Question Answering Using XML-Tagged Documents

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

Abstract

The official submission for CL Research's question-answering system (DIMAP-QA) for TREC-11 only slightly extends its semantic relation triple (logical form) technology in which documents are fully parsed and databases built around discourse entities. We were unable to complete the planned revision of our system based on a fuller discourse analysis of the texts. We have since implemented many of these changes and can now report preliminary and encouraging results of basing our system on XML markup of texts with syntactic and semantic attributes and use of XML stylesheet functionality (specifically, XPath expressions) to answer questions.

The official confidence-weighted score for the main TREC-11 QA task was 0.049, based on processing 20 of the top 50 documents provided by NIST. Our estimated mean reciprocal rank was 0.128 for the exact answers and 0.227 for sentence answers, comparable to our results from previous years. With our revised XML-based system, using a 20 percent sample of the TREC questions, we have an estimated confidence-weighted score of 0.869 and mean reciprocal rank of 0.828. We describe our system and examine the results from XML tagging in terms of question-answering and other applications such as information extraction, text summarization, novelty studies, and investigation of linguistic phenomena.

1 Introduction

In previous years, DIMAP-QA was based on full parsing of the 10 or 20 NIST-supplied top documents of TREC texts and extracting semantic relation triples into a database from which answers were then found (Litkowski, 2001; Litkowski, 2002a). As noted previously, our system was intended to be part of a larger system of discourse analysis of the texts, which had not been sufficiently implemented to serve as the basis for question-answering. In addition, although our idea of capturing semantic relation triples (identifying discourse entities and their relations to other discourse elements) seemed sound, the use of a traditional database structure made it difficult to represent and exploit the structural properties of natural language.

Extensible Markup Language (XML) provides a more natural mechanism for representing texts. A valid XML document is a tree and we can readily design our entire representation on this tree structure. The entire TREC collection (or any subset of documents) can be represented as one tree; the next level of the tree represents each document. At the next level, each document may be represented as a set of sentences, each of which may then be subdivided into sentence segments or clauses (elementary discourse units), which are then broken down into traditional parse trees, ending in leaf nodes corresponding to the words in the sentences. Each node in the tree may have associated attribute names and values.

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. For question-answering, an XPath expression (query) essentially specifies search criteria that return a string answer, based on node types and attributes. For example, for question 1553 ("who makes Magic Chef refrigerators?"), a single XPath expression looks for a sentence containing "Magic Chef" and an anaphor with an antecedent, finds the sentence containing the antecedent, notices that the antecedent is a predicate to "is", and retrieves the subject of the predicative, "Maytag", as the answer to the question.

Section 2 presents the TREC QA problem description. Section 3 describes our system: sentence splitting, parsing, discourse and sentence analysis, and database development and XML tagging. Section 4 briefly describes question-answering against the document databases. Section 5 provides a detailed description of procedures used to answer questions from XML-tagged documents. Section 6 presents and analyzes our official results and the unofficial results achieved using the XSLT approach. Section 7 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

Participants in the main TREC-11 QA track were provided with 500 unseen questions to be answered from 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). 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.

Participants in the main task were required to answer the 500 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 46 questions for which no answer exists in the collection. Answers for the 500 questions were to be sorted according to a participant's confidence. NIST evaluators next judged whether an answer was correct, inexact, unsupported, or incorrect. The submissions were then scored as the sum with I from 1 to 500 of the number of correct answers up to I divided by I, and this sum divided by 500, called a confidence-weighted score (CWS).

CL Research performed two runs for the main task. However, we mistakenly submitted the second run, based on using the top 20 documents (rather than one for the top 10 and one for the top 20), for both run tags. The discussion of our official submission will thus present results for only one run.

3 System Description

The CL Research question-answering system consists of four major components: (1) a sentence splitter that separated the source documents into individual sentences; (2) a parser which took each sentence and parsed it, resulting in a parse tree containing the constituents of the sentence; (3) a parse tree analyzer that identified important elements of the sentence and created semantic relation triples stored in a database and a set of discourse constituents (sentences and clauses, discourse entities, verbs and prepositions) used to create an XML-tagged version of each document; and (4) two question-answering programs, one using the database and one using the XML documents.

3.1 Sentence Splitting

Sentence splitting proceeded as described in previous years (Litkowski, 2002a; Litkowski, 2001). Using the new AQUAINT collection posed some difficulties because of idiosyncratic markup (described below). Unlike previous years, this phase of processing was completely robust.

