Use of Metadata for Question Answering and Novelty Tasks

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

Abstract

CL Research's question-answering system for TREC 2003 was modified away from reliance on database technology to the core underlying technology of using massive XML-tagging for processing both questions and documents. This core technology was then extended to participate in the novelty task. This technology provides many opportuinities for experimenting with various approaches to question answering and novelty determination.

For the QA track, we submitted one run and our overall main task score was 0.075, with scores of 0.070 for factoid questions, 0.000 for list questions, and 0.160 for definition questions. For the passage task, we submitted two runs, our better score was 0.119 for the factoid questions. These scores were all considerably below the medians for these tasks. We have implemented further routines since our official submission, improving our scores to 0.18 and 0.23 for the exact answer and passages tasks, respectively. For the Novelty track, we submitted four runs for task 1, one run for task 2, five runs for task 3, and one run for task 4; our submissions for tasks 2 and 4 were identical. For task 1, our best run received an F-score of 0.483 for relevant sentences and 0.410 for new sentences. For task 2, our F-score was 0.788 for new sentences. For task 3, our best F-score was 0.558 for relevant sentences and 0.419 for new sentences. For task 4, our F-score was 0.655 for new sentences. On average, our F-scores were somewhat above the medians on all tasks. We describe our system and examine our results from the perspective of exploiting the metadata in the XML tags.

1 Introduction

In TREC 2002, CL Research examined the potential of using XML-tagged documents for question answering (Litkowski, 2003a). In particular, we saw how the use of hand-developed XPath expressions could obtain extremely good results when compared with the best sytems. However, as noted, the challenge was the automatic creation of XPath expressions. While this was the focus of our participation in the TREC 2003 questions answering (QA) track, this effort was performed in the context of creating a new Knowledge Management System (KMS) from the GUI interface we used in last year's explorations. KMS was first used in CL Research's participation in text summarization in the 2003 Document Understanding Conference (Litkowski, 2003b). KMS is continuing to evolve as we explore the possibilities of using XML-tagging to perform various natural language processing (NLP) tasks.

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 and we can readily design our entire representation on this tree structure. We can create XML representations for 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 NLP tasks. This paper describes some of our findings emerging from CL Research's participation in TREC.

A key part of the XML design philosophy is the ability to transform an XML file into usable output for display or other purposes (e.g., populating a database). This is accomplished via XML stylesheet language transformations (XSLT). XSLT is based on the creation of XPath expressions, which specify the path from the top of the XML tree to some intermediate or leaf node. XPath expressions are useful in QA, since they provide a simple mechanism for homing in on answers. Just as important, we have found it essential to be able to move about in the XML tree from a particular node, based on relations of one node to others (e.g., moving to a clause containing a noun phrase).

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 anticipated next steps for improving the question-answering capability and for using XML-tagged documents in other applications such as information extraction, text summarization, novelty studies, and investigation of linguistic phenomena.

2 Problem Description

The TREC 2003 QA and Novelty tasks used the AQUAINT Corpus of English News Text on two CD-ROMs, (about one million documents), containing documents from *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 500 unseen questions to be answered in a "main" task from the corpus. Participants were given the option of using their own search engine or of using the results of a "generic" search engine. CL Research chose the latter, relying on the top 50 documents retrieved by the search engine. These top documents were provided simultaneously with the questions. The 500 questions included a type: factoid (417, requiring a short, exact answer), list (37, requiring a list of answers), and definition (50, requiring a set of core elements and/or acceptable peripheral information concerning the term to be defined). An additional task, using only the 413 factoid questions, was to submit "passages" containing the answer.

Participants in the main task were required to answer the 413 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 30 questions for which no answer exists in the collection. For these questions, NIST evaluators next 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 definitions questions ("Who is Vlad the Impaler" or "What are fractals"), 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 main task and two runs for the passages task.

