Evolving XML and Dictionary Strategies for Question Answering and Novelty Tasks

Kenneth C. Litkowski
CL Research
9208 Gue Road
Damascus, MD 20872
ken@clres.com

Abstract

CL Research participated in the question answering and novelty tracks in TREC 2004. The Knowledge Management System (KMS), which provides a single interface for question answering, text summarization, information extraction, and document exploration, was used for these tasks. Question answering is performed directly within KMS, which answers questions either from a repository or the Internet. The novelty task was performed with the XML Analyzer, which includes many of the functions used in the KMS summarization routines. These tasks are based on creating and exploiting an XML representation of the texts used for these two tracks. For the QA track, we submitted one run and our overall score was 0.156, with scores of 0.161 for factoid questions, 0.064 for list questions, and 0.239 for "other" questions; these scores are significantly improved from TREC 2003. For the novelty track, we submitted two runs for task 1, one run for task 2, four runs for task 3, and one run for task 4. For most tasks, our scores were above the median. We describe our system in some detail, particularly emphasizing strategies that are emerging in the use of XML and lexical resources for the question answering and novelty tasks.

1 Introduction

In TREC 2002, CL Research examined the potential of using XML-tagged documents for question answering (Litkowski, 2003a) and showed that hand-developed XPath expressions could obtain extremely good results when compared with the best sytems. In TREC 2003 (Litkowski, 2004), initial efforts at the automatic creation of XPath expressions achieved very limited results. As noted last year, these efforts were embedded in CL Research's Knowledge Management System (KMS), which provides an integrated framework for question answering, text summarization (including novelty assessment), information extraction, and document exploration. As indicated in these previous papers, use of XML characterizations of texts raises many challenging research issues. This paper describes evolving XML and dictionary strategies employed in KMS, with specific reference to CL Research's participation in TREC 2004 question answering and novelty tracks. In particular, these strategies have emerged from the fact that the hand-developed XPath expressions are idiosyncratically specific and are not generalizable.

KMS was initially developed as an interface to provide integrated access to XML representations of documents for question answering and summarization. In the past year, it has been extended to include a wrapper for web question answering, use of a search engine for access to a document collection (such as the AQUAINT collection), integrated word-sense disambiguation, and mechanisms for document exploration based on an ontological analysis of documents.

At the root of KMS, Extensible Markup Language (XML) provides a natural mechanism for representing texts, from small snippets like titles through extensive collections of texts. A valid XML document is a tree that can represent questions, topic descriptions (such as used in the novelty task), individual documents, and collections of documents. Generally, the representations create nodes for sentences, clauses, phrases, and words. Each node in the tree will generally have associated attribute names and values. A major challenge is determining an appropriate set of metadata (tags, attributes, and values) for different tasks. The central approach in performing these tasks focuses on nodes in the XML representation, primarily sentences and noun phrases.

Another central component of KMS is the use of lexical resources. Each task (question answering, summarization, or document exploration) involves an attempt to "understand" text. Understanding requires semantic analysis to characterize and process entities, events, and relations. KMS now includes many specialized lexical resources: a dictionary used in parsing, a machine-readable dictionary, WordNet, the Specialist Lexicon of the Unified Medical Language

System, specially developed verb and preposition dictionaries, and a Roget-style thesaurus.

Section 2 presents the TREC QA and Novelty problem descriptions. Section 3 describes the KMS, specifically components for processing texts and for performing particular NLP tasks. Section 4 provides our question answering results, particularly our experience in handling different types of questions. Section 5 describes our novelty experiments, particularly identifying insights about the nature of the task as they emerged. Section 6 describes the underlying functionality used in performing the question answering and novelty tasks.

2 Problem Description

The TREC 2004 QA and Novelty tasks used the AQUAINT Corpus of English News Text on two CD-ROMs, about one million newswire documents from the *Associated Press Newswire*, *New York Times Newswire*, and *Xinhua News Agency*. These documents were stored with SGML formatting tags (XML compliant).

For the QA track, participants were provided with 65 targets, primarily names of people, groups, and organizations, viewed as entities for which definitional information was to be assembled. For each target, a few factual questions were posed, totaling 230 factoid questions for the 65 targets (e.g., "Who founded the Black Panthers organization?" and "Where was it founded?"). One or two list questions for each target were also posed for most of the targets (e.g., "Who were members of the Black Panthers?"); there were 56 list questions. Finally, for each target, "other" information was to be provided, simulating an attempt to "define" the target. Each target was used as a search query against the AQUAINT corpus and the top 50 documents were provided, along with a list of the top 1000 documents.

Participants were required to answer the 230 factoid questions with a single exact answer, containing no extraneous information and supported by a document in the corpus. A valid answer could be NIL, indicating that there was no answer in the document set; NIST included 22 questions for which no answer exists in the collection. For these factoid questions, NIST evaluators judged whether an answer was correct, inexact, unsupported, or incorrect. The submissions were then scored as percent of correct answers. For the list questions, participants returned a set of answers (e.g., a list of chewing gums); submissions were given F-scores, measuring recall of the possible set of answers and the precision of the answers returned. For the "other" questions, participants provided a set of answers. These answer sets were also scored with an F-score, measuring whether the answer set contained certain "vital" information and how efficiently peripheral information was captured (based on answer lengths).

