Semantic Search via XML Fragments: A High-Precision Approach to IR

Jennifer Chu-Carroll[1], John Prager[1], Krzysztof Czuba[2], David Ferrucci[1], and Pablo Duboue[1]

[1]IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA

[2]Google Inc., 1440 Broadway, 21st Floor, New York, NY 10018, USA

{jencc,jprager,ferrucci,duboue}@us.ibm.com; kczuba@google.com

ABSTRACT

In some IR applications, it is desirable to adopt a high precision search strategy to return a small set of documents that are highly focused and relevant to the user's information need. With these applications in mind, we investigate semantic search using the XML Fragments query language on text corpora automatically pre-processed to encode semantic information useful for retrieval. We identify three XML Fragment operations that can be applied to a query to *conceptualize*, *restrict*, or *relate* terms in the query. We demonstrate how these operations can be used to address four different query-time semantic needs: to *specify target information type*, to *disambiguate keywords*, to *specify search term context*, or to *relate select terms* in the query. We demonstrate the effectiveness of our semantic search technology through a series of experiments using the two applications in which we embed this technology and show that it yields significant improvement in precision in the search results.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Languages, Experimentation

Keywords

Semantic Search, XML Retrieval, Question Answering

1. INTRODUCTION

In some IR applications, in particular those in which search results are directly presented for user consumption, it may be highly desirable to employ a search strategy that strongly favors precision at the possible cost of a loss in recall. Take, for example, an intelligence analyst trying to find out whether there is nuclear technology transfer between North Korea and Syria. It may be preferable for a system to present to the analyst (at least initially) a small set of documents to confirm incidents of such technology transfers, than to return a much larger and noisier document set from which a list of all transfers of this nature can be compiled.

With this class of applications in mind, we investigate, on unstructured text documents, the use of semantic search techniques, which have primarily been adopted in search on the Semantic Web or in collections of highly structured XML documents (cf. [7, 4]). Rather than rely on document authors to encode semantic information as metadata, we employ automatic named entity and relation recognizers to pre-process text corpora and identify the semantic information we intend to leverage to improve search performance.

In our applications, there are some frequently-occurring query-time semantic needs, such as *specifying target information type*, *disambiguating keywords*, *specifying search term context*, and *relating select query terms*. To construct high-precision semantic queries to address these needs, we adopt the XML Fragments query language [3] because of its expressiveness and the availability of a search engine that supports the query language. In a nutshell, XML Fragments allow XML tags to surround query terms as a way to constrain the semantics of those terms. We identify three XML Fragment operations that can be applied to a query to *conceptualize*, *restrict*, or *relate* terms in the query, and we demonstrate how they can be utilized to address our query-time semantic needs. Since these operations constrain the underlying query, it is expected to yield more focused and more precise search results. Obviously, the impact of these operations on the precision and recall of retrieval results is directly affected by the accuracy of the named entity and relation recognizers used to analyze the corpus. In the case of perfect annotation, there should be a significant increase in precision with no loss in recall. We show that even in the face of imperfect annotations, our experiments yield significant improvement in precision in the search results.

The rest of the paper is organized as follows. Section 2 gives a brief overview of XML Fragments, while Section 3 describes how we pre-process our corpora to encode semantic information. In Section 4 we discuss the three XML Fragment operations and their four use cases. Section 5 presents experimental setup and results. Finally, we discuss related work and conclude.

SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.

```
<Book>
   <Title>My Life</Title>
   <Author>
      <FirstName>Bill</FirstName>
      <LastName>Clinton</LastName>
   </Author>
   <Publisher>Knopf</Publisher>
   <PubDate>2004</PubDate>
</Book>
<Book>
   <Title>
      The Survivor: Bill Clinton in the White House
   </Title>
   <Author>
      <FirstName>John</FirstName>
      <LastName>Harris</LastName>
   </Author>
   <Publisher>Random House</Publisher>
   <PubDate>2005</PubDate>
</Book>
```

Figure 1: Two Sample Book Documents

2. XML FRAGMENTS OVERVIEW

XML Fragments were initially developed to enable viewing and editing of parts of an XML document [6]. [3, 2] introduced the idea of using XML Fragments as a document-centric means to specifying queries for searching XML documents and showed it to be sufficiently expressive to address most information needs in multiple domains. In this section we give a brief overview of the subset of the XML Fragments query capabilities relevant to the discussion in this paper.

Consider a canonical example on the retrieval of book information encoded as XML documents in Figure 1. The XML Fragment query *[[<Book> Bill Clinton </Book>]]*, which allows *Bill* and *Clinton* to appear anywhere inside a *Book* tag, retrieves both documents, while *[[<Book><Author> Bill Clinton </Author></Book>]]* retrieves only the first document. In the above and subsequent examples, we denote the boundaries of XML Fragment queries with double square brackets (*[[. . .]]*) for better readability.

