Extracting Exact Answers to Questions Based on Structural Links*

Wei Li, Rohini K. Srihari, Xiaoge Li, M. Srikanth, Xiuhong Zhang, Cheng Niu
Cymfony Inc.
600 Essjay Road, Williamsville, NY 14221. USA.
{wei, rohini, xli, srikanth, xzhang, cniu}@cymfony.com

Keywords: Question Answering, Information Extraction, Semantic Parsing, Dependency Link

Abstract

This paper presents a novel approach to extracting phrase-level answers in a question answering system. This approach uses structural support provided by an integrated Natural Language Processing (NLP) and Information Extraction (IE) system. Both questions and the sentence-level candidate answer strings are parsed by this NLP/IE system into binary dependency structures. Phrase-level answer extraction is modelled by comparing the structural similarity involving the question-phrase and the candidate answer-phrase.

There are two types of structural support. The first type involves predefined, specific entity associations such as Affiliation, Position, Age for a person entity. If a question asks about one of these associations, the answer-phrase can be determined as long as the system decodes such pre-defined dependency links correctly, despite the syntactic difference used in expressions between the question and the candidate answer string. The second type involves generic grammatical relationships such as V-S (verb-subject), V-O (verb-object).

Preliminary experimental results show an improvement in both precision and recall in extracting phrase-level answers, compared with a baseline system which only uses Named Entity constraints. The proposed methods are particularly effective in cases where the question-phrase does not correspond to a known named entity type and in cases where there are multiple candidate answer-phrases satisfying the named entity constraints.

Introduction

Natural language Question Answering (QA) is recognized as a capability with great potential. The NIST-sponsored Text Retrieval Conference (TREC) has been the driving force for developing this technology through its QA track since TREC-8 (Voorhees 1999). There has been significant progress and interest in QA research in recent years (Voorhees 2000, Pasca and Harabagiu 2001).

QA is different than search engines in two aspects: (i) instead of a string of keyword search terms, the query is a natural language question, necessitating question parsing, (ii) instead of a list of documents or URLs, a list of candidate answers at phrase level or sentence level are expected to be returned in response to a query, hence the need for text processing beyond keyword indexing, typically supported by Natural Language Processing (NLP) and Information Extraction (IE) (Chinchor and Marsh 1998, Hovy, Hermjakob and Lin 2001, Li and Srihari 2000). Examples of the use of NLP and IE in Question Answering include shallow parsing (Kupiec, 1993), semantic parsing (Litkowski

* This work was partly supported by a grant from the Air Force Research Laboratory's Information Directorate (AFRL/IF), Rome, NY, under contracts F30602-00-C-0037 and F30602-00-C-0090.

1999), Named Entity tagging (Abney et al. 2000, Srihari and Li 1999) and high-level IE (Srihari and Li, 2000).

Identifying exact or phrase-level answers is a much more challenging task than sentence-level answers. Good performance on the latter can be achieved by using sophisticated passage retrieval techniques and/or shallow level NLP/IE processing (Kwok et al. 2001, Clarke et al. 2001). The phrase-level answer identification involves sophisticated NLP/IE and it is difficult to apply only IR techniques for this task (Prager et al. 1999). These two tasks are closely related. Many systems (e.g. Prager et al 1999; Clark et al 2001) take a two-stage approach. The first stage involves retrieving sentences or paragraphs in documents as candidate answer strings. Stage Two focuses on extracting phrase-level exact answers from the candidate answer strings.

This paper focuses on methods involving Stage Two. The input is a sentence pair consisting of a question and a sentence-level candidate answer string. The output is defined to be a phrase, called answer-point, extracted from the candidate answer string. In order to identify the answer-point, the pair of strings are parsed by the same system to generate binary dependency structures for both specific entity associations and generic grammatical relationships. An integrated Natural Language Processing (NLP) and Information Extraction (IE) engine is used to extract named entities (NE) and their associations and to decode grammatical relationships. The system searches for an answer-point by comparing the structural similarity involving the question-phrase and a candidate answer-phrase. Generic grammatical relationships are used as a back-off for specific entity associations when the question goes beyond the scope of the specific associations or when the system fails to identify the answer-point which meets the specific entity association constraints. The proposed methods are particularly helpful in cases where the question-phrase does not correspond to a known named entity type and in cases where there are multiple candidate answer-points to select from.