CL Research submitted one run for the question-answering track.

For the Novelty track, participants were provided with descriptions of 50 topics (labeled as "event" or "opinion") and a set of 25 to 30 documents for each topic, with as many as 5 not relevant. These documents were further broken down into sentences. There were four tasks. For the first task, participants were to identify sentences relevant to the topic and then to identify which of these sentences provided novel or new information. For task 2, participants were given the relevant sentences from all documents and asked to identify those which were new. For task 3, participants were provided with the relevant and new sentences from the first five documents and asked to identify the relevant and new sentences for the remaining 20 documents. For task 4, participants were given the relevant sentences from all documents and the new sentences from the first 5 documents and asked to identify the new sentences in the last 20 documents. The tasks were spread out over a three week period, with participants given additional information at the end of the first week.

CL Research submitted two runs for task 1, one run for task 2, four runs for task 3, and one run for task 4.

3 The Knowledge Management System

The CL Research KMS is a graphical interface that enables users to create repositories of files (of several file types) and to perform a variety of tasks against the files. The tasks include question answering, summarization (from headlines to summaries of any length), information extraction, and document exploration. The text portions of files are processed into an XML representation and the actual performance of each task revolves around an XML-based analysis of the texts. In the past year, KMS has been extended to include web-based question answering (acting as a wrapper to Google) and indexing of document repositories (using the publicly available Lucene indexing and search engine).

KMS uses lexical resources as an integral component in performing the various tasks. Specifically, KMS employs dictionaries developed using its DIMAP dictionary maintenance programs,

available for rapid lookup of lexical items. CL Research has created DIMAP dictionaries for a machine-readable version of the *Oxford Dictionary of English*, WordNet, the Unified Medical Language System (UMLS) Specialist Lexicon (which provides a considerable amount of syntactic information about lexical items), *The Macquarie Thesaurus*, and specialized verb and preposition dictionaries. These lexcial resources are used seamlessly in a variety of ways in performing various tasks, described in more detail below.

KMS consists of a large number of modules. These modules are also used in two additional programs that perform more specialized processing. The DIMAP Text Parser uses the parsing and XML-generation components for background processing of large numbers of texts, such as the document sets for the QA and novelty tasks. The XML Analyzer, primarily used to enable detailed examination of XML files for diagnostic purposes, includes special processing routines to perform the novelty task, primarily for mapping NIST sentences to KMS sentences.

When performing a task with KMS, results are maintained in XML representations. For example, all answers to questions are retained and saved answers can be viewed without going through question answering again. Answers include the file they came from, the document within the file, the score, an exact answer, and the sentence from which they were extracted. (During development, as question answering routines are modified, new answer sets replace existing answer sets.)

Improving the performance of KMS in its various tasks requires improved characterization of the texts, development of appropriate strategies for performing the tasks, and the use of efficient mechanisms for evaluating performance. XML provides a convenient formalism for representation and analysis, but this approach involves frequent reformulations to capture and recognize large numbers of linguistic phenomena. Improvements come from first dealing with low-level phenomena, sometimes idiosyncratically, and then, as patterns emerge, from generalizing over several phenomena. In general, improved characterization of linguistic phenomena leads to improved performance.

3.1 Text Processing

As described in previous years (see Litkowski, 2004 and references therein), the text processing component consists of three elements: (1) a sentence splitter that separates the source documents into individual sentences; (2) a parser which takes each sentence and parses it, resulting in a parse tree containing the constituents of the sentence; and (3) a parse tree analyzer that identifies important discourse constituents (sentences and clauses, discourse entities, verbs and prepositions) and creates an XML-tagged version of the document.

The rendition of raw text into XML is not completely accurate. Problems arise from improper sentence splitting, the inability of the parser to handle many sentences, incomplete and inaccurate mapping of the parse output into the underlying data structures, and inaccurate rendition of the underlying data into a complete XML representation. Sentence splitting is over 99 percent accurate. Poor parses occur in about 50 percent of the sentences, but the parse output generally provides chunks (such as noun phrases), so that the various underlying data are reasonably complete. However, only about 75 percent of the underlying is ultimately rendered into XML, primarily as a result of difficulties in handling clauses.

In the past year, two major changes were made in the XML representation. Before, discourse entities were represented solely as units, with various attributes (such as the base form of plural heads or whether it contained an ordinal or degree adjective) in the node for the entire unit. These entities have now been given substructure, identifying and characterizing the component words (adjectives, conjunctions, adverbs, and nouns) and these words have been disambiguated using WordNet. In addition, the possible semantic types of nouns and verbs have been identified, using WordNet's broad semantic groups.

For TREC, there was an XML file for each of the 65 targets in the QA track and for each of the 50 topics in the novelty track. XML tagging of these files resulted in files that were 9.1 times the size of the original documents, up from 6.7 during TREC 2003.

