype* = 0), or containing no useful information (*stype* = −1). Table 2 shows the mapping of SGML tags to *stype* values of 0 and 1. All other tags, e.g., <DOCNO>, <AUTHOR>, etc., have *stype* = −1. Note that the choice for these mappings is heuristic—based on our manual inspection of several documents of each type—and may need to be amended for other document sets. The complete set of SGML tags for each type of document is defined using a document type definition (DTD) file. The name of the DTD file associated with each file type is also listed in Table 2. The DUC documents already contain the SGML tags needed by QCS, but Medline documents are not SGML-encoded, and a separate preprocessing step is required to map Medline fields to SGML encoded text (the field names are used as the SGML tag names in QCS). A generic document type is also used for all other documents that are not originally SGML-encoded. An additional preprocessing step for such documents is required for mapping at least one sentence to *stype* = 1 (using the SGML tag <TEXT>) and one to *stype* = 0 (using the SGML tag <SUBJECT>)—at least one sentence of each *stype* is required by the current summarization module in QCS.

Embedding the information (i.e., the *stype* of each sentence) in the document itself instead of creating a processing module in the summarization algorithm creates the flexibility of using the information throughout the various stages of the QCS system. It also enables expansion of the types of sentence classification without affecting the implementation of the summarization module.

Currently, QCS uses a vector space model [27] for document representation in the querying, clustering, and summarization modules. In such a model, a set of m documents containing n distinct terms can be represented by an $m \times n$ term-document matrix A. Terms in QCS are all the (white space delimited) words in a document with the exception of a pre-designated list of stop words. The list of stop words currently used in QCS is the one provided with the implementation of the query module [16].

Table 2. Mapping of SGML tags to *stype* values in QCS.

Document Type (DTD filename)	SGML Tag	*stype*
Generic	<TEXT>	1
	<SUBJECT>	0
Acquaint (acquaint.dtd)	<TEXT>	1
	<HEADLINE>	0
Associated Press (ap.dtd)	<TEXT>	1
	<HEAD>	0
San Jose Mercury News (sjmn.dtd)	<TEXT>, <LEADPARA>	1
	<CAPTION>, <DESCRIPT>, <HEADLINE>, <MEMO>	0
Los Angeles Times (latimes.dtd)	<TEXT>	1
	<HEADLINE>, <SUBJECT>, <GRAPHIC>	0
Federal Register (fr.dtd)	<TEXT>, <SUMMARY>, <SUPPLEM>, <FOOTNOTE>	1
	<DOCTITLE>	0
Foreign Broadcast Information Service (fbis.dtd)	<TEXT>	1
	<TI>, <H1>, ..., <H8>	0
Medline	<ABSTRACT>	1
	<TITLE>	0
Wall Street Journal (wsj.dtd)	<TEXT>, <LP>	1
	<HL>	0
Financial Times (ft.dtd)	<TEXT>	1
	<HEADLINE>	0

The value of an entry of the matrix A is a product of three scaling terms:

$$a_{ij} = \tau_{ij} \cdot \gamma_i \cdot \delta_j \qquad (i = 1, ..., m; j = 1, ..., n) \tag{1}$$

where τ_{ij}, γ_i, and δ_j, are the *local weight*, *global weight*, and *normalization factor*, respectively. These parameters are chosen so that the value a_{ij} best represents the importance (or weight) of term i in document j for a particular document set. The j^{th} column of A, a_j, is the *feature vector* for document j. The various scaling options for a term-document matrix in QCS are presented in Table 3. The values f_{ij} and f_i are the number of times term i appears in document j and the number of times term i appears in the entire document collection, respectively. The local *binary* weighting is used when it is important whether or not a term appears in a document (as is the case with a document set with very little overlap in terms across the document set), whereas the *log* weighting would be used to damp the effects of large differences in term frequencies within a single document. Global weighting reduces the weight of terms that occur frequently within a document or across several documents while giving a greater weight to terms that occur infrequently. See, for example, [17] for more information. Finally, the normalization factor is used to remove any bias based on document size by scaling each document feature vector to unit length in the Euclidean norm. This standard *tf.idf* (term frequency, inverse document frequency) scheme, along with normalization, is used in the examples presented in this paper.

The indexing of the terms and documents is performed in QCS using the General Text Parser (`GTP`) [16]. `GTP` was chosen for use in QCS since it includes tools for parsing documents and representing them in a vector space model along with a retrieval tool that is currently used in the querying module. Minor changes were necessary to provide an interface to the term-document matrix consistent with that needed by the clustering module.

Currently, the indexing is done "offline"; it is performed once as a preprocessing step for a static document set or during system downtime for a dynamic set. The reason for this is that the parsing and indexing are too computationally expensive to be done in real-time.

2.2 Querying Documents

The method used for query-based document retrieval in QCS is Latent Semantic Indexing (LSI) [8]. LSI attempts to reveal latent relationships caused by term ambiguity, while preserving the most characteristic features of each document. It does this by approximating the matrix A by a rank-p matrix A_p computed using a singular value decomposition (SVD) of A.

We represent a query using a query vector q, with m components, just as a document can be represented by a feature vector. A query vector is typically much

Table 3. Scaling factors for a term-document matrix

	Local Weights (τ_{ij})
Term Frequency	f_{ij}
Binary	$\chi(f_{ij}) = \begin{cases} 0 & f_{ij} = 0 \\ 1 & f_{ij} > 0 \end{cases}$
Log	$\log(f_{ij} + 1)$
	Global Weights (γ_i)
None	1
Normalized	$\left(\sum_i f_{ij}^2\right)^{-1/2}$
Inverse Document Frequency (IDF)	$\log\left(n / \sum_j \chi(f_{ij})\right)$
IDF Squared(IDF2)	$\log\left(n / \sum_j (\chi(f_{ij}))^2\right)$
Entropy	$1 - \sum_j \frac{\left(f_{ij} / \sum_k f_{ik}\right) \log\left(f_{ij} / \sum_k f_{ik}\right)}{\log n}$
	Normalization (δ_j)
None	1
Normalized	$\left(\sum_i (\tau_{ij}\gamma_i)^2\right)^{-1/2}$

more sparse than a feature vector (since it contains far fewer terms than an average document) and does not necessarily use the same scaling scheme. For comparing query vectors and document vectors in LSI, the query vector is projected into the p-dimensional subspace spanned by the columns of A_p, and we denote the projected vector as q_p.

The relevance of a document to a query is measured by the cosine similarity score, s, between q_p and the column of A_p corresponding to that document. For example, the relevance of document j to the query is computed as

$$s_j = \frac{q_p^T (a_p)_j}{\|q_p\| \, \|(a_p)_j\|} \,, \tag{2}$$

where $(a_p)_j$ is the jth column of A_p. Note that $0 \leq s_j \leq 1$.

The querying module in QCS is called `GTPQUERY` and is part of the `GTP` system. `GTPQUERY` parses the query (using the method used to parse the document set), normalizes the resulting vector, and calculates the cosine similarity scores. A very helpful feature implemented in `GTPQUERY` is the ability to use different low-rank approximations without having to recompute the SVD. Since we store the components of the SVD rather than the reassembled low-rank approximation, a user is able to choose the rank of the approximation to be used for each query up to the number of singular values computed during the SVD computation by `GTP`. If all of the singular values are stored, the user has the option of performing queries ranging from exact matches (using all of the singular values) to extremely conceptual matches (using just a few singular values). In the current implementation of QCS, all of the singular values are computed and stored for each document collection. Note that for larger collections, though, the number of singular values computed may be limited by the computational resources available.

The documents matching a query can be chosen by specifying either the number of documents to be retrieved or a cutoff for the query score. In the current implementation of QCS, 100 documents are returned in order to have a large enough subset of documents to guarantee good clustering and summarization output. The potential downside to this is that, depending on the specific query, many of the retrieved documents may have very low query scores. This may need to be adjusted based on the document set and/or distribution of query scores.

2.3 Clustering Documents

In QCS, we use the information derived from the query processing phase to cluster documents into a variable number of clusters, each representing a single topic. Throughout this section, we assume that the querying module has identified a set of N documents for further processing.

Our clustering of the N documents is a partitioning into k disjoint subsets, $\pi_1, \ldots, \pi_k$, based on cosine similarity of the N document feature vectors, $\{d_1, \ldots, d_N\}$. The *coherence* of the cluster π_j can be defined as

$$\sum_{d_i \in \pi_j} d_i^T c_j \ , \tag{3}$$

where d_i is assumed to be normalized (i.e, $\|d_i\| = 1$) and c_j is the normalized centroid of cluster π_j containing n_j documents:

$$c_j = \frac{\frac{1}{n_j} \sum_{d_i \in \pi_j} d_i}{\|\frac{1}{n_j} \sum_{d_i \in \pi_j} d_i\|} \ . \tag{4}$$

We want to choose the clusters π_j to maximize the sum of the coherence functions. This is one of the classical approaches to k-means clustering and can be shown to be equivalent to minimizing the radii of the clusters.

To perform the clustering in QCS, we currently use the spherical k-means algorithm [11] employing first variation and splitting [10]. This is an iterative method for maximizing the coherence functions of document feature vectors and includes efficient computation of feature vector similarities (the main computational bottleneck in many implementations of k-means algorithms) and the ability to choose a range for the number of clusters into which the feature vectors will be partitioned. Comparisons of this clustering algorithm to the classical k-means algorithm on large document sets indicate significant decreases in computational time coupled with insignificant degradation in the quality of the clusters [9].

Clustering can be a computational bottleneck unless a good initial guess is provided. In QCS, we use 5 initial (seed) clusters and allow the results to be partitioned into as many as $N/10$ final clusters. The seeding of the initial clusters is based on the query scores (i.e., the cosine similarity scores) of the documents with cluster i, $(i = 1, \ldots, 5)$, containing documents with scores satisfying

$$0.2(i-1)(s_{\max} - s_{\min}) + s_{\min} < s \leq 0.2(i)(s_{\max} - s_{\min}) + s_{\min} \ , \tag{5}$$

where $s_{\max}$ and $s_{\min}$ are the maximum and minimum scores, respectively, of the documents returned from the query module. This seeding has proven useful for the document collections used in QCS to date, but may not work well for all sets of documents. The best use of the similarity scores in seeding the initial clusters remains an open question.

The clustering of documents in QCS is performed using `GMEANS` v1.0 [10]. Only slight modifications to the original code were necessary to insure that the interface to the data in the vector space model matched both the query and summarization modules. The GMEANS software includes several distance measures; only spherical k-means has been tested extensively in QCS. The other distance measures are Euclidean distance, Kullback-Leibler divergence, and diametric distance. More testing on the use of these distance measures will help determine their usefulness in producing good clusters for use in summarization.

Once a set of clusters of documents has been determined, the list of documents in each cluster is then passed to the summarization module.

2.4 Summarizing Documents and Clusters

The summarization module in QCS is based on the methods presented in [5] and its implementation from the DUC 2003 evaluation [13]. The algorithm proceeds in two steps: trimming sentences and then choosing the sentences to include in a summary. The sentence trimming algorithms are the work of Schlesinger, first documented in [13].

2.4.1 Choice of Summary Sentences

The choice of sentences to include in the summary is done in two phases; first, single document extract summaries are produced for each document in the cluster, and then sentences from these summaries are considered for inclusion in the summary of the document cluster.