The rest of the paper is structured as follows: Section 1 presents the NLP/IE engine used, sections 2 discusses how to identify and formally represent what is being asked, section 3 presents the algorithm on identifying exact answers leveraging structural support, section 4 presents case studies and benchmarks, and section 5 is the conclusion.

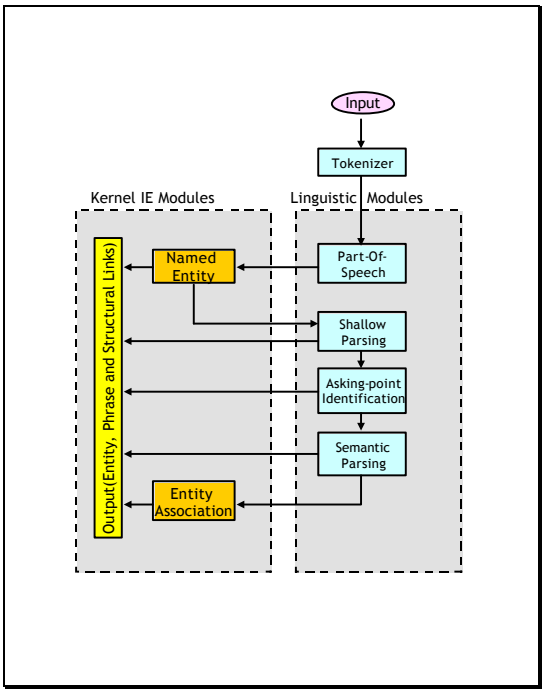

Figure 1: InfoXtract™ NLP/IE System Architecture

1 NLP/IE Engine Description

The NLP/IE engine used in the QA system described here is named InfoXtract™. It consists of an NLP component and IE component, each consisting of a set of pipeline modules (Figure 1). The NLP component serves as underlying support for IE. A brief description of these modules is given below.

- *Part-of-Speech Tagging*: tagging syntactic categories such as noun, verb, adjective, etc.
- *Shallow Parsing*: grouping basic linguistic units as building blocks for structural links, such as Basic Noun Phrase, Verb Group, etc.
- *Asking-point Identification*: analysis of question sentences to determine what is being asked

- *Semantic Parsing*: decoding grammatical dependency relationships at the logical level between linguistic units, such as Verb-Subject (V-S), Verb-Object (V-O), Head-Modifier (H-M) relationships; both active patterns and passive patterns will be parsed into the same underlying logical S-V-O relationships
- *Named Entity Tagger*: classifying proper names and other phrases to different categories such as Person, Organization, Location, Money, etc.
- *Entity Association Extractor*: relating named entities with predefined associations such as Affiliation, Position, Age, Spouse, Address, etc.

The NE tagger in our system is benchmarked to achieve close to human performance, around or above 90% precision and recall for most categories of NE. This performance provides fundamental support to QA. Many questions require a named entity or information associated with a named entity as answers. A subset of the NE hierarchy used in our system is illustrated below:

Person: woman, man
Organization: company, government, association, school, army, mass-media
Location: city, province, country, continent, ocean, lake, etc.
Time Expressions: hour, part-of-day, day-of-week, date, month, season, year, decade, century, duration
Numerical Expressions: percentage, money, number, weight, length, area, etc.
Contact expressions: email, address, telephone, etc.

The Entity Association module correlates named entities and extracts their associations with other entities or phrases. These are specific, predefined relationships for entities of person and organization. Currently, our system can extract the following entity associations with high precision (over 90%) and modest recall ranging from 50% to 80% depending on the size of grammars written for each specific association.

Person: affiliation, position, age, spouse, birth-place, birth-time, etc.
Organization: location, staff, head, products, found-time, founder, etc.

Entity associations are semantic structures very useful in supporting QA. For example, from the sentence *Grover Cleveland , who in June 1886 married 21-year-old Frances Folsom,…*the IE engine can identify the following associations:

Spouse: Grover Cleveland ←Frances Folsom
Spouse: Frances←Grover Cleveland
Age: Frances Folsom←21-year-old

A question asking about such an association, say, Q11: *Who was President Cleveland 's wife*, will be parsed into the following association link between a question-phrase 'Who' and the entity 'Cleveland' (see Section 2): *Spouse: Cleveland → Who*. The semantic similarity between this structure and the structure *Spouse: Grover Cleveland → Frances Folsom* can determine the answer point to be 'Frances Folsom'.