Questions in the QA track and the title, description, and narrative in the novelty track are processed using the same text processing mechanisms. The XML representations of the track "topic" statements are used in performing the question answering or novelty detection. Since these topic statements are not as voluminous as the documents for these topics, they tend to be somewhat smaller and less elaborate in complexity.

The tagging process is in continual development. Improvements arise in the first instance from detecting bugs in the parser, extracting more information from the parse results, and detecting bugs in the creation of the lists for sentences, clauses, noun phrases, verbs, and prepostions. In the second instance, improvements arise from improved use of lexical resources for

characterizing the various elements. Finally, improvements occur as the system is expanded to cover more linguistic phenomena, particularly in characterizing semantic relations between various discourse elements.

3.2 KMS Question Answering Strategies

As indicated above, KMS has been modified to include a wrapper for a Google search. Users can select to have questions answered using a repository of documents or from Google. As with answers against a repository, web answers are maintained in their own file, in this case showing the web address of the document answering the question. KMS can use the TREC question set and have them submitted to Google, or a user can pose any question and the question is saved in a repository of questions.

Answering a question using Google (or any web-based search engine) goes through a slightly different strategy than against a repository. The main difference is how the question is analyzed and formulated for submission to Google. Given the breadth of the Internet, there is a strong likelihood that a question can be answered using a canonical form. For example, to the "When was John Glenn born?", it is likely that "John Glenn was born" will exist as an exact phrase and that a search for the entire phrase as given will return documents where the displayed Google results will contain the answer.

Each question must be reformulated into an exact query. KMS analyzes the question type and its constituent terms to determine what is likely to be the best form. For the most part, the key transformation involves the placement, number, and tense of the verb. For example, "How did James Dean die?" has to be transformed to "James Dean died". KMS uses the UMLS lexicon for this purpose, since it contains information about the "regularity" of a verb form and how it's various tenses are formed. Special routines are used for "definition" questions (e.g., "What is a cataract?") to make use of Google's "define:" operator.

In answering a question from Google, it is usually unnecessary to go to the source document. In its search results, Google displays a snippet that usually consists of one or two sentences, along with some additional partial sentences. KMS examines the snippet and extracts the whole sentence containing the exact phrase. This single sentence is then treated as a document and is parsed and processed into an XML representation. Since Google usually returns 10 hits per page, the corpus to answer a question usually consists of 10 single-sentence documents. (For definition questions, Google may return many more "documents", each of which is in the form of a definition from some source; a definition may contain more than one sentence, frequently several when the source is a technical glossary.) After extracting these sentences, the usual KMS mechanisms are employed to analyze the questions and extract the answers, with results displayed according to the score for each sentence.

In initial testing of the KMS Google search, using the TREC 2002 and TREC 2003 factoid questions, the number of exact answers, the number of sentences (or passages), and the mean reciprocal ranks were all equivalent or better than CL Research's official results. For example, for TREC 2002 questions, exact answers were obtained for 60 of 500 questions. These results were achieved despite many shortcomings such as cases where no search string was formulated, sentence extraction from the Google results was not made properly, and question answering routines were still in their initial stages of development.

The use of Lucene in KMS is in the preliminary stages of development. Lucene comes into play when KMS is unable to answer a question with the repository, i.e., no answers are generated. In such cases, the search terms are reformulated, based in part on the Google search query, as the search specification for a Lucene query. The query is submitted to Lucene and document numbers are obtained as the answers. These document numbers are compared with those in the repository and the top 5 that are not in the repository are used to create an auxiliary file from the AQUAINT CDs. This file is then parsed and processed into an XML representation and the typical question-answering routines are invoked in an attempt to answer the question from these new documents. This procedure has not yet been tested on TREC 2002 and TREC 2003 questions, where more than 25 percent of the questions do not have answers in the top 50 documents provided by NIST. In TREC 2004, the procedures using Lucene were invoked in only 9 of the 230 factoid questions and succeeded in providing answers to only 2 of these questions.

3.3 Evaluation Mechanisms

The development and assessment of strategies for question answering and novelty detection using XML representations and lexical resources is quite complex. The process essentially requires a detailed examination of the XML representation of individual sentences to identify discourse structure, syntactic, and semantic characteristics that indicate how an answer might be

extracted or the relevance and novelty of the sentence can be judged. As a result, scoring programs have been constructed to facilitate the development and assessment of strategies. These programs not only provide a mechanism for gauging progress, but since they involve accessing the XML representations of the documents, their continued development also leads to the identification of new procedures that can be incorporated in performing the tasks themselves.

The QA scoring program is centered on the answer patterns (as Perl-compatible regular expressions) developed from the NIST judgment sets. Generally, these are easily developed for factoid and list questions, where exact answers are required, although the NIST-supplied list answers need to be examined for conformance to the judgment set. For definition or "other" questions, the nuggets identified as vital or "okay" seldom reflect the answers judged correct and it is more difficult to develop Perl regular expressions for them. In scoring answers, the program determines whether the answer satisfies the Perl regular expression.

In addition to the answer patterns, this program takes the list of questions and the files of answers as input. As indicated above, the answer files are structured in XML format. For ease of development, questions are subdivided by type, generally corresponding to the automatic identification of question type used in KMS.

