

Towards Light Semantic Processing for Question Answering

Benjamin Van Durme*, **Yifen Huang***, **Anna Kupść*⁺**, **Eric Nyberg***

*Language Technologies Institute, Carnegie Mellon University

⁺Polish Academy of Sciences

{vandurme,hyifen,aniak,ehn}@cs.cmu.edu

Abstract

The paper¹ presents a lightweight knowledge-based reasoning framework for the JAVELIN open-domain Question Answering (QA) system. We propose a constrained representation of text meaning, along with a flexible unification strategy that matches questions with retrieved passages based on semantic similarities and weighted relations between words.

1 Introduction

Modern Question Answering (QA) systems aim at providing answers to natural language questions in an open-domain context. This task is usually achieved by combining information retrieval (IR) with information extraction (IE) techniques, modified to be applicable to unrestricted texts. Although semantics-poor techniques, such as surface pattern matching (Soubbotin, 2002; Ravichandran and Hovy, 2002) or statistical methods (Ittycheriah et al., 2002), have been successful in answering factoid questions, more complex tasks require a consideration of text meaning. This requirement has motivated work on QA systems to incorporate knowledge processing components such as semantic representation, ontologies, reasoning and inference engines, e.g., (Moldovan et al., 2003), (Hovy et al., 2002), (Chu-Carroll et al., 2003). Since world knowledge databases for open-domain tasks are unavailable, alternative approaches for meaning representation must be adopted. In this paper, we present our preliminary approach to semantics-based answer detection in the JAVELIN QA system (Nyberg et al., 2003). In contrast to other QA systems, we are trying to realize a formal model for a lightweight semantics-based open-domain question answering. We propose a constrained semantic representation as well as an explicit unification

framework based on semantic similarities and weighted relations between words. We obtain a lightweight robust mechanism to match questions with answer candidates.

The organization of the paper is as follows: Section 2 briefly presents system components; Section 3 discusses syntactic processing strategies; Sections 4 and 5 describe our preliminary semantic representation and the unification framework which assigns confidence values to answer candidates. The final section contains a summary and future plans.

2 System Components

The JAVELIN system consists of four basic components: a question analysis module, a retrieval engine, a passage analysis module (supporting both statistical and NLP techniques), and an answer selection module. JAVELIN also includes a planner module, which supports feedback loops and finer control over specific components (Nyberg et al., 2003). In this paper we are concerned with the two components which support linguistic analysis: the question analysis and passage understanding modules (Question Analyzer and Information Extractor, respectively). The relevant aspects of syntactic processing in both modules are presented in Section 3, whereas the semantic representation is introduced in Section 4.

3 Parsing

The system employs two different parsing techniques: a chart parser with hand-written grammars for question analysis, and a lexicalized, broad coverage skipping parser for passage analysis. For question analysis, parsing serves two goals: to identify the finest answer focus (Moldovan et al., 2000; Hermjakob, 2001), and to produce a grammatical analysis (f-structure) for questions. Due to the lack of publicly available parsers which have suitable coverage of question forms, we have manually developed a set of grammars to achieve these goals. On the other hand, the limited coverage and ambiguity in

¹The authors appear in alphabetical order.

these grammars made adopting the same approach for passage analysis inefficient. In effect, we use two distinct parsers which provide two syntactic representations, including grammatical functions. These syntactic structures are then transformed into a common semantic representation discussed in Section 4.

```
( (Brill-pos VBN)
  (adjunct (
    (object (
      (Brill-pos WRB)
      (atype temporal)
      (cat n)
      (ortho When)
      (q-focus +)
      (q-token +)
      (root when)
      (tokens 1)))
    (time +)))
  (cat v)
  (finite +)
  (form finite)
  (modified +)
  (ortho founded)
  (passive +)
  (punctuation (
    (Brill-pos "."))
    (cat punct)
    (ortho ?)
    (root ?)
    (tokens 6)))
  (qa (
    (gap (
      (atype temporal)
      (path (*MULT* adjunct
            object))))
    (qtype entity)))
  (root found)
  (subject (
    (BBN-name person)
    (Brill-pos NNP)
    (cat n)
    (definite +)
    (gen-pn +)
    (human +)
    (number sg)
    (ortho "Wendy's")
    (person third)
    (proper-noun +)
    (root wendy)
    (tokens 3)))
  (tense past)
  (tokens 5))
```

Figure 1: *When was Wendy's founded*: KANTOO f-structure

3.1 Questions

The question analysis consists of two steps: lexical processing and syntactic parsing. For the lexical processing step, we have integrated several external resources:

the Brill part-of-speech tagger (Brill, 1995), BBN Identifier (BBN, 2000) (to tag named entities such as proper names, time expressions, numbers, etc.), WordNet (Fellbaum, 1998) (for semantic categorization), and the KANTOO Lexifier (Nyberg and Mitamura, 2000) (to access a syntactic lexicon for verb valence information).