For the Novelty track, participants were provided with descriptions of 50 topics (labeled as "event" or "opinion") and a set of 25 documents relevant to each topic. 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 four runs for task 1 (using different amounts of information from the topic description and with a different threshhold for judging relevance), one run for task 2, five runs for task 3 (using different threshholds for taking into account information in the identified relevant sentences), and one run for task 4.

3 The Knowledge Management System

The CL Research KMS consists of two major components, a tasking component and a text processing component. The tasking component sets up the tasks to be performed based on the NLP task (such as question answering, text summarization, or novelty detection). The text processing component processes documents into an XML representation. For question answering, the tasking component will parse and process a question into an XML representation, invoke the text processing component to handle any documents that might contain the answers, and answers the question. For novelty detection, the tasking component will parse and process the title, the description, and the narrative into an XML representation, invoke the text processing component for relevant documents, and then process the sentences in the documents against the topic description to identify relevant and novel sentences.

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 remainder of this section details these elements, focusing on document processing; these same elements are used in parsing and processing questions and topic descriptions into XML representations.

3.1 Sentence Splitting

Sentence splitting proceeds as described in previous years (Litkowski, 2002a; Litkowski, 2001). For TREC 2003, we were able to process the full set of 50 documents for each question quite rapidly, unlike previous years where we were more limited in speed by the reliance on database technology, where processing speed slowed exponentially as the database grew. Instead, processing grew linearly with the size of the document collection. Overall, we processed 25,000 documents from which 724,165 sentences were identified and presented to the parser. Thus, we processed an average of 29.0 sentences per document (compared to 25.7 in TREC 2002, 22.8 in TREC-10, 28.9 in TREC-9 and 31.9 in TREC-8). Overall, we had an average of 1448 sentences to consider for each question. For the novelty task, only 1250 documents were processed, at 25 documents for each of 50 topics.

Our sentence splitter has remained largely the same, but some improvements have been made, primarily in the recognition of abbreviations and initials associated with a name. In previous years, this had resulted in improper splitting. In TREC 2003, we observed that the documents for the novelty task, which were provided already split into "sentences", had many sentences improperly split. We rejoined these sentences prior to processing these texts.

3.2 Parser

We continued our use of the Proximity parser, described in more detail in our previous papers (Litkowski, 2002a; Litkowski, 2001). As described there, the parser output consists of bracketed parse trees, with leaf nodes describing the part of speech and lexical entry for each sentence word. Annotations, such as number and tense information, may be included at any node. Usable output was generated by the parser for 99.9 percent of the sentences that were processed.

3.3 Sentence and Discourse Analysis

The sentence parsing in KMS is part of a broader system designed to provide a discourse analysis of an entire text. After each sentence is identified and parsed, its parse tree is traversed in a depth-first recursive function. During this traversal, each non-terminal and terminal node is analyzed, making use of parse tree annotations and other functions and lexical resources that provide "semantic" interpretations of syntactic properties and lexical information.

At the top node in the tree, just prior to iteration over its immediate children, the principal discourse analysis steps are performed. Each sentence is treated as an "event" and added to a list of events that constitute the discourse. We first update data structures used for anaphora resolution. Next, we perform a quick traversal of the parse tree to identify discourse markers (e.g., subordinating conjunctions, relative clause boundaries, and discourse punctuation) and break the sentence down into elementary discourse units. We also identify and maintain a list of the sentence's verbs at this stage, to serve as the bearers of the event for each discourse unit.

After the initial discourse analysis, the focal points in the traversal of the parse tree are the noun phrases. When a noun phrase is encountered, its constituents are examined and its relationship to other sentence constituents are determined. Each noun phrase is added to a list of discourse entities for the entire text, that is, a "history" list. As each noun phrase is encountered, it is compared to discourse entities already on the history list. This comparison first looks for a prior mention, in whole or in part, to determine whether the new entity is a coreferent of a previous entity (particularly valuable for named entities). If the new entity is an anaphor, an anaphoric resolution module is invoked to establish the antecedent. A similar effort is made to find antecedents for definite noun phrases. The noun phrase's constituents are examined for numbers, adjective sequences, possessives (which are also subjected to the anaphoric resolution module), genitive determiners (which are made into separate discourse entities), leading noun sequences, ordinals, and time phrases. Finally, an attempt is made to assign a semantic type to the head noun of the phrase using WordNet or an integrated machine-readable dictionary or thesaurus.