Single document summaries are produced using a hidden Markov model (HMM) [3, 24] to compute the probability that each sentence is a good summary sentence. The highest probability sentences are chosen for the summary. The 13-state HMM shown in Figure 1, built to extract six *primary* sentences and an arbitrary number of additional supporting sentences, is used to compute these probabilities. Currently, this 13-state HMM and an additional 5-state HMM (3 primary sentence states and 2 supporting sentence states) are used in QCS for different document collections. The ability to use a different extraction model for each document collection allows for the application of QCS to a wide range of document formats and genres.

The HMMs in QCS use features based upon "signature" and "subject" terms occurring in the sentences. The signature terms are the terms that are more likely to occur in the document (or document set) than in the corpus at large. To identify these terms, we use the log-likelihood statistic suggested in [14] and first used in summarization in [19]. The statistic is equivalent to a mutual information statistic and is based on a 2-by-2 contingency table of counts for each term. The subject terms are those signature terms that occur in sentences with $stype = 0$, e.g., headline and subject leading sentences.

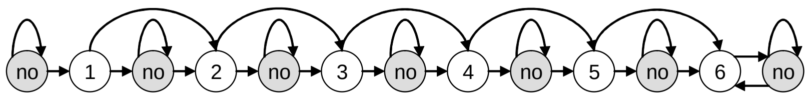


Figure 1. The state space of the 13-state HMM used in the QCS summarizer.

The HMM features are

- $\log(n_{\text{sig}} + 1)$, where n_{sig} is the number of signature terms in the sentence,
- $\log(n_{\text{subj}} + 1)$, where n_{subj} is the number of subject terms in the sentence,
- the position of the sentence in the document, built into the state-structure of the HMM.

The two term-based features are normalized component-wise to have mean zero and variance one. In addition, the features for sentences with *stype* $= 0$ or -1 are coerced to be -1, which forces these sentences to have an extremely low probability of being selected as summary sentences.

Multidocument summaries are created for each cluster by choosing a subset of the sentences identified by the HMM. If we want a summary containing w words, we consider the highest probability sentences from documents in that cluster, cutting off when the number of words exceeds $2w$. We form a term-sentence matrix, B, similar in structure to the term-document matrix A used in the querying and clustering modules, containing a column for each of these sentences. The columns of B are scaled so that the Euclidean norm equals the probability assigned to the sentence by the HMM.

In order to remove redundant sentences, a pivoted QR algorithm is applied to the scaled term-sentence matrix. We first choose the sentence whose corresponding column in B has maximum norm. Then, within the matrix B, we subtract from each remaining column the component in the direction of the column for this chosen sentence. This process is iterated until the number of words in the collection of chosen sentences exceeds the desired length w. For more details, see [5].

2.4.2 Sentence Trimming

The HMM tends to select longer sentences due to the features currently used. Because of this, for a 100-word summary, the pivoted QR algorithm typically selects 2 or 3 sentences from all those first selected by the HMM. We hypothesized that if we could shorten sentences, by removing less important information from them, we could increase the number of sentences in a summary and, therefore, add additional information to the summary.

As an inexpensive alternative to full parsing and comprehension, we identified trimming patterns using "shallow parsing" techniques, keying off lexical cues based on part-of-speech (POS) tags in our preprocessed data. The following eliminations were considered:

- lead adverbs and conjunctions;
- gerund phrases;
- restricted relative-clause appositives;
- intra-sentential attribution.

We define a *token* to be a white-space delimited word with all punctuation removed and use the simple heuristic that if the number of tokens to be deleted is greater than or equal to the number of tokens to be retained, the elimination is not performed.

Lead adverbs and conjunctions include POS-tagged adverbs that are comma-delimited from the remainder of the sentence along with conjunctions such as "and" and "but". They do not tend to add substantial information and often hinder the flow of the summary when the preceding sentence of the document was not also selected.

Gerund phrases often comment on, rather than advance, a narration and therefore tend to be incidental. Restricted relative-clause appositives usually provide background information which could be eliminated. While attributions can be informative, we decided that they could be sacrificed in order to include other, hopefully more important, information in the summary.

An example of each of the three phrase/clause eliminations is given in Figure 2.4.2.

---

a. **Example of a gerund phrase to be removed:** More than 800 lives were lost when the 21,794 tonne ferry, *sailing from the Estonian capital Tallinn to Stockholm,* sank within minutes early yesterday morning in the Baltic Sea 40 km south west of the Finnish island of Uto.

b. **Example of a restricted relative-clause appositive to be removed:** The Menendez family lived in the Princeton Area until 1986*, when they moved to California.*

c. **Example of an attribution to be removed:** *The federal Government's highway safety watchdog said Wednesday that* the Ford Bronco II appears to be involved in more fatal roll-over accidents than other vehicles in its class and that it will seek to determine if the vehicle itself contributes to the accidents.

Figure 2. Examples of phrase/clause eliminations in the summarization module of QCS.

Our DUC 2003 submission, which used the same summarizer as in QCS, used these phrase/clause eliminations in a *post*-processing mode. Sentence selection was first made by the HMM and QR algorithms. These sentences were then trimmed and one or more sentences were added if space was made available. Based on the DUC 2003 results, we hypothesized that we would see added benefit if we applied these transformations as a *pre*-processing step applied to all sentences in the documents, before summary sentence selection was performed. This was tested in DUC 2004 and results were superior to the submission using the post-processing version. See [6] for details.

We also experimented with removing two types of sentences. The first type of sentence is one that begins with an imperative. This type is not currently removed since a lead imperative so rarely occurred it was not worth looking for it. The second type of sentence is one containing a personal pronoun at or near the start. While these sentences negatively impact a summary's readability, eliminating them adversely affected the quality of the summary's information content. We are working on a solution to the anaphora problem to resolve this issue.

2.5 The QCS Client-Server Architecture

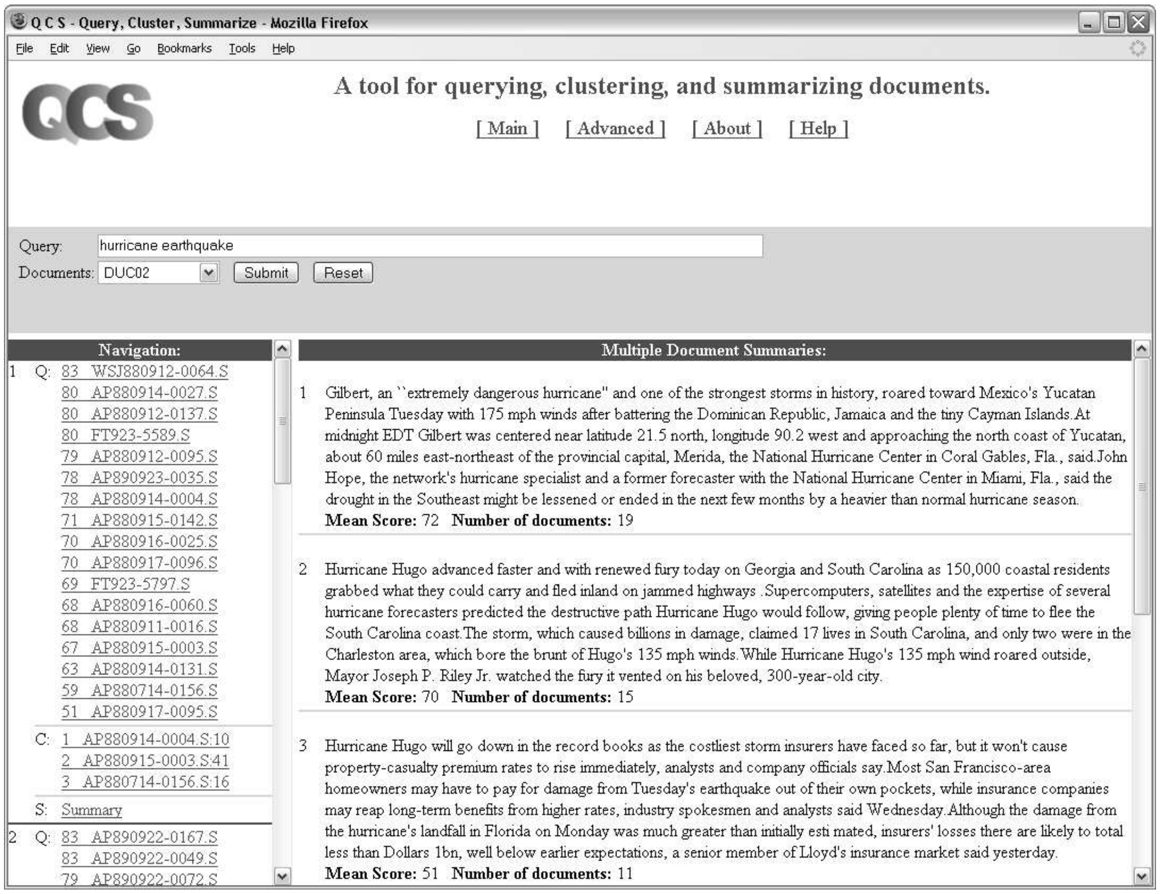
A screen shot of the QCS user interface is presented in Figure 3. There are three main frames in the interface: the query form, the navigation bar, and the results frame.

The query form contains an input field for entering a query and a field for selecting the document set on which to perform the query. Currently, the document sets from the 2002–2004 DUC evaluations and a Medline document set are available for online use.

The *navigation bar* contains links to the documents and is organized to reflect the output from the querying, clustering and summarization modules. For each cluster, query scores and document names are given, with hyperlinks to the text of the documents in the "Q" subsection. In the "C" subsection, links are given to the documents containing the sentences used in the multidocument summary, along with the index of the sentence within the original document. Lastly, in the "S" subsection, a link to the multidocument summary for the cluster is presented.

The *results frame* displays information requested through the navigation bar. The default output is multidocument summaries (also chosen using the "S" links). Other options include the text of individual documents (chosen using the "Q" links) or individual documents with summary sentences highlighted (chosen using the "C" links).

Note that all instances of the query scores presented in the QCS user interface are scaled to the interval $[0, 100]$ for readability.


Q C S - Query, Cluster, Summarize - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
QCS
A tool for querying, clustering, and summarizing documents.
[Main] [Advanced] [About] [Help]
Submit
Reset

Query: hurricane earthquake
Documents: DUC02

Navigation:

```
1  Q:  83  WSJ880912-0064.S
       80  AP880914-0027.S
       80  AP880912-0137.S
       80  FT923-5589.S
       79  AP880912-0095.S
       78  AP890923-0035.S
       78  AP880914-0004.S
       71  AP880915-0142.S
       70  AP880916-0025.S
       70  AP880917-0096.S
       69  FT923-5797.S
       68  AP880916-0060.S
       68  AP880911-0016.S
       67  AP880915-0003.S
       63  AP880914-0131.S
       59  AP880714-0156.S
       51  AP880917-0095.S
   C:  1  AP880914-0004.S:10
       2  AP880915-0003.S:41
       3  AP880714-0156.S:16
   S:  Summary
2  Q:  83  AP890922-0167.S
       83  AP890922-0049.S
       79  AP890922-0072.S
```

Multiple Document Summaries:

1 Gilbert, an ``extremely dangerous hurricane'' and one of the strongest storms in history, roared toward Mexico's Yucatan Peninsula Tuesday with 175 mph winds after battering the Dominican Republic, Jamaica and the tiny Cayman Islands.At midnight EDT Gilbert was centered near latitude 21.5 north, longitude 90.2 west and approaching the north coast of Yucatan, about 60 miles east-northeast of the provincial capital, Merida, the National Hurricane Center in Coral Gables, Fla., said.John Hope, the network's hurricane specialist and a former forecaster with the National Hurricane Center in Miami, Fla., said the drought in the Southeast might be lessened or ended in the next few months by a heavier than normal hurricane season.
Mean Score: 72 **Number of documents:** 19

2 Hurricane Hugo advanced faster and with renewed fury today on Georgia and South Carolina as 150,000 coastal residents grabbed what they could carry and fled inland on jammed highways .Supercomputers, satellites and the expertise of several hurricane forecasters predicted the destructive path Hurricane Hugo would follow, giving people plenty of time to flee the South Carolina coast.The storm, which caused billions in damage, claimed 17 lives in South Carolina, and only two were in the Charleston area, which bore the brunt of Hugo's 135 mph winds.While Hurricane Hugo's 135 mph wind roared outside, Mayor Joseph P. Riley Jr. watched the fury it vented on his beloved, 300-year-old city.
Mean Score: 70 **Number of documents:** 15

3 Hurricane Hugo will go down in the record books as the costliest storm insurers have faced so far, but it won't cause property-casualty premium rates to rise immediately, analysts and company officials say.Most San Francisco-area homeowners may have to pay for damage from Tuesday's earthquake out of their own pockets, while insurance companies may reap long-term benefits from higher rates, industry spokesmen and analysts said Wednesday.Although the damage from the hurricane's landfall in Florida on Monday was much greater than initially esti mated, insurers' losses there are likely to total less than Dollars 1bn, well below earlier expectations, a senior member of Lloyd's insurance market said yesterday.
Mean Score: 51 **Number of documents:** 11

Figure 3. The QCS user interface.

