# Experiments in Questions and Relationships at The University of Iowa

David Eichmann,[1,2] Padmini Srinivasan,[1,3]

[1]School of Library and Information Science

[2]Computer Science Department

[3]Department of Management Sciences

The University of Iowa

{david-eichmann, padmini-srinivasan}@uiowa.edu

The University of Iowa participated in the genomics and question answering tracks of TREC-2005. This paper covers only our work in question answering.

## 1 – Overview

Our general approach to question answering is one that attempts to front-load much of the evaluation of corpus semantics, allowing for a more search engine-like user experience. By shifting natural language parsing forward in the process, we can amortize this very expensive step against a number of downstream extraction processes that mine the text for named entities, relationships, etc. Redefinition of extraction specifications hence does not require reparsing of the source text. As for last year, we use a tgrep-like extraction grammar designed for predicate-based extensibility using it in mapping sentence parse trees to relational structure. This overall approach handles not only factoid answers, but definitional answers and those requiring inference across multiple extracted relationships.

Each document in the corpus is decomposed into doc-id / sentence pairs, with the sentence being the unit of analysis from that point. Each sentence is then POS-tagged and fed to the CMU link grammar parser. The parse tree for the sentence is then attributed with the POS tags for each word. Processing both queries and documents using this scheme allows us to establish both the nature of the query (using a fairly typical taxonomy) and the nature of the needed answer. This is particularly useful with respect to identification of candidate phrases in sentences and scoring of these phrases against the goal of the query. Sentences are then matched against the set of extraction patterns, populating a set of relations used to answer queries derived from the questions.

The availability of the parse tree for the phrase allows for elision of subordinate clauses that can cause answers to span too long a string and for extraction of likely answers through heuristic matching of, for example, a subordinate clause immediately trailing a mention of a candidate named entity.

### Promotion Patterns

A significant benefit of our syntactic approach to relationship extraction is the ability to detect entities that have a relationship, but are buried deep within separate syntactic contexts. Consider the sentence "Fred Smith, who is married to Jane Smith (the current CEO of Acme Inc.), today was named CFO of the biggest banking firm of southern Lithuania, Moneygrubbers Holding Corp." The primary relationship of the sentence involves the named entities at opposite ends of the sentence, and there are three intervening named entities separating them. However, the upper syntactic structure of the sentence is just a noun phrase (Fred Smith and a SBAR involving him) followed by a verb phrase (characterizing Fred's new role with his employer).

Our first phase in relationship extraction is hence an attribute grammar based promotion of named entity tagging of the syntax tree, supporting the recognition of lexically-distant connections. Our promotion system is specification driven. An input file first defines both common and named entities to be promoted:

```
# common entities
entity: person
entity: organization
...
# named entities (classes available for instantiation)
entity: edu.uiowa.entity.Person
entity: edu.uiowa.entity.Organization
...
```

A set of generic promotion rules are usually then provided, which will apply to all defined entities. Each rule involves a grammar fragment, optionally constrained with entity tags and preceded by a clause number indicating which entity tag is to be promoted to the containing clause:

```
# generic
1 [NP NN:* ]
```

```
1 [NP [NP:* ] [PP ] ]
1 [NP [NP:* ] , [SBAR ] ]
3 [NP DT JJ NN:* ]
```

For example here, the third rule promotes the entity tag of the first noun phrase of a containing noun phrase when the containing NP is comprised of a NP, a comma and a SBAR (a form of subordinate phrase). Hence Fred's entity tag is now associated with the outermost NP of the sentence. More specific promotions are readily done by additional constraints:

```
# persons
1 [NP [NP:edu.uiowa.entity.Person ] , [NP:person ] ]
5 [NP NNP NNP JJ NN:organization NN:person ]
...
# organizations
1 [NP [NP:edu.uiowa.entity.Organization ] , [NP ] , ]
...
# place names
1 [NP NNP:edu.uiowa.entity.PlaceName , [PP IN [NP ] ] ]
```

We have found that limiting the pattern specification to three levels of the syntax tree, the containing clause and the immediate two levels below it, appears quite sufficient to capture the necessary semantics as the promoter algorithm recursively walks the tree bottom-up.