If a noun phrase is part of a prepositional phrase, a special preposition dictionary is invoked in an attempt to disambiguate the preposition and identify its semantic type. This module identifies the attachment point of the preposition and uses information about the syntactic and semantic characteristics of the attachment point and the prepositional object for this disambiguation. The preposition "definitions" in this dictionary are actually function calls that check for such things as literals and hypernymy relations in WordNet. A list of all prepositions encountered in the text is maintained as the text is processed. (See Litkowski (2002b) for further details.)

Predicative adjective phrases, relative clauses, subordinate clauses, and appositives are also flagged as the parse tree is traversed. The attachment points and spans of relative clauses and appositives are noted.

As indicated above, the text analysis module develops four lists at the same time as the semantic relation triples: (1) events (the discourse segments), (2) entities (the discourse entities), (3) verbs, and (3) semantic relations (the prepositions). Each document consists of one or more tagged segments, which may include nested segments. Each discourse entity, verb, and preposition in each segment is then tagged. A segment may also contain untagged text, such as adverbs and punctuation. Each item on each list has an identification number (used in many of the functions of the text analysis module). As indicated above, the discourse analysis assigns attributes to each segment (and subsegment), discourse entity, verb, and preposition.

For segments, the attributes include the sentence number (if the segment is the full sentence), a list of subsegments (if any), the parent segment (if a subsegment), the text of the segment, the discourse markers in the sentence, and a type (e.g., a "definition" sentence or "appositive"). For discourse entities, the attributes include its segment, position in the sentence, syntactic role (subject, object, prepositional object), syntactic characteristics (number, gender, and person), type (anaphor, definite or indefinite), semantic type (such as person, location, or organization), coreferent (if it appeared earlier in the document), whether the noun phrase includes a number or an ordinal, antecedent (for definite noun phrases and anaphors), and a tag indicating the type of question it may answer (such as who, when, where, how many, and how much). For verbs, the attributes include its segment, position in the sentence, the subcategorization type (from a set of 30 types), its arguments, its base form (when inflected), and its grammatical role (when used as an adjective). For prepositions, the attributes include its segment, the type of semantic relation it instantiates (based on disambiguation of the preposition) and its arguments (both the prepositional object and the attachment point of the prepositional phrase).

After all sentences in a document have been processed, the four lists are used to create an XML-tagged version of the document. The XML tagging is performed for each segment within the XML element **segment**, with the attributes listed in the tag opening. The tag content is initialized to the segment text and we proceed to mark up this text according to the text contained within each subsegment, discourse entity (**discent**), verb (**verb**), and preposition (**semrel**) in the segment. As these XML elements are generated, their attributes are added to the tag opening.

The resultant XML-tagged text for individual documents were combined into one overall file of documents, each with a tag for the document number. For TREC, there was an XML file for each of the 500 questions in the QA track and for each of the 50 topics in the novelty track. As of the submission date for the QA track, XML tagging results in nearly a sevenfold expansion of the documents. The 94 MBs of the top 50 documents generated 632 MBs of XML-tagged text.

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.

4 Question-Answering Using XML-Tagged Documents

As described for TREC 2002 (Litkowski, 2003a), question-answering against the XML files essentially involves describing a path (XPath) from the top of the document tree (in this case, a file of 50 TREC documents) to a discourse entity (in our case, to a **discent** node) which is returned as the answer. In TREC 2002, we demonstrated that the desired **discent** nodes were present for almost all the questions, with the residual few cases not present because of parsing or processing bugs. Further, we showed that an XPath expression could be developed by hand to extract these answers to a degree better than the best performing system. (This is slightly misleading since the measure used in TREC 2002 was a confidence weighted score that was higher than simply the percentage of correct answers, as used in TREC 2003.) We concluded our discussion last year by indicating that the challenge was the automatic creation of XPath expressions.