The client in QCS consists of dynamically-created HTML pages. Using this approach makes the QCS system as portable as possible from the perspective of its users. The dynamic HTML pages are generated by Java servlets that are deployed via an Apache Tomcat Java Server (v.4.1.12). The interface between the QCS server (consisting of all of the C/C++ code) is handled using the Java Native Interface (JNI) in QCS. For QCS, this allows the computationally intensive code to be developed in C and C++ code that can be highly optimized on a given hardware platform, while still allowing for the greatest amount of portability for the user interface.

The current implementation of QCS can be found at `http://stiefel.cs.umd.edu:8080/qcs/` (`http://128.8.128.181:8080/qcs/`).

3 Example of the QCS System

We present here an example of the entire QCS system from the standpoint of the user. The example uses the query *hurricane earthquake* in finding documents in the DUC 2002 document collection. The DUC 2002 collection consists of 567 documents and the QCS preprocessing modules identified 7767 unique terms across the collection.

Table 4 shows the highest query similarity scores along with the first "subject" sentence (i.e., first sentence with $stype = 0$) from each document. In this example, a rank-50 approximation of A (i.e., $p = 50$) was used in computing the similarity scores. Clearly, QCS has found several documents about hurricanes. Furthermore, there are no clear examples of documents relating to earthquakes in these documents. However, some of the subject sentences are rather uninformative, and it would be difficult to classify the documents on the basis of these alone. Given just this kind of information (as is typically the case with query tools), a user would have many documents to read and no idea whether or not the high-ranking documents contained redundant information.

The results of clustering the 100 top scoring documents returned by the querying module using an upper limit of 10 clusters are presented in Table 5. The clustering algorithm split the original 5 seed clusters into 10 clusters, and the table shows the number of documents and the mean query score for each cluster. For this example, a majority of the documents are in the 5 clusters with the highest mean query scores; this is representative of most of our tests and may be biased by our initial seeding scheme. However, it is unclear if and how this behavior would change if a different initial cluster seeding is used.

Table 6 presents the subject sentences of the top 3 scoring documents in each of the top 3 clusters, illustrating the contents of each cluster. It is clear from the subject lines that the documents in the first cluster relate to Hurricane Gilbert and those in the third cluster relate to insurance claims associated with hurricanes. However, from the subject lines alone, is is difficult to determine the focus of the documents in the second cluster; they could relate to forecasting or a specific hurricane which hit a historic city or something else.

Figure 3 shows the multidocument summaries for the top 5 scoring clusters. We see that the subject lines in Table 6 for the first and third clusters were indeed indicative of the topics of those clusters, as further illustrated by the summaries. From the summary for the second cluster, we see that the documents in that cluster focus on Hurricane Hugo. Note that the name *Hugo* did not appear in the subject lines of the top query results (Table 4) or top cluster results (Table 6), and only is indicated as the topic of the second cluster through the multidocument summary. Moreover, the name *Hugo* only appears in the subject line of the document in the second cluster which has the lowest query score (47).

Table 4. Query results in the *hurricane earthquake* example.

Score	*Subject Sentence*
90	Hurricane Latest in String of Disasters to Hit Historic City
85	Hurricane Forecasters Carry on Amid Chaos
85	Forecasting Aided by Supercomputers, but Still an Uncertain Science
84	Killer Storm Hits South Carolina Coast
83	Scientists: Warming Trends Could Mean Fiercer Hurricanes
82	City Sends Money to Charleston in Repayment of 211-year-old Debt
82	150,000 Take Off as Hugo Takes Aim at Ga., Carolina
82	Loss of Life Low because People Were Prepared
81	Hurricane Gilbert Heading for Jamaica with 100 MPH Winds
80	Gilbert: Third Force 5 Hurricane This Century

Summaries for the top 5 clusters are shown in Figure 3 to illustrate the ease of finding information about *earthquakes* even though most of the top scoring results focused on *hurricanes*. In fact, the highest scoring document related to earthquakes in this example is found in position 39 in the query results with a score of 51. The potential savings to the user in using QCS in this example is that only 3 summaries would need to be read before finding information about earthquakes (instead of 38 subject lines or even full documents). Furthermore, the documents related to earthquakes are clustered to differentiate between those related to an earthquake in California (cluster 4) and those related to one in Iran (cluster 5).

The flow of the summaries is representative of the output of QCS for the queries tested. They do not read like human-generated summaries, but the hope is that they are sufficient to inform a user of the content of the documents contained in each cluster. Note that in some cases, the summaries can be misleading, most notably for clusters containing documents covering two or more related but distinct topics.

This example illustrates the usefulness of providing document clusters and cluster summaries in presenting query results to a user. We undertake a systematic evaluation of the QCS system in the next section.

Table 5. Clustering results in the *hurricane earthquake* example.

	Initial (Seed) Clusters		*Final Clusters*	
Cluster	*Documents*	*Mean Query Score*	*Documents*	*Mean Query Score*
1	26	76	19	72
2	11	62	15	70
3	25	44	11	51
4	20	31	15	41
5	18	13	17	34
6			6	20
7			8	17
8			3	17
9			3	13
10			3	08

Table 6. Top scoring documents (using query scores) from the top scoring clusters (using mean query scores) in the *hurricane earthquake* example.

Score	*Subject Sentence*
Cluster 1	
83	Hurricane Gilbert Heading for Jamaica With 100 MPH Winds
80	Gilbert: Third Force 5 Hurricane This Century
80	Hurricane Hits Jamaica With 115 mph Winds; Communications Disrupted
Cluster 2	
83	Forecasting Aided By Supercomputers, But Still An Uncertain Science
83	Hurricane Latest in String of Disasters to Hit Historic City
79	Hurricane Forecasters Carry On Amid Chaos
Cluster 3	
67	Hurricane batters southern US but lets insurers off lightly
67	US insurers face heaviest hurricane damage claims
66	UK Company News: GA says hurricane claims could reach 'up to Dollars 40m'

Cluster 1
Gilbert, an "extremely dangerous hurricane" and one of the strongest storms in history, roared toward Mexico's Yucatan Peninsula Tuesday with 175 mph winds after battering the Dominican Republic, Jamaica and the tiny Cayman Islands. At midnight EDT Gilbert was centered near latitude 21.5 north, longitude 90.2 west and approaching the north coast of Yucatan, about 60 miles east-northeast of the provincial capital, Merida, the National Hurricane Center in Coral Gables, Fla., said. John Hope, the network's hurricane specialist and a former forecaster with the National Hurricane Center in Miami, Fla., said the drought in the Southeast might be lessened or ended in the next few months by a heavier than normal hurricane season.

Cluster 2
Hurricane Hugo advanced faster and with renewed fury today on Georgia and South Carolina as 150,000 coastal residents grabbed what they could carry and fled inland on jammed highways. Supercomputers, satellites and the expertise of several hurricane forecasters predicted the destructive path Hurricane Hugo would follow, giving people plenty of time to flee the South Carolina coast. The storm, which caused billions in damage, claimed 17 lives in South Carolina, and only two were in the Charleston area, which bore the brunt of Hugo's 135 mph winds. While Hurricane Hugo's 135 mph wind roared outside, Mayor Joseph P. Riley Jr. watched the fury it vented on his beloved, 300-year-old city.

Cluster 3
Hurricane Hugo will go down in the record books as the costliest storm insurers have faced so far, but it won't cause property-casualty premium rates to rise immediately, analysts and company officials say. Most San Francisco-area homeowners may have to pay for damage from Tuesday's earthquake out of their own pockets, while insurance companies may reap long-term benefits from higher rates, industry spokesmen and analysts said Wednesday. Although the damage from the hurricane's landfall in Florida on Monday was much greater than initially estimated mated, insurers' losses there are likely to total less than Dollars 1bn, well below earlier expectations, a senior member of Lloyd's insurance market said yesterday.

Cluster 4
A major earthquake rocked northern California Tuesday evening, collapsing part of the San Francisco Bay Bridge and shaking Candlestick Park and buildings up to 95 miles away. Tuesday's earthquake, the strongest on the San Andreas fault since the San Francisco quake on April 18, 1906, came in a place that had been identified by scientists just last year as the most likely spot for a major jolt in Northern California within the next 30 years. A violent earthquake rocked Northern California during Tuesday evening's rush hour, caving in a section of the San Francisco Bay Bridge, terrifying World Series fans in Candlestick Park and shaking buildings as far as 200 miles away.

Cluster 5
Iran said today that it would welcome relief offered by its bitter enemy, the United States, to help victims of the earthquake that has killed as many as 35,000 people, the State Department said in Washington. State Department officials said the government gave $300,000 worth of supplies to the American Red Cross for shipment to Iran – including 1,000 hard hats, 1,000 pairs of leather gloves, 10,000 face masks, 2,940 wool blankets and about 500 tents. Orange County's Iranian community launched an ambitious effort Sunday to collect half a million dollars in money, medicine, tents, blankets and sleeping bags for hundreds of thousands of injured and homeless people in earthquake-ravaged Iran.

Figure 4. Multidocument summaries (∼100 words) for the top 5 scoring clusters in the *hurricane earthquake* example.

4 Experiments

In this section, we describe the results of several sets of experiments performed to test QCS on various document collections. We first present the results of timing experiments involving the processing of document collections, setting up and running QCS on collections of various sizes, and partial parallel implementations of the QCS system. We then describe our framework for evaluating the performance of QCS and present the results of several tests performed within this framework using data from the 2002–2004 DUC evaluations.

Tests were performed on a Sun Ultra60 with a 450 MHz processor and 512 Mb of RAM running Solaris 8.

4.1 Timing Tests

The first set of experiments focused on the computational time required for document preprocessing and calls to QCS using the document collections from the 2002–2004 DUC evaluations and a subset of Medline documents focusing on gastrointestinal stromal tumors.

The timing results, *in minutes*, for the offline preprocessing of the documents are presented in Table 7. Note that the data for the DUC 2003 evaluation is split into three sets of documents, one for each of the summarization tasks (2, 3, and 4) from that year where the goal was to generate 100-word summaries. Also, the DUC 2004 document collection that was modified using the natural language processing (NLP) techniques described in Section 2.4 is denoted as DUC04-PRE. The times for `GTP` preprocessing (document parsing, term indexing, and SVD computation) appear to increase linearly with respect to the document collection size, but it is not as clear how well the NLP methods (part-of-speech and sentence tagging) scale with larger collections. Because the main computational bottleneck in the QCS document preprocessing involves the NLP methods, we plan to study the scalability of these methods with the goal of more efficient processing of very large document collections, specifically in the context of massively parallel computing.