For TREC-11, the top 20 documents (as ranked by the search engine) were analyzed for the main task, with one database containing only the processing for the top 10 documents and the other for the full 20 documents. Overall, this resulted in processing 10,000 documents from which 257,276 sentences were identified and presented to the parser. Thus, we used an average of 25.7 sentences per document (compared to 22.8 in TREC-10, 28.9 in TREC-9 and 31.9 in TREC-8) or 257 sentences for the 10-document set and 514 for the 20-document set.

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 Discourse and Sentence Analysis

The sentence parsing in the CL Research system is part of a broader system designed to provide a discourse analysis of an entire text or set of texts. We are using this system for processing encyclopedia articles, historical texts, scientific articles[1], as well as the news wire texts in TREC and the RST treebank (Linguistic Data Consortium, 2002). Frequently, the

[1]See http://www.clres.com/sa-articles.xml.

input has already been tagged (e.g., in SGML) and our processing may result in additional tagging.

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. The relationship analysis gives rise to a **semantic relation triple**, which consists of a discourse entity (the noun phrase itself), a syntactic or semantic relation which characterizes the entity's role in the sentence, and a governing word to which the entity stands in the semantic relation. A triple is generally equivalent to a logical form (where the operator is the semantic relation) or a conceptual graph, except that a semantic relation is not strictly required, with the driving force being the discourse entity.

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 the noun phrases are encountered, we attempt to identify the syntactic or semantic role they play in the sentence. These include "SUBJ," "OBJ", "TIME," "NUM," "ADJMOD," and the prepositions heading prepositional phrases. Relative clauses and appositives are inherently modifiers of their attachment points.

The governing word was generally the word in the sentence that the discourse entity stood in relation to. For "SUBJ," "OBJ," and "TIME," this was generally the main verb of the sentence. For prepositions, the governing word was generally the noun or verb that the prepositional phrase modified. (Because of the context-sensitive dynamic parsing goals that were added when a verb or a governing noun was recognized, it was possible to identify what was modified.) For the adjectives and numbers, the governing word was generally the noun that was modified.

A semantic relation and a governing word were not identified for all discourse entities. Notwithstanding, a list of every discourse entity is maintained with a unique identifier and all characteristics that can be associated with them.

3.4 Database Development and XML Tagging

The text analysis module generates two types of output: (1) a database of semantic relation triples and (2) an XML tagging of the text. Either type of output is optional. For the database, each semantic relation triple is added as it is generated. Overall, 2,306,698 semantic relation triples were created in parsing the

257,276 sentences, an average of 9.0 triples per sentence (compared to 9.7 in TREC-10).

Although we have achieved some degree of success with the database approach, we have found that it is difficult to work with. The table of semantic relation triples is not intuitive because the flat structure removes the tree structure that is inherent in a grammar-based parser. For simple questions, answering is a matter of forming a join between a question database parsed and analyzed in the same way as the document database. With greater complexity, and with a document database where a simple join does not produce an answer, the logic required to examine a path of relations becomes more difficult.

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, the output consisted of groups of ten documents from the NIST-provided top documents for each question. Since we only processed the top 20 documents, we had 500 XML files for the top ten documents and 500 for documents ranked 11th through 20th. These are the files used for answering the TREC questions.

4 Question-Answering Using Document Databases

For TREC-11, the question-answering against the document databases was little changed from previous years. We refer to our earlier detailed descriptions (Litkowski, 2002a; Litkowski, 2001) and provide only a brief overview here.

For TREC-11, a database of documents was created for each question, as provided by the NIST generic search engine. A single database was created for each question in the main task. The question-answering consisted of matching the database records for an individual question against the database of documents for that question.

The question-answering phase consists of four main steps: (1) detailed analysis of the question to set the stage for detailed analysis of the sentences according to the type of question, (2) coarse filtering of the records in the database to select potential sentences, (3) extracting possible short answers from the sentences, with some adjustments to the score, based on matches between the question and sentence database records and the short answers that have been extracted and (4) making a final evaluation of the match between the question's key elements and the short answers to arrive at a final score for the sentence. The sentences and short answers were then ordered by decreasing score. The short answer for each question (an "exact" answer), its score, and its sentence (the "justification") were printed to a file. This file was then sorted by score to create a "confidence-ordered" answer set submitted to NIST.