As we proceeded into the development of the functionality for XPath expression creation for TREC 2003, we found it necessary to situate the use of the XPath expressions into a broader set of routines. Generally, the overall architecture for answering questions consists of four components: (1) question analysis, (2) creating of a broad XPath expression that will retrieve appropriate sentences, (3) appending to this basic XPath expression a specific XPath expression request for discourse entities (with routines specific to question type), and (4) retrieving candidate answers using the XPath expression and submitting these answers to a post-processng routine (also question-type specific) to evaluate the answers. We have found that some looping through these components may be necessary, based on results that are obtained. For the most part, this involves revising the specific XPath expression, but some looping may also occur in evaluating answers when a pass through the candidates results in a narrowing and subsequent passes can examine the relative quality of different answers.

The complexity of these processes was unexpected, with the result that our implementation was incomplete at the time of our submission. In particular, we were unable to convert many of the details of our candidate answer evaluations (which are question-type specific) into appropriate processing steps making use of the XPath and XML node processing capabilities. We have since implemented many of these routines. Not surprisingly, the complexity of the routines varies with the question type. Several general principles have emerged. Again, not surprisingly, the principles reflect those developed for question answering in our previous work and in the work of other TREC participants. These prinicples are couched within a more general architecture that implements linguistic and semantic processing, rather than simply implementing pattern recognition routines. Although we have not ahcieved the performance of our hand generation of XPath expressions, we are confident that continued development of the XML mechanisms will serve not only question answering, but also other NLP tasks.

We describe our observations to date in the following areas: (1) the question analysis phase, (2) the creation of the basic XPath expression to retrieve sentences, (3) question-type specific XPath expression creation ,and (4) evaluation of candidate answers.

4.1 Question Analysis

We have grounded our analysis of questions on a rigorous hierarchical analysis of dictionary definitions for question words, specifically for **what**, **which**, **who**, **when**, **where**, **why**, and **how**. The definition for **what** is at the top of the hierarchy, "asking for information specifying something", with **which** being a slight variant, "asking for information specifying one or more people or things from a definite set". The others are defined in terms using the **what** primitive: "what person" (**who**), "at what time" (**when**), "at what location" (**where**), "for what reason" (**why**), and "by what means" (**how**). There are three additional varieties of **how**: **how much** ("what amount or price"), **how many** ("what number"), and **how adj** ("used to ask about the degree or extent of something").

Our question analysis identifies the type of question, either directly according to the question word or by analyzing the question string and the semantics of key discourse entities in the question. Thus, "what year" is converted into a **when** question, "what is the length of np" is converted into a **how adj** question, and "what country" is converted into a **which** question (or equivalently, a **whatNP** question). Sentences that have no question elements (which are given a "qelem" attribute in the XML tagging), such as "List types of chewing gum", are converted into **whatNP** questions. Very simple **what** or **who** questions "Who is Vlad the Impaler?" are converted into **whatIsDef** or **whoIsDef** questons.

4.2 Basic Sentence XPath Expression

Each sentence that is processed in KMS is enclosed in the metadata tag **segment**. Clauses also have this tag, but are distinguished from the full sentence via attributes. A full sentence is assigned an ID number and may have an associated list of subsegments, while a clause has a parent. All questions give rise to an XPath expression that selects sentences and begin with

//segment[@sent].

This will retrieve all sentences in all documents in a file.