Table 8 presents the run time of QCS using several different queries. The query terms were chosen such that at least 100 documents in the corresponding collection contained at least one of the terms and that a range of the number of clusters were produced across all document collections. The collection name, the query term(s) used, and the times for setting up the problem, and the three phases of processing (Q, C, and S) are shown. Note that these times are in *seconds*, not minutes as in Table 7, and represent the amount of time a user of QCS waits for results once a query has been submitted.

The results indicate that the major portion of the computation performed in QCS

Table 7. Timing results for preprocessing steps.

				Matrix	*Time (minutes)*	
Collection	*Size (Mb)*	*Docs*	*Terms*	*Nonzeros*	*GTP*[†]	*NLP*[‡]
DUC02	3.77	567	19464	119791	2:27	43:17
DUC03-2	1.99	298	11907	63751	1:25	23:01
DUC03-3	1.94	326	13426	62093	1:29	19:59
DUC03-4	9.92	1105	38037	277975	5:23	55:08
DUC04	2.83	500	16037	106854	2:04	25:20
DUC04-PRE	2.73	500	15780	102976	1:56	25:20
MEDLINE	2.42	738	12039	74596	1:39	26:15

[†]Document parsing, term indexing, and SVD computation using `GTP`.

[‡]Part-of-speech/sentence tagging using `LT POS` and *stype* tagging using Perl.

Table 8. Average timing results for 10 runs of QCS.

			Time (seconds)			
Collection	*Query*	*Clusters*[†]	*Setup*	*Q*	*C*	*S*
DUC02	floods	3	0.19	0.05	9.68	12.88
DUC03-2	president	5	0.22	0.04	9.14	22.81
DUC03-3	government	2	0.22	0.05	8.88	10.26
DUC03-4	technology	10	0.21	0.10	17.55	98.38
DUC04	party	6	0.22	0.06	7.72	39.73
DUC04-PRE	party	6	0.21	0.06	8.68	36.73
MEDLINE	cajal kit	10	0.24	0.08	7.90	21.72

[†]Number of clusters with mean score above 20.

can be performed offline so the user experiences response times in seconds rather than minutes. Note that the most recently developed and least optimized piece of code in QCS is the summarization module, and this accounts for the bulk of user time. Furthermore, the problem setup code and the implementations of the querying and clustering algorithms scale fairly well with respect to the number of clusters generated. This indicates a clear plan for the first steps in attempting speedups in the overall QCS system.

A parallel version of `GTP`, `PGTP`, has also been incorporated into QCS. `PGTP` uses the Message Passing Interface (MPI) library specification for implementing the code in parallel. We used the MPICH implementation of MPI. The part of `GTP` that can be efficiently performed in parallel is the computation of the SVD of the term-document matrix [4].

Timing results for `PGTP` using 14 Sun Ultra10 workstations are presented in Figure 5. The specific times presented in the figure are the real, or wall clock, time and the user, or computational, time required to compute the SVD in parallel for the term-document matrix produced from the DUC document set. From the figure, we can see that there is definitely a speedup when using more than one processor. Note that there is an unexpected *four*-fold increase in speedup between using 1 and 2 processors; in this experiment the data and associated temporary storage required for computing the SVD of the term-document matrix exceeded the physical memory available on a single processor but fit on 2 processors. The best speedup factor in real time over using a single processor is more than 6 (4 processors), and for user time it is more than 25 (4 processors). However, for this small set of documents, we can see that there is essentially no difference in user time when using more than 3–4 processors. Results presented in [4] show a similar leveling off behavior for `PGTP` run on several larger test sets (more than 130,000 documents and 270,000 terms).

4.2 Experiments with QCS on Small Topic-Related Document Collections

The second set of experiments focused on the interplay between the querying, clustering and summarization modules in QCS. We evaluated the system measuring the effect of replacing a machine generated component with the "gold-standard" equivalent.

We evaluated both single and multi-document summaries. In each case we compared machine summaries with human model summaries using the Recall-Oriented Understudy for Gisting Evaluation (`ROUGE`) v1.5.5 summarization evaluation tool [18]. We report here the `ROUGE-1`, `ROUGE-2`, and `ROUGE-SU4` recall scores, as these are the scores reported in several of the most recent DUC evaluations [7]. These scores range from 0 to 1 and reflect the similarity—with higher score reflecting more similarity—between two summaries. The `ROUGE-1` and `ROUGE-2` scores are based on the overlap of

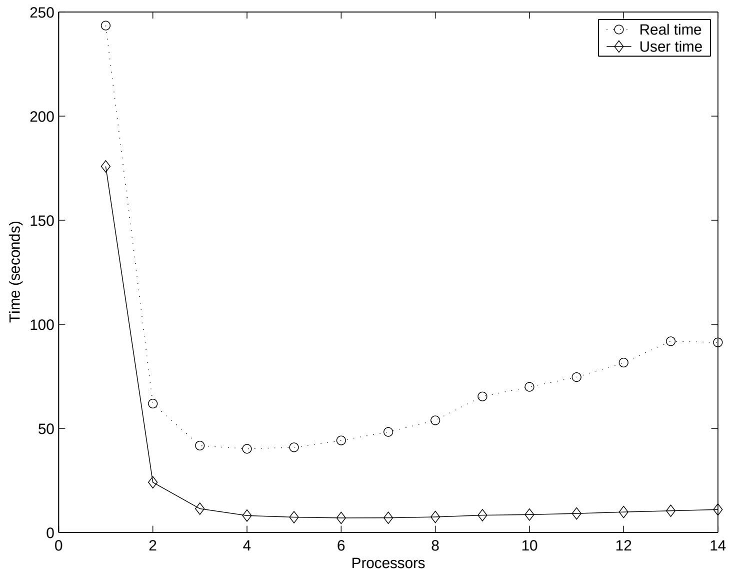


Figure 5. Timing of `PGTP` indexing the DUC 2002 documents (14 workstations).

unigrams and bigrams (using words as tokens), respectively, between automatically-generated summaries and human-generated summaries.[1] The `ROUGE-SU4` score is also based on the overlap of bigrams between summaries, but allows for gaps to occur between tokens (skip-bigram), with a maximum gap length of 4 tokens, and includes unigram co-occurrence statistics as well. Token stemming is performed using the Porter stemmer algorithm, and four-way cross-validation was performed for summaries generated using QCS and those using the summarization module independently.

4.2.1 Experiments with Single Document Summaries

We designed an experiment to measure the effects of the clustering algorithm on single-document summaries. Recall that the summarization component uses signature terms—terms identified as representative of the document—and the performance of the algorithm is greatly influenced by the quality of the signature terms. The experiment was to compare the quality of the summary when signature terms were taken from "ground-truth" clusters versus when the clustering information is withheld and the documents are treated in isolation.

[1]The specific `ROUGE` parameters used to produce the `ROUGE-2` and `ROUGE-SU4` scores are as follows: `ROUGE-1.5.5.pl -n 2 -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -a` which generates recall, precision, and F-measure average scores (averaged across sentences in each summary) along with 95% confidence intervals for each summary.

For this test, we turned to the DUC02 data sets. These data contain 567 documents which are clustered into 59 topics. There are 1112 human model summaries, with approximately 2 summaries per document. In Table 9, we see that indeed the `ROUGE-1`, `ROUGE-2`, and `ROUGE-SU4` scores are significantly better when the summarization algorithm is given clusters.

Table 9. Single document `ROUGE` measures with and without clusters.

Clusters Given	*Method*	*Mean*	*95% CI Lower*	*95% CI Upper*
YES	`ROUGE-1`	0.44865	0.44045	0.45665
NO	`ROUGE-1`	0.43335	0.42498	0.44132
YES	`ROUGE-2`	0.18766	0.17891	0.19688
NO	`ROUGE-2`	0.17499	0.16615	0.18352
YES	`ROUGE-SU4`	0.21119	0.20353	0.21911
NO	`ROUGE-SU4`	0.20000	0.19210	0.20749

4.2.2 Experiments with Multidocument Summaries

The goal of these experiments was to determine whether the best machine-generated summary produced for a given DUC cluster is one using all of the documents for that cluster or a subset of those documents. In the cases where a better summary could be produced with fewer documents, we also ran experiments to determine if QCS is able to generate such summaries by incorporating document querying and clustering into the summarization process.

In these experiments, a multidocument summary was produced using the summarization module of QCS for each possible subset of two or more documents from each cluster. Since each DUC collection contained 10 documents, there were a total of 1013 subsets generated for each. Next, several queries for each cluster were generated from the cluster topic descriptions included as part of the DUC evaluations and used to run QCS. Finally, the output of QCS was compared to the human summaries and machine-generated summaries generated by the variant of the summarization module in QCS for each year of DUC.

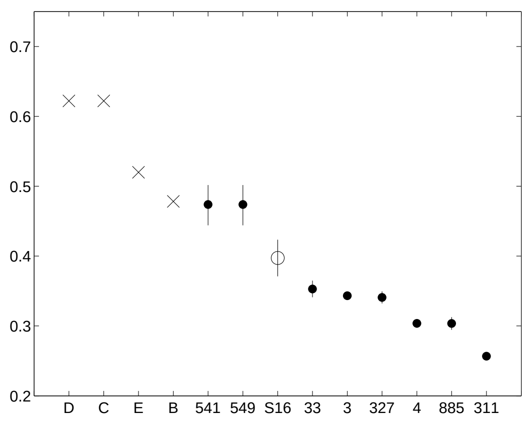
We used the topic descriptions for each cluster provided in the DUC 2003, Task 2 description to generate queries to be used in QCS. Three queries were generated for each cluster using the words from the 1) topic headline; 2) topic headline and seminal event description; and 3) topic headline, seminal event description, and topic explication. Our intent in using these different queries was to simulate a range of queries containing different amounts of information—from an ambiguous query with a few key words, query 1, to a query reflecting all known information on a particular subject of interest, query 3.

We present the results of the experiments for the d30003t and d31033t clusters. The topic of cluster d30003t is the arrest and trial of Chilean dictator General Augusto Pinochet in 1998, and cluster d31033t contains documents about the anti-trust case in the late 1990s against the Microsoft Corporation. To study the effects of cluster size on the quality of summaries produced by QCS, we ran QCS using each of the three queries and allowing for up to as many as $k = 2, \ldots, 9$ subclusters to be formed for each of the DUC clusters. Note that with such small document collections (10 documents for each cluster), the clustering module failed in several instances where too many singleton clusters were formed (i.e., when $k > 5$ maximum number of clusters were allowed).