Scoring an answer set is based on the NIST question type (factoid, list, or definition). For factoids, the rank and number of answers is provided for both exact and sentence answers. The summary score shows how many exact answers have been obtained in the first position and also computes the mean reciprocal rank of the answers. For list questions, the scoring program shows the number of correct answers, the number of answers returned by KMS, the number of these answers that are correct, the instance precision, the instance recall, and the F-score. For definition or "other" questions, the scoring program shows the number of correct nuggets, the number of answers returned, the number of answers that are correct, the nugget precision, the nugget recall, and the F-score. Since answers to definition questions include both exact answers and the sentences in which they are contained, the scoring program computes these scores for either type. For list and definition questions, the scoring follows the NIST specification and the average F-score over all questions is determined.

The scoring program is also designed to enable more detailed examination of the answers for each question. Clicking on an individual question brings up a display that provides all the details for each question: the question number, the question, the XPath expression that was used in answering the question, the Google search string that was generated, the list of correct answers, and each answer that was returned by KMS. Each answer shows the score, the document and sentence number, the answer rank, the exact answer, the sentence answer, and whether the answer was judged correct against the Perl patterns. Another button enables the document collection for a question to be examined in full for the presence of the Perl pattern(s). Each sentence of the collection is examined in turn and all sentences containing each pattern are identified with their document and sentence numbers.

The information provided in the scoring program provides the basis for more detailed analysis and more efficient improvement in the KMS routines. The search strings and XPath expressions can be used in the XML Analyzer. The document and sentence numbers can be used to focus on the XML representation for the individual sentence. In addition to aiding development, the scoring program provides feedback to the evolution of KMS. For example, in displaying KMS results, functionality has been added to enable a user to examine the context within which an answer occurs and to look at the detailed XML representation.

As a result of working with the QA scoring program, several issues have emerged with regard to the TREC scoring metrics. By focusing only on the top answers, the overall precision and recall of a system is not assessed. The NIST judgment sets provide (at least an initial) full identification of all the instances in the AQUAINT collection of where answers occur. A more complete assessment of a system's performance would use these as the basis for computing an overall F-score. Such a metric would identify how many instances of correct answers are found; such a metric is important to provide an indication of the confidence a user might place in the answers, as well as allowing the system to estimate how many answers support one another.

The novelty scoring program has been constructed on similar principles. The input to this program consists of the novelty topic file, the answer file identifying documents and sentences judged relevant or novel, and the NIST qrels files containing the assessors' judgments of which sentences are relevant and novel. The scoring corresponds exactly to the Perl script NIST provides for evaluating a run. For each topic, the scoring program shows the number of relevant or novel sentences as judged by the NIST assessors, the number of sentences returned by the system, the number of matches, the precision, the recall, and the F-score. The overall precision, recall, and F-score are also computed.

Selecting an individual topic uses the NIST topic file to show the topic type, title, description, and narrative. By checking or unchecking options, the scoring program then shows "mistakes" for either the relevant or new assessments. The mistakes include either the sentences that have not been recalled (i.e., recall) or the sentences that have been erroneously included (i.e., precision). These sentences are obtained from the XML representation of the document set corresponding to a topic, and show the document and sentence numbers that can then be used in the KMS development environment. This scoring program will be extended to provide more details on why individual sentences were assessed incorrectly.

4 TREC 2004 Question-Answering

CL Research submitted one run for the TREC 2004 question-answering track. All answers were generated in KMS. The question set provided by NIST was first converted into a form for parsing and processing into an XML representation. The top 50 documents for the 65 targets were also processing into an XML representation. The questions are displayed in KMS, each with a checkbox used to select which questions are to be answered. All questions were selected and answered automatically, generating an answer file as described above. A Perl script was then used to create an answer file in the format required for a QA track submission.

Conversion of the NIST question set involved minor formatting changes (using a different tag set) and a more considerable anaphora replacement algorithm. For TREC 2004, this was performed in a Perl script, rather than attempting to use the potential capabilities of KMS for capturing anaphoric references. The Perl script identified all referring expressions in the questions, including anaphors such as *her*, *it*, and *their* and definite noun phrases such as "the organization." The script kept track of the type of anaphors so that the "other" question could be converted to either "Who is" or "What is". The revised question set was added to KMS as a question list from which question selection could be made.

As described in Litkowski (2004), the question answering process in KMS begins with an analysis of the question. The analysis first assesses the question type and determines whether the surface form indicates that the type can be changed to one that can best be handled by a different set of routines than might at first be indicated. For example, "what cities" would be changed to a **where** question and "what year" would be changed to a **when** question. This assessment is quite principled and is based on an analysis of the head noun in the question element. Specifically, the WordNet file numbers are used to characterize the head noun into one of 25 types. Question analysis also involves identification of a focal noun, a key noun, an expected hypernym of an answer, a key verb, and the types of the other terms used in the question (such as modifiers and quoted material). For some questions, this analysis can be complex. For example, "what kind of community is a kibbutz?" indicates that an answer should have a hypernym of "community".