In addition to XML tags, XML Fragments support classical operators such as "+", "–", and *phrase*. The phrase operator applies to content within XML tags, such as *[[<Title> "White House" </Title>]]*, while "+" and "–" apply also to XML Fragments themselves. For example, *[[<Book><Title> Bill +Clinton </Title> </Book>]]* requires that all books retrieved have titles containing *Clinton*, while *[[<Book> +<PubDate> </PubDate> </Book>]]* retrieves only those books with publication dates in their records. Additionally, the language supports numeric comparison operators represented as XML tags. For instance, *[[<Book> <PubDate> <.GE.> 1999 </.GE.> </PubDate> </Book>]]* retrieves books published in or after 1999. For further details on the XML Fragment query language, see [3].

3. CORPUS ANALYSIS FOR XML RETRIEVAL

Much of the effort in the document-centric view of XML search has focused on developing query languages and search methodologies for the retrieval of XML documents, an example of which was discussed in the previous section. While this search paradigm has been shown to be effective in retrieving existing XML documents where the semantics of text segments are clearly encoded, the majority of electronic documents available today are significantly less structured than the examples shown in Figure 1. We believe that XML search can be more broadly applicable if existing unannotated documents can be automatically processed to include useful semantic information which can subsequently be used to improve search results. In particular, we are interested in annotating semantic information, such as named entities and relations, that can be reasonably reliably extracted from text, and in exploring alternative ways in which this semantic information may be exploited by XML search to improve retrieval performance. For example, Figure 3 shows how the unstructured text in Figure 2 may be annotated with named entities, such as *Person*, *Date*, and *College*, and relations, such as *BirthDateOf* and *AlmaMater*, of general interest (some annotations are omitted for space reasons).

There are ample techniques and systems in the area of information extraction for creating named entity and relation annotations analogous to those shown in Figure 3 (cf. [1, 19]). In this paper, we will focus on the alternative ways in which we exploit the XML Fragment query language to perform semantic search over a text corpus pre-processed with a set of named entity and/or relation annotators. We assume that the annotators adopt a common type system,

President Clinton was born William Jefferson Blythe IV on August 19, 1946, in Hope, Arkansas, three months after his father died in a traffic accident. When he was four years old, his mother wed Roger Clinton, of Hot Springs, Arkansas. In high school, he took the family name.

Clinton was graduated from Georgetown University and in 1968 won a Rhodes Scholarship to Oxford University. He received a law degree from Yale University in 1973, and entered politics in Arkansas.

Figure 2: Sample Excerpt From Clinton's Biography

<***BirthPlaceOf***> <***BirthDateOf***> <***Alias***> <***Person***> President Clinton </***Person***> was born <***Person***> William Jefferson Blythe IV </***Person***></***Alias***> on <***Date***> August 19, 1946 </***Date***> </***BirthDateOf***>, in <***City***> Hope, Arkansas </***City***> </***BirthPlaceOf***>, three months after his father died in a traffic accident. When he was four years old, <***SpouseOf***> <***Person***> his mother </***Person***> wed <***Person***> Roger Clinton </***Person***> </***SpouseOf***>, of <***City***> Hot Springs, Arkansas </***City***>. In high school, he took the family name.

<***AlmaMater***> <***Person***> Clinton </***Person***> was graduated from <***College***> Georgetown University </***College***> </***AlmaMater***> and in <***Date***> 1968 </***Date***> won a Rhodes Scholarship to <***College***> Oxford University </***College***>. <***AlmaMater***> <***Person ref="Clinton"***> He </***Person***> received a law degree from <***College***> Yale University </***College***> </***AlmaMater***> in <***Date***> 1973 </***Date***>, and entered politics in <***UsState***> Arkansas </***UsState***>.

Figure 3: Sample Annotations of Text in Figure 2

and that the annotations, along with lemmatized terms in the corpus, are stored in the search index. This technique is an extension of Predictive Annotation in [15] and of [14].

4. USING XML FRAGMENTS FOR SEMANTIC SEARCH

In our work, we focus on enriching query expressiveness to address four query-time semantic needs and to yield higher precision search results. We have identified three operations for this purpose that one may apply using the XML Fragment query syntax with its classical operators [2]:

- The **conceptualization** operation, which generalizes a lexical string to an appropriate concept in the type system represented by that string. For example, the query *animal* returns documents containing the word "animal", while the conceptual query *[[<Animal> </Animal>]]* retrieves documents containing the annotation *Animal*, which applies to all subtypes of the concept *animal*, e.g., *lion*, *owl*, and *salmon*.

- The **restriction** operation, which constrains the XML tags in which keywords must appear to be considered relevant. For example, *[[<Animal> bass </Animal>]]* returns documents in which the literal "bass" is used in its fish sense, while *[[<Instrument> bass </Instrument>]]* retrieves those where it represents a musical instrument. Note that in this operation, the span of the specified annotation needs to contain the span of the keywords, but need not exactly match it. For instance, *[[<Animal> bass </Animal>]]* will match both highlighted instances in the sentence *Fish in the* ***<Animal> bass </Animal>*** *family include* ***<Animal> striped bass </Animal>***.