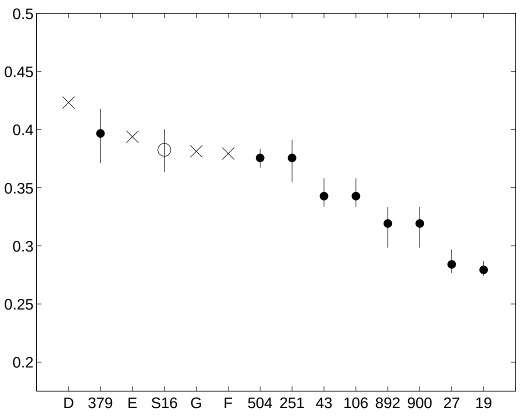
Figures 6–8 present the `ROUGE-1`, `ROUGE-2` and `ROUGE-SU4` recall scores for the human ($\times$), summarization module ($\circ$) and QCS ($\bullet$) summaries (over all runs where $k = 2, \ldots, 9$ subclusters were formed). The scores appear in descending order of average score from left to right, and include 95% confidence intervals for the machine-generated systems. Note that there are no confidence intervals for the human summaries since each human summary is scored once per cluster against all other human summaries. To remain consistent with the DUC evaluations, the summary labels appearing along the horizontal axes in the figures correspond to the summary labels used in the DUC evaluations (A–J for the humans and S# for the system number assigned to the variant of the summarization module submitted to DUC.) These results suggest that an improvement in summary quality can be made using QCS in place of the summarization module alone. In all but one case— cluster d31033t using the DUC04 collection—at least one summary returned by QCS has a higher average score than those of the summaries produced using the summarization module. However, the results suggest only marginal improvement, as illustrated in the overlap of the confidence intervals for the scores.

Figures 9–11 present the same `ROUGE` scores as a function of the number of clusters formed in the clustering module of QCS. The dotted lines denote the score(s) of the summaries generated by the different variants of the summarization module and submitted to the DUC evaluations. These results suggest that the number of clusters formed in QCS affects the quality of the summary produced. Although the improved QCS summaries are not generated using the same number of clusters across all of the experiments, the appearance of trends in the soring data between summary quality and the number (and thus size) of QCS clusters suggests a potential relationship that may be leveraged using QCS.

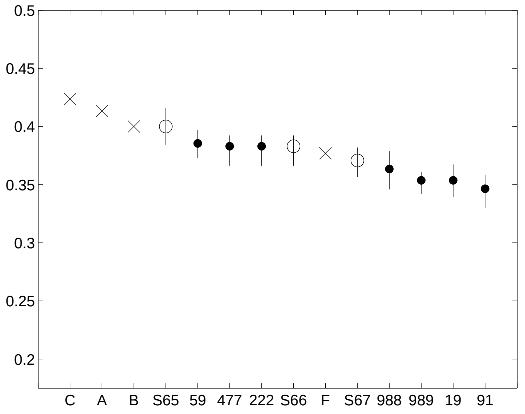
We conclude from this experiment that the clustering of documents used for multidocument summarization can greatly affect the quality of the summary produced. Specifically, determining subclusters (i.e., subtopic detection) is critical for accurately reflecting the information conveyed by a set of documents through automatically generated summaries. Furthermore, we have demonstrated that the use of clustering as a preprocessing step used before performing automatic summarization can help improve summaries generated.



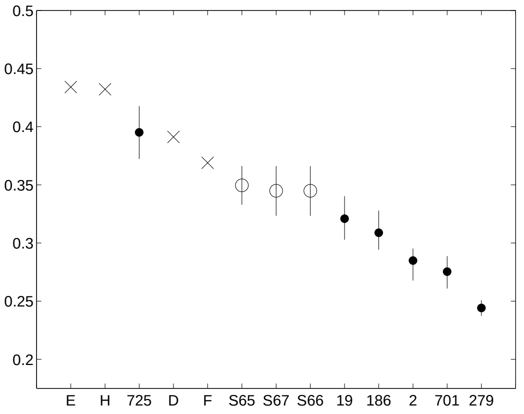
(a) d30003t, DUC03-2



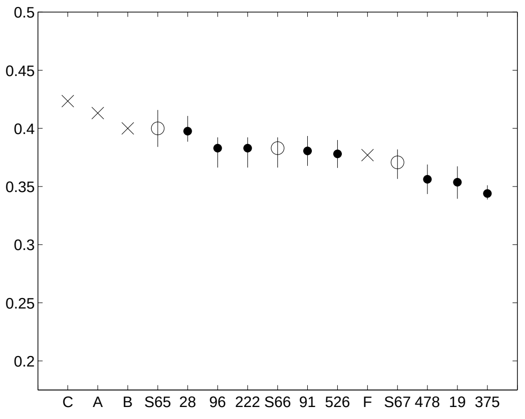
(b) d31033t, DUC03-2



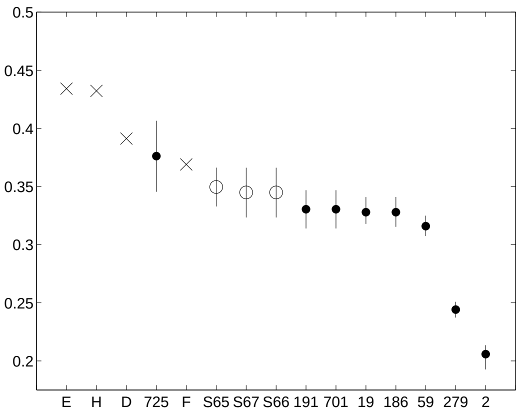
(c) d30003t, DUC04



(d) d31033t, DUC04

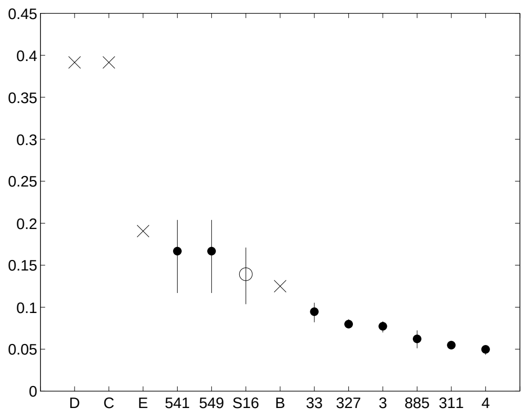


(e) d30003t, DUC04-PRE

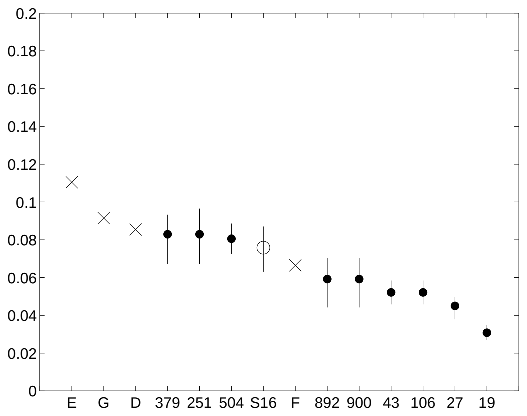


(f) d31033t, DUC04-PRE

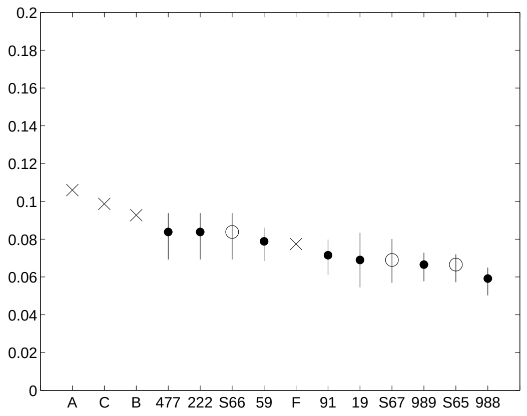
Figure 6. `ROUGE-1` recall scores plotted with 95% confidence intervals (lines) for the human (×), summarization module (○) and QCS (●) summaries for clusters d30003t and d31033t. The scores appear in descending order of average score from left to right.



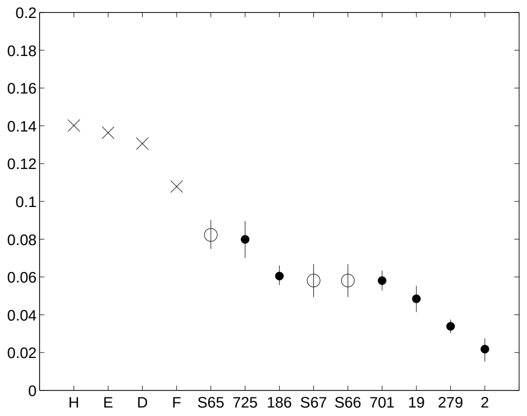
(a) d30003t, DUC03-2



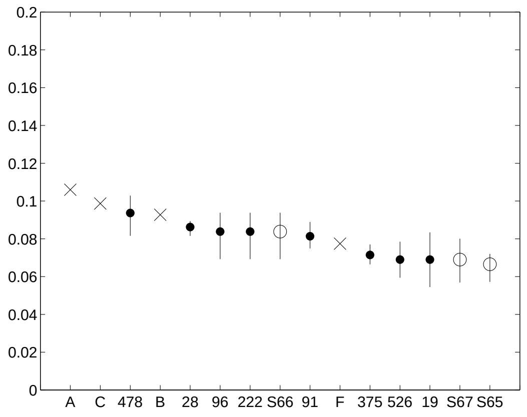
(b) d31033t, DUC03-2



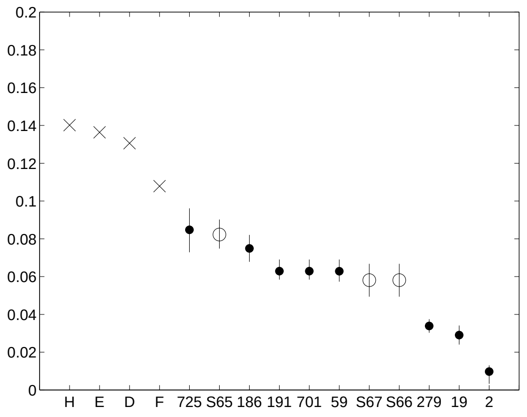
(c) d30003t, DUC04



(d) d31033t, DUC04

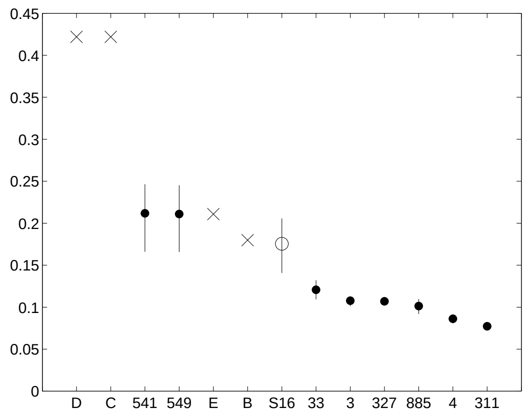


(e) d30003t, DUC04-PRE

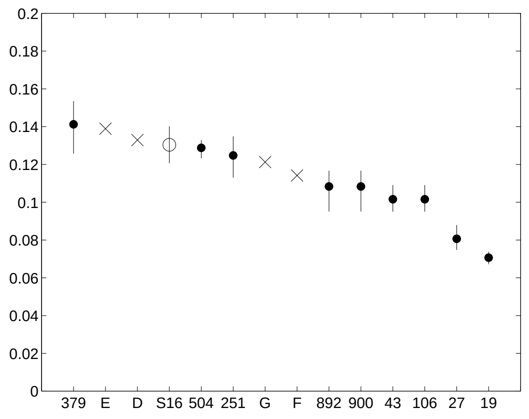


(f) d31033t, DUC04-PRE

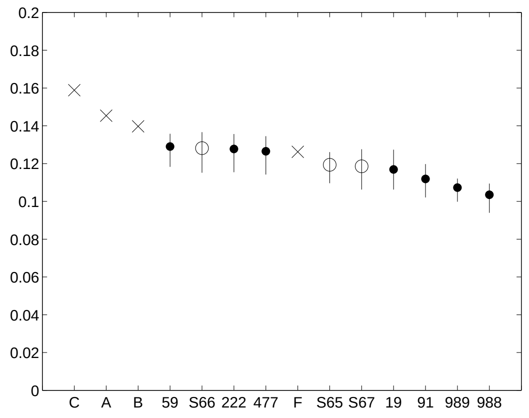
Figure 7. `ROUGE-2` recall scores plotted with 95% confidence intervals (lines) for the human (×), summarization module (○) and QCS (●) summaries for clusters d30003t and d31033t. The scores appear in descending order of average score from left to right.