The hand-written grammars employed in the project are based on the Lexical Functional Grammar (LFG) formalism (Bresnan, 1982), and are used with the KANTOO parser (Nyberg and Mitamura, 2000). The parser outputs a functional structure (f-structure) which specifies the grammatical functions of question components, e.g., subject, object, adjunct, etc. As illustrated in Fig. 1, the resulting f-structure provides a deep, detailed syntactic analysis of the question.

3.2 Passages

Passages selected by the retrieval engine are processed by the Link Grammar parser (Grinberg et al., 1995). The parser uses a lexicalized grammar which specifies links, i.e., grammatical functions, and provides a constituent structure as output. The parser covers a wide range of syntactic constructions and is robust: it can skip over unrecognized fragments of text, and is able to handle unknown words.

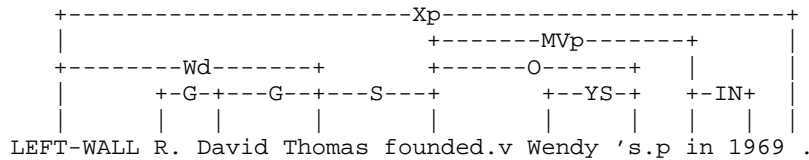
An example of the passage analysis produced by the Link Parser is presented in Fig. 2. Links are treated as predicates which relate various arguments. For example, *O* in Fig. 2 indicates that *Wendy's* is an object of the verb *founded*. In parallel to the Link parser, passages are tagged with the BBN Identifier (BBN, 2000), in order to group together multi-word proper names such as *R. David Thomas*.

4 Semantic Representation

At the core of our linguistic analysis is the semantic representation, which bridges the distinct representations of the functional structure obtained for questions and passages. Although our semantic representation is quite simple, it aims at providing the means of understanding and processing broad-coverage linguistic data. The representation uses the following main constructs:²

- **formula** is a conjunction of literals and represents the meaning of the entire sentence (or question);
- **literal** is a predicate relation over two terms; in particular, we distinguish two types of literals: **extrinsic literal**, a literal which relates a label to a label, and **intrinsic literal**, a literal which relates a label to a word;

²The use of terminology common in the field of formal logic is aimed at providing an intuitive understanding to the reader, but is not meant to give the impression that our work is built on a firm logic-theoretic framework.



Constituent tree:

```
(S (NP (NP R. David Thomas)
  (VP founded
    (NP (NP Wendy 's))
    (PP in
      (NP 1969))))
.)
```

Figure 2: *R. David Thomas founded Wendy's in 1969.*: Link Grammar parser output

- **predicate** is used to capture relations between terms;
- **term** is either a **label**, a variable which refers to a specific entity or an event, or a **word**, which is either a single word (e.g., *John*) or a sequence of words separated by whitespace (e.g., for proper names such as *John Smith*).

The BNF syntax corresponding to this representation is given in (1).

```
(1) <formula> := <literal>+
<literal> := <pred>(<term>, <term>)
<term> := <label>|<word>
<word> := |[a-zA-Z0-9\s]+|
<label> := [a-z][0-9]+
<pred> := [A-Z_-]+
```

With the exception of the unary ANS predicate which indicates the sought answer, all predicates are binary relations (see examples in Fig. 3). Currently, most predicate names are based on grammatical functions (e.g., SUBJECT, OBJECT, DET) which link events and entities with their arguments. Unlike in (Moldovan et al., 2003), names of predicates belong to a fixed vocabulary, which provides a more sound basis for a formal interpretation. Names of labels and terms are restricted only by the syntax in (1). Examples of semantic representations for the question *When was Wendy's founded?* and the passage *R. David Thomas founded Wendy's in 1969.* are shown in Fig. 4.