- The **relation** operation, in which the annotation represents a relation that holds between terms covered by the annotation. These relations may be syntactic, e.g., *[[<SubjectVerb> Unabomber kill </SubjectVerb>]]*, semantic, e.g., *[[<Kill> Unabomber <Person> </Person> </Kill>]]*, or pragmatic, e.g., *[[<HasNegativeOpinion> Clinton war on Iraq </HasNegativeOpinion>]]*. In addition, the XML Fragment query syntax allows for nesting of relation and entity annotations, e.g., *[[<Visit> <Person> John </Person> <Person> Victoria </Person> </Visit>]]*. This generally matches documents where John and Victoria visited one another but excludes those where John visited the city of Victoria.

Using XML Fragments to enable the conceptualization, restriction, and relation operations allows us to enhance the expressiveness of user queries and to obtain better search results. We have utilized these three operations to express four different query-time semantic needs, i.e. to specify target information type, to disambiguate keywords, to specify search term context, and to specify relations between select terms, which we discuss in detail in the following sections. Note that the ways we applied the operations place additional constraints on the queries compared to keyword queries, thus we expect them to result in more focused, higher precision queries at the possible expense of recall.[1]

[1]Another application of the *conceptualization* operation corresponds to walking up a hierarchy to generalize concepts in the query, resulting in a more recall-oriented query. We do not address this use of the operation in this paper.

4.1 Target Information Type Specification

The conceptualization operation, indicated by the XML Fragment *[[<tag></tag>]]*, enables semantic search queries that explicitly specify the semantic type of the information the user is seeking. The inclusion of the target information type ensures that each returned document contains *some* information of the target type that could potentially satisfy the user's information need. This capability is particularly useful when 1) the query consists of low *idf* keywords, and 2) the target information does not frequently co-occur with these keywords. For example, suppose the user wants to find the zip code for the White House. In a typical newswire corpus, the query *"white house"* will retrieve a large number of documents. However, few, if any, of them are likely to mention its zip code, which co-occurs with the search terms fairly infrequently. Adding *"zip code"* to the query is unlikely to improve the results as the most common context in which the correct answer is mentioned is as part of the address: The White House, 1600 Pennsylvania Avenue NW, Washington, DC 20500, without explicit indication that *20050* represents the zip code.[2] On the other hand, the query *[[+"white house" +<Zipcode></Zipcode>]]* will only retrieve documents containing both "white house" and a *Zipcode* annotation. This query will successfully retrieve documents containing the above sentence, assuming *20050* was annotated as a *Zipcode*, a fairly straightforward task given the regularity of most street address expressions.

In addition to the zip code example, target information types suitable for this operation include, for instance, *PhoneNumber*, *StreetAddress*, *URL*, and *Date* where the lexical formats of the target strings for the most part unambiguously identify their semantic types. Thus, they are often not cued by relevant keywords that would facilitate their retrieval. Without the conceptualization operation, it would be difficult, if not impossible to locate such information in a large corpus. In other cases, information may equally likely be expressed in textual description or through an established convention. For instance, one may say "John Grisham's 'The Testament' (1999) was a bestseller." Here being able to recognize "1999" as a *PubDate* enables retrieval of those documents where publication dates are expressed in this convention and which would otherwise be difficult to retrieve.

We have utilized this capability of specifying target information type in two applications. The first is the Semantic Analysis Workbench (SAW) for exploring different search paradigms, one of which supports semantic search with conceptualization capability [11]. In the SAW, the user can include desired concepts in the query to focus search either by directly typing *"<tag></tag>"* as part of the query or selecting *tag* from a dropdown list of available types. The second application is an open-domain question answering (QA) system in which the concept included in the query represents the semantic type of the answer the user is seeking [15]. In this system, the search query is automatically generated after the user's question is analyzed to determine, among other things, the answer semantic type and the salient terms in the question. For instance, the question *"What is the telephone number for the University of*

[2]In our reference corpus of one million news articles, the queries *+"white house"* combined with either *+"zip code"* or *+zipcode* yielded empty hitlists.

Kentucky?" results in the following XML Fragment query: *[[+<PhoneNumber></PhoneNumber> +"University of Kentucky"]]*, while *"How long is the Rio Grande?"* yields *[[+<Length> </Length> +"Rio Grande" long]]*.

4.2 Search Term Disambiguation

The restriction operation can be applied to disambiguate terms based on their word senses, thus focusing search toward retrieving documents more likely to match the user's information need. This operation is particularly useful when the query terms have multiple senses in the corpus and when either the senses are fairly evenly distributed or when the user intends a minority word sense. For example, the string "Victoria" can be annotated as a *City* (city name in many countries), *County* (county name in a few countries), *State* (in Australia), *Person* (as a first name), *Royalty* (British Queen), or *Deity* (Roman goddess). In order to focus search on specific interpretations of "Victoria", for instance as a city or royalty, we can use the restriction operation to disambiguate the search term by generating the query *[[<City> Victoria </City>]]* or *[[<Royalty> Victoria</Royalty>]]*, respectively. Our use of the restriction operation here can be viewed as a coarser grained application of WSD in IR adopted by earlier systems (cf. [13, 17]).