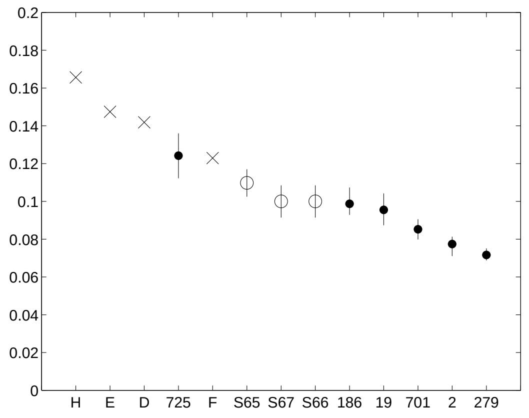
(a) d30003t, DUC03-2



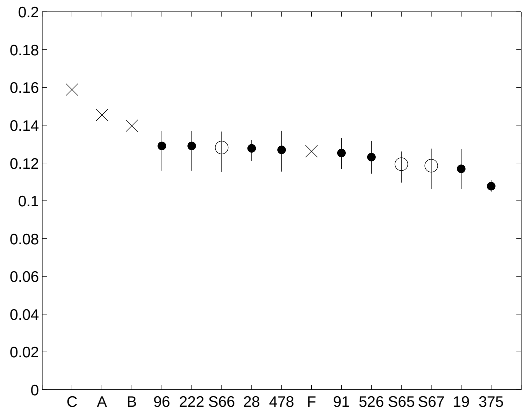
(b) d31033t, DUC03-2



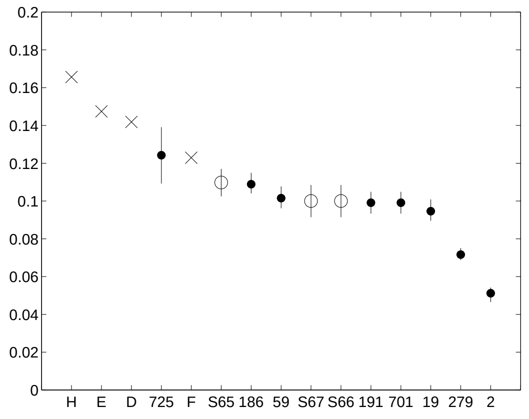
(c) d30003t, DUC04



(d) d31033t, DUC04

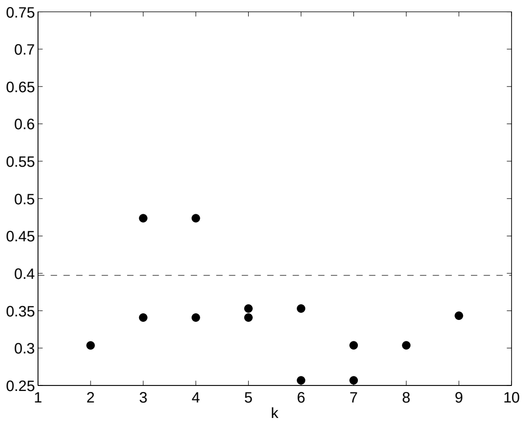


(e) d30003t, DUC04-PRE

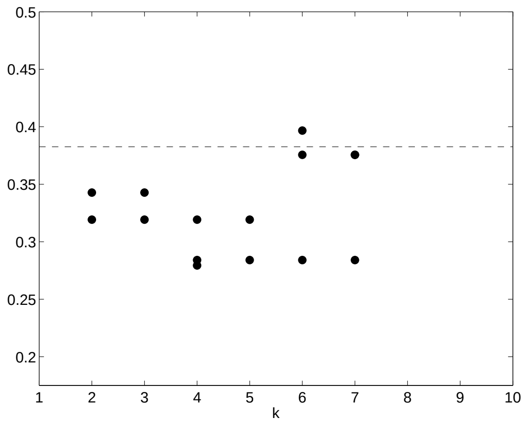


(f) d31033t, DUC04-PRE

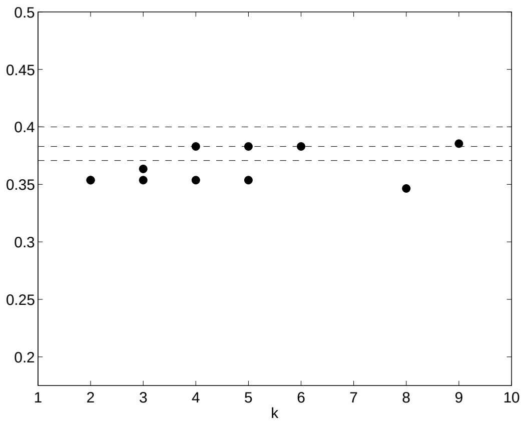
Figure 8. `ROUGE-SU4` recall scores plotted with 95% confidence intervals (lines) for the human (×), summarization module (○) and QCS (●) summaries for clusters d30003t and d31033t. The scores appear in descending order of average score from left to right.



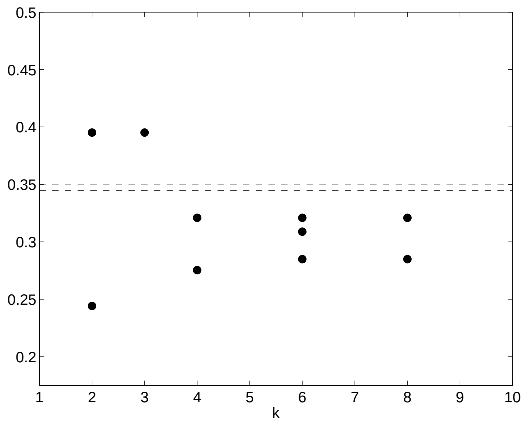
(a) d30003t, DUC03-2



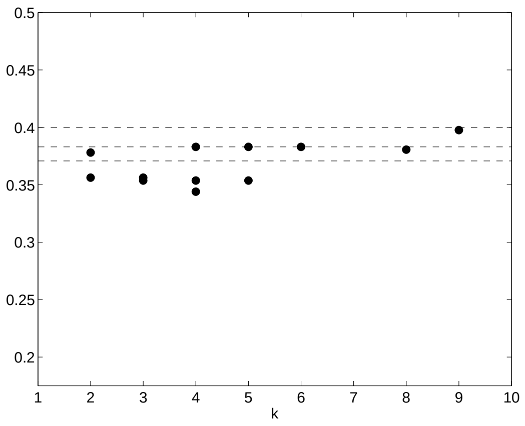
(b) d31033t, DUC03-2



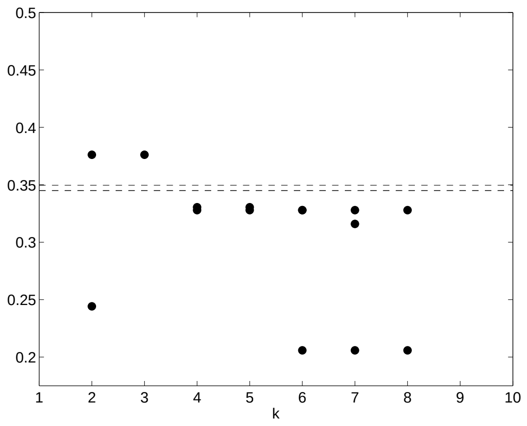
(c) d30003t, DUC04



(d) d31033t, DUC04

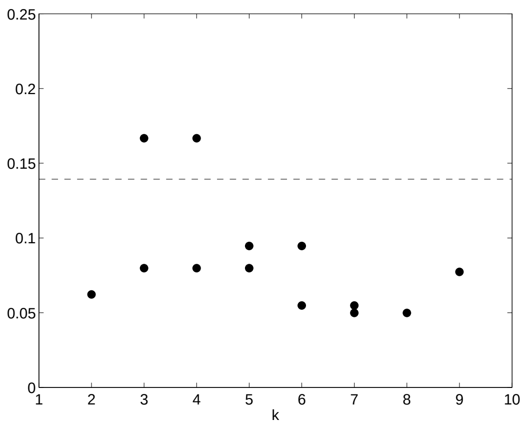


(e) d30003t, DUC04-PRE

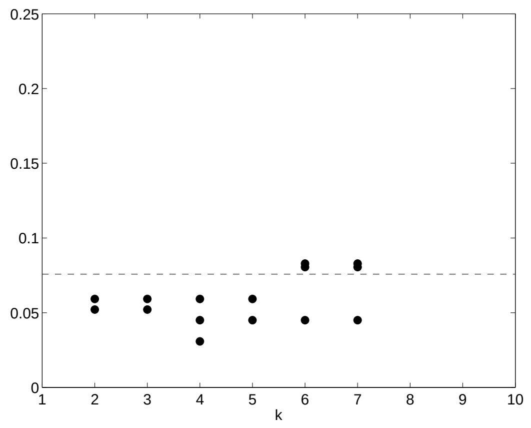


(f) d31033t, DUC04-PRE

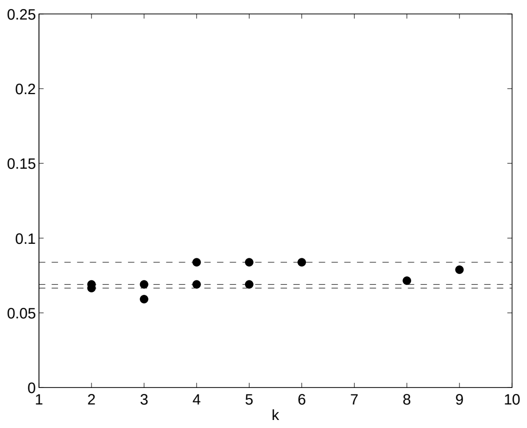
Figure 9. `ROUGE-1` recall scores for the QCS summaries for clusters d30003t and d31033t as a function of the number of clusters formed (k). The dotted lines represent the corresponding scores for the summarization module summaries.



