Information Extraction Supported Question Answering[*]

Rohini Srihari and Wei Li

Cymfony Inc.
5500 Main Street
Williamsville, NY 14221, U.S.A.

rohini@cymfony.com, wei@cymfony.com
phone: (716) 565-9114 fax: (716) 565-0308

15 October, 1999

Abstract

This paper discusses the use of our information extraction (IE) system, Textract, in the question-answering (QA) track of the recently held TREC-8 tests. One of our major objectives is to examine how IE can help IR (Information Retrieval) in applications like QA. Our study shows: (i) IE can provide solid support for QA; (ii) low-level IE like Named Entity tagging is often a necessary component in handling most types of questions; (iii) a robust natural language shallow parser provides a structural basis for handling questions; (iv) high-level domain independent IE, i.e. extraction of multiple relationships and general events, is expected to bring about a breakthrough in QA.

1 Introduction

Natural language QA (Question Answering) is an ideal test bed for demonstrating the power of IE (Information Extraction). In our vision, there is a natural co-operation between IE and IR (Information Retrieval).

An important question then is, what type of IE can support IR in QA and how well does it support it? This forms the major topic of this paper. We structure the remaining part of the paper as follows. In Section 2, we first give an overview of the underlying IE technology that Cymfony has been developing. We then present in Section 3 the use of this technology to implement the prototype for the QA Track. In Section 4, we examine question types and discuss their relationship with IE tasks. Finally, in Section 5, we propose a more sophisticated QA system supported by 3 levels of IE.

[*] This work was supported in part by the following grants from the Air Force, Rome Laboratories: AFRL/IFKRD Phase 2 (Contract No. F30602-98-C-0043) and AFRL/IFKRD Phase 1 (Contract No. F30602-99-C-0102).

2 Overview of Textract IE

The last decade has seen great advances and interest in the area of IE. In the US, the DARPA sponsored Tipster Text Program [Grishman 1997] and the Message Understanding Conferences (MUC) [MUC-7 1998] have been the driving force for developing this technology. In fact, the MUC specifications for various IE tasks have become *de facto* standards in the IE research community. It is therefore necessary to present our IE effort in the context of the MUC program.

MUC divides IE into distinct tasks, namely, NE (Named Entity), TE (Template Element), TR (Template Relation), CO (Co-reference), and ST (Scenario Templates) [Chinchor & Marsh 1998]. Our proposal for three levels of IE is modeled after the MUC standards using MUC-style representation. However, we have modified the MUC IE task definitions in order to make them more useful and more practical.

More precisely, we propose a *hierarchical*, three-level architecture for developing a kernel IE system which is *domain-independent* throughout.

In fact, for level-1 IE, Cymfony has already developed *Textract 1*.0, a state-of-the-art NE tagger [Srihari 1998]. *Textract 1.0* has obtained a score of 91.24% in combined precision and recall (i.e. F-measures), when tested on the MUC-7 dry run data using the MUC-provided scorer. Our tagging speed, approximately 100 MB/hour on a Pentium system, is also comparable to that of the few deployed NE systems, like NetOwl [Krupka & Hausman 1998] and Nymble [Bikel et al 1997].

It is to be noted that, in our definition of NE, we significantly expanded the type of information to be extracted. In addition to all the MUC defined NE types (*person, organization, location, time, date, money* and *percent*), the following entities are also identified by our existing NE tagger:

- duration, frequency, age
- number, fraction, decimal, ordinal, math equation
- weight, length, temperature, angle, area, capacity, speed, rate
- product, software
- address, email, phone, fax, telex, www
- name (default, i.e. proper name which does not belong to any of the above category)

Sub-type information like *company, government agency, school* (belonging to the type *organization*) and *military person, religious person* (belonging to *person*) are also identified. These new types or sub-types of named entities provide a better foundation for defining multiple relationships between the identified entities and for supporting question answering functionality. For example, the key to a question processor is to identify the **asking point** (*who, what, when, where,* etc.). In many cases, the asking point corresponds to an NE beyond the MUC definition, e.g. the *how*-type questions: *how long* (*duration* or *length* depending on the question context), *how far* (*length*), *how often* (*frequency*), *how old* (*age*), etc. Therefore, an extended NE tagset is helpful for sophisticated IE and QA.