After the question analysis, a basic XPath expression is created for the question. This XPath expression specifies what a sentence should contain. It indicates that only nodes that are of type **segment** and having a **sent** attribute are to be used, i.e., that sentences are treated as the primary unit of analysis. This specification also identifies words that must be present in the sentence. These words are taken from the focal and key nouns, the hypernym, and other key elements. They are put together as a set of alternatives and thus act as a very minimal filter on acceptable sentences. This specification essentially acts as passage retrieval of sentences from the entire set of documents.

At this point in the question-answering process, further analysis becomes question-type specific. The basic XPath expression is customized to look for (primarily) specific types of discourse entities (nodes of type **discent**) in the set of sentences. In some cases, the specification asks for verbs (nodes of type **verb**) or prepositions (nodes of type **semrel**). In all cases, some further restrictions are placed on the content or attributes of the node.

For questions containing ordinal or degree modifiers, the discourse entity might be required to have an ordinal or degree modifier (**ord** or **deg** attributes, regardless of the value of these attributes). The specification might require a specific syntactic role (**synrole** equal to *subj*). Frequently, the specific value of the **discent** node or its **antecedent** may be required; an exact match may be specified or the value may require only that a specific word be part of the value. In some cases, the specification may require a **discent** that has a **synrole** of *subj* preceding a **verb** having a value or **base** equal to the key verb. In general, however, this last level of specification is proving to be too restrictive, requiring that an answer adhere to a detailed set of requirements. Instead, less restrictive specifications are made, relegating the identification of the more specific requirements to the analysis performed in evaluating potential answers.

After the basic XPath expression has been refined based on the question type, the query is submitted to

the document set and a set of potential answers is obtained. These answers are sent to question-type specific routines for detailed analysis (described in Section 6). Each potential answer is scored and a decision is ultimately made to keep or discard it. Each answer to be kept is evaluated finally for whether it duplicates an existing answer. Finally, the various characteristics of the answer are assembled into an array of answers, sorted by score, and including the document and sentence numbers and the exact and sentence answer. After all potential answers are evaluated, the final answer set is constructed, displayed in KMS, and saved to the answer file.

Table 1 shows the summary score for the CL Research QA run, along with the median score for all participating teams, broken down by major question type. The table also shows the current score based on changes made to KMS after the official results were provided in late September.

Table 1. Summary Score

	Factoid (0.170)	"Other" (0.184)	List (0.094)	Overall
Official	0.161	0.239	0.064	0.156
Current	0.191	0.239	0.106	0.182

As can be seen, CL Research scored somewhat higher than the median for the "other" component and somewhat less for the factoid and list components, which have since been improved slightly to above the medians. These results are considerably improved over CL Research's performance in TREC 2003 with an overall score of 0.075. This improvement was achieved with only about 80 hours of effort in modifying the core QA routines. The improvement to the current level was made with an additional 20 hours of effort. As will be described in Section 6, the improvements have come from the development of new mechanisms for manipulating and analyzing the XML representation. These new mechanisms have generally been developed in routines for analyzing specific question types, and have not yet been applied, where possible, to other question types. As indicated below, considerable opportunity still remains for exploiting the XML approach to question answering.

Tables 2 and 3 show the official and improved scores for the factoid questions by question type. The improvements in Table 3 indicate the procedure followed in making changes to KMS, viz., working on question types that have the lowest score until they are no longer the lowest.

Table 2. Factoid Questions (Official)

Question Type	Number	Correct	Accuracy	MRR
How	3	0	0.000	0.000
HowMany	20	2	0.100	0.150
HowMeas	8	0	0.000	0.000
HowMuch	2	0	0.000	0.000
WhatIs	39	4	0.103	0.109
WhatNP	35	7	0.200	0.219
WhatVP	6	1	0.167	0.167
When	57	13	0.228	0.270
Where	28	4	0.143	0.168
Who	4	0	0.000	0.000
WhoIs	15	0	0.000	0.013
WhoVP	11	3	0.273	0.273
Why	2	0	0.000	0.000
Total	230	34	0.148	0.171

Table 3. Factoid Questions (Improved)

Question Type	Number	Correct	Accuracy	MRR
How	3	1	0.333	0.333
HowMany	20	2	0.100	0.150
HowMeas	8	1	0.125	0.125
HowMuch	2	1	0.500	0.500
WhatIs	39	4	0.103	0.109
WhatNP	35	7	0.200	0.219
WhatVP	6	1	0.167	0.167
When	57	13	0.228	0.270
Where	28	4	0.143	0.168
Who	4	1	0.250	0.250
WhoIs	15	2	0.133	0.161
WhoVP	11	3	0.273	0.273
Why	2	1	0.500	0.500
Total	230	41	0.178	0.202

5 TREC 2004 Novelty

Our participation in the Novelty track had two main components, one implementing special procedures to handle the various tasks, and the other implementing the procedures for actually performing the tasks. Our procedures were largely unchanged from those used in TREC 2003, except for the use of verbs in addition to discourse entities in identifying the relevance of sentences and the use of a test for communication verbs when "opinion" was part of the narrative specification.