5 Question-Answering Using XML-Tagged Documents

As described earlier, question-answering against XML files essentially involves describing a path (XPath) from the top of the tree(s) to a discourse entity (in our case, to a **discent** node) which is returned as the answer. To do this, a question is converted into an XPath expression used to select nodes in the files. For example, for question 1593 ("What percent of Egypt's population lives in Cairo?"), an XPath expression is

```
//segment[contains(.,'Cairo')]
//discent[contains(.,'percent') and
@tag='howmany']
```

The first double slash says to find any node in all documents being searched that are marked as **segment** elements and contains the word "Cairo". The second double slash says to find all **discent** elements that are descendants of such segments containing the word "percent" and that have an attribute **tag** with value equal to "howmany". This XPath expression will return zero or more nodes from however many documents are processed.

In general, question-answering consists of the following steps: (1) analyze the question and convert it into an XPath expression; (2) load the XML file(s) and select the nodes satisfying the XPath expression; and (3) if necessary, score and/or evaluate the nodes returned and present them to the user. The second step is the easiest, consisting of a loop over the files being processed, with a single statement to load the file and another single statement to select the nodes.

The first step, determining the XPath expression, is more difficult. As can be seen for q1593, not all the question elements are present in the query. This may be characterized as a "backoff" strategy, beginning with all the terms in the query and removing some that are not necessary or are too restrictive. For q1593, including all the terms will result in zero nodes. This is frequently the case, with questions often providing much more information than is likely to appear in one sentence. The third step, evaluating the nodes selected, is generally not as complicated; a well-formulated XPath expression generally returns only a couple of answers, although there are some question types that require more extensive processing. We will describe our observations about the first and the third steps in more detail below.

As indicated earlier, we were not able to implement our question-answering against the XML-tagged documents for our official submission. Using these documents has required an entirely new conceptual approach, involving the resolution of many intertwined issues. This new approach has been evolving since our submission; many refinements are necessary and many possibilities for making these changes have been emerging.

To begin with, the whole tagging process described in general terms above requires dealing with virtually the full panoply of natural language processing, including tokenization, sentence splitting, parsing, word-sense disambiguation, anaphora resolution, and discourse analysis. While we have developed a system that comprehends all these components, many of the components have not yet been implemented to the state of the art. For example, our anaphora resolution module is currently estimated at 55 percent correct, whereas the state of the art has been attaining levels over 80 percent. Also, our typing and characterization of prepositional semantic relations is currently operating at about 20 percent (see Litkowski (2002b) for our lexicographic approach to this problem), so that we have to rely on the preposition itself as the bearer of information about the semantic relation. Further, our discourse structure analysis is an initial implementation, presently handling only appositives and relative clauses.

A second major issue to be faced is the selection of tags and their attribute names and values. This issue involves identifying what information will be useful and then developing techniques for extracting the information, using whatever other resources may be available (such as dictionaries and thesauruses). An important question, given our semantic predilections, is what semantic classes to use for characterizing discourse entities. Another important question is how to group information: what sentence parts should be grouped together and which modifiers should be separated or put into attributes of a discourse entity.

Dealing with these issues (identifying problems with the functioning of our XML output generation and examining representational alternatives) is very complex and requires the development of mechanisms for analyzing them. This has led to two steps in our development cycle: (1) the development of an analysis interface for assessing problems and (2) the use of the TREC questions as guidance for inadequacies in our representations. As will be suggested below, the use of XSLT has demonstrated not only a capability for dealing with these issues, but also provides a strong indication that an XML representation of text will be extremely useful for a wide range of applications, including question-answering.

5.1 Step 1: An XML Analysis Interface

The generation of 1000 XML files each containing 10 TREC documents provides a large amount of data; the XML files are approximately five times the size of the TREC documents. The XML files can be viewed (with retention of the nested structure) in Microsoft's Internet Explorer, but this does not allow any systematic examination of the data. Conventionally, those working with XML files develop XML stylesheets for portraying the data (XSLT), perhaps embedded in interactive browser web pages. However, this requires a prior design, something not yet developed for the files generated here. Moreover, XLST is somewhat involved and not convenient for the analysis required here. Instead, we developed a GUI interface which enables lower-level access to the XML data and provides an easier development vehicle for the kinds of exploration needed here. Lessons learned from this interface can guide future development of applications using XML-tagged documents.