One may argue that in some cases, inclusion of extra disambiguating keywords can yield similar search results without the XML Fragment disambiguation capability. For example, the query *+city +Victoria* is likely to steer search results in the same direction as *[[<City> Victoria </City>]]* and similarly *+Queen +Victoria* may be used in lieu of *[[<Royalty> Victoria </Royalty>]]*. We agree with this observation with respect to these examples where very specific information is sought; however, we maintain that our semantic search approach offers a more desirable solution than the use of disambiguating keywords in that it 1) provides a uniform query representation, 2) eliminates the need for query-time keyword selection for disambiguation, and 3) enables higher precision query with potentially lower loss in recall. We illustrate these advantages by way of the following example. Suppose a user is interested in finding out about geographic locations named "Victoria". With XML Fragments, formulating a query involves finding the semantic type in the type system that covers all geographic entities, in this case, *GPE* (GeoPolitical Entity). This results in the query *[[<GPE> Victoria </GPE>]]* which is analogous in form to the semantic query representing the city of Victoria. On the other hand, without using the restriction operator, it is up to the user to come up with a list of geographic "units" for disambiguation and to construct a query such as *[[+Victoria +<>city state county</>]]*.[3] With semantic search, this burden of knowing what kinds of location Victoria can be is shifted to the developers of the named entity recognizers (the domain experts) and to inferences within the type system (*city*, *state* and *county* are all subtypes of *GPE*). Furthermore, requiring disambiguating keywords as in the example above may result in a loss in recall when the keywords selected by the user is incomplete or when the geographic "unit" of "Victoria" is implicitly conveyed in the document (e.g., "Victoria, British Columbia").

We employed the restriction operation for disambiguation in the same two applications described in the previous section. In the SAW, the semantic search paradigm also supports the restriction operation for disambiguation. This capability is useful in this end user application when the user enters a keyword query and, upon examining the search results, discovers that a large number of irrelevant documents are returned because a keyword is ambiguous in the target corpus. The user can then refine the original query and formulate a disambiguating XML Fragment query using the restriction operator, resulting in a set of documents that better addresses his information need. In the QA application, our named entity recognizer is invoked during the question analysis process and the restriction operation is applied to all keywords/phrases found to be potentially ambiguous. For example, for the question *"When was George Washington born?"*, the disambiguating XML Fragment query (with lemmatized keywords) *[[+<Date></Date> bear +<Person> +George +Washington </Person>]]* eliminates matches with *<College> George Washington University </College>* and *<Facility> George Washington Bridge </Facility>*.

4.3 Search Term Context Specification

In addition to being used for keyword disambiguation, the *restriction* operation can also be used to specify the context in which keywords should occur. In this case, the XML tags surrounding a keyword are not used to identify the meaning of the keyword, but are used to specify meta information which will be difficult, if not impossible, to express with keyword queries. Such meta information represents the *context* of the keyword and may include syntactic information (*Subject/Object*), semantic information (*Agent/Patient*), and discourse information (*Request/Suggest*).

There is growing interest in identifying and retrieving texts along the fact versus opinion dimension as well as the strength and polarity of opinions (cf. [23, 24]). The identification of opinion texts sets the *context* for the words appearing in these texts. With proper query support, this in turn enables the user to retrieve documents where keywords appear in particular contexts, for instance, the war on Iraq being discussed in a positive light or as part of a negative opinion about US foreign policy. A sample query to retrieve relevant documents using XML Fragments in this setting may be *[[<PositiveOpinion> "war on Iraq" </PositiveOpinion>]]*. Note that although this query is syntactically equivalent to the previous case in which XML Fragments are used for keyword disambiguation, in this case, the semantic tags are used to specify the context in which the keywords occur, and regardless of context, the phrase "war on Iraq" is unambiguous and refers to the same event.

In our work on the AQUAINT 2005 opinion pilot, the goal was to establish a baseline performance for the task without making use of opinion-specific annotations. Our approach was based on the observation that many opinions in documents are expressed as direct quotes from a person's speech, which are annotated by our existing named entity recognizer with the semantic type *Quotation*. Thus by setting the context of the subject matter to appear within a *Quotation* in the query we are targeting a select subset of all opinions regarding the subject matter, i.e., those expressed in direct speech. For example, for the following question, taken from the opinion pilot test set, *"What was the Pentagon panel's position with respect to the dispute over the US Navy training range in the island of Vieques?"*, our system generates the following XML Fragment query which includes

[3]The empty tags represent a disjunction. In this case, one of *city*, *state*, or *county* must occur in the document.

the identified opinion holder and the subject matter constrained in the context of a *Quotation*: *[[+Pentagon +panel +<Quotation> +US +Navy training range island +Vieques </Quotation>]]*. We recognize that our attempt to increase precision may result in a loss in recall, as our approach targets only opinions expressed as direct speech. We also note that we do not require that there be any connection, other than textual proximity, between the opinion holder and the subject matter, which is a potential loss in precision.[4]

4.4 Relation Specification

The relation operation can be used to express relations that must hold between query terms. These relation queries are inherently more precise than keyword queries because of this added constraint. For example, to find out about the biological weapons Iraq owns, the following XML Fragment relation query may be constructed: *[[<WeaponOwner> +Iraq +<BioWeapon> </BioWeapon> </WeaponOwner>]]*. A document matches this query if it contains a text span which is annotated as *WeaponOwner* and contains the term Iraq and an embedded text span annotated as *BioWeapon*. As would be expected from text search queries, the terms within the relation annotation are neither order nor type specific (unless specified), so *[[<Visit> Mary Sue </Visit>]]* will retrieve documents in which Mary visited Sue, Sue visited Mary, Mary Sue (one person) visited some person or place, or some person visited Mary Sue, among others.