We next add further restrictions on what sentences we would like to examine in further detail based on the elements of the XML representation of the question, in particular the discourse entities. At this point, we want to remain fairly generous in keeping sentences, so we look for any sentences containing any of the words in the discourse entities in the question, except for stop words. For example, for question 1401 in TREC 2002, "What is the democratic party symbol?", the basic XPath expression is

//segment[@sent and (contains(.,'democratic') or contains(.,'party') or contains('symbol'))]

Identification of the words to be included in this part of the XPath expression is somewhat involved, but provides a first screen to identify sentences where an answer might be found. When evaluating answers for each question type, we determine whether changing "or" or "and" retrieves any sentences, and allow this most restrictive screen if it returns any sentences.

We are continuing to study alternatives to this basic XPath expression, such as using regular expressions, allowing query expansion using synonyms, and weighting the importance of terms.

4.3 Question-Specific XPath Expressions

As indicated above, each question type has its own specific processing to characterize the discourse entity that is sought in the XPath expression. This generally entails adding to the basic XPath expression a specification for a **discent**, with the following appended:

//discent

In all cases, the attributes of the desired discourse entity are specified via qualifiers. Since we have tagged discourse entities with many attributes during text processing, we can make use of this metadata when adding qualifiers.

For the more "complex" question types (**when**, **where**, **how much**, **how many**, and **how adj**), the specification qualifier for the discourse entity is actually simpler, such as

//discent[@tag='when'], or

//discent[@tag='where'].

For a question like "How many time zones", the qualifier would add the head noun "time zones"

//discent[@tag='howmany' and contains(.,'time zones')]

to obtain a discourse entity that has a numeric quantifier.

For the more "basic" question types, **what**, **whatNP**, and **who**, analysis of the question is necessary to identify appropriate qualifiers. For **who** questions, the discourse entity would be required to have a **semtype** attribute with a value of "person". When a question asks for the "first" or the "biggest", the qualifier would require that the discourse entity have an **ordMod** (ordinal modifier) with the same numerical value (**ord**) as required (e.g., "2" for second) or a **degMod** (degree modifer) of the same level (e.g., "est" for superlative).

Many questions would give rise to qualifiers on the discourse entity string, frequently asking that either the entity itself or its antecedent (if a pronoun) contain the key or focal noun in the question. This is done frequently with **what** and **who** questions, where the qualification will retrieve a discourse entity that is essentially identical with what is being asked for, rather than the discourse entity that is the answer. This is done so that in the next (evaluation) step, we examine the context surrounding such a key noun, looking for appositives, verb subjects, or verb or prepositional objects. (Frequently, In our official submission, for example, in a question like "what band", our answer was the word "band", because we had not fully implemented our routines for examining the context surrounding the focal noun.)

Another common qualifier on the discourse entity was a specification of the context, where we would look for a discourse entity in relation to a verb. For example, in "Who sang the Tennessee Waltz" looking for a discourse entity that had a **synrole** attribute of "subj" and preceded the verb either equal to "sing" or with a **base** attribute equal to "sing".

In general, the development of these qualifiers is quite tricky. There is a tradeoff between specifying the characteristics of the actual discourse entity or specifying sufficient qualifiers to obtain a reasonable candidate set of discourse entities that can be evaluated, as described in the next section.

4.4 Question-Specific Evaluation of Candidate Answers

In general, the search query returned a considerable number of candidate answers. We then use question-specific routines to examine the context for text elements that have a specific relation to each candidate answer. (For all questions, as mentioned above, we first determined whether we could convert the "or" to "and" to retrieve sentences containing all the terms in the search query to reduce the number of candidates.)

For **What** and **Who** questions, the answer is sought in discourse entities that stand in a copular or appositive relationship to the candidate answers. The candidate answer is first screened to make sure that it does not contain a mismatch in degree or ordinal from one that is specified in the question (e.g., "highest" or "second"). An answer is first sought in a copular relationship, and if not found, the candidate is examined to determine whether it has a following appositive. For example, "What is the national airline of Spain?", the discourse entity "Spain's state airline" is followed by "Iberia". A similar set of tests is used in answering **WhatIsDef** and **WhoIsDef** questions, with the distinctions that these questions allow multiple answers.