Our development environment (known as XMLPartner) provides powerful tools for low-level access to the XML data. A well-structured XML file has the form of a completely hierarchical tree, wherein nodes contain the data and the attributes.[2] In our system, an XML file of any size (with extremely large files using a buffered stream) is loaded with one statement. Similarly, a search for nodes providing the answer to some query (the XPath expression conforming to the XML Path Language) is accomplished with one statement. This enables us to focus on development of queries and examination of search results (perhaps with further search statements). We have developed surrounding GUI components to facilitate examination of different aspects of the XML data (referred to below as XML Analyzer).

5.1.1 Global Examination of Data

In the first place, we used XML Analyzer to examine (and sometimes extract) interesting phenomena in the text. XML Analyzer can be used as a concordancer; a suitable XPath expression can extract all sentences in our TREC XML files (80 MB) that begin with "After" in four minutes. Similarly, we can find all discourse entities that contain a capitalized word, to examine whether we have assigned them an appropriate named entity type. In general, we use this basic capability to examine words, the entities and sentences in which they occur, and their attributes.

We display results of a search with the entity (if requested), the document title (the document number for TREC documents), and the sentence containing the entity. When we are searching only for sentences, no entity is given. A user can select a sentence and ask to see all the entities in that sentence. A user can select an entity and request all other entities which co-refer to it or have it as an antecedent.

5.1.2 Detailed Investigation of Discourse Entities

The XML Analyzer can be used to examine details about particular discourse entities. For example, question 1502 asks "when was President Kennedy killed". In the NIST top 10 documents for this question, a search on "Kennedy" in discourse entities identified 152 occurrences (the Kennedy clan). Narrowing the search to those also containing "Edward" gave 7 instances; expanding this to include entities where "Edward" was contained in the antecedent attribute identified an additional 14 instances. An examination of the attributes of these 21 instances showed 14 as the subject, one as the object, one as a possessive pronoun and three as a genitive determiner, and two as a prepositional object.

Use of the XML Analyzer in this way suggests that a user can examine the different relations in which an entity participates. For those as subject, we can examine the verbs to determine what kind of actions the subject performs (for action verbs) or what properties the subject has (for stative verbs). For those as possessive pronoun or genitive determiner, the user can examine what kinds of possessive relationships the entity can have (e.g., as brother, his back, or his commitment). For those as prepositional object, we can examine the relations the entity has with other entities. More generally, this suggests the possibility of an interactive web page allowing a user to explore the different relations in which a discourse entity participates, perhaps moving to other discourse entities with which it shares a relation.

5.2 Step 2: Answering Questions with XPath Expressions

As our first step in developing techniques for answering questions, we examined whether the

[2]Indexing of XML documents includes traditional indexing for information retrieval, but is also "XML-aware", meaning that searches can be efficiently performed on any XML tags, attributes, and values. We envision that XML output generated by our system would be subjected to XML-aware indexing.

answers (as contained in the patterns) occurred as discourse entities in our XML output. For virtually all cases, the answers were present in distinct entities; in those where they were not, we identified several bugs we were able to correct in our XML output processing.

This process generalizes well with our interface: create an XPath expression, determine whether it leads to appropriate discourse entities, and if not, make changes in some part of our system, either correcting bugs or altering our XML representation. This process has involved learning the intricacies of XPath expressions, which have proved capable of returning the exact answer to almost all TREC questions.

We developed XPath expressions for a contiguous 20 percent sample of the TREC questions, providing a basis for drawing conclusions. In general, the XPath expressions are highly confirmatory of techniques developed over the years in the QA track. The XPath expressions show that simple string patterns are quite effective and that syntactic and semantic information can be quite useful. Our development of these expressions shows that characterizing the patterns in the underlying text via XML elements and attributes is worthwhile for QA, and potentially other applications. We demonstrate this by showing the XPath expressions for several question types. In each of these cases, the development of an XPath expression proceeds by (1) further characterization of the question type, (2) development of a query component that selects segments, and (3) refinement of the query in specifying characteristics of the discourse entity.