As in the case for using the restriction operation for disambiguation, the use of the relation operation has one key advantage in that it shifts the burden of synonym expansion as a query-time process performed by the user or search engine to the relation annotator development process. For example, while it is reasonable to suggest that, without the relation operator, the user or search engine frontend may expand an initial query such as *[[+Iraq +<BioWeapon></BioWeapon> +own]]* into *[[+Iraq +<BioWeapon></BioWeapon> +<> own have possess </>]]*, it is much less likely that either would come up with a query that would retrieve the sentence *The UN weapons inspection team confirmed the existence of Iraq's anthrax stockpile*. This is because using the term *stockpile* to denote ownership is specific to this application domain, and thus we believe that the task of making the link between them explicit is best left to the domain expert involved in annotator development.

A natural question that arises with using XML Fragments to express relations is, once the relations are annotated in the corpus, why not extract all relations and their arguments and store the triplets in a database for efficient retrieval? This approach has been taken by many researchers, including [11, 7], and we argue that both approaches have their merits. While SQL queries are more precise and potentially more efficient than text search, our approach of using XML Fragments for relation queries offers more flexibility and can retrieve relevant documents/relations that SQL queries against a database recording extracted relations cannot. The key difference is in that a database records only the extracted relevant information, i.e., relations and their arguments, while the semantic search index records the begin and end offsets of relations with complete information over the text spans they cover. For example, in our system *Iraq possesses 33kg of 80 percent enriched uranium* is annotated as *<WeaponOwner> <Nation>Iraq</Nation> possesses 33kg of 80 percent <NucWeaponAgent> enriched uranium </NucWeaponAgent> </WeaponOwner>*. From the *WeaponOwner* annotation, a record for the database would contain the relation name, *WeaponOwner*, as well as its two arguments, *Iraq* and *enriched uranium*, thus losing the additional information *33kg* and *80 percent*.[5] Using XML Fragments, the following query (with lemmatized keywords) can retrieve documents, including the above example, stating that Iraq possesses uranium that is 75 percent enriched or higher: *[[<WeaponOwner> +Iraq +<.GE.> 75 </.GE.> +percent +enrich +uranium </WeaponOwner>]]*.

We have made use of the relation operation in our SAW system, where the user can formulate XML Fragment queries containing relations. Because of the additional constraint imposed by the relation, these queries typically retrieve fewer relevant documents than an otherwise identical query with the relation tag replaced by one or more keywords. This enables the user to quickly home in on a few relevant documents with little or no noise. Our QA system provides the capability of automatically generating XML Fragment queries containing relations from natural language questions. For example, the question *"When did Nixon visit China?"* generates the following relation XML Fragment query *[[<HoldsDuring> +<At> +Nixon +China </At> +<Date> </Date></HoldsDuring>]]*.

5. EVALUATION

Aside from enhancing the expressiveness of user queries, our use of the three XML Fragment operations to include target information type, to disambiguate keywords, to specify search term context, and to specify relations, results in more constrained queries. We hypothesize that these semantic search queries, even in the face of imperfect annotations, will still result in significant improvement in precision in retrieval results. This section describes our experiments for validating this hypothesis and their results for each of our four uses of XML Fragments for semantic search. Where possible, we use standard test and judgment sets in our evaluation.

5.1 Target Information Type Specification

5.1.1 Experimental Setup

In this experiment we evaluate the impact of using the conceptualization operation to specify target information type. The reference corpus is the 3GB AQUAINT corpus used in the TREC QA track since 2002, which contains just over one million news articles between 1996-2000.

The corpus is pre-processed with a named entity recognizer that identifies about 100 entity types, ranging from common types such as *Person*, *Organization* and *Date*, to specific types such as *PhoneNumber* and *URL* for addressing particular information needs. The corpus is then processed by the indexer and both lemmatized keywords and the identified semantic types are stored in the index.

[4]This issue can be remedied by specifying a relation between these two entities using the relation operation discussed in the next section. We leave developing such a relation annotator and utilizing the resulting annotations for future work.

[5]It is possible to include this information in the database by adding extra arguments to the relation. The point here, however, is that there will always be text where extra, previously unanticipated information is provided and thus not recorded in the database entry.