For **WhoIsDef** questions, documents in the collection that were talking about a person with the same last name but a different first name were excluded from consideration. This was particularly important when a candidate answer was an anaphor, so it was important to establish that its antecedent was the person in the question. For these questions, we also looked for particular constructions, such as verb phrases beginning with "won" or "discovered" or prepostional phrases beginning with "of" (identifying a person's affiliation).

For **WhatNP** and **List** questions, the key noun constitutes a hypernym and what is being sought is a hyponym. We examine whether a candidate answer node (i.e., a discourse entity) is itself a hyponym, i.e., a noun phrase such as "Labor Party" in answering the question "What party led Australia from 1983 to 1996?". This includes examining the head noun of a phrase to determine, in WordNet, whether it is a hyponym of the key noun in the question. Another test examines other discourse entities in the same sentence to determine if they have a hypernym that matches the one required by the question (e.g., "Philadelphia" is a city in answering the question "What city is the Liberty Bell currently located in?").

For basic **How** questions, the focus is on the verb (e.g., "die"). The candidate answers are the verbs with the base form ("die" from "died"). If the verbs don't appear in sentences containing other key words from the question, an expanded set of synonyms is used (e.g., "assassinate", "murder", "kill", "shoot"). Then, if the verb form is not one that lexicalizes "death" (a derived form in WordNet), the sentence is examined for semantic components (or frame elements) in prepositional phrases that identify the cause of death (e.g., "died of cancer", "died in a plane crash", "from a ruptured abdominal aneurysm"). Other constraints in the question are examined to ensure that the key noun is the subject of the verb (e.g., making sure an anaphor refers to "Marty Robbins" and not "Jerome Robbins").

For **when** questions, the candidate answers were retrieved simply by asking for discourse entities that had the **tag** "when". For questions that asked specifically for a particular type of time ("what year", "what day", or "what month") (which had previously been recategorized as **When** questions rather than **What** questions), the candidate time entities were scrutinized specifically to ensure they were of the proper type. For these question, when the key noun was a phrase ("Cold Mountain" or "International Volunteers Day"), the sentence containing the candidate answer was examined for the occurrence of each word to give higher scores for those containing more words in the phrase.

For **Where** questions, all candidate answers had either a **tag** value of "where" or a **semtype** of "location". were those tagged as being of type "where" or in which the key noun was contained in WordNet, we tested whether a candidate answer was also present in the WordNet definition. We also tested whether a candidate answer had a hypernym in WordNet that was equivalent to the type of geographic location that was sought (e.g., in questions like "what city").

For **HowMany**, **HowMeas**, and **HowMuch** questions, the candidate answers were discourse entities containing numbers and that had already been tagged as one of these three types during the creation of the XML-tagged files. The specific tag was selected during this process based on the semantic characteristics of the modified noun. This considerably simplified the search for candidate answers.

4.5 TREC 2003 QA Results

We submitted one run for the main QA task and two runs for the passage task. Our overall main task score was 0.075, with scores of 0.070 for factoid questions, 0.000 for list questions, and 0.160 for definition questions. For the passage task, we had some difficulty in matching up our answer to the original text, since our sentence splitting often modified the original text, making it difficult to determine the offset and length of our answer. Our first run contained our computations of the offsets and lengths that we had constructed when the splitting was performed; for this run, our socre was 0.087. We then wrote a script that enabled us to examine the correctness of the offset and length of our passage and to make adjustments where necessary (37 percent of the cases). This second run, while not fully automatic, would have been the correct version if we had submitted the actual passage rather than the offset and length. For this run, our score was was 0.119 for the factoid questions. These scores were all considerably below the medians for these tasks.

