LCC Tools for Question Answering

Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu and Orest Bolohan

Language Computer Corporation
Richardson, TX 75080
email: moldovan@languagecomputer.com

Abstract

The increased complexity of the TREC QA questions requires advanced text processing tools that rely on natural language processing and knowledge reasoning. This paper presents the suite of tools that account for the performance of the PowerAnswer question answering system. It is shown how questions, answers and world knowledge are transformed first in logic representation, followed by a systematic and rigorous logic proof that validly answers questions posed to the QA system. At TREC QA 2002, PowerAnswer obtained a confidence-weighted score of 0.856, answering correctly 415 out of 500 questions.

1 Introduction

To cope with the continuously increasing difficulty of the TREC QA task, LCC has enhanced the processing capability of the QA system by improving some modules and strengthening the set of tools involved in answer extraction. This paper presents a general overview of the main tools and focuses on some of them.

PowerAnswer, the QA system developed at LCC, searches for answers from large collections of texts by combining syntactic, semantic, lexical and world knowledge information sources. PowerAnswer consists of three main parts: question processing, document retrieval and answer extraction. In turn, each part consists of smaller modules that work collectively to produce question answers. The relative performance of these modules was described in [Moldovan et al 2002].

Questions and relevant document paragraphs are parsed and transformed into logic forms that together with world knowledge axioms extracted from the WordNet glosses are fed to a logic prover. For each question, the result is a set of ranked answers accompanied by their measures of beliefs.

Advanced QA requires sophisticated text processing tools based on the state-of-the-art NLP and reasoning methods. The LCC tool set includes: Name Entity Recognizer, Syntactic Parser, Logic Form Transformer, Word Sense Desambiguator, Lexical Chainer, Logic Prover and others. In addition, since PowerAnswer operates in comercial environments, it has a set of support tools such as System Manager, PowerAnalytics, PowerIndex, Power Ontology, and Document Manager that enhance its functionality. The main focus of this paper is on Lexical Chainer and Logic Prover.

2 Name Entity Recognizer

Name Entity Recognition systems identify named-entities such as people, organizations, dates, places, quantities and others in text documents. LiteNE, the LCC Name Entity Recognizer is implemented as a cascade of finite-state automata interleaved with other preprocessing and coreference resolution.

The *tokenizer* is the first module of the LiteNE system. The task of the tokenizer is to break the document into lexical entities (tokens). Generally, a token is a word or a punctuation sign, but in some cases it may be a word fragment. For example, the word "U.S.A." is broken into six tokens: "U", ".", "S", ".", "A", and ".". It is the task of the next module to identify the above sequence as a complex word representing an abbreviation of "United States of America".

The *Lexicon* module identifies words and complex words that have lexical and semantic meaning. This is done by inspecting both dictionaries and gazetteers. Dictionaries contain open-domain lexico-semantic information, e.g. "house" is an artifact, or to a lesser extent domain-specific information, e.g. "mail bomb" is a kind of bomb. Gazetteers typically store well-known entity names, such as locations, e.g. "Dallas" is a city in the state of "Texas" part of the country "United States of America".

The *Preprocessor* identifies simple lexical entries that are not stored in lexicon or gazetteers. Some of the items identified by the preprocessor are: phone numbers "1-(800) 888-7777", money "$4.75", dates "December 25", times "8:30am", measures such as "10kg", "one hundred degrees", and others.

The core *Name Entity Recognizer* assigns lexical features to words or groups of words such as locations, organizations, persons, addresses, and others. Proper names are particularly useful for extraction systems since they point to objects about which we need to identify properties, relations, events. The technique is to use capitalization if available. Some of the most frequently used methods are Hidden Markov Models and finite state automata patterns. With the help of dictionaries these techniques are able to recognize that "John Smith" is a proper name, and "John Hopkins" is a University; or that "Austin Ventures" is a company, "Austin, Texas" is a city and "Austin Thomas" is a name. Machine learning methods were used to train the LiteNE system. NE-recognition benefits by morphological analysis by looking up in a dictionary for all morphological variations of words.