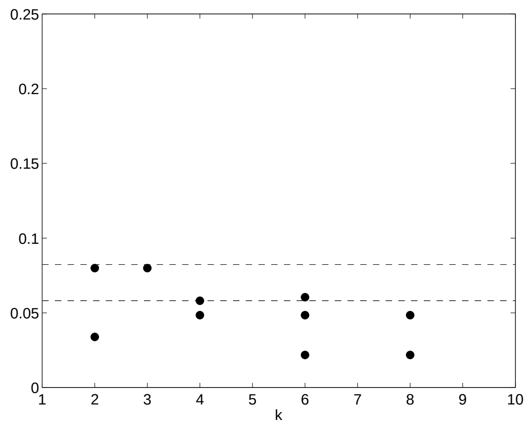
(a) d30003t, DUC03-2



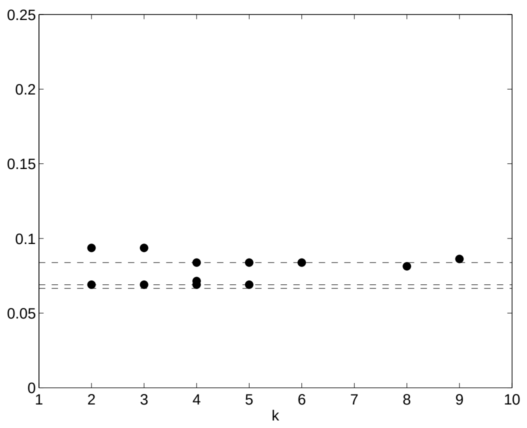
(b) d31033t, DUC03-2



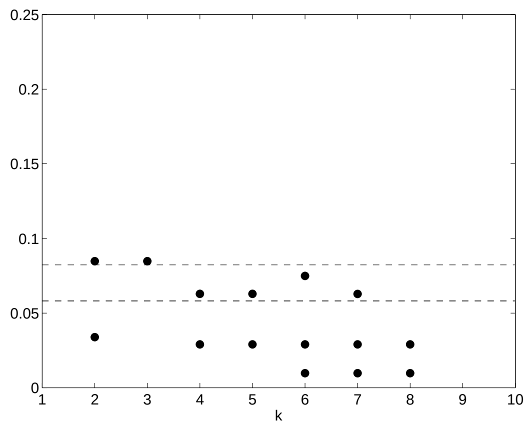
(c) d30003t, DUC04



(d) d31033t, DUC04

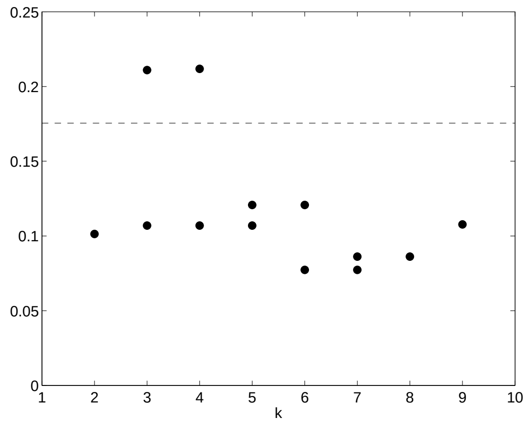


(e) d30003t, DUC04-PRE

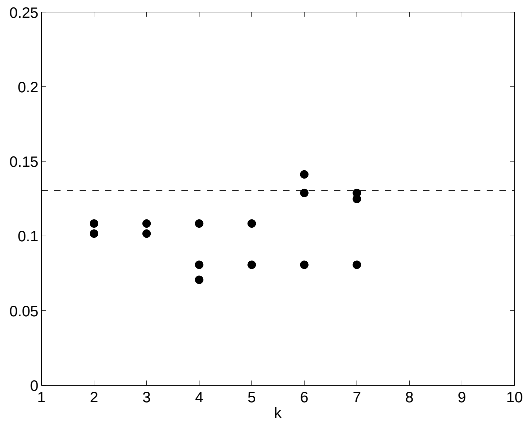


(f) d31033t, DUC04-PRE

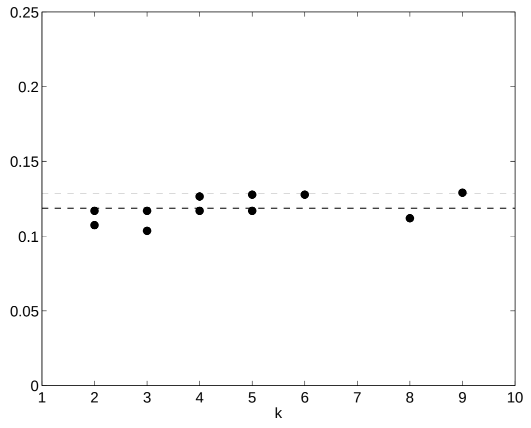
Figure 10. `ROUGE-2` recall scores for the QCS summaries for clusters d30003t and d31033t as a function of the number of clusters formed (k). The dotted lines represent the corresponding scores for the summarization module summaries.



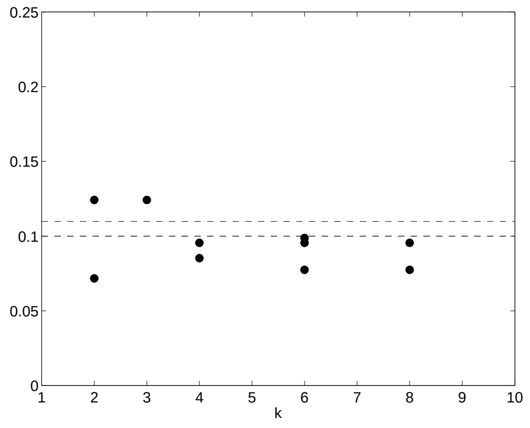
(a) d30003t, DUC03-2



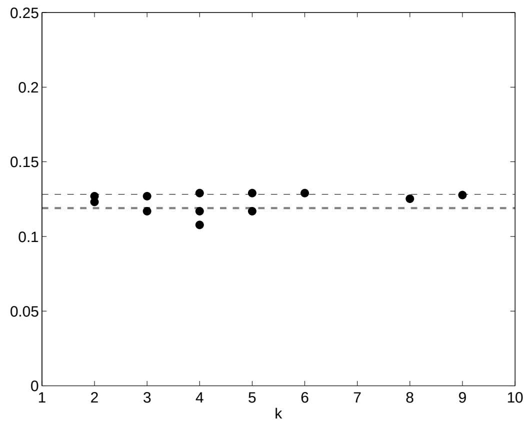
(b) d31033t, DUC03-2



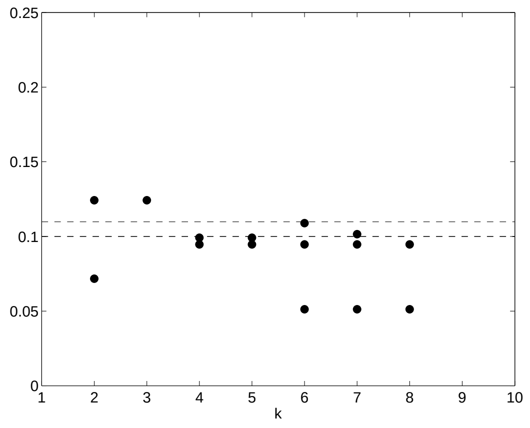
(c) d30003t, DUC04



(d) d31033t, DUC04



(e) d30003t, DUC04-PRE



(f) d31033t, DUC04-PRE

Figure 11. `ROUGE-SU4` recall scores for the QCS summaries for clusters d30003t and d31033t as a function of the number of clusters formed (k). The dotted lines represent the corresponding scores for the summarization module summaries.

4.3 Experiments with QCS on a Larger Diverse Document Collection

In the final set of experiments, we focused on the effects of querying and clustering on summarization, both independently and in the full QCS system, on a larger collection of documents covering a wide variety of topics. The collection consisted of documents from all clusters in the DUC 2003 Task 4 evaluation data where 1) a topic description was provided, 2) a summary generated using the summarization module was submitted to the DUC 2003 Task 4 evaluation, and 3) four human-generated summaries were provided. There were 28 clusters which met this criteria, resulting in a collection of 625 files.

For each of the 28 clusters, we generated several summaries using four different methods, as well as the method we submitted to the DUC 2003 evaluation. The first method is the full QCS system. As in the experiments in the previous section, queries were derived from each topic description. The topic descriptions for the DUC03-4 data included a title, two short descriptions, and a topic narrative. Four queries were created for each topic description using 1) title only, 2) descriptions only, 3) title and descriptions, and 4) all topic information. Using the default QCS setup, up to 10 multidocument summaries were generated per query.

The second method, denoted *QL*, combines the QCS query module and lead sentence extraction to generate one multidocument summary per query. Given a query, a subset of documents is retrieved and ordered by query score. A multidocument summary is then produced using the lead sentence with $stype = 1$ from each of the top scoring documents until the total number of words in these sentences exceeds 100. As in the experiments in the previous section, four queries derived from the topic descriptions were used to retrieve a subset of the 625 documents. In many of the DUC evaluations, similar lead-sentence summaries have been used as baselines, representing a summarization approach requiring minimal text and/or natural language processing. However, since the DUC evaluation data consists of newswire documents, such baseline summaries have performed fairly well compared to many more sophisticated approaches in several of the DUC evaluations [23, 7].

The third method, denoted *QS*, is similar to the QL method, but uses the QCS summarization module instead of lead-sentence extraction to generate a summary. Again, given a query, a subset of documents is retrieved and ordered by query score. The top scoring document and those documents with query scores within 30% of the top score are collected into a cluster and a single multidocument summary is generated for this cluster using the QCS summarization module.

The final method, denoted *CS*, combines the clustering and summarization modules from QCS to generate several multidocument summaries. Given a cluster of n documents, the clustering module generates a maximum of $k = \min\{10, n/2\}$ subclusters starting with 2 randomly seeded initial subclusters. Multidocument summaries for each of the resulting k subclusters are then generated using the QCS summarization module.

Figures 12–14 present the `ROUGE-1`, `ROUGE-2`, and `ROUGE-SU4` recall scores for all of these systems for each of the 28 DUC clusters. For QL (□) and QS (+), four summaries associated with each of the DUC clusters were produced (one for each query); for CS (◊), an average of 9.14 summaries were produced per DUC cluster (due to the varying number of subclusters generated); and for QCS (•), an average of 33.5 summaries associated with each DUC cluster generated (using the four summaries and generating up to 10 clusters). The results presented in Figures 12–14 only show the top scoring summary for each of the QCS, QL, QS, and CS methods.

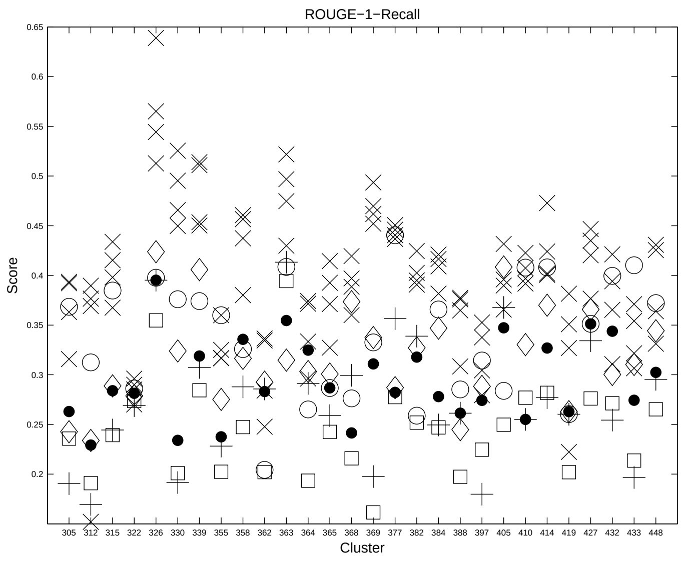


Figure 12. `ROUGE-1` recall scores for DUC03 Task 4 data for the human (×) and summarization module (○) summaries, along with the top scoring QCS (•), QL (□), QS (+), and CS (◊) summaries.

Table 10 presents the results of pairwise comparisons of the top scoring summaries generated by the five methods. The entry in the row labeled QCS and the column labeled S, for instance, indicates that S had a better `ROUGE-2` score on 57% of the 28 instances. There is much variability in scores across the different experiments, as shown in Figures 12–14. However, the pairwise comparisons of methods using `ROUGE-1`, `ROUGE-2`, and `ROUGE-SU4` suggests the following overall performance ordering: S, CS, QCS, QS, and QL. Although QCS is not the top-performing method throughout all of the experiments, we note that it outperforms S and CS at least 25% of the time using any of the `ROUGE` scores and it outperforms S and C as much as 43% and 32% of the time, respectively, evaluated using `ROUGE-2` scores. Furthermore, both S and CS relied on human intervention to obtain the relevant documents.