As indicated earlier, we had not yet implemented many of the routines described in the previous section at the time of submission. With their implementation and further detailed examination of the results, we can provided an updated assessment of our progress in answering questions using the XML-tagging of documents and the use of XPath specifications for obtaining the exact answer. Tables 1 and 2 show our results for the passages task and the exact answer task, respectively.

Table 1. Factoid Questions (Passage)

Question Type	Number	No docs	Correct	Accuracy	Un Mrr	Adj Mrr	Fnd
How	36	9	3	0.11	0.16	0.21	0.37
HowMany	45	4	22	0.54	0.51	0.56	0.59
HowMeas	45	16	0	0.00	0.04	0.05	0.41
HowMuch	5	1	0	0.00	0.00	0.00	0.00
What	98	33	9	0.14	0.13	0.20	0.49
WhatNP	139	42	24	0.25	0.20	0.28	0.39
When	39	12	9	0.33	0.28	0.40	0.74
Where	1	0	1	1.00	1.00	1.00	1.00
Who	5	2	1	0.33	0.20	0.33	0.67
Total	413	119	69	0.23	0.20	0.28	0.47

Table 2. Factoid Questions (Exact Answer)

Question Type	Number	No docs	Correct	Accuracy	Un Mrr	Adj Mrr	Fnd
How	36	9	2	0.07	0.11	0.14	0.26
HowMany	45	4	21	0.51	0.50	0.55	0.59
HowMeas	45	16	0	0.00	0.02	0.04	0.28
HowMuch	5	1	0	0.00	0.00	0.00	0.00
What	98	33	5	0.08	0.06	0.20	0.20
WhatNP	139	42	17	0.18	0.14	0.31	0.31
When	39	12	8	0.30	0.24	0.63	0.63
Where	1	0	1	1.00	1.00	1.00	1.00
Who	5	2	0	0.00	0.00	0.00	0.33
Total	413	119	54	0.18	0.15	0.22	0.34

In these tables, the first column shows our breakdown of the question types and the second column the number of each question type. The third column shows the number of questions for which there were no answers in the top 50 documents provided by NIST. Since we used this set, we feel that an evaluation of our performance should take into account only those questions for which we have a possibility of obtaining an answer. The fourth column shows the number of correct answers using the NIST criteria. The fifth column shows the accuracy based on the number correct divided by the total number of questions less the number with no documents. The sixth column shows the mean reciprocal rank of our answers for those cases in which an answer was obtained in the top five answers, without adjusting for the questions with no answers. The seventh column shows the mean reciprocal rank, with adjustment for the questions with no answers. The eighth column shows the percentage of cases in which our answer set contained the correct answer (i.e., without regard to the rank of our answer).

Our overall accuracy for the passages task is 0.167 (69/413) and for the exact answer task is 0.131 (54/413). These are somewhat better than our official scores of 0.119 and 0.070, but they would still not affect our low rank among participating teams. We suggest that a better picture of our results is given by our accuracy scores after adjusting for questions with no answers in the top 50 documents. Our score of 0.23 for the passages task would place us in the 4th position, and our score of 0.18 for the exact answer task would place us in the 12th position. We would not suggest this should be used as the measure to be used in ranking teams, but only to indicate that the information retrieval component is important and that, notwithstanding, the basic mechanisms we have employed are viable. In addition, the final column of the tables further indicate the viability of our approach, since they indicate that we are finding the answers, but have further work to boost them in our ranking system.

For all question types, we have not yet implemented the full range of tests that we used in past years when using a database representation of questions and documents. We believe that our further implementation will lead to significant improvements. In addition, we note that many of our routines were implemented for specific question types. Many of these routines can be usefully employed for other question types and can be further generalized. Work in this area is also likely to lead to some overall improvements.

5 Novelty Detection Using XML-Taggged Documents

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 performing the tasks.

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. 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 different amounts of information, 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.

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, five 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 will be selected from among a wider than used in task 1, since these were conditioned by the greater recall of relevance sentences. 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.