Part of Speech Tagging is useful for subsequent text analysis stages. This involves specifying the part of speech of each word. POS tagger combines rule-based and statistical methods and achieves an accuracy around 96%.

3 Syntactic Parser

In advanced QA, like in many other NLP applications, syntactic parsing plays a major role in the overall system accuracy, especially since subsequent steps rely on it. In the last few years, LCC developed its own parser, and a version was trained for QA, meaning it has the capability of parsing questions as well as free text. It is a probabilistic parser which has been improved over the years.

Parser identifies simple noun phrases ("the fast red car"), verb phrases ("is being obverved daily"), and also particles that may be significant in subsequent text analysis. It recognizes phrases and solves the attachment of prepositional phrases and close subordination. Full syntactic parsing takes long time to process even a small number of text documents. Since we reduce documents to relevant passages we can afford to fully parse these passages. The quality of parser affects the accuracy of subsequent steps.

Coreference Resolution is the task of determining that a noun phrase refers to the same entity as another noun phrase. This involves equating various forms of personal proper names, for example "President Bush", "George Bush", "the 43rd President of US", etc. There are other more complex forms of coreference such as definite or indefinite noun phrase and pronoun coreference that have been implemented. Also, a light form of temporal coreference resolution has been implemented.

4 Logic Form Representation

The logic form (LF) is an intermediary step between syntactic parse and the deep semantic form. The LF codification acknowledges syntax-based relationships such as: (1) syntactic subjects, (2) syntactic objects, (3) prepositional attachments, (4) complex nominals, and (5) adjectival/adverbial adjuncts.

There are two criteria that guide our approach: (1) the notation be as close as possible to English, and (2) the notation be syntactically simple. Our approach is to derive the *LF directly from the output of the syntactic parser.* The parser resolves the structural and syntactic ambiguities.

The basis of integrating a Logic Form representation into the PowerAnswer system is that all questions and relevant paragraphs are transformed into an unambiguous logic representation. The term Answer Logic Form (ALF) refers to the candidate answers in logic form. Candidate answers returned by the Answer Extraction module are classified as free text due to the unpredictable nature of their grammatical structure. The term Question Logic Form (QLF) refers to the questions posed the Question Answering system in logic form.

Essentially there is a one to one mapping of the words of the text into the predicates in the logic form. The predicate names consist of the base form of the word concatenated with the part of speech of the word. Each noun has an argument that is used to represent it in other predicates. One of the most important features of the Logic Form representation is the fixed-slot allocation mechanism of the verb predicates. This allows for the Logic Prover to see the difference between the role of the subjects and objects in a sentence that is not answerable in a keyword based situation.

Logic Forms are derived from the grammar rules found in the parse tree of a sentence. There are far too many grammar rules in the English language to efficiently and realistically implement them all. We have observed that the top ten most frequently used grammar rules cover 90% of the cases for WordNet glosses. This is referred to as the 10-90 rule. Below we provide a sample sentence and its corresponding LF representation. More details regarding the transformation of text into logic forms are presented in [Moldovan and Rus 2001].

Example:
Heavy selling of Standard & Poor's 500-stock index futures in Chicago relentlessly beat stocks downward.

LF:
heavy_JJ(x1) & selling_NN(x1) & of_IN(x1,x6) & Standard_NN(x2) & &_CC(x13,x2,x3) & Poor_NN(x3) &'s_POS(x6,x13) & 500-stock_JJ(x6) & index_NN(x4) & future_NN(x5) & nn_NNC(x6,x4,x5) & in_IN(x1,x8) & Chicago_NN(x8) & relentlessly_RB(e12) & beat_VB(e12,x1,x9) & stocks_NN(x9) & downward_RB(e12)