Note that our semantic representation reflects the 'canonical' structure of an active sentence. This design decision was made in order to eliminate structural differences between semantically equivalent structures. Hence, at the semantic level, all passive sentences correspond to their equivalents in the active form. Semantic representation of questions is not always derived directly from the f-structure. For some types of questions, e.g., definition

<i>When was Wendy's founded?</i>	<i>R. David Thomas founded Wendy's in 1969.</i>
ROOT(x6, Wendy's)	ROOT(x6, Wendy's)
ROOT(x2, found)	ROOT(x2, ound)
ADJUNCT(x2,x1)	ADJUNCT(x2,x1)
OBJECT(x2,x6)	OBJECT(x2,x6)
SUBJECT(x2,x7)	SUBJECT(x2,x7)
	ROOT(x7, R. David Thomas)
TYPE(x2, event)	TYPE(x2, event)
TENSE(x2, past)	
ROOT(x1,x9)	ROOT(x1, 1969)
TYPE(x1, time)	TYPE(x1, time)
ANS(x9)	

Figure 4: An example of question and passage semantic representation

questions such as *What is the definition of hazmat?*, specialized (dedicated) grammars are used, which allows us to more easily arrive at an appropriate representation of meaning. Also, in the preliminary implementation of the unification algorithm (see Section 5), we have adopted some simplifying assumptions, and we do not incorporate sets in the current representation.

The present formalism can quite successfully handle questions (or sentences) which refer to specific events or relations. However, it is more difficult to represent questions like *What is the relationship between Jesse Ventura and Target Stores?*, which seek a relation between entities or a common event they participated in. In the next section, we discuss the unification scheme which allows us to select answer candidates based on the proposed representation.

5 Fuzzy Unification

A unification algorithm is required to match question representations with the representations of extracted passages which might contain answers. Using a precursor

predicate	example	comments
ROOT	ROOT(x13, John)	the root form of entity/event x13
OBJECT	OBJECT(x2,x3)	x3 is the object of verb or preposition x2
SUBJECT	SUBJECT(x2,x3)	x3 is the subject of verb x2
DET	DET(x2,x1)	x1 is a determiner/quantifier of x2
TYPE	TYPE(x3, event)	x3 is of the type <i>event</i>
TENSE	TENSE(x1, present)	x1 is a verb in <i>present</i> tense
EQUIV	EQUIV(x1,x3)	semantic equivalence: apposition: "John, a student of CMU" equality operator in copular sentences: "John is a student of CMU"
ATTRIBUTE	ATTRIBUTE(x1,x3)	x3 is an adjective modifier of x1: adjective-noun: "stupid John" copular constructions: "John is stupid"
PREDICATE	PREDICATE(x2,x3)	copular constructions: "Y is x3" ROOT(x2, be) SUBJECT(x2,Y) PREDICATE(x2,x3)
POSSESSOR	POSSESSOR(x2,x4)	x4 is the possessor of x2 "x4's x2" or "x2 of x4"
AND	AND(x3,x1) AND(x3,x2)	"John and Mary laughed." ROOT(x1, John) ROOT(x2, Mary) ROOT(x4, laugh) TYPE(x4, event) AND(x3,x1) AND(x3,x2) SUBJECT(x4,x3)
ANS	ANS(x1)	only for questions: x1 indicates the answer

Figure 3: Examples of predicates

to the representation presented above, we constructed an initial prototype using a traditional theorem prover (Kalman, 2001). Answer extraction was performed by attempting a unification between logical forms of the question and retrieved passages. Early tests showed that a unification strategy based on a strict boolean logic was not as flexible as we desired, given the lack of traditional domain constraints that one normally possesses when considering this type of approach. Unless a retrieved passage exactly matched the question, as in Fig. 4, the system would fail due to lack of information. For instance, knowing that *Benjamin killed Jefferson*. would not answer the question *Who murdered Jefferson?*, using a strict unification strategy.

This has led to more recent experimentation with probabilistic models that perform what we informally refer to as **fuzzy unification**.³ The basic idea of our unification strategy is to treat relationships between question terms as a set of weighted constraints. The confidence score assigned to each extracted answer candidate is related to the number of constraints the retrieved passage satisfies, along with a measure of similarity between the relevant terms.

5.1 Definitions

In this section, we present definitions which are necessary for discussion of the similarity measure employed by our fuzzy unification framework.