**Extraction Patterns**

Once the containing entity tags have been promoted up the syntax tree, we invoke our extraction pattern recognition engine on the root of the tree. This engine was based originally on a number of pattern recognition tools collectively known as tgrep. We have implemented all documented tgrep functions in our engine and have additionally implemented both regular expression matching of nodes and reflection-based runtime specification of predicate functions. These enhancements allow for an extensible framework for domain-specific semantic constraints to be imposed on sentence structure during the extraction of entities and their relationships.

The definition of extraction patterns is done similarly to the promotion patterns, via a specification file. The first set of definitions establishes the name space for available predicates and the implementations they are to be dynamically bound to:

```
<comparator>
  <functor>isPerson</functor>
  <class>edu.uiowa.syntaxMatch.personComparator</class>
</comparator>
<comparator>
  <functor>isDirection</functor>
  <class>edu.uiowa.syntaxMatch.directionComparator</class>
</comparator>
```

One or more pattern sets are then provided, each set comprising of one or more pattern definitions and mappings to an extraction table:

```
<pattern-set>
  <pattern>
    <patString>NP <1(NNP) <2[PP <1(/in|of/) <(NP)]</patString>
  </pattern>
  <pattern>
    <patString>NP <1(NNP) <2(NNP) <3[PP <1(/in|of/) <(NP)]</patString>
  </pattern>
</pattern-set>
```

This pattern set matches a noun phrase (the first NP in each patString) comprised of either one or two NNPs (proper nouns) in positions one and two, immediately followed by a PP (prepositional phrase) which must begin with 'in' or 'of' immediately followed by a NP. Parenthesis indicate fragments to be extracted from the tree - in this case the NNPs, the prepositional term and the contained NP.

The above patterns are basically stock tgrep notation. An example of the power allowed by our extension features can be seen in this pattern set:

```
<pattern-set>
  <pattern>
    <patString>(NP) $. [VP <1/is/ <2[PP <[NP <1(NP)
                    <2[PP <<(isDirection(/NN|RB/)) <<(NP)]]]]</patString>
  </pattern>
  <pattern>
    <patString>PP <2[NP <1(NP) <3[NP <<(NP </miles|kilometers/)
                       <[PP <(isDirection(/NN|RB/))
                            <<(NP !<</miles|kilometers/)]]]</patString>
```

```
      </pattern>
    </pattern-set>
```

which extracts two entities and their relative position relationship.

New domains are readily retargeted with the following methodology:

1. Parse and entity tag a sample corpus.
2. Generate three-level syntax fragments (as show in the pattern promotion examples).
3. Cluster the syntax fragments into equivalence classes.
4. In decreasing frequency order, check the actual text of the fragments for 'interestingness'; for those useful in relationship declaration; generating an appropriate extraction pattern for each equivalence class.

This approach maximizes payoff on high frequency patterns, while minimizing the effort of the human pattern specifier, since they now only need to inspect samples from the equivalence classes.

## 2 – Main Task

Our main task runs for this year were done with a set of 200 extraction rules, which we estimate to be perhaps only one percent of the probable pool of result-generating rules (i.e., those that populate the database with information that eventually is bound to some question in this class). Hence our pattern sets are insufficiently rich to provide sufficient coverage of potential questions, and hence the number of correct answers we generate is modest. However, as shown in Table 1, there is interesting potential in the low levels of unsupported and inexact answers relative to correct answers. When we do generate an answer, that answer is responsive.

Note here that as a design choice, we do not return a 'guess' – NIL is returned any time that the system fails the retrieve a match to the generated database query that corresponds to the question. This has the obvious side effect of resulting in a comparatively high level of NIL answer recall, particularly given our level of correct answers. As we populate the framework with additional extraction rules, this approach will more closely model a semantics of 'there doesn't seem to be an answer.'

**Table 1: QA Track, Main Task**

| Run | Factoid | | | | | | List | Other |
|---|---|---|---|---|---|---|---|---|
| | U | X | R | Accuracy | # NIL returned | NIL P | Avg. F | Avg. F |
| UIowaQA0501 | 0 | 1 | 17 | 0.047 | 317 | 0.050 | 0.000 | 0.012 |
| UIowaQA0502 | 0 | 1 | 15 | 0.041 | 309 | 0.045 | 0.000 | 0.014 |
| UIowaQA0503 | 0 | 1 | 17 | 0.047 | 318 | 0.050 | 0.000 | 0.015 |

## 3 – Document Ranking Task

Our implementation for the (required) document ranking task was extremely simplistic. For each question, the unnormalized occurrence count for query terms appearing in documents was used to rank those documents in descending order. Given our completely sentential approach for the main task, we had to construct something to generate this output, and didn't want to devote much effort to it

Tables 2, 3 and 4 show our overall performance, interpolated recall - precision averages and precision at N results. Interestingly, this very simple approach frequently is at or above the median for participating systems. Either few teams did much with this task, or it is very difficult to get off base-line performance for the task.