5 Lexical Chains

A major problem in QA is that often an answer is expressed with words different from the question keywords. In such cases it is useful to find topically related words to the question keywords. By exploiting the information in the WordNet glosses, the connectivity between the synsets is dramatically increased. When a word in a gloss is semantically disambiguated, it points to the synset it belongs to. We call this extended WordNet (XWN) [Harabagiu, Miller and Moldovan 1999]. In the context of XWN, or any other lexical database, topical relations can be expressed as *lexical chains*. These are sequences of semantically related words that link two concepts. Lexical chains have been used in computational linguistics to study: discourse, coherence, inference, implicatures malapropisms and others [Morris and Hirst 1991], [Hirst and St-Onge 1998], [Harabagiu and Moldovan 1998b]. Lexical chains improve the performance of question answering systems in two ways: (1) increase the document retrieval recall and (2) improve the answer extraction by providing the much needed world knowledge axioms that link question keywords with answers concepts.

It is possible to establish some connections between synsets via topical relations. We developed software that automatically provides connecting paths between any two WordNet synsets S_i and S_j up to a certain distance. The meaning of these paths is that the concepts along a path are topically related.

In Table 1 we show a few examples of morphologically related words that appear in synsets and their glosses which are brought to bear by the topical relations.

Synset	**Gloss**	**Morphological relation**
laughter:n#1	(the sound of laughing:v#1)	noun - verb
immediately:r#3	(bearing an immediate:a#2 relation)	adverb - adjective
insure:v#4	(take out insurance:n#3 for)	verb - noun
parental:adj#2	(..characteristic of or befitting a parent:n#1)	adjective - noun

Table 1: Examples of new morphological relations revealed by the topical relations

Examples

Below we provide the most relevant lexical chains that link some selected TREC 2002 questions with their answers.

Q1403: When was the internal combustion engine invented ?
Answer: The first internal - combustion engine was built in 1867
Lexical chains:
(1) invent:v#1 → HYPERNYM → create_by_mental_act:v#1 → HYPERNYM → create:v#1 → HYPONYM → build:v#1

Q1404: How many chromosomes does a human zygote have ?
Answer: 46 chromosomes that lie in the nucleus of every normal human cell
Lexical chains:
(1) zygote:n#1 → HYPERNYM → cell:n#1
(2) zygote:n#1 → HYPERNYM → cell:n#1 →HAS_PART → nucleus:n#1

Q1411: What Spanish explorer discovered the Mississippi River ?
Answer: Spanish explorer Hernando de Soto reached the Mississippi River
Lexical chains:
(1) discover:v#7 → GLOSS → reach:v#1

Q1462: Where is the oldest synagogue in the United States ?
Answer: Newport is marking the 350th anniversary of the founding of Trinity Church , and is also home to the nation 's oldest synagogue
Lexical chains:
(1) United_States:n#1 → HYPERNYM → North_American_country:n#1 → HYPERNYM → country:n#1 → GLOSS → nation:n#1

Q: 1518 What year did Marco Polo travel to Asia ?
Answer: Marco Polo divulged the truth after returning in 1292 from his travels , which included several months on Sumatra.
Lexical chains:
(1) travel_to:v#1 → GLOSS → travel:v#1 → RGLOSS → travel:n#1
(2) travel_to#1 → GLOSS → travel:v#1 → HYPONYM → return:v#1
(3) Sumatra:n#1 → ISPART → Indonesia:n#1 → ISPART → Southeast_Asia:n#1 → ISPART → Asia:n#1

Q: 1540 What is the deepest lake in America ?
Answer: Rangers at Crater Lake National Park in Oregon have closed the hiking trail to the shore of the nation 's deepest lake
Lexical chains:
(1) America:n#1 → HYPERNYM → North_American_country:n#1 → HYPERNYM → country:n#1 → GLOSS → nation:n#1