In order to allow KMS to process the Novelty texts and build XML representations for them, we first added wrappers to the NIST provided texts to make them XML compliant. We then processed the files with KMS. Similarly, we added wrappers to the topic

file to make it XML compliant, and then processed it with KMS, processing as text the title, description, and narrative fields. For tasks 2, 3, and 4, we converted the NIST-provided qrels files of relevant and new sentences into XPath expressions that would select the corresponding sentences from the XML versions of the NIST texts.

5.1 Determination of Relevance

The basic relevance judgment for a sentence was determined by examining its discourse entities and antecedents if a discourse entity had an antecedent (anaphors, coreferents, or definite noun phrases). Each word, except words on a stop list, was compared to the list of words obtained from the topic. Verbs in the topic specification were also used (in their base form). When the topic specification indicated that the topic type was "Opinion" or when an Event topic type included a word that asked for opinions, a test was made on verbs in the sentences to determine if they were "communication" verbs, as identified by WordNet. The basic criterion for selection of a sentence as relevant was whether a sentence had two or more hits.

For task 1, the basic criterion was applied using all information from the title, the description, and the narrative. When a sentence in the topic description contained the word "irrelevant" or "not relevant", words in the sentence were removed from the list for measuring hits. One run used two or more hits and the other required three or more hits.with one run using only the title, one run using the title and the description, and a third run using the title, the description, and the narrative (with words in sentences containing the word "irrelevant" or the phrase "not relevant" excluded from the list). A fourth run used all the information as in the third run but required three or more hits. (In TREC 2003, we had used varying amounts of information, but found that using all information provided the best overall results, so we eliminated those distinctions in TREC 2004.)

For task 3, where relevant sentences were provided for the first five documents, a frequency list was developed for words in discourse entities or antecedents. The total number of words in this list was also determined. For each sentence, a frequency score was computed as the sum of the frequency count for all word in the sentence on the frequency list divided by the total number of words on this list. Then, the basic criterion was modified. Sentences with two or more hits were still selected as relevant, but also sentences with a frequency score greater than a specified level were also selected as relevant. For task 3, four runs were submitted based on different frequency scores, 0.01, 0.02, 0.03, 0.04, and 0.05. The lower scores allow more sentences, thus increasing recall.

5.2 Determination of Novelty

Once a set of sentences had been selected as relevant, they were considered in order to determine novelty. Sentences that were exact duplicates were first eliminated. Next, each discourse entity was evaluated for novelty against an accumulating list of all unique discourse entities encountered thus far. Again, antecedents were used in preference to the actual discourse entity for anaphors, coreferents, and definite noun phrases that had a non-empty antecedent attribute. If a discourse entity from the current sentence being evaluated was not found, it was added to the growing history list and the sentence was accepted as novel. In evaluating a discourse entity, if all of its words were present in a discourse entity already on the history list, the candidate was viewed as old information. If all discourse entities in a sentence were present on the history list, the sentence being evaluated was characterized as overlapping with prior information and thus eliminated from the set of novel sentences.

For task 1, the novel sentences were selected from the relevant sentences that had been determined as described above. For task 2, where all relevant sentences were given, only these were considered in determining novelty; this task thus provides a reasonable characterization of the novelty component by itself. For task 3, the novel sentences were selected from among a wider set than those used in task 1, since these were conditioned by the greater recall of relevance sentences; sentences from the qrels provided for the first five sentences were removed from consideration for novelty. For task 4, we submitted the same results as for task 2, since our novelty routines at this time contained no processing that would take into account sentences that had been identified as being novel; the submission only removed sentences from the first five documents.

5.3 Novelty Track Results

For the Novelty track, we submitted two runs for task 1, one run for task 2, for runs for task 3, and one run for task 4; our submissions for tasks 2 and 4 were identical.

Our results for Task 1 are shown in Table 4 for relevance and Table 5 for novelty.

Table 4. Task 1 (Relevance)			
Run	**Precision**	**Recall**	**F-Score**
clr04n1h2	0.33	0.68	0.394
clr04n1h3	0.37	0.50	0.367

Last year, the comparable run to clr04n1h2 received a precision of 0.69, a recall of 0.45, and an F-score of 0.483. The comparable run to clr04n1h3 received a precision of 0.72, a recall of 0.24, and an F-score of 0.316.

Table 5. Task 1 (Novelty)			
Run	**Precision**	**Recall**	**F-Score**
clr04n1h2	0.15	0.62	0.216
clr04n1h3	0.17	0.46	0.213

Last year, the comparable run to clr04n1h2 received a precision of 0.50, a recall of 0.40, and an F-score of 0.410. The comparable run to clr04n1h3 received a precision of 0.51, a recall of 0.24, and an F-score of 0.278.

For task 1, our best run received an F-score of 0.394 for relevant sentences and 0.216 for new sentences; this was higher than the median of 0.379 for relevant sentences and 0.193 for novel sentences. The second run had a lower score than the median for relevance, but was still higher than the median for novelty. These results clearly demonstrate that the more restrictive run requiring three or more hits had a significantly lower recall.

