A Question Answering System Supported by Information Extraction[*]

Rohini Srihari
Cymfony Inc.
5500 Main Street
Williamsville, NY14221
rohini@cymfony.com

Wei Li
Cymfony Inc.
5500 Main Street
Williamsville, NY14221
wei@cymfony.com

Abstract

This paper discusses an information extraction (IE) system, *Textract*, in natural language (NL) question answering (QA) and examines the role of IE in QA application. It shows: (i) Named Entity tagging is an important component for QA, (ii) an NL shallow parser provides a structural basis for questions, and (iii) high-level domain independent IE can result in a QA breakthrough.

Introduction

With the explosion of information in Internet, Natural language QA is recognized as a capability with great potential. Traditionally, QA has attracted many AI researchers, but most QA systems developed are toy systems or games confined to lab and a very restricted domain. More recently, Text Retrieval Conference (TREC-8) designed a QA track to stimulate the research for real world application.

Due to little linguistic support from text analysis, conventional IR systems or search engines do not really perform the task of *information* retrieval; they in fact aim at only *document* retrieval. The following quote from the QA Track Specifications (www.research.att.com/~singhal/qa-track-spec.txt) in the TREC community illustrates this point.

> *Current information retrieval systems allow us to locate documents that might contain the pertinent information, but most of them leave it to the user to extract the useful information from a ranked list. This leaves the (often unwilling) user with a relatively large amount of text to consume. There is an urgent need for tools that would reduce the amount of text one might have to read in order to obtain the desired information. This track aims at doing exactly that for a special (and popular) class of information seeking behavior: QUESTION ANSWERING. People have questions and they need answers, not documents. Automatic question answering will definitely be a significant advance in the state-of-art information retrieval technology.*

Kupiec (1993) presented a QA system MURAX using an on-line encyclopedia. This system used the technology of robust shallow parsing but suffered from the lack of basic information extraction support. In fact, the most siginifcant IE advance, namely the NE (Named Entity) technology, occured after Kupiec (1993), thanks to the MUC program (MUC-7 1998). High-level IE technology beyond NE has not been in the stage of possible application until recently.

AskJeeves launched a QA portal (www.askjeeves.com). It is equipped with a fairly sophisticated natural language question parser, but it does not provide direct answers to the asked questions. Instead, it directs the user to the relevant web pages, just as the traditional search engine does. In this sense, AskJeeves has only done half of the job for QA.

We believe that QA is an ideal test bed for demonstrating the power of IE. There is a natural co-operation between IE and IR; we regard QA as one major intelligence which IE can offer IR.

[*] This work was supported in part by the SBIR grants F30602-98-C-0043 and F30602-99-C-0102 from Air Force Research Laboratory (AFRL)/IFED.

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 1, we first give an overview of the underlying IE technology which our organization has been developing. Section 2 discusses the QA system. Section 3 describes the limitation of the current system. Finally, in Section 4, we propose a more sophisticated QA system supported by three levels of IE.

1 Overview of Textract IE

The last decade has seen great advance 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 modelled 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, 3-level architecture for developing a kernel IE system which is domain-independent throughout.

The core of this system is a state-of-the-art NE tagger [Srihari 1998], named *Textract 1.0.* The *Textract* NE tagger has achieved speed and accuracy comparable to that of the few deployed NE systems, such as *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 types/sub-types of information are also identified by the *TextractNE* module:

- 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 proper name)

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 sub-types 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*+adjective questions: *how long* (*duration* or *length*), *how far* (*length*), *how often* (*frequency*), *how old* (*age*), etc.

Level-2 IE, or CE (Correlated Entity), is concerned with extracting pre-defined multiple relationships between the entities. Consider the person entity as an example; the *TextractCE* prototype is capable of extracting the key relationships such as *age, gender, affiliation, position, birth_time, birth_place, spouse, parents, children, where_from, address, phone, fax, email, descriptors*. As seen, the information in the CE represents a mini-CV or profile of the entity. In general, the CE template integrates and greatly enriches the information contained in MUC TE and 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* (or *how* often) and *where*. By general events, we refer to argument structures centering around verb notions plus the associated information of time/frequency and location. We show an example of our defined GE extracted from the text 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

Figure 1 is the overall system architecture for the IE system *Textract* that our organization has been developing.