Table 1: Target Information Type Evaluation Results

	R-Prec	MAP	Exact Precision
Baseline	0.4219	0.4329	0.0817
w/ Target	0.4342	**0.4505***	**0.1124****

$^{*}p < 0.01$; $^{**}p < 0.001$

For our test set, we used the 50 questions and relevant judgments in the TREC 2005 QA track document task [22].[6] Each question is processed by the question analysis component of our QA system [16] to identify one or more answer types and a set of salient keywords. The keywords and answer types are used to generate an XML Fragment query, where the conceptualization operator is applied to the answer type to specify target information. For our baseline run, a query is constructed from the keywords alone. In other words, the only difference between the baseline run and the run using semantic search is the presence of the concept *[[+<tag></tag>]]* in the latter run that corresponds to the semantic type of the answer.

5.1.2 Results and Discussion

Table 1 shows the results of this experiment in which up to 1000 documents were returned for each query. In contrast to the INEX evaluation metrics [10] in which scoring is based on the relevance of an element (part of an XML document determined by the document structure), we evaluate our results at the document level because the annotations in our documents do not convey structural information. We report both the "Precision at R" score (R-Prec, where R is the number of relevant documents for each question), and the MAP score, which were the two official evaluation metrics used in the TREC 2005 QA track document task. Our results show a 2.9% and 4.1% relative improvement in R-Prec and MAP scores, respectively, where the MAP score improvement is statistically significant at the $p < 0.01$ level (using the Wilcoxon Signed Rank Test in this and subsequent evaluations). Although both of these metrics reward retrieving relevant documents and ranking them high in the hitlist, neither of them takes into account the total number of documents retrieved, which can be a critical element in an end user application. Since all of the semantic search operations discussed in this paper constrain the query and presumably lead to fewer matches for each query, we include a third evaluation metric in the table, the Exact Precision score as returned by the trec_eval script, which is the exact precision over the retrieved document set. This measure favors queries that return the same relevant documents with a smaller hitlist. Our Exact Precision score demonstrates that including target information type in the query resulted in a 37.5% improvement in precision, significant at $p < 0.001$.

5.2 Search Term Disambiguation

5.2.1 Experimental Setup

In this experiment, where we evaluate the impact of using the restriction operation to disambiguate keywords, we used the same corpus, index, and test set as in the previous experiment. Our QA system was reconfigured to generate XML Fragment queries that include, in addition to target information type, disambiguating XML tags for query terms that 1) are considered likely to be ambiguous based on a predetermined occurrence threshold, and 2) have specific desired word senses as indicated in the question. We compare this run to one without disambiguation capabilities, i.e., the outcome of the semantic search run in the previous section.

[6]In order to focus our experiment on the impact of semantic search, the anaphoric expressions in questions with referents in previous questions or answers have been manually resolved so all questions are self-contained.

Table 2: Disambiguation Evaluation Results

	R-Prec	MAP	Exact Precision
Baseline	0.4464	0.4357	0.1443
w/ Disambiguation	0.4658	0.4409	0.1443

5.2.2 Results and Discussion

In manually examining the queries generated by the two runs, we found that most terms in this standard test set were not considered ambiguous relative to the reference corpus. In fact, out of the 50 questions, only 2 questions resulted in different queries. Table 2 shows the evaluation results on this subset of questions that yielded a difference. Our results show a 4.3% improvement in R-Prec score and a slight gain in MAP score; however, in both cases, the improvement is not statistically significant due to the small sample size.

5.3 Search Term Context Specification

5.3.1 Experimental Setup

This experiment evaluates the impact of using the restriction operation to specify search term context, and is again based on the same corpus and index. The test set is 46 out of the 50 questions in the AQUAINT 2005 opinion pilot which were of the general form "What does *OpinionHolder* think about *SubjectMatter*?" Each question was processed by the question analysis component of our QA system to extract *OpinionHolder* and *SubjectMatter*, along with keywords in the question that do not fall into either category. As described in Section 4.3, the restriction operation is applied to *SubjectMatter* to constrain the context in which it should appear. In this case, the query consists of *OpinionHolder*, the additional relevant keywords, and an XML Fragment where the *Quotation* tag is wrapped around *SubjectMatter*. For each question, the system returned up to two passages each containing 1-3 sentences.

For the baseline run, instead of wrapping *Quotation* tags around *SubjectMatter*, we merely added *[[+<Quotation></Quotation>]]* as the target information type. In other words, the query still required that a direct quote be present in the document, although the subject matter does not necessarily have to be present within the quote.

5.3.2 Results and Discussion

We manually judged the two runs based on the list of "vital" and "okay" nuggets [22] used by the NIST assessors in the opinion pilot evaluation.[7] "Vital" nuggets are those that must be present in the returned passages while "okay" nuggets are considered relevant but not essential. In the opinion evaluation, an F-score is computed based on the

[7]Since some passages in our current runs were potentially previously unseen by the assessors, we augmented the assessors' list in our best attempt to reproduce the assessor's thinking, adding 20 nuggets to their list of 233 nuggets.

Table 3: Context Specification Evaluation Results

	# Vital Nuggets	# Okay Nuggets
Baseline	14	4
w/ Context	**28***	5

$^*p < 0.05$

number of "vital" nuggets returned (recall) and a length allowance determined by the total number of "vital" and "okay" nuggets returned (precision).