The results for task 1 changed quite significantly for TREC 2004 compared with TREC 2003. In tuning our system with TREC 2003 topics, we had increased our performance to a level in the top four for both relevance and novelty. This was achieved primarily through a considerable increase in recall, without too much of a decrease in precision. While the change in recall held firm in TREC 2004 (over 0.20 improvement), precision descreased dramatically. Although we did not have the detailed precision and recall for other participants in TREC 2004, the overall F-scores for all participants declined significantly. It is likely that this dramatic decrease stems from the sharply reduced number of relevant and novel sentences identified in TREC 2004, from 15557 to 8343 for relevant sentences and 10226 to 3454 novel sentences. It would seem that the NIST assessors became more familiar with the novelty task and were more demanding in accepting sentences as relevant and novel. This suggests that systems will correspondingly need to achieve comparable levels of specificity.

Our results for Task 2 are shown in Table 6. Run clr04n2 is our official submission, and run clr04n2a is a revised submission.

Table 6. Task 2 (Novelty)			
Run	**Precision**	**Recall**	**F-Score**
clr04n2	0.28	0.93	0.410
clr04n2a	0.45	0.86	0.574

Last year, the comparable run to clr04n2 received a precision of 0.71, a recall of 0.91, and an F-score of 0.788. In examining our results for run clr04n2, we observed that the number of sentences returned (13047) considerably exceeded the number of relevant sentences provided in the qrels (8343). We found that this arose from code that had not been removed and that had expanded the reduced set identified by the qrels back to the original full set of sentences. After removing this code and scoring the revised run, our results were changed significantly, up to the level of the average F-score for all participating teams.

Notwithstanding the change from the revised run, our results are still significantly lower than those achieved in TREC 2003. Again, the decline in performance arises primarily from the lower precision.

Our results in Task 3 are shown in Table 7 (relevance) and Table 8 (novelty).

Table 7. Task 3 (Relevance)			
Run	**Precision**	**Recall**	**F-Score**
clr04n3h1f1	0.30	0.85	0.405
clr04n3h1f2	0.30	0.85	0.405
clr04n3h2f1	0.31	0.81	0.404
clr04n3h2f2	0.32	0.78	0.410

Last year, the best run for Task 3 received a precision of 0.48, a recall of 0.77, and an F-score of 0.541. Again, our results show that our efforts to improve recall were offset by significant declines in precision.

Table 8. Task 3 (Novelty)			
Run	**Precision**	**Recall**	**F-Score**
clr04n3h1f1	0.13	0.77	0.203
clr04n3h1f2	0.13	0.77	0.203
clr04n3h2f1	0.13	0.73	0.206
clr04n3h2f2	0.14	0.71	0.209

Last year, the best run for Task 3 comparable run to clr04n2 received a precision of 0.33, a recall of 0.73, and an F-score of 0.408. It is possible that the low scores on novelty were affected by the errant code, but we have not yet rerun this submission to examine this possibility.

Our results for the relevance portion of Task 3 were all higher than the median of all participants (0.376). For novelty, our results were slightly lower than the median for all runs.

Our resutls for Task 4 are shown in Table 9.

Table 9. Task 4 (Novelty)			
Run	**Precision**	**Recall**	**F-Score**
clr04n4	0.22	0.92	0.332

Last year, our results for Task 4 received a precision of 0.53, a recall of 0.91, and an F-score of 0.655. Again, we are certain that the errant code significantly affected our precision, since the number of sentences identified as novel (13011) is much larger than the number identified as relevant.

6 Functions Supporting Evaluation

As indicated above, evaluation of potential answers for both question answering and novelty is performed in individual functions that are specific to the type of question or the assessment of relevance or novelty (which is subsumed under KMS' summarization routines). Within each of these broad functions, various low-level functions are used to add or deduct points from scores that eventually determine whether an answer is retained or discarded. These functions can be grouped into (1) XML functions, (2) linguistic functions, (3) dictionary access functions (DIMAP and WordNet), (4) summarization functions, and (5) miscellaneous functions.

6.1 XML Functions

Basic XML functions are used to maneuver around the XML trees representing the documents and to examine characteristics of nodes in the tree. The development environment provides several of these functions, the most important of which is used to select nodesets using XPath expressions. The XPath expressions may be applied from any given node in the XML trees, perhaps starting from the root node, and frequently used in examining the child nodes of other specific nodes, such as sentence nodes. XPath expressions are used for selecting sentence nodes (passage retrieval) and selecting sentence elements (clauses, discourse entities, verbs, prepositions), frequently specifying attribute names and values. Usually, routines are implemented for iterating over the nodesets.

While XPath expressions may be used for a variety of other tests, simple functions are used instead to accomplish these objectives. Two basic functions ask whether a node is of a given type and whether a given node has a particular attribute and value. Two other basic functions obtain the value of a node (i.e., its text) or the value of an attribute. Another commonly used function is used to identify the sentence node from one of its children, such as a discourse entity that is being evaluated as a potential answer. Retrieving the sentence node facilitates examination of the context surrounding the node being evaluated. Another common function retrieves the document number and sentence containing a given node. While this function is used essentially in providing an answer, it is also used to compile lists of document numbers and sentence numbers (such as in summarization).