5.2.1 WhatIs and WhatNP Questions

What questions have the highest frequency, constituting more than 40 percent of the questions, and have the most subtypes. Four principal varieties are: (1) "**What (is|was) (the NP ... | NP called | the ORD NP | NP1's NP2 | NP)?**", where **NP** is a noun phrase and **ORD** is an ordinal (e.g., 'first'); (2) "**What NPA (is (NP2 | PP) | did (NP2)? V (PREP)?)**", where **NPA** is **NP1** or **NP1's NP3**, **PP** is a prepositional phrase, **PREP** is a preposition, and the internal '?' indicates an optional element; (3) "**What is NP's (real | original | nick) name?**", and (4) "**What (do NP V | does NP stand for)?**", where **V** is a verb.

For the most general variety ("**What (is|was) the NP ... ?**"), a canonical answer would be "**X (is|was) the NP ... ?**". Examples are "What is the oldest college bowl game?" (1529), "What is the most populated country in the world?" (1544), and "What is the text of an opera called?" (1583). A suitable XPath expression can ask **//segment[contains(.,'(is|was) the NP ...')]**, i.e., a simple string match, or perhaps suitable subsets of the NP. To get at the specific discourse entity, the XPath expression would continue with **//discent[contains(.,'NP') and @synrole='obj'] /preceding-sibling::verb[.='was'] /preceding-sibling::discent[@synrole='subj']**, which says "find a discourse entity containing NP with syntactic role 'object' that is preceded by a verb equal to 'was' and that is preceded by a discourse entity with syntactic role 'subject'". This discourse entity is the answer to the question.

Another possibility for the general "**What (is|was) the NP ... ?**", as well as the third variety above "**What is ... name?**" and the second alternative of the fourth variety "**What does NP stand for?**", is a search for a relative clause, appositive, or parenthetical. As mentioned earlier, our text analysis and XML-tagging modules generally identify these as subsegments. Our segment search for these can be formulated as **//segment[contains(.,'NP') and (child::segment or contains(.,', or') or contains(.,'('))]**, which looks for a segment that contains the NP and contains either a nested segment or a simple string (a comma and "or" or an opening parenthesis). In these cases, the desired discourse entity would be obtained by first looking for **//discent[contains(.,'NP')]** and then either **/preceding-sibling::discent** or **/following-sibling::discent**. In the case asking what something stands for, the NP is usually an abbreviation or acronym. In this case, it is possible to build a more elaborate XPath expression that tests whether the letters of the answer node(s) correspond to the NP. With our low-level access to the answer nodes, however, it may be more efficient to perform this test in a post-processing phase that evaluates the answers.

A post-processing phase becomes even more important in handling the second variety listed above ("**What NPA (is (NP2 | PP) | did (NP2)? V (PREP)?)**"). For this variety, the segment search is generally of the form **//segment[contains(.,'NP2') and (child::verb[@root='V'] or contains(.,'PP'))]**, looking for elements of NP2, the root form of V, or elements of the prepositional phrase, usually the prepositional object, depending on the specific subvariety. The desired discourse entity may be present in the same discourse entity containing NPA (**//discent[contains(.,'NPA')]**), such as q1525, "What university did Thomas Jefferson found?", where our text processing created a single named entity "University of Virginia". However, in the more general case, the head noun of NPA would be a hypernym of an answer (such as q1499, "Which African country's major export is coffee?"). In this case, we might return all discourse entities in segments containing “major”, “export”, and “coffee” (//**discent**, i.e., without any further restrictions) and in post-processing ask whether any of them are hyponyms of “African country”, using WordNet as the basis for such tests.

5.2.2 When Questions

When questions comprise 20 percent of the TREC set. In our analyses, only two varieties were needed: (1) “**WHEN (was|did) NP VP?**”, where **VP** includes a verb and other constituents, and (2) “**WHEN was NP V?**”. **WHEN** can be either “When”, “What year”, or some other time question. Although the second variety is a subset of the first, the second is distinguished in having a verb as the final element (e.g., q1502, “What year was President Kennedy killed?”).

Since our parser’s grammar labels time expressions, identification of time patterns for our segment search was not necessary. Instead, for the first variety, the XPath expression was generally **//segment[contains(.,’NP’) and contains(.,’VP’)]**. For the second variety, synset expansion of the verb was necessary, **//segment[contains(.,’NP’) and (contains(.,’V’) or contains(.,’V1') or ... contains(.,’VN’))]**, where the **VI** are verbs from **V**’s synset. To get at the discourse entity, **//discent[@tag=’when’]** was sufficient.