Leve-2 IE, or CE (Correlated Entity), is concerned with extracting *pre-defined* multiple relationships between the entities. This represents a giant step forward from existing deployed IE systems such as *NetOwl, IdentiFinder* [MUC-7 1998] as well as *Cymfony Textract 1.0*, which only output isolated named entities. With CE extraction, salient information is made available to a user since individual, isolated named entities are inter-related. Cymfony has recently implemented a CE prototype. Consider the *person* entity for example; our CE prototype *Textract 2.0* is capable of extracting the following key relationships:

- **name:** including aliases
- **title:** e.g. Mr.; Prof; etc.
- **subtype:** e.g. MILITARY; RELIGIOUS; etc
- **age:**
- **gender**: e.g. MALE; FEMALE
- **affiliation**:
- **position:**
- **birth_time:**
- **birth_place:**
- **spouse:**
- **parents:**
- **children:**
- **where_from:**
- **address:**
- **phone:**
- **fax:**
- **email:**
- **descriptors:**

As shown, the information in the CE represents a mini-CV of the person. In general, our CE template integrates and greatly enriches the information contained in MUC TE and TR. In terms of relationships, there are only a couple of relationships (*employee_of, location_of*) defined in MUC TR.

The final goal of our IE effort is to further extract *open-ended* general events (GE, or level 3 IE) for information like *WHO* did *WHAT* (to *WHOM*) *WHEN* and *WHERE*. By general events, we refer to argument structures centering around verb notions plus the associated information of time and location. GE is dramatically different from the MUC ST task because it is open-ended and domain independent, while ST is pre-defined and highly domain dependent.

Currently, Cymfony is n the stage of research and prototype implementation for this high level IE technology. We show an example of our defined GE extracted from the text (MUC-7 data) below:

> *Julian Hill, a research chemist whose accidental discovery of a tough, taffylike compound revolutionized everyday life after it proved its worth in warfare and courtship, died on Sunday in Hockessin, Del.*

```
[1]     <GE_TEMPLATE> :=
        PREDICATE:        die
        ARGUMENT1:        Julian Hill
        TIME:             Sunday
        LOCATION:         Hockessin, Del.
```

We show below the overall system architecture for the IE system *Textract* Cymfony has been developing.

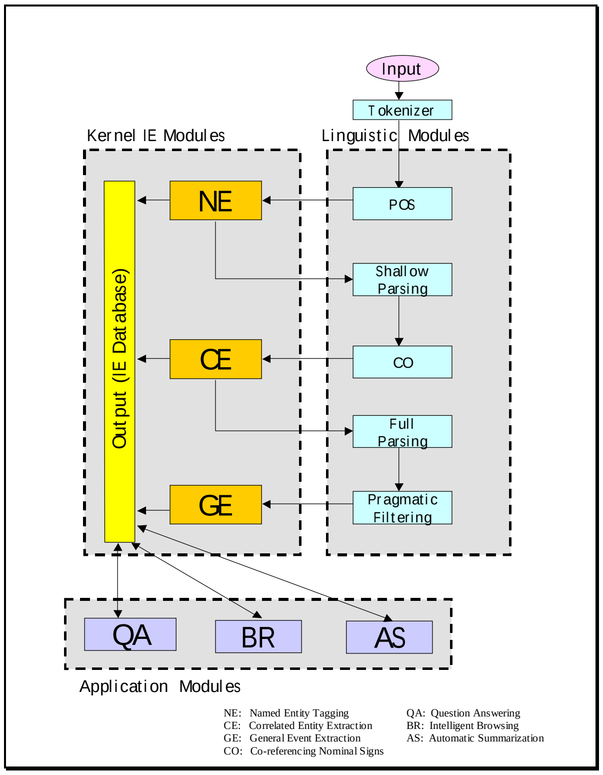

Figure 1: Textract IE System Architecture

As shown, the core of the system consists of 3 kernel IE modules and 6 linguistic modules. These modules remain domain independent. The multi-level linguistic modules serve as an underlying support system for different levels of IE. The IE results are stored in a database which is the basis for IE-related applications like QA, BR (Intelligent Browsing and threading), and AS (Automatic Summarization).[1]

It can be shown that each level of IE has immediate application in enhancing the function of IR systems. For example, NE, CE, and GE all support QA. In other words, we do not need to wait until a more sophisticated module is completed before we can try to port our component technology to QA. This is what we have done in TREC QA by employing only NE and a Shallow Parser.