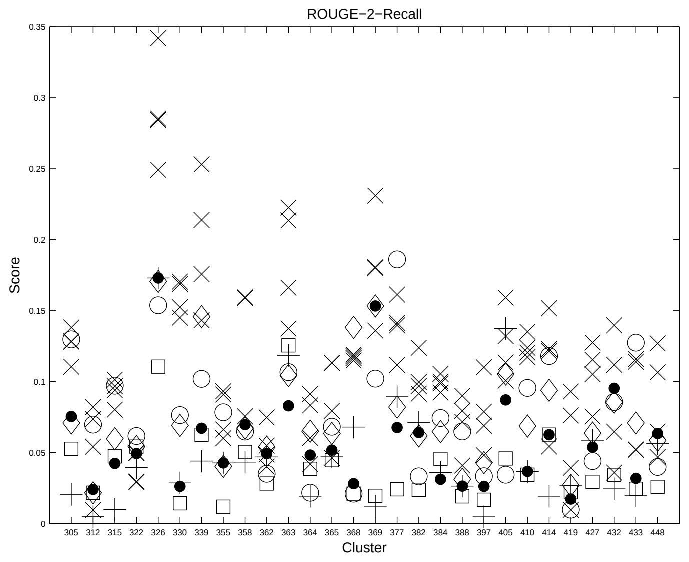


Figure 13. ROUGE-2 recall scores for DUC03 Task 4 data for the human (×) and summarization module (○) summaries, along with the top scoring QCS (•), QL (□), QS (+), and CS (◊) summaries.

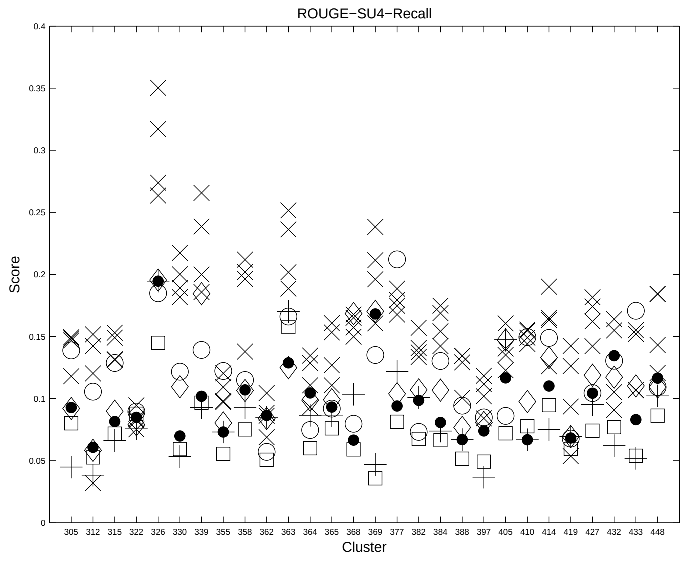


Figure 14. ROUGE-SU4 recall scores for DUC03 Task 4 data for the human (×) and summarization module (○) summaries, along with the top scoring QCS (•), QL (□), QS (+), and CS (◊) summaries.

Table 10. Comparison of `ROUGE` scores of top scoring summaries for DUC03-4 data. The entry in each table is the percent of times that the system corresponding to the column outperformed the system corresponding to the row using the `ROUGE-1`, `ROUGE-2`, and `ROUGE-SU4` scores.

	S	CS	QCS	QS	QL
S	–	39	25	21	0
CS	61	–	29	14	4
QCS	75	71	–	21	7
QS	79	86	79	–	32
QL	100	96	93	68	–

`ROUGE-1`

	S	CS	QCS	QS	QL
S	–	46	43	32	14
CS	54	–	32	21	11
QCS	57	68	–	32	21
QS	68	79	68	–	32
QL	86	89	79	68	–

`ROUGE-2`

	S	CS	QCS	QS	QL
S	–	43	36	29	0
CS	57	–	32	14	7
QCS	64	68	–	21	11
QS	71	86	79	–	39
QL	100	93	89	61	–

`ROUGE-SU4`

We conclude from these experiments that QCS preforms well in producing summaries for automatically generated clusters of documents, rivaling summaries generated using manual processing of data. The benefit of using QCS over such methods is that it is a fully automatic system for document retrieval, organization, and summarization.

5 Future Directions

Optimization of the code through the use of persistent variables would improve performance. For instance, loading the term-document matrix as a persistent data array would speed up query processing, but inconsistencies in how `GTP`, `GMEANS` and the `HMM+QR` modules access the documents present challenges in implementing this.

A prototype Application Programming Interface (API) has been developed to function as a wrapper around the various components of the QCS system. Incorporating a different algorithm for any one of the components of the QCS system would provide insight on the robustness of the modularity and scalability of the system.

Modifications to the parsing and indexing routines of the current QCS system could improve the performance of the querying module. Specifically, explicit term stemming, query expansion, and allowing phrases (such as multiword person names, company names, countries, etc., i.e., named entities) to be terms have been shown in the IR literature to help improve query-based IR tools.

Only one of the distance measures from `GMEANS` is currently used, and testing the others would be useful. A more rigorous analysis of the range for the number of starting clusters as well as the upper limit on the number of clusters is also planned.

As a possible alternative to `GMEANS`, a support vector machine (SVM) could be implemented to perform the categorical clustering of the query results. As there are several implementations of SVM's currently available, this could serve as a test of the modularity of the QCS system as described above.

The process of creating a summary document for each of the k clusters is inherently parallelizable. Work has already begun in preparing the existing code for parallelization.

It can be seen in Tables 7 and 8 that processing all of the documents in a corpora with a POS-tagger is extremely costly in terms of time required, even if it need be done only once. Also, current state-of-the-art POS-taggers have a 2–3% error rate. These errors then contribute to poor trimming decisions. For these reasons, we decided to eliminate the POS-tagging and make the trimming decisions based on the word and punctuation patterns in a sentence rather than relying on POS tags. Creating new patterns that found the desired text to eliminate has been a non-trivial process and was the major effort until it was recently completed. Our DUC 2006 submission uses these new patterns and the results from that will help with an evaluation. Ongoing work includes new trimming patterns, anaphora resolution as mentioned in Section 2.4.2, and using named entities more effectively.

The current interface to the QCS system does not allow the user to choose any of the algorithmic parameters specified throughout this report except by recompiling the system. Detailed analysis via parameter estimation techniques could highlight the parameters that most greatly affect the quality of the multidocument summaries. Once this is accomplished, fields for specifying these important parameters could be added to the user interface. As is typical in several IR tools, the plan is to include a link to a more advanced interface to include these parameter fields, so as to not hinder users who are satisfied with results produced using default values.

Finally, we aim to research scalability issues for QCS applied to very large document collections by incorporating new modules into QCS which support parallel computation and developing a framework for parallel operations (e.g., data distribution, load balancing, operation queuing, etc.).

6 Conclusions

QCS is a tool for document retrieval that presents results in a format so that a user can quickly identify a set of documents of interest. The results include a multidocument summary of each cluster of documents, a summary of each individual document, a pointer to each document, and pointers to documents from which the multidocument extract summary was derived. Results of using QCS on the DUC document set illustrate the usefulness of this system; in particular, we provide evidence of the value of clustering as a tool for increasing the quality of the summaries.

The QCS system has been developed as a completely modular tool, enabling new methods to be integrated into the system as improvements are made in the areas of query, clustering, and summarizing documents. It has been developed as a client-server application in which the client can be run from any platform that can process HTML documents, which currently includes most major computing platforms.

References

[1] G. Aas and M. A. Chase. HTML-Parser, 2003. http://search.cpan.org/~gaas/HTML-Parser/.

[2] Apache Software Foundation. Apache Tomcat, 2003. http://tomcat.apache.org/.

[3] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. Maximization technique occurring in statistical analysis of probabilistic functions in markov chains. *The Annals of Mathematical Statistics, 41(1):164–171*, 41(1):164–171, 1970.

[4] M. W. Berry and D. I. Martin. Parallel SVD for scalable information retrieval. In *Proc. Intl. Workshop on Parallel Matrix Algorithms and Applications*, Neuchatel, Switzerland, 2000.

[5] 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, 2001.

[6] J. M. Conroy, J. D. Schlesinger, J. Goldstein, and D. P. O'Leary. Left-brain/right-brain multi-document summarization. In *Proc. Document Understanding Conference*, 2004.

[7] H. T. Dang. Overview of DUC 2005. In *Proc. Document Understanding Conference*, 2005.

[8] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *J. Am. Soc. Inform. Sci.*, 41(6):391–407, 1990.

[9] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. K. R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001.

[10] I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, 2002.

[11] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Mach. Learn.*, 42(1):143–175, 2001.

[12] D. M. Dunlavy, J. M. Conroy, T. J. O'Leary, and D. P. O'Leary. Clustering and summarizing Medline abstracts. In *BISTI 2003 Symposium on Digital Biology: The Emerging Paradigm*. National Institutes of Health Biomedical Information Science and Technology Initiative (BISTI), 2003.

[13] D. M. Dunlavy, J. M. Conroy, J. D. Schlesinger, S. A. Goodman, M. E. Okurowski, D. P. O'Leary, and H. van Halteren. Performance of a three-stage system for multi-document summarization. In *Proc. Document Understanding Conference*, 2003.

[14] T. Dunning. Accurate methods for statistics of surprise and coincidence. *Comput. Linguist.*, 19:61–74, 1993.

[15] W. N. Francis and H. Kucera. *Frequency Analysis of English Usage: Lexicon and Grammar.* Houghton Mifflin Company, Boston, MA, 1982.

[16] J. T. Giles, L. Wo, and M. W. Berry. GTP (General Text Parser) software for text mining. In H. Bozdogan, editor, *Statistical Data Mining and Knowledge Discovery*, pages 455–471. CRC Press, Boca Raton, 2003.

[17] T. G. Kolda and D. P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing in information retrieval. *ACM T. Inform. Syst.*, 16(4):322–346, 1998.

[18] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization (WAS 2004)*, Barcelona, Spain, 2004.

[19] C.-Y. Lin and E. Hovy. The automatic acquisition of topic signatures for text summarization. In *Proc. Document Understanding Conference*, 2002.

[20] M. J. Maña-López, M. de Beunaga, and J. M. Gómez-Hidalgo. Multidocument summarization: An added value to clustering in interactive retrieval. *ACM T. Inform. Syst.*, 22:215–241, 2004.

[21] K. R. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, J. L. Klavans, A. Nenkova, C. Sable, B. Schiffman, and S. Sigelman. Tracking and summarizing news on a daily basis with Columbia's Newsblaster. In *Proceedings of the Human Language Technology Conference*, 2002.

[22] A. Mikheev. Tagging sentence boundaries. In *Proc. NAACL Conference*, pages 264–271, Seattle, WA, 2000. Morgan Kaufmann.

[23] P. Over and J. Yen. An introduction to DUC-2004: Intrinsic evaluation of generic news text summarization systems. In *Proc. Document Understanding Conference*, 2004.

[24] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *P. IEEE*, 77:257–285, 1989.

[25] D. R. Radev, S. Blair-Goldensohn, Z. Zhang, and R. S. Raghavan. Newsinessence: A system for domain-independent, real-time news clustering and multi-document summarization. In *Proc. HLT Conference*, San Diego, CA, 2001.

[26] D. R. Radev, W. Fan, and Z. Zhang. Webinessence: A personalized web-based multi-document summarization and recommendation system. In *Proc. NAACL Workshop on Automatic Summarization*, Pittsburgh, PA, 2001.

[27] G. Salton. *Automatic Text Processing – The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley, 1989.

DISTRIBUTION:

2 MS 9018
Central Technical Files, 8944

2 MS 0899
Technical Library, 4536