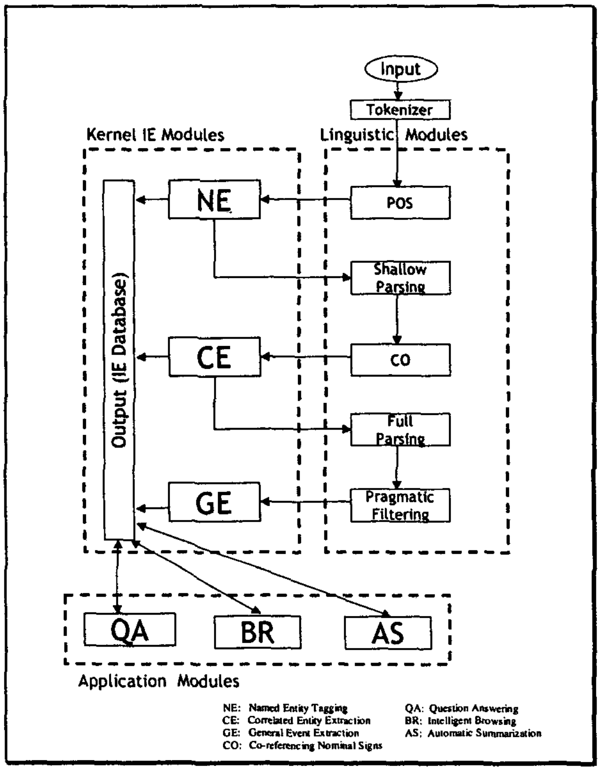

Figure 1: Textract IE System Architecture

The core of the system consists of three kernel IE modules and six linguistic modules. 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 (Browsing, threading and visualization) and AS (Automatic Summarization). The approach to IE taken here, consists of a unique blend of machine learning and FST (finite state transducer) rule-based system [Roche & Schabes 1997]. 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 [Srihari 1998, Li & Srihari 2000].

2 NE-Supported QA

This section presents the QA system based on Named Entity tagging. Out of the 200 questions that comprised the TREC-8 QA track competition, over 80% asked for an NE, e.g. *who* (PERSON), *when* (TIME | DATE), *where* (LOCATION), *how far* (LENGTH). Therefore, the NE tagger has been proven to be very helpful. 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. Nevertheless, using even crude 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.)", in most cases, the QA system was able to extract text portions which contained answers in the top five list.

Figure 2 illustrates the system design of *TextractQA* 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 and their common location within a same sentence.

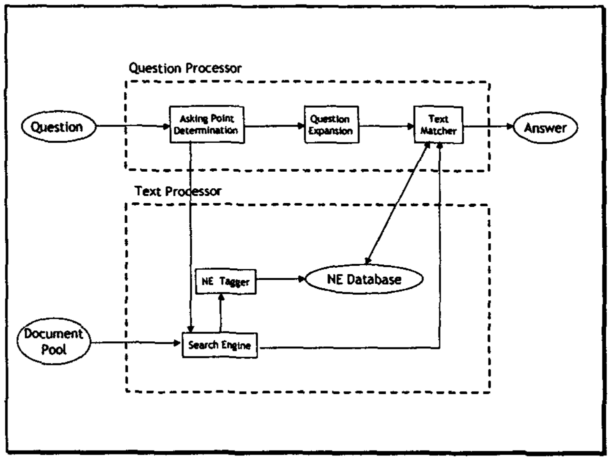

Figure 2: Textract/QA 1.0 Prototype Architecture

The general algorithm for question answering is as follows:

Process Question
    Shallow parse question
    Determine Asking Point
    Question expansion (using word lists)
Process Documents
    Tokenization, POS tagging, NE Indexing
    Shallow Parsing (not yet utilized)
Text Matcher
    Intersect search engine results with NE
    rank answers

2.1 Question Processing

The Question Processing 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*. 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.

[3] *Why did David Koresh ask the FBI for a word processor?*

The system then maps it to the following question template :

```
[4] asking_point:  REASON
    key_word:      {ask, David, Koresh,
                    FBI, word, processor }
```

The question processor 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). It is fairly easy to exhaust the second set of rules as interrogative question words/phrases form a closed set. In comparison, the development of the first set of rules are continuously being fine-tuned and expanded. This strategy of using two set of rules leads to the robustness of the question processor.

The first set of rules are based on shallow parsing results of the questions, using Cymfony FST based Shallow Parser. 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) -->
    CITY|COUNTRY|COMPANY
[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 in capitalized string of words. Examples include *Name a film that has won the Golden Bear in the Berlin Film Festival.*

The word lists *org_w* and *person_w* are currently manually maintained based on inspection of large volumes of text. An effort is underway to automate the learning of such word lists by utilizing machine learning techniques.