Given a user query Q , where Q is a formula, we retrieve a set of passages \mathbf{P} . Our task is to find the best passage $P_{best} \in \mathbf{P}$ from which an answer candidate can be extracted. An answer candidate exists within a passage P if the result of a fuzzy unification between Q and P results in the single term of $\text{ANS}(x0)$ being ground in a term from P .

$$(2) P_{best} = \operatorname{argmax}_{P \in \mathbf{P}} \mathbf{p} \operatorname{sim}(Q, P)$$

The restriction that an answer candidate must be found within a passage P must be made explicit, as our notion of fuzzy unification is such that a passage can unify against a query with a non-zero level of confidence even if one or more constraints from the query are left unsatisfied. Since the final goal is to find and return the best possible answer, we are not concerned with those passages which seem highly related yet do not offer answer candidates.

In Section 4, we introduced extrinsic literals where predicates serve as relations over two labels. Extrinsic literals can be thought of as relations defined over distinct

³Fuzzy unification in a formal setting generally refers to a unification framework that is employed in the realm of fuzzy logics. Our current representation is of an ad-hoc nature, but our usage of this term does foreshadow future progression towards a representation scheme dependent on such a formal, non-boolean model.

entities in our formula. For example, $\text{SUBJECT}(x1, x2)$ is an extrinsic literal, while $\text{ROOT}(x1, |\text{Benjamin}|)$ is not. The latter has been defined as an intrinsic literal in Section 4 and it relates a label and a word.

This terminology is motivated by the intuitive distinction between intrinsic and extrinsic properties of an entity in the world. We use this distinction as a simplifying assumption in our measurements of similarity, which we will now explain in more detail.

5.2 Similarity Measure

Given a set of extrinsic literals P^E and Q^E from a passage and the question, respectively, we measure the similarity between Q^E and a given ordering of P^E as the geometric mean of the similarity between each pair of extrinsic literals from the sets Q^E and P^E .

Let \mathbf{O} be the set of all possible orderings of P^E , O an element of \mathbf{O} , Q_j^E literal j of Q^E , and O_j literal j of ordering O . Then:

$$(3) \operatorname{sim}(Q, P) = \operatorname{sim}(Q^E, P^E) \\ = \max_{O \in \mathbf{O}} \left(\prod_{j=0}^n \operatorname{sim}(Q_j^E, O_j) \right)^{\frac{1}{n}}$$

The similarity of two extrinsic literals, l^E and $l^{E'}$, is computed by the square root of the similarity scores of each pair of labels, multiplied by the weight of the given literal, dependent on the equivalence of the predicates p, p' of the respective literals $l^E, l^{E'}$. If the predicates are not equivalent, we rely on the engineers tactic of assigning an epsilon value of similarity, where ϵ is lower than any possible similarity score⁴. Note that the similarity score is still dependent on the weight of the literal, meaning that failing to satisfy a heavier constraint imposes a greater penalty than if we fail to satisfy a constraint of lesser importance.

Let t_j and t'_j be the respective j -th term of $l^E, l^{E'}$. Then:

$$(4) \operatorname{sim}(l^E, l^{E'}) = \operatorname{weight}(l^E) * \\ \begin{cases} (\operatorname{sim}(t_0, t'_0) * \operatorname{sim}(t_1, t'_1))^{\frac{1}{2}}, p=p' \\ \epsilon, \text{otherwise} \end{cases}$$

The weight of a literal is meant to capture the relative importance of a particular constraint in a query. In standard boolean unification the importance of a literal is uniform, as any local failure dooms the entire attempt.⁵ In a non-boolean framework the importance of one literal vs. another becomes an issue. As an example, given a question concerning a murder we might be more interested in the suspect's name than in the fact that he was tall. This

⁴The use of a constant value of ϵ is *ad hoc*, and we are investigating more principled methods for assigning this penalty.

⁵That is to say, classic unification is usually an all or nothing affair.

idea is similar to that commonly seen in information retrieval systems which place higher relative importance on terms in a query that are judged a priori to possess higher information value. While our prototype currently sets all literals with a weight of 1.0, we are investigating methods to train these weights to be specific to question type.

Per our definition, all terms within an extrinsic literal will be labels. Thus, in equation (10), t_0 is a label, as is t_1 , and so on. Given a pair of labels, b and b' , we let I, I' be the respective sets of intrinsic literals from the formula containing b, b' such that for all intrinsic literals $l^I \in I$, the first term of l^I is b , and likewise for b', I' .