Table 1 gives a concise comparison of the IE task definitions in our architecture and those defined in MUC.

[1] More precisely, the CE technology supports intelligent browsing/threading in the sense that a viewer has access to the collected information about any entity (CE) he is interested in and can jump freely between inter- related entities. The GE results can be used as basis for automatic summarization. When it is adapted to a particular domain, *Textract* also has a wide variety of application potential (e.g. intelligence, police archive, commercial banking, medical records, classified ads, personalized news abstraction, etc.). However, we believe the key application area is information retrieval.

Our modules	Corresponding MUC definitions	Remark
NE	NE	more NE types defined
CE	TE, TR	more TR types defined in CE
GE	ST	substituting domain dependent ST by our domain independent GE
CO	CO	our CO allows for non-deterministic output

Table 1: Comparison of IE Task Definitions

Our approach to IE consists of a unique blend of machine learning and FST (finite state transducer) rule-based system [Srihari 1998]. By combining machine learning with an FST rule-based system, we are able to exploit the best of both paradigms while overcoming their respective weaknesses.

3 NE-Supported QA

This section presents our approach to QA based on Named Entity tagging. We used our NE tagger *Textract 1.0* for the TREC-8 QA Track and obtained very encouraging results of 66.0% accuracy.

An analysis showed that over 80% out of the 200 questions asked for an NE as a response, e.g. *who* (PERSON), *when* (TIME | DATE), *where* (LOCATION), *how long* (DURATION | LENGTH), *how far* (LENGTH). Therefore, our NE tagger has been proven to be very helpful. For example, in answering questions like "*when did something happen*", the answer should at least contain an NE of time (or date). Questions of the type "*who did this*" usually require an answer containing an NE of person. Of course, the NE of the targeted type is only *necessary* but not complete in answering such questions because NE by nature only extracts isolated individual entities from the text. More sophisticated technology like CE or GE is required in locating the targeted answer. Since the CE prototype *Textract 2.0* was not yet available; we were unable to use this new technology in enhancing our QA performance. Nevertheless, using even simplistic methods like "the nearest NE to the queried key words" or "the NE and its related key words within the same line (or same paragraph, etc.)", NE can help narrow down the targeted text portions which contain potential answers. This is the basic strategy which we employed for our QA prototype.

Would it be possible to achieve the same effect using a traditional search engine without NE support? The answer is largely negative because it is difficult to form a query pertaining to the information about *when* or any NE in general. Although most search engines allow users to form some types of queries using wild cards for pattern matching, they still do not help very much in locating the targeted NE (and associating the NE with something else). This is due to many types of NE (except, perhaps, the NE of *percent*) being expressed in many variable forms in natural language, in fact many more varieties than one usually expects. Even a sophisticated user may not able to readily come up with good patterns to cover the great number of varieties of NE when forming a query to the system for an answer.[2]

Figure 2 illustrates the system design of Textract/QA Prototype. There are two components for the QA prototype: Question Processor and Text Processor. The Text Matcher module links the two processing results and tries to find answers to the processed question. Matching is based on keywords, plus the NE type (if the question pertains to an entity) and their common location within a same sentence.

[2] In addition, not all types of NE can be captured by pattern matching effectively. A considerable number of NEs of *person*, *organization* and *location* appear in texts with no obvious surface patterns to be captured. That is exactly the rational behind our hybrid approach to IE combining pattern matching rules and statistical learning [Srihari 1998].