The Semantic Parsing module decodes the grammatical dependency relationships: V-S, V-O, V-C (Verb-Complement), H-M of time, location, reason, manner, purpose, result, etc. This module extends the shallow parsing module through the use of a cascade of handcrafted pattern matching rules. Manual benchmarking shows results with the following performance:

H-M:	Precision 77.5%
V-O:	Precision 82.5%
V-S:	Precision 74%
V-C:	Precision 81.4%

In our semantic parsing, not only passive patterns will be decoded into the same underlying structures as active patterns, but structures for verbs such as *acquire* and for de-verbal nouns such as *acquisition* lead to the same dependency links, as shown below.

AOL acquired Netscape in 1998. →
V-S: acquired ← AOL
V-O: acquired ← Netscape
H-M: acquired ← in 1998 (time-modifier)

Netscape was acquired by AOL in 1998. →
V-S: was acquired ← by AOL

V-O: was acquired ← Netscape
H-M: was acquired ← in 1998 (time-modifier)

the acquisition of Netscape by AOL in 1998… →
V-S: acquisition ← by AOL
V-O: acquisition ← of Netscape
H-M: acquired ← in 1998 (time-modifier)

These links can be used as structural support to answer questions like *Who acquired Netscape* or *which company was acquired by AOL.*

Obviously, our semantic parser goes one step further than parsers which only decode syntactic relationships. It consumes some surface structure variations to provide the power of comparing the structural similarity at logical level. However, compared with the entity association structures which sits at deep semantic level, the logical SVO (Subject-Verb-Object) structures still cannot capture semantic relations which are expressed using different head verbs with different structures. An example is the pair : *X borrows Y from Z* versus *Z lends Y to X.*

2 Asking Point Link Identification

Asking point link identification is a crucial step in a QA system. It provides the necessary information decoded from question processing for a system to locate the corresponding answer-points from candidate answer strings.

The Asking-point (Link) Identification Module is charged with the task of parsing *wh*-phrases in their context into three categories: NE Asking-point, Asking-point Association Link and Asking-point Grammar Link. *Asking Point* refers to the question phrases with its constraints that a corresponding answer-point should satisfy in matching. *Asking-point Link* is the decoded binary relationship from the asking point to another unit in the question.

The identification of the NE asking point is essentially mapping the *wh*-phrase to the NE types or subtypes. For example, *which year* is mapped to [which year]/NeYear, *how old* mapped to [how old]/NeAge, and *how long* mapped to [how long]/NeLength or [how long]/NeDuration, etc.

The identification of the Asking-point Association Link is to decide whether the incoming question asks about a predefined association relationship. For Asking-point Association Link, the module needs to identify the involved entity and the asked association. For example, the Asking-point Association Link for *How old is John Smith* is the AGE relationship of the NePerson *John Smith*, represented as *AGE: John Smith ← [how old]/NeAge.*

The *wh*-phrases which may or may not be mapped to NE asking points and whose dependency links are beyond predefined associations lead to Asking-point Grammar Links, e.g. *How did Julian Hill discover nylon?* This asking-point link is represented as *H-M: discover ← [How]/manner-modifier.* As seen, an asking-point grammar link only involves generic grammatical constraints: in this case, the constraints for a candidate answer-point to satisfy during matching are H-M link with 'discover' as head and a phrase which must be a modifier of manner.

These three types of asking points and their possible links form a natural hierarchy that can be used to facilitate the backoff strategy for the answer-point extraction module (see Section 3): Asking-point Association Link → Asking-point Grammar Link → NE Asking Point. This hierarchy defines the sequence of matching steps which should be followed during the answer-point extraction.

The backoff from Asking-point Association Link to Asking-point Grammar Link is necessary as the latter represents more generic structural constraints than the former. For example, in the sentence *where is IBM located,* the Asking-point Association Link is *LOCATION: IBM ← [where]/NeLocation* while the default Grammar Link is *H-M: located ← [where]/location-modifier.* When the specific association constraints cannot be satisfied, the system should attempt to locate an answer-point by searching for a location-modifier of the key verb 'located'.

The NE asking point constraints are also marked for asking-point association links and those asking-point grammar links whose *wh*-phrases can be

mapped to NE asking points. Backing off to the NE asking point is required in cases where the asking-point association constraints and grammatical structural constraints cannot be satisfied. For *How old is John Smith*, the asking-point grammar link is represented as *H-M: John Smith ← [how old]/NeAge.* If the system cannot find a corresponding AGE association or a modifier of NeAge for the entity *John Smith* to satisfy the structural constraints, it will at least attempt to locate a candidate answer-point by enforcing the NE asking point constraints NeAge. When there is only one NeAge in the answer string, the system can extract it as the only possible answer-point even if the structural constraints are not honored.