6 Logic Prover

Usefulness of a Logic Prover in Question Answering
The LCC Logic Prover renders a deep understanding of the relationship between the question text and answer text. The Logic Prover captures the syntax-based relationships such as the syntactic objects, syntactic subjects, prepositional attachments, complex nominals, and adverbial/adjectival adjuncts provided by the LF representation. In addition to the LF representations of questions and candidate answers, the Logic Prover needs world knowledge axioms to link questions to answers. For this, the Logic Prover uses the Lexical Chains to bring to the forefront the most important logic axions needed in a proof. In XWN, an axiom is the LF expression of a synset and its gloss. With this deep and intelligent representation, the Logic Prover effectively and efficiently re-ranks candidate answers by their correctness and ultimately eliminates incorrect answers. In this way, the Logic Prover is a powerful tool in boosting the accuracy of the PowerAnswer system. Moreover, the trace of a proof constitutes a justification for that answer.

LCC's Logic Prover
The base of LCC's Logic Prover is Otter, an automated reasoning system developed at Argonne Labs. Extensions were made to customize Otter to the Question Answering task. The inference rule sets are based on hyperresolution and paramodulation. Hyperresolution is an inference rule that does multiple binary resolution steps in one, where binary resolution is an inference mechanism that looks for a positive literal in one clause and negative form of that same literal in another clause such that the two literals can be canceled, resulting in a newly inferred clause. Paramodulation introduces the notion of equality substitution so that axioms representing equality in the proof do not need to be explicitly included in the axiom lists. Additionally, similar to hyperresolution, paramodulation combines multiple substitution steps into one.

The search strategy used is the Set of Support Strategy, which partitions the axioms used during the course of a proof into those that have support and those that are considered auxiliary. The axioms with support are placed in the Set of Support (SOS) list and are intended to guide the proof. The auxiliary axioms are placed in the Usable list and are used to help the SOS infer new clauses. This strategy restricts the search such that a new clause is inferred if and only if one of its parent clauses come from the Set of Support. The axioms that are placed in the SOS are the candidate answers, the question negated (to invoke the proof by contradiction), axioms related to linking named entities to answer types, and axioms related to decomposing conjunctions, possessives, and complex nominals. Axioms placed in the Usable list are the WordNet axioms and other axiom based outside world knowledge.

The Logic Prover will continue trying to find a proof until one of two conditions is met; either the Set of Support becomes empty or a refutation is found.

Examples of answer justification in action:

Example 1.

Question 1797: How did Adolf Hitler die ?
QLF: manner_AT(e1) & adolf_nn(x2) & hitler_nn(x3) & nn_nnc(x4,x2,x3) & die_vb(e1,x4,x1)

Question Axiom:
-(exists e1 x1 x2 x3 x4 (adolf_nn(x2) & hitler_nn(x3) & nn0_nnc(x4,x2,x3) & die_vb(e1,x4,x1))).

Answer:
It was Zhukov 's soldiers who planted a Soviet flag atop the Reichstag on May 1 , 1945 , a day after Adolf Hitler committed suicide.

We introduce a psuedo-verb for suicide since the WordNet gloss for the noun suicide lets us infer that suicide is an act and therefore can be treated as a verb.

ALF:
It_PRP(x14) & be_VB(e1,x14,x2) & Zhukov_NN(x1) & 's_POS(x2,x1) & soldier_NN(x2) & plant_VB(e2,x2,x3) & Soviet_JJ(x3) & flag_NN(x3) & atop_IN(e2,x4) & Reichstag_NN(x4) & on_IN(e2,x8) & May_NN(x5) & 1_NN(x6) & 1945_NN(x7) & nn_NNC(x8,x5,x6,x7) & day_NN(x9) & Adolf_NN(x10) & Hitler_NN(x11) & nn_NNC(x12,x10,x11) & commit_VB(e3,x12,x13) & suicide_NN(x13) & suicide_VB(x13,x19,x12)

Answer Axiom:
exists e1 e2 e3 e4 x1 x10 x11 x12 x13 x14 x17 x18 x19 x2 x3 x4 x5 x6 x7 x8 x9 (it_prp(x14) & be_vb(e1,x14,x2) & zhukov_nn(x1) & _s_pos(x2,x1) & soldiers_nn(x2) & planted_vb(e2,x2,x3) & soviet_jj(x3) & flag_nn(x3) & atop_in(e2,x4) & reichstag_nn(x4) & on_in(e2,x8) & may_nn(x5) & 1_nn(x6) & 1945_nn(x7) & nn_nnc(x8,x5,x6,x7) & day_nn(x9) & adolf_nn(x10) & hitler_nn(x11) & nn_nnc(x12,x10,x11) & commit_vb(e3,x12,x13) & suicide_nn(x13) & suicide_vb(x13,x19,x12)).