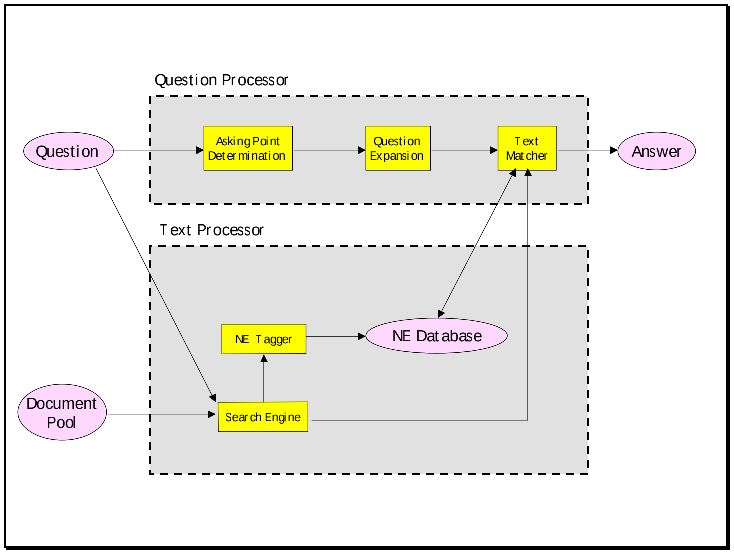

Figure 2: Textract/QA 1.0 Prototype Architecture

Question Processing

The Question Processor consists of two modules: Asking Point Determination and Question Expansion. The results are a list of keywords plus the information for asking point. For example, the question:

[2] *Who won the 1998 Nobel Peace Prize?*

contains the following keywords: *won, 1998, Nobel, Peace, Prize*. The **asking point** *Who* refers to the NE type *person*, which is determined by the module Asking Point Determination. The output before question expansion is a simple 2-feature template as shown below:

[3]	*asking_point*:	PERSON
	key_word:	{won, 1998, Nobel, Peace, Prize}

The following is an example where the asking point does not correspond to any type of NE in our definition:

[4] *Why did the Cultural Revolution occur in China?*

The system then maps it to the following question template:

[5]	*asking_point*:	REASON
	key_word:	{occur, Cultural, Revolution, China}

Basically, the module Asking Point Determination scans the question to search for question words (*wh*-words) and maps them into corresponding NE types/sub-types or pre-defined notions like REASON. We adopt two sets of pattern matching rules for this purpose: (i) structure based pattern matching rules; (ii) simple key word based pattern matching rules (regarded as default rules).

The first set of rules are based on shallow parsing results of the questions using our existing FST based *Textract English Shallow Parser*; the same shallow parser is also used in our IE system, as shown in Figure 1. This parser identifies basic syntactic constructions like BaseNP (Basic Noun Phrase), BasePP (Basic Prepositional Phrase) and VG (Verb Group).

The following is a sample of the first set of rules:

[6] Name NP(city|country|company|mountain|river)
--> CITY|COUNTRY|COMPANY|MOUNTAIN|RIVER

[7] Name NP(person_w) --> PERSON

[8] Name NP(org_w) --> ORGANIZATION

[9] Name NP(NOT person_w, NOT org_w) --> NAME

Rule [6] checks the head word of the NP. It covers cases like *VG[Name] NP[a country] that VG[is developing] NP[a magnetic levitation railway system]*. Rule [7] works for cases like *VG[Name] NP[the first private citizen] VG[to fly] PP[in space]* as *citizen* belongs to the word class *person_w*. Rule [9] is a catch-all rule: if the NP is not of class person (*person_w*) or organization (*org_w*), then the asking point is a proper name (default NE), often realized in English as a capitalized string of words. Examples include *Name a film that has won the Golden Bear in the Berlin Film Festival*.

We used the following pattern transformations to expand our ruleset:

(Please) name NP[X]
--> what/which Aux(be) (the name of) NP[X]
--> NP(what/which...X)

In other words, the 4 rules are expanded to 12 rules. For example, Rule [10] below corresponds to Rule [6]; Rule [11] is derived from Rule [7]:

[10] what/which Aux(be) NP(city|country|company|mountain|river)
--> CITY|COUNTRY|COMPANY|MOUNTAIN|RIVER

[11] NP(what/which ... person_w) --> PERSON

Rule [10] extracts the asking point from cases like *NP[What] Aux[is] NP[the longest river] PP[in the United States]*. Rule [11] covers the following questions: *NP[What costume designer] VG[decided] that NP[Michael Jackson] VG[should only wear] NP[one glove]*, *NP[Which former Ku Klux Klan member] VG[won] NP[an elected office] PP[in the U.S.]*, *NP[What Nobel laureate] VG[was expelled] PP[from the Philippines] PP[before the conference] PP[on East Timor]*, *NP[What famous communist leader] VG[died] PP[in Mexico City]*, etc.