Since the two runs each returned two passages of similar average length per question, the final F-score is simply a function of the recall of "vital" and "okay" nuggets in each run. Given the same judgment set, we simply report the number of "vital" and "okay" nuggets returned by each run as shown in Table 3. Our results show that using semantic search to specify the context of the subject matter, we retrieved twice as many "vital" nuggets as our baseline run, thus effectively doubling the recall as well as precision.

5.4 Relation Specification

5.4.1 Experimental Setup

In this experiment, where we evaluate the impact of specifying relations between search terms, we used a corpus in the national intelligence domain from the Center for Non-Proliferation Studies (CNS) because 1) there is no standard test set for evaluating relation semantic queries, and 2) our relation annotators focused on this domain. This corpus contains over 37,000 documents and is about 75MB in size.

The corpus was pre-processed with our named entity and relation recognizers, and the resulting annotations were indexed along with the lemmatized terms as in the AQUAINT corpus. The relation recognizer identifies 10 relations in this application domain, including *ProducesWeapon* (country X produces weapon Y), *ImportExport* (country X imports/exports goods to/from country Y), and *DestructionPotential* (weapon X can kill Y people).

Our test set for this experiment targets the 10 relations relevant to the reference corpus. For each relation, we constructed semantic queries by randomly instantiating its arguments. For example, for *ProducesWeapon* we generated *[[<ProducesWeapon> +<Nation></Nation> +weapon +grade +uranium </ProducesWeapon>]]* (countries that produce weapon-grade uranium) and *[[<ProducesWeapon> +Russia +<BioWeapon> </BioWeapon> </ProducesWeapon>]]* (biological weapons produced by Russia). We constructed 2-3 semantic queries for each relation, resulting in a total of 25 queries. The baseline queries which do not make use of relations were generated by replacing the relation in each query with one or more keywords to express the semantics of the relation. For example, the baseline queries for the above two relation queries are *[[+<Nation></Nation> +weapon +grade +uranium +produce]]* and *[[+Russia +<BioWeapon> </BioWeapon> +produce]]*, respectively.

For our evaluation runs, each query returned up to 10 documents, which correspond to the typical "first page" search results that a user examines. The documents returned by each question pair in the two runs are pooled and the documents manually judged by one of the authors as relevant or irrelevant (without knowledge of which of the two runs retrieved a particular document). The pooled judgment set is used in the final evaluation for both runs.

Table 4: Relation Specification Evaluation Results

	R-Prec	MAP	Exact Prec	#docs/Q
Baseline	0.3895	0.4147	0.3530	9.72
With Relation	0.4139	0.4108	**0.6108***	**6.32****

$^*p < 0.005$; $^{**}p < 0.00005$

5.4.2 Results and Discussion

Table 4 shows our experimental results. While there is a 6.3% increase in R-Prec score and a very slight drop in MAP score, the relation queries achieved a 73% relative improvement in Exact Precision, which is significant at the $p < 0.005$ level. This result confirms our hypothesis and is expected because 1) our relation annotators were developed to favor precision over recall, and 2) the application of the relation operator constrains the query and results in a more precise hitlist. While the overall results are as expected, the statistical insignificance of the R-Prec score improvement warrants some explanation. Because both the relation query and the relation annotators targeted high precision, in a fair number of cases, this resulted in a loss in recall and hence a decrease in R-Prec scores. While the substantial gain in R-Prec scores in other questions are sufficient to result in a net improvement, the improvement is not sufficiently uniform across questions to be statistically significant.

The last column in Table 4 shows the average number of documents returned for each query. In the baseline run, the average is very close to the maximum number of documents requested, with only two queries retrieving fewer than the maximum. In the relation run, however, all but three queries returned fewer than the maximum. This result is significant in applications where it is desirable to provide the user with a very targeted, short hitlist that may only contain partial evidence for the information sought. For example, if the user wants to know whether or not Iraq has a heavy water stockpile, returning a few documents that confirm its existence may be sufficient to meet the user's information need. On the other hand, a more extensive hitlist with higher recall that details the storage facilities and quantities may be appropriate under a difference scenario. Based on these observations and our evaluation results, we believe that the relation operation is a powerful semantic search query operation that is suitable for applications in which high recall is not initially an important factor and where the cost of processing large numbers of return documents is high.

6. RELATED WORK

Semantic search is deeply related to the concept and vision of the Semantic Web. Efforts on semantic search in this context assume a yet-to-be-constructed, large, distributed semantic network containing documents pre-annotated with metadata and focus on methods for the retrieval of relevant information from such documents [7, 8].

Research in XML query and retrieval can be divided into *document-centric*, where an XML document is viewed as text with markup, and the *data-centric* view, where an XML document is just a free form database. Research in the document-centric view has focused on marrying traditional IR techniques over text spans and tree retrieval techniques over XML tags [5, 2], as well as extending document representation to support stand-off *annotations* [12] that allow for the crossing of tags. In our work, we build upon these results

to demonstrate how the XML Fragments query language can leverage a collection of annotated XML documents to address different query-time semantic needs.