Wordnet Relations:
Suicide is a manner of killing.
Suicide_NN(e1) → kill_NN(e1) & manner_AT(e1)
Axiom:
all e1 (suicide_nn(e1) → kill_nn(e1) & manner_at(e1)).

Pseudo-Verb Wordnet Gloss:
Suicide is the act of killing yourself
Gloss Logic Form:
suicide_VB(e1,x1,x2) ↔ kill_VB(e2,x1,x2) & yourself_PRP(x2) *Axiom:* all e1 x1 x2 (suicide_vb(e1,x1,x2) → kill_vb(e1,x1,x2) & yourself_nn(x2)).

Wordnet Gloss:
To kill is to cause to die
Gloss Logic Form:
kill_VB(e1,x1,x2) ↔ cause_VB(e2,x1,e3) & die_VB(e3,x2,x4)
Axiom:
all e1 e2 e3 x1 x2 x4 (kill_vb(e1,x1,x2) ↔ cause_vb(e2,x1,e3) & die_vb(e3,x2,x4)).

Linguistic Axioms:
Link the noun kill to the verb kill
all e1 e2 x1 x2 (kill_nn(e1) & kill_vb(e2,x1,x2) → kill_vb(e1,x1,x2)).

Make the yourself predicate relfexively used in the verb kill
all e1 x1 x2 (kill_vb(e1,x1,x2) & yourself_nn(x2) → yourself_nn(x1)).

The relevent steps in the proof:
1 [] -manner_at(x15) | -adolf_nn(x2) | -hitler_nn(x3) | -nn_nnc(x4,x2,x3) | -die_vb(x15,x4,x1).
(The question negated to invoke a proof by contradiction)
18 [] adolf_nn($c16).
19 [] hitler_nn($c15).
20 [] nn_nnc($c14,$c16,$c15).
22 [] suicide_nn($c13).
23 [] suicide_vb($c13,$c9,$c14).
(The Logic Prover selects the above clauses from the answer)
25 [] -suicide_nn(x16) | manner_at(x16).
(The Logic Prover selects the axiomatic knowledge extracted from Wordnet that suicide is a manner of killing)
29 [] -kill_vb(x23,x1,x2) | die_vb(x23,x2,$c23).
(The Logic Prover selects the WordNet gloss for kill implies die)
30 [] -suicide_vb(x24,x1,x3) | kill_vb(x24,x1,x3).
(The Logic Prover selects the WordNet gloss for suicide implies kill) 32 [hyper,22,25] manner_at($c13).
35 [hyper,23,30] kill_vb($c13,$c9,$c14).
36 [hyper,35,29] die_vb($c13,$c14,$c23).
39 [hyper,1,32,18,19,20,36] $F.

In the final step the terms for adolf_nn($c16), hitler_nn($c15), nn_nnc($c14,$c16,$c15), manner_at($c13), and die_vb($c13,$c14,$c23) are hyperresolved with the negated question to yield a full proof by contradiction.

Example 2.

Question 1512: What is the age of our solar system ?

QLF:
_quantity_AT(x2) & age_NN(x2) & of_IN(x2,x3) & solar_JJ(x3) & system_NN(x3)

Question Axiom:
-(exists x1 x2 x3 (_quantity_at(x2) & age_nn(x2) & of_in(x2,x3) & solar_jj(x3) & system_nn(x3))).