As seen, shallow parsing helps us to capture a variety of natural language question expressions. However, there are cases where some simple key word based pattern matching would be enough to capture the asking point. That is our second set of rules. These rules are used when the first set of rules have failed to produce results. The following is a sample of such rules:

[12]	who/whom -->	PERSON
[13]	when -->	TIME/DATE
[14]	where/what place -->	LOCATION
[15]	what time (of day) -->	TIME

[16]	what day (of the week) -->	DAY
[17]	what/which month -->	MONTH
[18]	what age/how old -->	AGE
[19]	what brand -->	PRODUCT
[20]	what -->	NAME
[21]	how far/tall/high -->	LENGTH
[22]	how large/big/small -->	AREA
[23]	how heavy -->	WEIGHT
[24]	how rich -->	MONEY
[25]	how often -->	FREQUENCY
[26]	how many -->	NUMBER
[27]	how long -->	LENGTH/DURATION
[28]	why/for what -->	REASON

At times, the rule keeps ambiguity unresolved and produces non-deterministic output for the asking point. For example, *how long* maps to NE type *duration* | *length*; the Text Matcher will attempt to find sentences which contain either an NE of *duration* or an NE of *length* and at the same time maximally satisfying the key word based matching constraints. Obviously, this is a naive approach; however, this simple approach is more robust and has proven to be very helpful in our TREC experiments.

The second module in the Question Processor is Question Expansion (corresponding to *query expansion* in IR). Currently, this module is a simple lexical lookup procedure. It attempts to find synonyms to the key verbs (plus their morphological variants) and/or synonyms to the non-NE notion for the asking point. After expansion, the template for the template [3] would be:

```
[29]    asking_point:  PERSON
        key_word:      {won |win | gain | gained | get | got | acquire | acquired |
                        obtain | obtained,
                        1998, Nobel, Peace, Prize}
```

The template corresponding to [5] is expanded to the one shown below:

```
[30]    asking_point:  {because | because of | due to | thanks to | for the reason  |
                        since | as | for | in order to |  in order that | to VB
                       }
        key_word:      {occur | occurred | happen | happened | take place | took place,
                        Cultural, Revolution, China}
```

The last item in the *asking_point* list attempts to find an infinitive by checking the word *to* followed by a verb (with the part-of-speech tag VB). As we know, infinitive verb phrases are often used in English to explain a reason for some action.

Text Processing

On the text processing side, we first send the question directly to a search engine in order to narrow down the document pool to the first *n*, say 200, documents for NE processing. We employed the TREC-supplied search engine results from the AT&T engine for this purpose.

The Text Matcher attempts to match the question template with the processed documents for both the asking point and the key words. There is a preliminary ranking algorithm built-into the matcher in order to find the most probable answers. The primary rank is a count of how many unique keywords are contained within a sentence. The secondary ranking is based on the order that the keywords appear in the sentence compared to their order in the question. The third ranking is based on whether there is an exact match or a variant match for the key verb.

Finally, we implemented a procedure to filter the extracted sentences representing potential answers to a string of 50 bytes or less, as required by the QA Track. This procedure ensures that the output must contain the information, NE or other key words, determined by the asking point.

Cymfony Textract/QA participated in TREC QA in the category of submitting text strings less than 50 bytes. Our accuracy was 66.0%. Considering we have only used NE technology to support QA in this run, 66.0% is a very encouraging result. There is considerable room for performance enhancement. We believe that high level IE like our defined task of CE and GE are more helpful in QA (see 5 for discussion).

4 IE Results as Answers to Questions

We study general question types and how they are related to IE in this section. This suggests our future research direction in terms of using IE for QA. We show that our proposed hierarchical IE system covers a wide range of questions which are not supported by the conventional MUC-defined IE tasks

An IE system performing MUC tasks is helpful in performing the QA function. For example, in answering questions like *where is some company located*, the TR information (*location_of*) provides a ready answer. The identification of different types of NE is also helpful in raising the retrieval hit rate in performing QA. However, there are many more frequently asked questions where MUC IE is of little help. These questions are of two major types, as shown below.

Type I: ask about X's Y (X is an entity, Y is its one aspect/feature, including relationship), e.g.

What is X's address, email and telephone number?
What is X's profession?
Who (or Which organization) is X's employer?
Who is X's wife?
Where is the company X located?
What products does the company X have?

Type II: ask about when, where, who/whom, what of an event, e.g.

When did something happen?
Where did something happen?
Who did that?
Whom did X meet?
What did X give to Y?
To whom did X give Y?