Much like similarity between two formulae, the similarity between two labels relies on finding the maximal score over all possible orderings of a set of literals.

Now let \mathbf{O} be the set of all possible orderings of I' , O an element of \mathbf{O} , I_j the j -th literal of I , and O_j the j -th literal of O . Then:

$$(5) \quad \text{sim}(b, b') = \max_{O \in \mathbf{O}} (\prod_{j=0}^n \text{sim}(I_j, O_j))^{\frac{1}{n}}$$

We measure the similarity between a pair of intrinsic literals as the similarity between the two words multiplied by the weight of the first literal, dependent on the predicates p, p' of the respective literals being equivilant.

(6)

$$\text{sim}(l^I, l^{I'}) = \text{weight}(l^I) * \begin{cases} \text{sim}(t_1, t'_1), p=p' \\ \epsilon, \text{otherwise} \end{cases}$$

The similarity between two words is currently measured using a WordNet distance metric, applying weights introduced in (Moldovan et al., 2003). We will soon be integrating metrics which rely on other dimensions of similarity.

5.3 Example

We now walk through a simple example in order to present the current framework used to measure the level of constraint satisfaction (confidence score) achieved by a given passage. While a complete traversal of even a small passage would exceed the space available here, we will present a single instance of each type of usage of the $\text{sim}()$ function.

If we limit our focus to only a few key relationships, we get the following analysis of a given question and passage.

(7) *Who killed Jefferson?*

ANS(x0), ROOT(x1,x0), ROOT(x2,|kill|),
 ROOT(x3,|Jefferson|), TYPE(x2,|event|),
 TYPE(x1,|person|), TYPE(x3,|person|), SUB-
 JECT(x2,x1), OBJECT(x2,x3)

(8) *Benjamin murdered Jefferson.*
 ROOT(y1,|Benjamin|), ROOT(y2,|murder|),
 ROOT(y3,|Jefferson|), TYPE(y2,|event|),
 TYPE(y1,|person|), TYPE(y3,|person|), SUB-
 JECT(y2,y1), OBJECT(y2,y3)

Computing the similarity between two formulae, (loosely referred to here by their original text), gives the following:

$$(9) \quad \text{sim}[\text{Who killed Jefferson?}, \text{Benjamin murdered Jefferson.}] = (\text{sim}[\text{SUBJECT}(x2,x1), \text{SUBJECT}(y2,y1)] * \text{sim}[\text{OBJECT}(x2,x3), \text{OBJECT}(y2,y3)])^{\frac{1}{2}}$$

The similarity between the given extrinsic literals sharing the predicate SUBJECT:

$$(10) \quad \text{sim}[\text{SUBJECT}(x2,x1), \text{SUBJECT}(y2,y1)] = (\text{sim}[x2, y2] * \text{sim}[x1, y1])^{\frac{1}{2}} * \text{weight}[\text{SUBJECT}(x2,x1)]$$

In order to find the result of this extrinsic similarity evaluation, we need to determine the similarity between the paired terms, $(x1,y1)$ and $(x2,y2)$. The similarity between $x1$ and $y1$ is measured as:

$$(11) \quad \text{sim}[x2, y2] = (\text{sim}[\text{ROOT}(x2,|kill|), \text{ROOT}(y2,|murder|)] * \text{sim}[\text{TYPE}(x2,|event|), \text{TYPE}(y2,|event|)])^{\frac{1}{2}}$$

The result of this function depends on the combined similarity of the intrinsic literals that relate the given terms to values. The similarity between one of these intrinsic literal pairs is measured by:

$$(12) \quad \text{sim}[\text{ROOT}(x2,|kill|), \text{ROOT}(y2,|murder|)] = \text{sim}[|kill|, |murder|] * \text{weight}[\text{ROOT}(x2,|kill|)]$$

Finally, the similarity between a pair of words is computed as:

$$(13) \quad \text{sim}[|kill|, |murder|] = 0.8$$

As stated earlier, our similarity metrics at the word level are currently based on recent work on WordNet distance functions. We are actively developing methods to complement this approach.

6 Summary and Future Work

The paper presents a lightweight semantic processing technique for open-domain question answering. We propose a uniform semantic representation for questions and passages, derived from their functional structure. We also describe the unification framework which allows for flexible matching of query terms with retrieved passages.