3 Answer Point Identification

The answer-point identification is accomplished through matching the asking-point to candidate answer-points using the following back-off algorithm based on the processing results of the question and the sentence-level candidate answer string.

(1) if there is Asking-point Association Link, call Match(asking-point association link, candidate answer-point association link) to search for the corresponding association to locate answer-point
(2) if step (1) fails and there is an asking-point grammar link, call Match(asking-point grammar link, candidate answer-point grammar link) to search for the corresponding grammar link to locate the answer-point
(3) if step (2) fails and there is an NE asking point, search for the corresponding NEs: if there is only one corresponding NE, then extract this as the answer-point else mark all corresponding NEs as candidate answer-points

The function Match(asking-point link, candidate answer-point link) is defined as (i) exact match or synonym match of the related units (synonym match currently confined to verb vs. de-verbal noun); (ii) match the relation type directly (e.g. V-S matches V-S, AGE matches AGE, etc.); (iii) match the type of asking point and answer point (e.g. NePerson asking point matches NePerson and its sub-types NeMan and NeWoman; 'how' matches manner-modifier; etc.): either through direct link or indirect link based on conjunctive link (ConjLink) or equivalence link (S-P, subject-predicative or appositive relations between two NPs).

Step (1) and Step (2) attempt to leverage the structural support from parsing and high-level information extraction beyond NE. It is worth noticing that in our experiment, the structural support used for answer-point identification only checks the binary links involving the asking point and the candidate answer points, instead of full template matching as proposed in (Srihari and Li, 2000).

Full template matching is best exemplified by the following example. If the incoming question is *Who won the Nobel Prize in 1991*, and the candidate answer string is *John Smith won the Nobel Prize in 199*1, the question template and answer template are shown below:

win
V-S: NePerson [Who]
V-O: NP [the Nobel Prize]
H-M: NeYear [1991]

win
V-S: NePerson [John Smith]
V-O: NP [the Nobel Prize]
H-M: NeYear [1991]

The template matching will match the asking point *Who* with the answer point *John Smith* because for all the dependency links in the trees, the information is all compatible (in this case, exact match). This is the ideal case of full template matching and guarantees the high precision of the extracted answer point.

However, in practice, full template matching is neither realistic for most of cases nor necessary for achieving the objective of extracting answer points in a two-stage approach. It is not realistic because natural language semantic parsing is such a challenging problem that a perfect dependency tree (or full template) which pieces together every linguistic unit is not always easy to decode. For

InfoXtract,, in most cases, the majority, but not all, of the decoded binary dependency links are accurate, as shown in the benchmarks above. In such situations, insisting on checking every dependency link of a template tree is too strong a condition to meet. On the other hand, it is actually not necessary to check all the links in the dependency trees for full template matching. With the modular design and work division between sentence level candidate answer string generation module (Stage One) and answer-point extraction from the candidate answer strings (Stage Two), all the candidate answer strings are already determined by previous modules as highly relevant. In this situation, a simplified partial template matching, namely, 'asking/answer point binary relation matching', will be sufficient to select the answer-point, if present, from the candidate answer string. In other words, the system only needs to check this one dependency link in extracting the answer-point. For the previous example, only the asking/answer point binary dependency links need to be matched as illustrated below:

V-S win←[Who]/NePerson
V-S win←[John Smith]/NeMan

Some sample results are given in section 4 to illustrate how answer-points are identified based on matching binary relations involving asking/answer points.

4 Experiments and Results

In order to conduct the feasibility study on the proposed method, we selected the first 100 questions from the TREC-8 QA track pool and the corresponding first candidate answer sentences for this preliminary experiment. The Stage One processing for generating candidate answer sentences was conducted by the existing ranking module of our QA system. The Stage Two processing for answer-point identification was accomplished by using the algorithm described in Section 3.

As shown in Table 1, out of the 100 question-answer pairs we selected, 9 have detected association links involving asking/answer points, 44 are found to have grammar links involving asking/answer points.