Answer:
The solar system is 4.6 billion years old

ALF:
solar_JJ(x5) & system_NN(x5) & 4.6_NN(x2) & billion_NN(x3) & year_NN(x4) & nn_NNC(x5,x2,x3,x4) & old_JJ(x5)

Answer Axiom:
exists e1 x1 x2 x3 x4 x5 x7 (solar_jj(x5) & system_nn(x5) & 4_6_nn(x2) & billion_nn(x3) & years_nn(x4) & nn_nnc(x5,x2,x3,x4) & old_jj(x5)).

Wordnet Gloss:
Old is having lived for a relatively long time or attained a specific age.

Gloss Logic Form:
old_JJ(x6) ↔ live_VB(e2,x6,x2) & for_IN(e2,x1) & relatively_JJ(x1) & long_JJ(x1) & time_NN(x1) & or_CC(e5,e2,e3) & attain_VB(e3,x6,x2) & specific_JJ(x2) & age_NN(x2)

Axiom:
all e2 e3 e5 x1 x2 x6 (old_jj(x6) ↔ live_vb(e2,x6,x2) &for_in(e2,x1) & relatively_jj(x1) & long_jj(x1) & time_nn(x1) & or_cc(e5,e2,e3) & attain_vb(e3,x6,x2) & specific_jj(x2) & age_nn(x2)).

Named entity axioms:
all x2 x3 x4 x5 (4_6_nn(x2) & billion_nn(x3) & years_nn(x4) & nn_nnc(x5,x2,x3,x4) → _quantity_at(x5)).

Linguistic axioms:
all x1 (_quantity_at(x1) & solar_jj(x1) & system_nn(x1) → of_in(x1,x1)).

The relevent steps in the proof:
1 [] -_quantity_at(x2) | -age_nn(x2) | -of_in(x2,x3) | -solar_jj(x3) | -system_nn(x3).
(The question negated to invoke a proof by contradiction)
2 [] solar_jj($c2).
3 [] system_nn($c2).
4 [] 4_6_nn($c5).
5 [] billion_nn($c4).
6 [] years_nn($c3).
7 [] nn_nnc($c2,$c5,$c4,$c3).
8 [] old_jj($c2).
(The Logic Prover selects the above clauses from the answer)
17 [] -old_jj(x6) | age_nn(x2).
(The Logic Prover selects the WordNet gloss for old implies age)
19 [] -4_6_nn(x2) | -billion_nn(x3) | -years_nn(x4) | -nn_nnc(x5,x2,x3,x4) | _quantity_at(x5).
(The Logic Prover selects the named entity axiom linking 4.6 billion years to a quantity)
20 [] -_quantity_at(x1) | -solar_jj(x1) | -system_nn(x1) | of_in(x1,x1).
(The Logic Prover selects the Linguistic axiom implying that if a noun or noun phrase is a quantity then the quantity is an attribute of that noun or noun phrase)
21 [hyper,8,17] age_nn(x).
30 [hyper,7,19,4,5,6] _quantity_at($c2).
31 [hyper,30,20,2,3] of_in($c2,$c2).
32 [hyper,1,30,21,31,2,3] $F.

In the final step the terms for _quantity_at($c2), of_in($c2,$c2), age_nn(x), solar_jj($c2), and system_nn($c2) are hyperresolved with the negated question to yield a full proof by contradiction.

When the proof fails, we devised a way to incrementally relax some of the conditions that hinder the completion of the proof. This relaxation process puts weights on the proof such that proofs weaker than a predefined threshold are not accepted.

7 Other Support Tools

System Manager provides immediate access to all key parameters and indicators for managing and administering the Question Answering system, e.g. answer rate, number of simultaneous questions, static or dynamic content collection, number of documents retrieved, memory size for Natural Language parsing, and many others. With its web-based, easy to use interface, the System Manager allows for all these parameters to be adjusted in real time. The result is an Answer service that follows the needs of the Customer Service department, scaling up and down for the optimal performance and resource allocation.