**Table 2: QA Track, Document Ranking**

| Category | Count |
|---|---|
| Retrieved Documents | 29201 |
| Relevant | 1575 |
| Relevant Returned | 1127 |
| Average Precision | 0.1467 |
| R-Precision | 0.1612 |

**Table 3: Document Ranking Interpolated Recall - Precision Averages**

| Recall At | Interpolated Precision |
|---|---|
| 0.00 | 0.4192 |
| 0.10 | 0.3355 |
| 0.20 | 0.2832 |
| 0.30 | 0.2056 |
| 0.40 | 0.1798 |
| 0.50 | 0.1654 |
| 0.60 | 0.0786 |
| 0.70 | 0.0591 |
| 0.80 | 0.0476 |
| 0.90 | 0.0266 |
| 1.00 | 0.0241 |

**Table 4: Document Ranking Precision**

| at N docs | Precision |
|---|---|
| 5 | 0.1640 |
| 10 | 0.1440 |
| 15 | 0.1507 |
| 20 | 0.1400 |
| 30 | 0.1213 |
| 100 | 0.0846 |
| 200 | 0.0634 |
| 500 | 0.0386 |
| 1000 | 0.0225 |

## 4 – Relationship Task

We approached the relationship task as a sentence retrieval task, on the assumption that sentence length in newswire documents was not so great as to create a problem with the length penalty aspects of the scoring algorithm. We reformulated each query as a keyword or prefix that was used to retrieve a set of sentences comprising the documents containing that string. Each sentence was then pattern-matched against a set of regular expressions and any matching sentence was retrieved as relevant.

Table 5 shows each keyword, set of patterns and the corresponding results. While our approach can fail completely even with a complex query (e.g., Qid 4), it can also be surprisingly successful with rather modest queries (e.g., Qids 7 & 13). Furthermore, in a number of cases it achieves very high recall. As shown in Figure 1, recall spreads over a wide range, with not unexpected low precision. A key focus for failure analysis will be whether the overall response length can be substantially reduced by limiting the number of sentences retrieved, and still have recall maintain its current level. Plotting response length against precision, as shown in Figure 2, results in a distribution as would be expected for a simple precision/recall trade-off. However, the second graph in Figure 2 shows that reasonable recall performance occurs across a rather wide range of response lengths. We see this as an interesting window of opportunity for performance tuning.