Table 1: Experiment Results

	detected	correct	fail	precision	recall
Association Links	9	8	1	89%	8%
Grammar Links	44	39	6	89%	39%
NE Points (Baseline)	76	41	35	54%	41%
Overall performance	86	71	14	83%	71%

As for NE asking points, 76 questions were identified to require some type of NE as answers. Assume that a baseline answer-point identification system only uses NE asking points as constraints, out of the 76 questions requiring NEs as answers, 41 answer-points were identified successfully because there was only one NE in the answer string which matches the required NE type. The failed cases in matching NE asking point constraints include two situations: (i) no NE exists in the answer string; (ii) multiple NEs satisfy the type constraints of NE asking points (i.e. more than one candidate answer-points found from the answer string) or there is type conflict during the matching of NE asking/answer points. Therefore, the baseline system would achieve 54% precision and 41% recall based on the standard precision and recall formulas:

Precision = Correct / Detected
Recall = Correct / Relevant.

In comparison, in our answer-point identification system which leverages structural support from both the entity association links and grammar links as well as the NE asking points, both the precision and recall are raised: from the baseline 54% to 83% for precision and from 41% to 71% for recall. The significant improvement in precision and recall is attributed to the performance of structural matching in identifying exact answers. This demonstrates the benefits of making use of sophisticated NLP/IE technology, beyond NE and shallow parsing.

Using grammar links alone, exact answers were identified for 39 out of the 44 candidate answer-points satisfying the types of grammar links in 100 cases. During matching, 6 cases failed either due to the parsing error or due to the type conflict

between the asking/answer points (e.g. violating the type constraints such as *manner-modifier* on the answer-point for 'how' question). The high precision and modest recall in using the grammar constraints is understandable as the grammar links impose very strong constraints on both the nodes and the structural type. The high precision performance indicates that grammar links not only have the distinguishing power to identify exact answers in the presence of multiple NE options but also recognize answers in the absence of asking point types.

Even stronger structural support comes from the semantic relations decoded by the entity association extraction module. In this case, the performance is naturally high-precision (89%) low-recall (8%) as predefined association links are by nature more sparse than generic grammatical relations.

In the following, we illustrate with some examples with questions from the TREC-8 QA task on how the match function identified in Section 3 applies to different question types.

Q4: *How much did Mercury spend on advertising in 1993?* → asking-point grammar link:
V-O spend ← [How much]/NeMoney
A: *Last year the company spent Pounds 12m on advertising.* → candidate answer-point grammar link:
V-O spent ← [Pounds 12m]/NeMoney
Answer-point Output: Pounds 12m

This case requires (i) exact match in its original verb form between *spend* and *spent*; (ii) V-O type match; and (iii) asking/answer point type NeMoney match through direct link.

Q63: *What nuclear-powered Russian submarine sank in the Norwegian Sea on April 7, 1989?* → asking-point grammar link:
H-M submarine ← [What]
A: *NEZAVISIMAYA GAZETA on the Komsomolets nuclear-powered submarine which sank in the Norwegian Sea five years ago:* → candidate answer-point grammar link:
H-M submarine ← Komsomolets
Answer-point Output: Komsomolets

This case requires (i) exact match of *submarine*; (ii) H-M type match; and (iii) asking/answer point match through direct link: there are no asking point type constraints because the asking point goes beyond existing NE. This case highlights the power of semantic parsing in answer-point extraction. Since there are no type constraints on answer point,[1] candidate answer points cannot be extracted without bringing in structural context by checking the NE type. Most of *what*-related asking points such as those in the patterns 'what/which…N', 'what type/kind of …N' go beyond NE and require this type of structural relation checking to locate the exact answer. The case below is another example.

Q79: *What did Shostakovich write for Rostropovich?* → asking-point grammar link:
V-O write ← [What]
A: *The Polonaise from Tchaikovsky's opera Eugene was a brief but cracking opener and its brilliant bluster was no sooner in our ears than forcibly contradicted by the bleak depression of Shostakovich's second cello concerto, Op. 126, a late work written for Rostropovich in 1966 between the thirteenth and fourteenth symphonies.* → candidate answer-point grammar link:
V-O written ← [a late work]/NP
S-P [Op. 126]/NP ← [a late work]/NP
Answer-point Output: Op. 126

This case requires (i) exact match in its original verb form between 'written' and 'write'; (ii) V-O type match; and (iii) asking/answer point match through indirect link based on equivalence link S-P. When there are no NE constraints on the answer point, a proper name or an initial-capitalized NP is preferred over an ordinary, lower-case NP as an answer point. This heuristic is built-in so that 'Op. 126' is output as the answer-point in this case instead of 'a late work'.

[1] Strictly speaking, there are some type constraints on the answer point. The type constraints are something to the effect of 'a name for a kind of ship' which goes beyond the existing NE types defined.