6.2 Linguistic Functions

Linguistic functions are combinations of basic XML functions used to test or retrieve other elements in a sentence. They usually begin with a particular node in the XML representation and examine characteristics of the particular node or examine its child nodes (if it is a composite node) or its siblings (i.e., words or nodes before or after the given node).

One group of functions perform basic syntactic tests. One function tests whether a verb is active or passive. Some simple linguistic functions retrieve the subject or object of a verb or look for a prepositional phrase of a given type in relation to a verb or noun phrase. For example, for a **WhoVP** question ("Who wrote X?"), if the verb is passive, the author might be sought in a prepositional phrase beginning with "by".

Functions that test for appositives are commonly used. These functions may look for appositives following a noun phrase or may look to see if the current node is in an appositive and modifies a noun phrase that should be the answer. Appositives may be identified in many ways, including *that*, *who*, *or*-appositives, comma-delimited phrases, and parentheticals, either tagged as such or using string pattern matches. As indicated above, the potential answers submitted to the evaluation routines for particular question types are frequently the focal point of the evaluation, and they may be replaced by a new node, such as an appostive.

Some linguistic functions look for the specific strings in the surrounding text, such as the phrase "such as" or a comma followed by an "or". A function is used for copular verbs to retrieve the noun phrase following the verb. Another verb function obtains the the verb and following noun phrase and modifiers of the noun phrase (such as "won five Olympic medals").

Several functions are used to "extend" an answer, e.g., obtaining two or more prepositional phrases unbroken by discourse markers such as commas. The appositive and copular functions are particularly useful in providing definitional information.

6.3 DIMAP and WordNet Functions

Since lexical resources are used heavily in KMS, many functions have been developed to access dictionary information. Most of these lexical resources are integrated using DIMAP data structures and are accessible via rapid lookup methods. Several dictionaries are accessible and used in tandem: a machine-readable dictionary, WordNet, the Unified Medical Language System Specialist Lexicon (which contains considerable syntactic characterizations of general vocabulary as well), and a specially constructed preposition dictionary used for characterizing semantic relations.

The machine-readable dictionary is an auxiliary dictionary that has been constructed with definition features and various hypernymic and other relational information characterizing a sense. These entries are examined for particular features. Several functions are designed to examine the dictionary entries and accumulate properties that can then be tested. These include measuring the overlap between definitions and the context of a word in documents (as frequently used in word-sense disambiguation), testing for the presence of particular words (such as "unit" to indicate that a preceding number is a unit of measure), and obtaining a person's first name.

WordNet has also been structured as a DIMAP dictionary and several functions are used to examine its contents. These include obtaining a list of a word's synonyms and testing for word equivalence. Other functions test for a semantic class, e.g., using WordNet tops or file numbers). Several functions examine hypernymic relations, in a node, other entities in a sentence, or in appositives. Other functions test other relations, particularly derivative forms (e.g., *die, dying, death, died*) and pertainyms (e.g., *old => age*), identifying and testing measure type, and unbundling hyphenated forms (e.g., *4-day vigil*).

6.4 Summarization Functions

Summarization functions gather information about the dominant theme of documents, identify key words, construct frequency counts, and assess the relative importance of a sentence in characterizing a document.

Frequency counts of words are generated. This involves replacing anaphors, coreferring expressions, and definite noun phrases with their antecedents. Phrases are split into words and each word is examined against a stop list. Similarly, frequency counts of whole discourse entities may be generated, e.g., so that the first names of people may be ignored in frequency counts. Key word lists are generated from these frequency counts to combine words into phrases.

Sentences can be scored against the frequency counts, e.g., to identify top sentences containing dominant themes or to identify sentences that are being measured against some topical focus. Lists of unique discourse entities are generated for use in assessing duplication of discourse entities with previous mentions and to assess the overall duplication of a sentence with those in a growing list. These functions are particularly relevant to novelty assessments.

6.5 Miscellaneous Functions

Miscellaneous functions test for the presence of key words, look for string pattern matches, examine person names in detail. In examining key words, functions ask whether a key noun or key verb is present in a sentence, whether all words in a compound focal noun are present, whether all qualifiers of the focal noun are present in the sentence or in a clause, and whether the proper ordinal or degree modifiers are present (e.g., *first* or *highest*). String matches may look for specific words following a hypernym (*called* or *such as*), look for titles works, testing for an acronym or a brand name, or look for complete date strings. Another function will examine whether the document is referring to the correct person (i.e., is a person's first name present, e.g., is the correct "Mosley" or "Patterson").

7 Summary

Our results on the QA and Novelty tracks indicate that our approach of using massively XML-tagged documents is worth continuing. There are many opportunities still to be investigated.

References

Litkowski, K. C. (2004). Use of Metadata for Question Answering and Novelty Tasks. In E. M. Voorhees & L. P. Buckland (eds.), *The Eleventh Text Retrieval Conference (TREC 2003)*. NIST Special Publication 500-255. Gaithersburg, MD., 161-170.