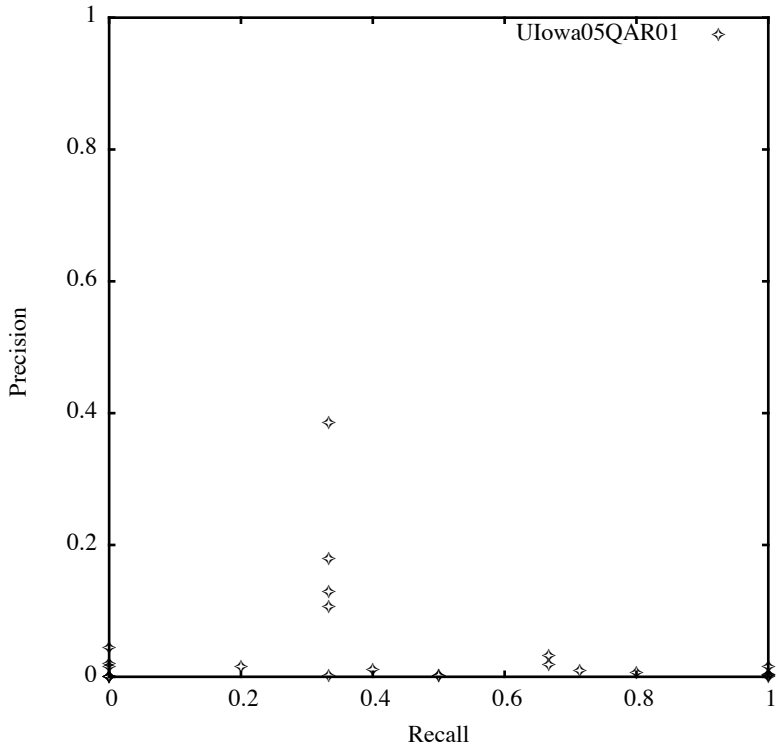


**Figure 1: Relationship Performance**

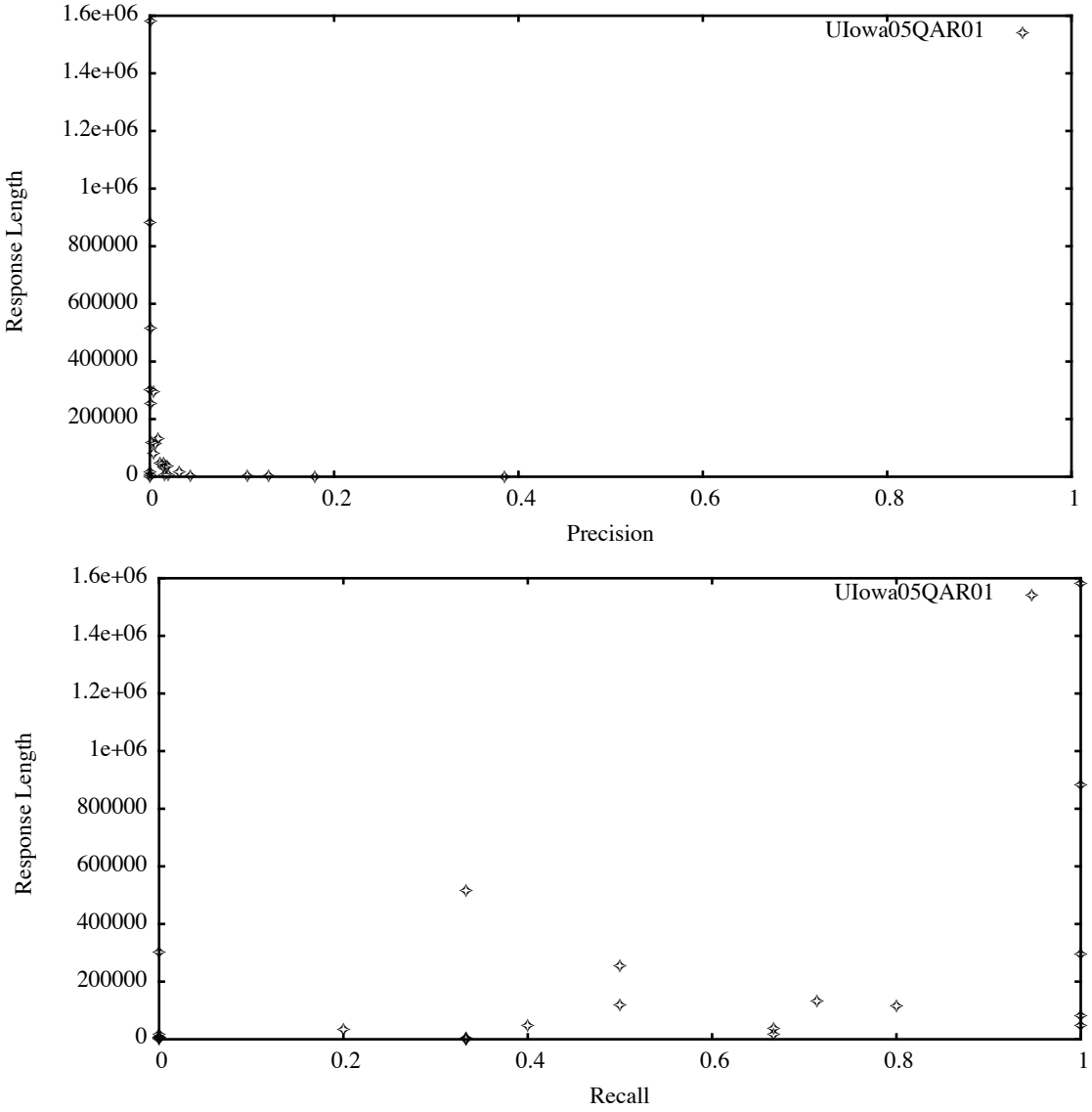


**Figure 2: Relationship Length vs. Precision and Recall**

**Table 5: Relationship Task**

| Qid | Query Word | Sentence Pattern | Vital Matches | Total Matches | Response Length | Recall | Prec. | F |
|---|---|---|---|---|---|---|---|---|
| 1 | qaeda | %Laden%<br>%Al Qaeda% | 5/7 | 12/18 | 131123 | 0.714 | 0.009 | 0.082 |
| 2 | columbi% | %arms%Columbi%<br>%Columbi%arms% | 0/3 | 0/6 | 17682 | 0.000 | 0.000 | 0.000 |
| 3 | bonaire | %Bonaire% | 0/3 | 1/8 | 2296 | 0.000 | 0.044 | 0.000 |
| 4 | mercosur | %Andean Community%Common Market%<br>'%Andean Community%MERCOSUR%<br>%CAN%Common Market%<br>%CAN%MERCOSUR%<br>%Common Market%%Andean Community<br>%MERCOSUR%Andean Community%<br>%Common Market%CAN%<br>%MERCOSUR%CAN% | 0/3 | 0/13 | 149 | 0.000 | 0.000 | 0.000 |
| 5 | liechtenstein | %cartel% | 1/3 | 1/6 | 560 | 0.333 | 0.179 | 0.307 |
| 6 | chech% | %support% | 1/3 | 3/15 | 515579 | 0.333 | 0.001 | 0.006 |
| 7 | auc | %AUC% | 1/3 | 5/7 | 3867 | 0.333 | 0.129 | 0.288 |
| 8 | angola | %Cuba%Angola%<br>%Angola%Cuba% | 0/2 | 1/9 | 5019 | 0.000 | 0.020 | 0.000 |
| 9 | congo | %Cuba%Congo%<br>%Congo%Cuba% | 1/3 | 3/5 | 2819 | 0.333 | 0.106 | 0.275 |