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 four 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) -->
CITY | COUNTRY | COMPANY
[11] NP(what/which ... person_w) -->
PERSON

Rule [10] extracts the asking point from cases like *NP[What] Aux[is] NP[the largest country] PP[in the world].* 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 has 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

In the stage of question expansion, the template in [4] would be expanded to the template shown in [29]:

[29] asking_point: {because|because of|
due to|thanks to|since|
in order|to VB}
key_word: {ask|asks|asked|asking,
David,Koresh,FBI,
word, processor}

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.

2.2 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 IE processing. Currently, this includes tokenization, POS tagging and NE tagging. Future plans include several levels of parsing as well; these are required to support CE and GE extraction. It should be noted that all these operations are extremely robust and fast, features necessary for large volume text indexing. Parsing is accomplished through cascaded finite state transducer grammars.

2.3 Text Matching

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 standard built-in 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.

In the TREC-8 QA track competition, Cymfony QA accuracy was 66.0%. Considering we have only used NE technology to support QA in this run, 66.0% is a very encouraging result.

3 Limitation

The first limitation comes from the types of questions. Currently only wh-questions are handled although it is planned that yes-no questions will be handled once we introduce CE and GE templates to support QA. Among the wh-questions, the *why*-question and *how*-question[1] are more challenging because the asking point cannot be simply mapped to the NE types/sub-types.

The second limitation is from the nature of the questions. Questions like *Where can I find the homepage for Oscar winners* or *Where can I find info on Shakespeare's works* might be answerable easily by a system based on a well-maintained data base of home pages. Since our system is based on the processing of the underlying documents, no correct answer can be provided if there is no such an answer (explicitly expressed in English) in the processed documents. In TREC-8 QA, this is not a problem since every question is guaranteed to have at least one answer in the given document pool. However, in the real world scenario such as a QA portal, it is conceived that the IE results based on the processing of the documents should be complemented by other knowledge sources such as e-copy of yellow pages or other manually maintained and updated data bases.

The third limitation is the lack of linguistic processing such as sentence-level parsing and cross-sentential co-reference (CO). This problem will be gradually solved when high-level IE technology is introduced into the system.

4 Future Work: Multi-level IE Supported QA

A new QA architecture is under development; it will exploit all levels of the IE system, including CE and GE.

The first issue is how much CE can contribute to a better support of QA. It is found that there are some frequently seen questions which can be better answered once the CE information is provided. These questions are of two types: (i) *what/who* questions about an NE; (ii) relationship questions.

Questions of the following format require CE templates as best answers: *who/what* is NE? For example, *Who is Julian Hill? Who is Bill Clinton? What is Du Pont? What is Cymfony?* To answer these questions, the system can simply retrieve the corresponding CE template to provide an "assembled" answer, as shown below.

Q:	Who is Julian Hill?	
A:	name:	Julian Werner Hill
	type:	PERSON
	age:	91
	gender:	MALE
	position:	research chemist
	affiliation:	Du Pont Co.
	education:	Washington University; MIT

Q:	What is Du Pont?	
A:	name:	Du Pont Co.
	type:	COMPANY
	staff:	Julian Hill; Wallace Carothers.

Questions specifically about a CE relationship include: *For which company did Julian Hill work?* (affiliation relationship) *Who are employees of Du Pont Co.?* (staff relationship) *What does Julian Hill do?* (position/profession relationship) *Which university did Julian Hill graduate from?* (education relationship), etc.[2]

The next issue is the relationships between GE and QA. It is our belief that the GE technology will result in a breakthrough for QA.

In order to extract GE templates, the text goes through a series of linguistic processing as shown in Figure 1. It should be noted that the question processing is designed to go through parallel processes and share the same NLP resources until the point of matching and ranking.

The merging of question templates and GE templates in 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, frequency and place of events (*when, how often, where*). This addresses

[1] For example, *How did one make a chocolate cake? How*+Adjective questions (e.g. *how long, how big, how old,* etc.) are handled fairly well.

[2] An alpha version of *TextractQA* supported by both NE and CE has been implemented and is being tested.

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 ware 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* 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 variable 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, a simple merging operation would suffice. Nevertheless, there are two important questions to be answered: (i) what if a different verb with the same meaning is used in the question from the one used in the processed text? (ii) what if the question asks about something beyond the GE (or CE) information? These are issues that we are currently researching.

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

Kupiec J. (1993) *MURAX: A Robust Linguistic Approach For Question Answering Using An On-Line Encyclopaedia,* "Proceedings of SIGIR-93 93" Pittsburgh, Penna.

Li, W & Srihari, R. 2000. *Flexible Information Extraction Learning Algorithm,* Final Technical Report, Air Force Research Laboratory, Rome Research Site, New York

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

Roche E. and Schabes Y. (1997) *Finite-State Language Processing,* MIT Press, Cambridge, MA

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