The first are entity-related questions. As any entity is related to its environment in multiple aspects, the couple of TRs defined by MUC only cover a fraction of this type of questions. The design of Correlated Entities is motivated by users' frequent needs to search for various aspects of information (Y) about an entity X, namely Type I questions: what is X's Y. As shown before, we have defined many more relationships which are deemed important for an entity in our CE template. It covers

Type I questions much better than MUC TR. In addition, it is expected that, with implementation of some trivial function on a threaded search, the type of questions which can be answered by our CE system can be extended from X's Y to more general questions of the following type: X1's X2's ...Y? For example, *What is Julian Werver Hill's wife's telephone number*? (equivalent to *What is Polly's telephone number*?) *Where is Werver Hill's affiliated company located*? (equivalent to: *Where is Du Pont Inc. located*?), etc.

The second are event-related questions, but they are about general events. The MUC ST information does not seem to be of help in answering general event questions as it is entirely domain dependent. For example, ST defined by [MUC-7 1998] is specific to the domain of air vehicle launch reports. It is defined to only capture "information about launch vehicle, the payload of that vehicle, the date and site of the launch, and information about the mission type, function and status". This type of extremely domain dependent information does not seem to have a place in general purpose IR/QA. The most powerful QA support, we believe, comes from our General Event design as it can address questions on open-ended domain independent events. Our effort in implementing GE is expected to produce a breakthrough in QA.

In summary, we have pointed out the limitation of the MUC IE standards in QA. We demonstrate that for each level of our domain independent IE, our design serves the purpose of QA much better.

5 Future Work: Multi-level IE Supported QA

Figure 3 illustrates the system architecture for our proposed QA development plan based on three levels of IE support from *Textract 3.0* (under development).

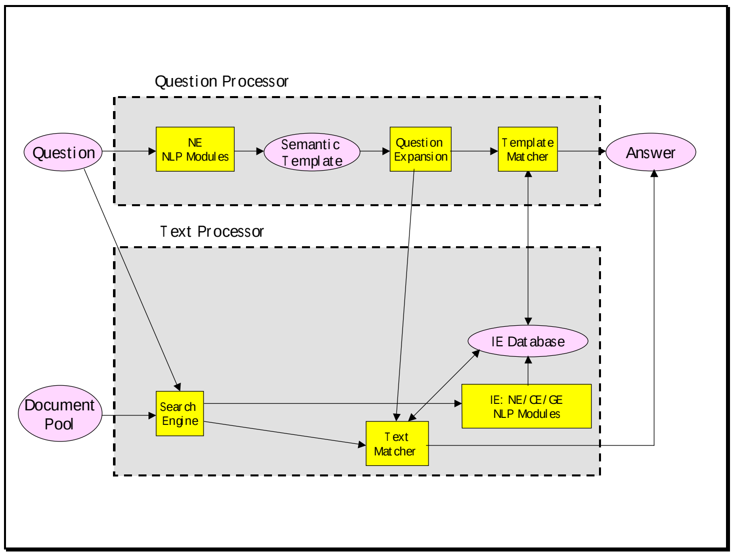

Figure 3: Textract/QA 3.0 System Architecture

To tackle the parsing of the questions (Q) and the extraction of an appropriate answer (A) from the free text, there are two sub-systems: (i) Question Processor; (ii) Text Processor. It should be noted that these two sub-systems go through parallel processes and share the same NLP resources until the point of matching and ranking. In fact, questions form a sub-domain of natural language phenomena, following the same syntax. In our design, a question goes through tokenization, POS tagging, NE tagging and shallow parsing and full parsing to generate the semantic template for the question. The only NLP module which does not have a place in question processing is CO since questions are normally related to each other to form a meaningful discourse. This design keeps the

QA system both modular and flexible. Both sub-systems can make use of the same linguistic and IE support.

The merging of question templates and GE templates in the Template Matcher are fairly straightforward. As they both undergo the same NLP processing, the resulting semantic templates are of the same form. Both question templates and GE templates correspond to fairly standard/predictable patterns (the PREDICATE value is open-ended, but the structure remains stable). More precisely, a user can ask questions on general events themselves (*did what*) and/or on the participants of the event (*who, whom, what*) and/or the time and place of events (*when*, *where*). This addresses by far the most types of general questions of a potential user.