| 10 | ecuador | %drug%Ecuador%<br>%Ecuador%drug% | 0/1 | 1/6 | 6357 | 0.000 | 0.016 | 0.000 |
| 11 | falklands | %Argentin%Britain%<br>%Britain%Argentin% | 2/3 | 5/9 | 15635 | 0.667 | 0.032 | 0.223 |
| 12 | farc | %Venez%<br>%Brazil% | 2/3 | 7/10 | 36564 | 0.667 | 0.019 | 0.152 |
| 13 | exocet | %Exocet% | 2/6 | 3/7 | 779 | 0.333 | 0.385 | 0.338 |
| 14 | smuggl% | '%smugg%Iraq%<br>%Iraq%smugg% | 5/5 | 7/10 | 47073 | 1.000 | 0.015 | 0.131 |
| 15 | weapon% | %Chin%Taiwan%<br>%Taiwan%Chin% | 5/5 | 11/11 | 295374 | 1.000 | 0.004 | 0.036 |
| 16 | india% | '%Israel% | 3/3 | 4/6 | 883239 | 1.000 | 0.000 | 0.005 |
| 17 | isra% | %Chin% | 6/6 | 14/16 | 1582522 | 1.000 | 0.001 | 0.009 |
| 18 | india% | %Isra%nuclear%<br>%nuclear%Isra% | 2/5 | 5/8 | 46072 | 0.400 | 0.011 | 0.087 |
| 19 | assad | %Rifaat% | 1/5 | 5/9 | 32667 | 0.200 | 0.015 | 0.091 |
| 20 | leban% | %Army% | 0/1 | 1/2 | 303041 | 0.000 | 0.000 | 0.000 |
| 21 | mitch | %Hurricane% | 1/2 | 3/7 | 253445 | 0.500 | 0.001 | 0.012 |
| 22 | oil-for-food | %oil-for-food% | 4/5 | 7/15 | 114616 | 0.800 | 0.006 | 0.057 |
| 23 | proliferation | %South% | 1/2 | 2/8 | 117542 | 0.500 | 0.002 | 0.017 |
| 24 | andres | %San Andres% | 0/3 | 0/7 | 5407 | 0.000 | 0.000 | 0.000 |
| 25 | castro | %brother% | 2/2 | 3/5 | 81942 | 1.000 | 0.004 | 0.035 |
| | | | | | | | Average F | 0.086 |