5.3 Novelty Track Results

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

Table 1. Task 1 (Relevance)

Run	Precision	Recall	F-Score
clr03n1t	0.71	0.25	0.309
clr03n1d	0.72	0.32	0.385
clr03n1n2	0.69	0.45	0.483
clr03n1n3	0.72	0.24	0.316

Table 2. Task 1 (Novelty)

Run	Precision	Recall	F-Score
clr03n1t	0.51	0.24	0.272
clr03n1d	0.51	0.31	0.331
clr03n1n2	0.50	0.40	0.410
clr03n1n3	0.51	0.24	0.278

For task 1, our best run received an F-score of 0.483 for relevant sentences and 0.410 for new sentences. For all runs, our F-scores were on average higher than the median. When examining the scores by the topic type, we found that on relevance, our F-scores were quite similar, but on novelty, there was a wide difference in our system's performance, achieving a much higher average F-score on event topics. For run clr03n1d, our F-score on event topics was 0.369 and for opinion topics, it was 0.282.

The results show clearly that more information describing the topic is valuable in increasing recall dramatically, while precision is only moderately different. Changing the number of hits for relevance determination clearly decreases the recall significantly, to less than what was achieved with less information.

Table 3. Task 2 (Novelty)			
Run	**Precision**	**Recall**	**F-Score**
clr03n2	0.71	0.91	0.788

Task 2 shows that the novelty component of our system is performing at a quite high level. This indicates that when the relevance determination is of high quality, we are able to discriminate novel information quite well. If our system improves its relevance assessment, or if our system operates in an environment where a user can provide relevance feedback, we can expect to identify novel information with considerable fidelity. This would be quite useful for text summarization.

Table 4. Task 3 (Relevance)			
Run	**Precision**	**Recall**	**F-Score**
clr03n3f01	0.48	0.84	0.558
clr03n3f02	0.48	0.77	0.541
clr03n3f03	0.48	0.72	0.527
clr03n3f04	0.48	0.68	0.513
clr03n3f05	0.48	0.63	0.493

Table 5. Task 3 (Novelty)			
Run	**Precision**	**Recall**	**F-Score**
clr03n3f01	0.33	0.79	0.419
clr03n3f02	0.33	0.73	0.408
clr03n3f03	0.33	0.69	0.401
clr03n3f04	0.33	0.65	0.395
clr03n3f05	0.33	0.61	0.383

Task 3 also indicates the value of relevance feedback, as well as the value of using frequency assessments in improving recall. We also made an additional five runs with the frequency score cutoff at values from 0.06 to 0.10, with the trends shown above occurring with them as well, with lower and lower values for recall, with a resultant degradation in the overall F-score.

Table 6. Task 4 (Novelty)			
Run	**Precision**	**Recall**	**F-Score**
clr03n1t	0.53	0.91	0.655

As indicated above, we submitted the same run for task 4 as for task 2; our lower F-score of 0.655 for new sentences is a direct reflection of the decreased performance brought about by the degradation of precision.

6 Summary

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

References

Litkowski, K. C. (2001). Syntactic Clues and Lexical Resources in Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Ninth Text Retrieval Conference (TREC-9)*. NIST Special Publication 500-249. Gaithersburg, MD., 157-166.

Litkowski, K. C. (2002a). CL Research Experiments in TREC-10 Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 122-131.

Litkowski, K. C. (2002b). Digraph Analysis of Dictionary Preposition Definitions. *Proceedings of the ACL SIGLEX Workshop: Word Sense Disambiguation*. Philadelphia, PA., 9-16.

Litkowski, K. C. (2003a). Question Answering Using XML-Tagged Documents. In E. M. Voorhees & L. P. Buckland (eds.), *The Eleventh Text Retrieval Conference (TREC 2002)*. NIST Special Publication 500-251. Gaithersburg, MD., 122-131.

Litkowski, K. C. (2002a). Text Summarization Using XML-Tagged Documents. In Proceedings of the DUC 2003 Document Understanding Conference.