To improve search performance, many researchers have investigated encoding in an index additional information such as syntactic information [18, 9, 20], word sense information [21, 13], as well as named entity information [15, 13, 20]. Although there is some conceptual overlap between our work and these previous efforts at the indexing stage, we have demonstrated innovative uses of the semantic index to significantly improve precision in search. Furthermore, none of these previous efforts attempted to operationalize an existing query language and systematically apply these operations to address different query-time semantic needs.

7. CONCLUSIONS

In this paper, we have presented our work on semantic search using the XML Fragments query language. We presented three operations that are supported by the XML Fragments query syntax: conceptualization, restriction, and relation. When applied to a traditional keyword query, each of these operations places additional constraints on the query, and thus leads to more focused and more precise results. These operations are therefore particularly valuable for applications in which the benefits of achieving high precision strongly outweighs potential loss in recall.

We further demonstrated the use of these three operators to express different query-time semantic needs. The conceptualization operation was used to specify target information type, the restriction operation was used to either disambiguate keywords or to specify search term context, while the relation operation was used to specify the relation between select terms. These uses of XML Fragment operators have been implemented and evaluated in multiple systems and, as we hypothesized, have led to significant increase in the precision of search results, thus validating their usefulness in the precision-centric class of applications.

8. ACKNOWLEDGMENTS

We would like to thank Eric Brown and Bill Murdock for helpful discussions. This work was supposed in part by the Disruptive Technology Office (DTO)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number H98230-04-C-1577.

9. REFERENCES

[1] D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder. In *Proc. 5th ANLP Conference*, 1997.

[2] A. Broder, Y. Maarek, M. Mandelbrod, and Y. Mass. Using XML to query XML – from theory to practice. In *Proceedings of RIAO*, 2004.

[3] D. Carmel, Y. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML documents via XML fragments. In *Proc. 26th SIGIR Conference*, 2003.

[4] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A semantic search engine for XML. In *Proc. 29th VLDB Conference*, 2003.

[5] N. Fuhr and K. Grosjohann. XIRQL: A query language for information retrieval in XML documents. In *Proc. 24th SIGIR Conference*, 2001.

[6] P. Grosso and D. Veillard. XML fragment interchange. W3C Candidate Recomendation 12 February 2001. http://www.w3.org/TR/xml-fragment.

[7] R. Guha, R. McCool, and E. Miller. Semantic search. In *Proc. 12th WWW Conference*, 2003.

[8] J. Heflin and J. Hendler. Searching the web with SHOE. In *AAAI Workshop on AI for Web Search*, 2000.

[9] B. Katz and J. Lin. Selectively using relations to improve precision in question answering. In *Proc. EACL Workshop on NLP for QA*, 2003.

[10] G. Kazai and M. Lalmas. INEX 2005 evaluation metrics. http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv6.pdf.

[11] A. Levas, E. Brown, J. Murdock, and D. Ferrucci. The Semantic Analysis Workbench (SAW): Towards a framework for knowledge gathering and synthesis. In *Proc. Int'l Conf. in Intelligence Analysis*, 2005.

[12] R. Mack, S. Mukherjea, A. Soffer, N. Uramoto, E. Brown, A. Coden, J. Cooper, A. Inokuchi, B. Iyer, Y. Mass, H. Matsuzawa, , and L. V. Subramaniam. Text analytics for life science using the unstructured information management architecture. *IBM Systems Journal*, 43(3), 2004.

[13] R. Mihalcea and D. Moldovan. Semantic indexing using WordNet senses. In *Proc. ACL Workshop on IR and NLP*, 2000.

[14] R. Mihalcea and D. Moldovan. Document indexing using named entities. *Studies in Informatics and Control*, 10(1), 2001.

[15] J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *Proc. 23rd SIGIR Conference*, 2000.

[16] J. Prager, J. Chu-Carroll, E. Brown, and K. Czuba. Question answering using predictive annoation. In *Advances in Open-Domain Question Answering*. Kluwer Academic Publishers, 2006.

[17] M. Sanderson. Retrieving with good sense. *Information Retrieval*, 2(1), 2000.

[18] A. Smeaton, R. O'Donnell, and F. Kelledy. Indexing structures derived from syntax in TREC-3: System description. In *Proc. 3rd TREC*, 1995.

[19] R. Srihari, W. Li, C. Nui, and T. Cornell. InfoXtract: A customizable intermediate level information extraction engine. *Journal of Natural Language Engineering*, 2006.

[20] J. Tiedemann. Integrating linguistic knowledge in passage retrieval for question answering. In *Proc. HLT/EMNLP Conference*, 2005.

[21] E. Voorhees. Using WordNet to disambiguate word sense for text retrieval. In *Proc. SIGIR*, 1993.

[22] E. Voorhees and H. Dang. Overview of the TREC 2005 question answering track. In *Proc. TREC*, 2006.

[23] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin. Learning subjective language. *Computational Linguistics*, 30(3), 2004.

[24] H. Yu and V. Hatzivassilogou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proc. EMNLP Conference*, 2003.