5.2.3 Where Questions

Where questions constituted 10 percent of the TREC set. Most of these questions followed the pattern “**Where is NP**”. There were some minor variations, e.g., with “where” replaced by “**At what place**” or with “**What NP ... LPrep?**”, where **LPrep** is a locative preposition. In all cases, the segment search was specified by **//segment[contains(.,’NP’)]** and the discourse entity search by **//discent[@tag=’where’]**.

5.2.4 Other Question Types

Who questions comprised more than 10 percent of the TREC set. Their complexity was comparable to the **What** questions; the discussion above on those questions generally covers the issues involved in the **Who** questions. **How** questions, including “how many”, “how much”, and “how measured”, comprise the remainder of the TREC set. The XPath expressions for these questions generally follow the patterns for the **When** and **Where** questions, replacing the **tag** value to “howmany”, “num”, “howmuch”, or “howmeas”, which were created during the text analysis.

6 TREC-11 Results and Analysis

6.1 Official Results Using Document Databases

CL Research submitted 2 runs for the main task, both using the document database approach used in previous years. Our intent was that one run would be based on the top 10 NIST documents and the other based on the top 20. However, an error in submission resulted in the set for the top 20 being submitted twice. Our official confidence-weighted score (CWS) was 0.049, with 36 correct answers, 10 inexact answers, and 2 unsupported answers.

Our official submission was significantly affected by an oversight in which *Associated Press Newswire* texts were not properly subjected to the sentence splitter. The effect (for 101 questions) was that whole paragraphs were evaluated and scored as a single sentence. The scoring used in our system gave a large number of these paragraphs unduly high scores. These paragraphs, from which an answer was extracted, were thus given a high ranking, significantly affecting the CWS. We have not yet reprocessed those texts to determine the overall effect on our document database submission. By changing the scores for these answers to the average of our scores for *Xinhua* and *New York Times* documents, the estimated CWS was changed to 0.080. However, the effect is likely to be more significant, since the selection of these paragraphs as answer sources precludes the possible selection of correct answers from lower ranked passages.

Notwithstanding this difficulty, our document-based question-answering produced results consistent with our system’s performance in the past two years. We calculated the mean reciprocal rank for our exact answers (0.128) and for the sentences (0.232) containing them. As indicated earlier, we made no significant changes in our document-based question-answering, so these results were expected.

6.2 Unofficial Results Using XML-Tagged Documents

To assess the potential benefit from using XML-tagged documents, we selected a contiguous set of 100 questions (1493-1592) and developed XPath expressions by hand for them to determine if we could obtain exact answers. This set was started after we had gained some familiarity with using the tagged documents and our XML Analyzer. We have not yet automated the creation of XPath expressions; we have found it necessary to develop an understanding of the patterns suitable for the different question types and varieties, as described in the previous section. We selected a contiguous set to compare our results to a contiguous subset of the full set of questions, rather than a random subset that might not generalize.

We applied the XPath expressions against the XML-tagged files for these 100 questions, first against the top 10 documents and then against the next 10 documents if we did not obtain an answer against the first 10. We constructed an answer set file conforming to the NIST specifications, using the first answer returned or NIL if no answers were returned. We did not use any scoring system to order the answers, but rather gave a score of 1005 to all non-NIL answers and 1000 to all NIL answers. As a result, sorting the answers by score ordered the answer file with the highest question number first. This answer file was then scored with the NIST Perl script.

Since many questions had no answers in the top 20 documents, we also formed a subset of 75 questions for which answers were present, but including the six questions in this subset for which no answers were present in the TREC collection, to test the effect of posing an XPath expression to the top 20 documents.

We developed the XPath expressions for these questions to conform as much as possible to linguistic intuitions, rather than just attempting to get the correct answer so that we will be able to develop appropriate mechanisms for automating the process. For the most part, the expressions have the simplicity described in section 5, with only a few requiring complex expressions. The expressions had a very high specificity in retrieving answers. The 75 XPath expressions returned a total of only 171 answers (2.3 per question), of which 97 were exact answers (1.3 per question). (For q1587, "What did Sherlock Holmes call the street gang that helped him crack cases?", which NIST characterized as not having an answer, the XPath expression returned "Baker Street Irregulars", although this answer would have been judged as unsupported.) Table 1 shows the confidence-weighted scores based on our official submission and based on the XML-based answers.

Table 1. Confidence-Weighted Scores for Question Samples

Sample	CWS
Official (100)	0.192
XML-based (100)	0.816
Official (75)	0.266
XML-based (75)	0.869

As can be seen, the question subset we have chosen is much better than our official results for the full set (0.049). In table 2, we show the mean reciprocal rank for these subsets.