For example, if a user is interested in company acquisition events, he can ask questions like: *Which companies were acquired by Microsoft in 1999? Which companies did Microsoft acquire in 1999?.* Our system will then parse these questions into the templates as shown below.

```
[31]    <Q_TEMPLATE> :=
        PREDICATE:          acquire
        ARGUMENT1:          Microsoft
        ARGUMENT2:          WHAT(COMPANY)
        TIME:               1999
```

If the user wants to know *when* did some acquisition happened, he can ask: *WHEN was Netscape acquired?* Our system will then translate it into the pattern below.

```
[32]    <Q_TEMPLATE> :=
        PREDICATE:          acquire
        ARGUMENT1:          WHO
        ARGUMENT2:          Netscape
        TIME:               WHEN
```

Note that WHO, WHAT, WHEN above are variables to be instantiated. Such question templates serve as search constraints to filter the events in our extracted GE template database. Because the question templates and the extracted GE template share the same structure (just compare the above question templates [31] [32] with the GE template in [1]), a simple merging operation would accomplish the function of providing the user with exactly the event and its related information he is searching for.

Nevertheless, there are two important questions waiting to be answered: (i) what should be done if a verb used in a question differs from one used in the processed text, even though both verbs convey the same meaning? (ii) what should be done if the question asks something beyond the GE (or CE) information?

The first question occurs when the question is *When was Netscape* ***acquired*** and the text is *Netscape was* ***bought*** *by AOL in 1998.* As discussed previously, although our GE extractor is designed to capture variations of surface structures (e.g. passive expressions) into the semantic structure (*argument structure*) of GE Template, the open ended PREDICATE slot is simply filled by the actual verb used (with inflection removed), *buy* in this case. The same thing happens to question parsing, resulting in *acquire* as the filler of the PREDICATE slot. In order to successfully merge synonyms like *acquire* and buy, we have designed a module Question Template Expansion based on ontology. The use of an on-line ontology is conceived in our interface system to automatically produce related patterns based on the question template. For example, from the ; from the template

X acquire/buy Y from Z, another patter using the trigger word *sell* is automatically produced: *Z sell Y to X* ; from the template *X own Y*, a derived template using synonymous trigger word (*have, possess*) is produced: *X have/possess Y* etc. This open-ended question answering feature supported by the GE results and lexical resources of an ontology is the ultimate goal for the QA application of our IE system.

The answer to the second question lies in our 'back-off' model, i.e. the model which we used in the TREC QA Track (presented fully in Figure 2 and included as a part in Figure 3). As this naive QA implementation is based on maximally satisfying the key word constraints plus the NE constraint, it is much more robust. Questions like *why*, *how* which are beyond the information in our GE (and CE) defined scope[3], will still get a list of potential, possibly less accurate, answers.

6 Acknowledgement

We would like to thank our colleagues: William Andrews, Shumin Liu, Lars Nilsson, Cheng Niu, Ray Wen, Hao Xu, and Wenjie Qiao for their assistance in completing the QA task. In addition, we wish to acknowledge Professor Gordon Cormack of the University of Waterloo, Waterloo, Ontario, for providing access to the on-line search engine.

7 References

Bikel, D.M. *et al*., 1997. Nymble: a High-Performance Learning Name-finder. *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Morgan Kaufmann Publishers, pp. 194-201.

Chinchor, N. & Marsh, E. 1998. MUC-7 Information Extraction Task Definition (version 5.1), *Proceedings of MUC-7*

Grishman, R., 1997. TIPSTER Architecture Design Document Version 2.3. Technical report, DARPA.

Krupka, G.R. & Hausman, K. 1998. IsoQuest Inc.: Description of the NetOwl (TM) Extractor System as Used for MUC-7, *Proceedings of MUC-7*

MUC-7, 1998. Proceedings of the Seventh Message Understanding Conference (MUC-7), published on the website http://www.muc.saic.com/

Srihari, R. 1998. A Domain Independent Event Extraction Toolkit, AFRL-IF-RS-TR-1998-152 Final Technical Report, published by Air Force Research Laboratory, Information Directorate, Rome Research Site, New York

[3] This is decided by the nature of IE. That is, an effective information extraction system always *ignores* some information and attempts to extract only salient or key information. In contrast, a user may ask questions about any information.