PowerAnalytics offers continuous customer feedback with up-to-date insight on customer needs, communication patterns and expectations. The advanced management module with friendly and easy-to-use web interfaces makes real-time service monitoring and content adjustment fast and powerful. Companies benefit from low maintenance costs and deep knowledge of business and customer service metrics in both standard and highly customized analytic reports.

PowerIndex makes content integration fast and easy. The system deploys advanced harvesting tools that access, collect and index large amounts of structured or unstructured information automatically, or on request. New and old knowledge is merged seamlessly and re-indexed most efficiently, so that fresh information is disseminated as soon as it is created.

PowerOntology creates deep concept ontologies starting from only a small number of concept seeds. For example, the WordNet ontology is enhanced with domain specific concepts through syntactic transformations, such as extending the seed concept with modified nouns, and through semantic chains, such as ISA-relations.

Document Manager allows for clear management of all structured and unstructured data. It keeps track of multiple document revisions, add-ons or deletions, data changes and document locations. Document Manager also integrates seamlessly with PowerIndex and offers a clear overview on the content resources and their immediate availability for fresh knowledge dissemination through our Question Answering and Information Extraction products.

8 Results at TREC 2002

The performance obtained by PowerAnswer at TREC QA 2002 in the main task is summarized in Table 2.

9 Conclusions

This paper introduced some of the tools that support the operation of LCC's QA system. In particular, it was demonstrated how questions, document paragraphs and world knowledge axioms are formally represented and how a logic prover methodically and efficiently generates correct answers.

Number wrong (W)	63
Number unsupported (U)	14
Number inexact (X)	8
Number right (R)	415
Precision of recognized no answer	0.578
Recall of recognizing no answer	0.804
Confidence-weighted score	**0.856**

Table 2: Performance over 500 questions

Essential in this framework is the extended WordNet which supplies the prover with world knowledge axioms. To cope with future questions that entail implicatures and other complex analyses, nonmonotonic reasoning methods need to be incorporated into the logic prover. It becomes more and more clear that QA is intimately linked with natural language processing, text mining, and reasoning on knowledge bases.

Acknowledgement
This work was supported in part by the ARDA AQUAINT program. We wish to thank Christine Clark, Mihai Surdeanu and Marius Pasca from LCC for their contribution to this work.

References

[Fellbaum 1998] Christiane Fellbaum. WordNet - *An Electronic Lexical Database*, MIT Press, Cambridge, MA, 1998.

[Harabagiu and Moldovan 1998b] S.M. Harabagiu and D.I. Moldovan. Knowledge Processing on an Extended WordNet. WordNet-An Electronic Lexical Database. The *MIT Press*, C. Fellbaum editor, pp 379-406, 1998.

[Harabagiu, Miller and Moldovan 1999] S. Harabagiu, G.A. Miller, and D.I. Moldovan. WordNet 2 - A Morphologically and Semantically Enhanced Resource. *Proceedings of ACL-SIGLEX99: Standardizing Lexical Resources*, Maryland, June 1999, pp.1-8.

[Hirst and St-Onge 1998] G. Hirst and D. St-Onge. Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. WordNet-An Electronic Lexical Database. The *MIT Press*, C. Fellbaum editor, pp 305-332, 1998.

[Moldovan and Rus 2001] D. Moldovan and V. Rus. Logic Form Transformation and its Applicability to Question Answering. In *Proceedings of ACL 2001.*

[Moldovan et al 2002] D. Moldovan, M. Pasca, S. Harabagiu and M. Surdeanu. Performance Issues and error Analysis in an Open-Domain Question Answering System. In *Proceedings of ACL 2002.*

[Moldovan and Novischi 2002] D. Moldovan and A. Novischi. Lexical Chains for Question Answering. In *Proceedings of COLING 2002*, pp.674-680.

[Morris and Hirst 1991] J. Morris and G. Hirst. Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text. In *Computational Linguistics*, Vol. 17 , no 1, pp. 21-48, 1991.