Table 2. Mean Reciprocal Ranks for 75 Question Sample

Sample	CWS
Official (first answer only)	0.160
XML-based (first answer only)	0.800
Uofficial (top 5 answers)	0.243
XML-based (top 5 answers)	0.828

In this table, the first two rows correspond to the percent of answers that are correct, while the second two rows consider the top 5 answers, as in previous years. These results, again, are higher than our overall results, where we answered only 36 questions correctly and our overall mean reciprocal rank was 0.128. Thus, this sample may overstate how much we would achieve with XPath expressions for all 500 questions.

In general, our results using the XML-tagged documents and XPath expressions were quite surprising. While our XML-tagging is comprehensive, it is far from complete, as indicated above. In addition, the question types and varieties did not seem to require an elaborate typing of answers (such as Harabigiu et al. (2002) or Hovy et al. (2002)). Rather, the XML-based approach seems closer to the pattern-matching methods described in Soubbotin (2002), Brill et al. (2002), and Ravichandran & Hovy (2002), with additional benefits achievable by having structural information available. However, it is not clear how much taggging is necessary for question-answering. This is an issue for further research; we believe our methodological approach is well-suited to examining this issue, using the many levels of detail available.

7 Future Developments

As mentioned earlier, many components of our XML-tagging system can be improved, including our discourse analysis, anaphora resolution, semantic typing, and disambiguation components. As these improvements are made, they can be examined specifically for their contribution to question-answering. In addition, we see the XML-tagging approach as having potential benefits for investigation of linguistic phenomena, information extraction, novelty detection, and text summarization.

We will be generalizing our XML Analyzer to handle arbitrary tagging systems used in tagging text, such as part-of-speech taggers, chunkers, word-sense taggers, and discourse taggers. This will entail only minor changes and will facilitate examination of linguistic phenomena, including the possibility of adding tags, one of the basic objectives of XSLT.

In developing XPath expressions to answer questions, the final component of the expression requests discourse entity nodes with specific properties. By focusing instead on all discourse entities having particular properties over a range of documents, the XML Analyzer can be reconfigured to act as an information extraction tool. A specification of the discourse entity type desired to propagate a database can be used to build suitable XPath expressions to extract this data.

The TREC top documents were noteworthy for frequently containing the same or a very similar document several times (perhaps differing only in the document number). The low-level functionality available to us for examining XML nodes makes it easy to recognize such duplication. This can be extended to recognize near duplication based on varying criteria, such as synonymy. Using the Kennedy example described earlier, for example, it would be straightforward to examine the various relations in which Kennedy participates for synonymy and novelty.

Finally, the low-level functionality also allows us to summarize the characteristics of a text at any node level (e.g., frequency, types of nodes, and novelty). These characteristics can then be used to create various text summaries, and indeed, to create new XML documents by combining nodes from the original document. For example, encyclopedia articles frequently discuss a topic by defining it in several places using a copulative and possessive properties; the corresponding nodes from the original article can be used to generate an overall definition.

8 Summary

CL Research has made a preliminary investigation of the feasibility of massive XML tagging of source documents for the purpose of answering questions. Our results strongly suggest that this is a viable approach. Further, the development of the infrastructure necessary to evaluate this approach suggests that XML tagging may be useful in several other text processing tasks.

References

Brill, E., Lin, J., Banko, M., Dumais, S., & Ng, A. (2002). Data-Intensive 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.

Harabigiu, S., Moldovan, D., Pasca, M., Surdeanu, M., Mihalcea, R., Girju, R., Rus, V., Lacatusu, F., Morarescu, P., & Bunescu, R. (2002). Answering complex, list, and context questions with LCC's Question-Answering Server. 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., 355-361.

Hovy, E., U. Hermjakob, & C. Lin. (2002a). The Use of External Knowledge in Factoid QA. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 644-652.

Linguistic Data Consortium (2002). The Rhetorical Structure Theory Discourse Treebank. ISBN 21-58563-223-6. Philadelphia, PA.

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.

Ravichandran, D. & E. Hovy. (2002). Learning Surface Text Patterns for a Question Answering System. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, PA., 41-7.

Soubbotin, M. M. (2002). Patterns of Potential Answer Expressions as Clues to the Right Answer. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 122-131.