One characteristic of the current representation is that it is built from grammatical functions and does not utilize a canonical set of semantic roles and concepts. Our overall approach in JAVELIN was to start with the simplest form of meaning-based matching that could extend simple keyword-based approaches. Since it was possible to extract grammatical functions from unrestricted text fairly quickly (using KANTOO for questions and the Link Grammar parser for answer passages), this framework provides a reasonable first step. We intend to extend our representation and unification algorithm by incorporating the Lexical Conceptual Structure Database (Dorr, 2001), which will allow us to use semantic roles instead of grammatical relations as predicates in the representation. We also plan to enrich the representation with temporal expressions, incorporating the ideas presented in (Han, 2003).

Another limitation of the current implementation is the limited scope of the similarity function. At present, the similarity function is based on relationships found in WordNet, and only relates words which belong to the same syntactic category. We plan to extend our similarity measure by using name lists, gazetteers and statistical cooccurrence in text corpora. A complete approach to word similarity will also require a suitable algorithm for reference resolution. Unrestricted text makes heavy use of various forms of co-reference, such as anaphora, definite description, etc. We intend to adapt the anaphora resolution algorithms used in KANTOO for this purpose, but a general solution to resolving definite reference (e.g., the use of “the organization” to refer to “Microsoft”) is a topic for ongoing research.

Acknowledgments

Research presented in this paper has been supported in part by an ARDA grant under Phase I of the AQUAINT program. The authors wish to thank all members of the JAVELIN project for their support in preparing the work presented in this paper. We are also grateful to two anonymous reviewers, Laurie Hiyakumoto and Kathryn Baker for their comments and suggestions for improving this paper. Needless to say, all remaining errors and omissions are entirely our responsibility.

References

- BBN Technologies, 2000. *IdentiFinder User Manual*.
- Joan Bresnan, editor. 1982. *The Mental Representation of Grammatical Relations*. MIT Press Series on Cognitive Theory and Mental Representation. The MIT Press, Cambridge, MA.
- Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21:543–565.
- Jenifer Chu-Carroll, John Prager, Christopher Welty, Krzysztof Czuba, and David Ferrucci. 2003. A multi-strategy and multi-source approach to question answering. In *TREC 2002 Proceedings*.
- Bonnie J. Dorr. 2001. LCS Database Documentation. HTML Manual. available from http://www.umiacs.umd.edu/~bonnie/LCS_Database_Documentation.html.
- Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.
- Dennis Grinberg, John Lafferty, and Daniel Sleator. 1995. A robust parsing algorithm for link grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, Prague, September.
- Benjamin Han. 2003. Text temporal analysis. Research status summary. Draft of January 2003.
- Ulf Hermjakob. 2001. Parsing and question classification for question answering. In *Proceedings of the Workshop on Open-Domain Question Answering at ACL-2001*.
- Eduard Hovy, Ulf Hermjakob, and Chin-Yew Lin. 2002. The use of external knowledge in factoid qa. In *Proceedings of the TREC-10 Conference*.
- Abraham Ittycheriah and Salim Roukos. 2003. IBM’s statistical question answering system – TREC-11. In *TREC 2002 Proceedings*.
- Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2002. IBM’s statistical question answering system – TREC-10. In *TREC 2001 Proceedings*.
- John A. Kalman. 2001. *Automated Reasoning with OTTER*. Rinton Press.
- Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. 2000. The structure and performance of an open-domain question answering system. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL-2000)*.
- Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu, and Orest Bolohan. 2003. LCC tools for question answering. In *TREC 2002 Proceedings*.
- Eric Nyberg and Teruko Mitamura. 2000. The KANTOO machine translation environment. In *Proceedings of AMTA 2000*.

Eric Nyberg, Teruko Mitamura, Jaime Carbonell, Jaime Callan, Kevyn Collins-Thompson, Krzysztof Czuba, Michael Duggan, Laurie Hiyakumoto, Ng Hu, Yifen Huang, Jeongwoo Ko, Lucian V. Lita, Stephen Murtagh, Vasco Pedro, and David Svoboda. 2003. The JAVELIN question answering system at TREC 2002. In *TREC 2002 Proceedings*.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a Question Answering system. In *Proceedings of the ACL Conference*.

Martin M. Soubbotin. 2002. Patterns of potential answer expressions as clues to the right answer. In *TREC 2001 Proceedings*.