Conclusion

This paper presented an approach to exact answer identification to questions using only binary structural links involving the question-phrases. Based on the experiments conducted, some preliminary conclusions can be arrived at.

- The Entity Association extraction helps in pinpointing exact answers precisely
- Grammar dependency links enable the system to not only identify exact answers but answer questions not covered by the predefined set of available NEs/Associations
- Binary dependency links instead of full structural templates provide sufficient and effective structural leverage for extracting exact answers

Some cases remain difficult however, beyond the current level of NLP/IE. For example,

> Q92: *Who released the Internet worm in the late 1980s?* → asking point link:
> V-S (released, NePerson[Who])
> A: *Morris, suspended from graduate studies at Cornell University at Syracuse, N,Y,, is accused of designing and disseminating in November, 1988, a rogue program or "worm" that immobilized some 6,000 computers linked to a research network, including some used by NASA and the Air Force.* → answer point link:
> V-S (disseminating, NePerson[Morris])

In order for this case to be handled, the following steps are required: (i) the semantic parser should be able to ignore the past participle postmodifier phrase headed by 'suspended'; (ii) the V-O dependency should be decoded between 'is accused' and 'Morris'; (iii) the V-S dependency should be decoded between 'designing and disseminating' and 'Morris' based on the pattern rule 'accuse NP of Ving'→ V-S(Ving, NP); (iv) the conjunctive structure should map the V-S ('designing and disseminating', 'Morris') into two V-S links; (v) 'disseminate' and 'release' should be linked somehow for synonym expansion. It may be unreasonable to expect an NLP/IE system to accomplish all of these, but each of the above challenges indicates some directions for further research in this topic.

We would like to extend the experiments on a larger set of questions to further investigate the effectiveness of structural support in extracting exact answers. The TREC-9 and TREC 2001 QA pool and the candidate answer sentences generated by both NLP-based or IR-based QA systems would be ideal for further testing this method.

5 Acknowledgement

The authors wish to thank Walter Gadz and Carrie Pine of AFRL for supporting this work. Thanks also go to anonymous reviewers for their valuable comments.

References

Abney, S., Collins, M and Singhal, A. (2000) *Answer Extraction*. In Proceedings of ANLP-2000, Seattle.

Chinchor, N. and Marsh, E. (1998) *MUC-7 Information Extraction Task Definition* (version 5.1), In "Proceedings of MUC-7". Also published at http://www.muc.saic.com/

Clarke, C. L. A., Cormack, G. V. and Lynam, T. R. (2001), *Exploiting Redundancy in Question Answering*. In Proceedings of SIGIR'01, New Orleans, LA.

Hovy, E.H., U. Hermjakob, and Chin-Yew Lin. 2001. The Use of External Knowledge of Factoid QA. In Proceedings of the 10th Text Retrieval Conference (TREC 2001), Gaithersburg, MD, U.S.A., November 13-16, 2001

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

Kwok, K. L., Grunfeld, L., Dinstl, N. and Chan, M. (2001), *TREC2001 Question-Answer, Web and Cross Language Experiments using PIRCS*. In Proceedings of TREC-10, Gaithersburg, MD.

Li, W. and Srihari, R. (2000) *A Domain Independent Event Extraction Toolkit*, Phase 2 Final Technical Report, Air Force Research Laboratory/Rome, NY.

Litkowski, K. C. (1999) *Question-Answering Using Semantic Relation Triples*. In Proceedings of TREC-8, Gaithersburg, MD.

Pasca, M. and Harabagiu, S. M. *High Performance Question/Answering*. In Proceedings of SIGIR 2001: pages 366-374

Prager, J., Radev, D., Brown, E., Coden, A. and Samn, V., *The use of predictive annotation for question*

answering in TREC8. In Proceedings of TREC-8, Gaithersburg, MD.

Srihari, R. and Li, W. (1999) *Information Extraction supported Question Answering.* In Proceedings of TREC-8, Gaithersberg, MD.

Srihari, R and Li, W. (2000b). *A Question Answering System Supported by Information Extraction.* In Proceedings of ANLP 2000, Seattle.

Voorhees, E. (1999), T*he TREC-8 Question Answering Track Report*, In Proceedings of TREC-8, Gaithersburg, MD.

Voorhees, E. (2000), *Overview of the TREC-9 Question Answering Track*, In Proceedings of TREC-9, Gaithersburg, MD.