# Question Answering Using Syntax-Based Concept Nodes

Demetrios G. Glinos and Fernando Gomez

School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL 32816

**Abstract.** This paper presents a syntax-based formalism for representing atomic propositions extracted from textual documents. A method is described for constructing a network of concept nodes for indexing such logical forms based on the discourse entities they contain. The decomposition of meaningful questions into Boolean combinations of question patterns using the same formalism is presented, in which free variables represent the desired answers. A method is described for using this formalism for robust question answering using the concept network and WordNet synonym, hypernym, and antonym relationships. Finally, the encouraging performance of an implementation of this formalism against the factoid questions from the 2005 Text Retrieval Conference, which operated upon the AQUAINT document corpus, is discussed.

## 1 Introduction

Question answering (QA) represents an intermediate stage along the path from document retrieval to text understanding. As an area of research interest, it serves as a proving ground where strategies for document processing, knowledge representation, question analysis, and answer extraction may be evaluated in real world information extraction contexts. Such contexts typically involve large text document collections, web content, multimedia documents, or domain-specific databases. In this paper, we present our formalism for processing text documents to produce a concept network for persistent knowledge, for analyzing questions to produce question patterns employing the same formalism, and for robust question answering using the concept network so obtained together with the WordNet lexical resource. We also discuss the performance of an implementation of our methods against the factoid questions from the 2005 Text Retrieval Conference (TREC) Question Answering Track.

## 2 Question Answering Systems

Current QA systems typically involve large text document collections and primarily use keywords or other attributes extracted from the question to restrict the subset of the document collection for consideration and also to extract an answer from the documents and/or passages retrieved [10]. However, systems vary widely in their approaches for achieving these ends.

G. Glinos D. and Gomez F. (2006).
Question Answering Using Syntax-Based Concept Nodes.
In *Proceedings of the 3rd International Workshop on Natural Language Understanding and Cognitive Science*, pages 81-90
DOI: 10.5220/0002477500810090

Some systems, such as that described in [7], are built around a named entity recognizer for selecting the relevant passages, to which surface patterns are applied to extract candidate answers. Other systems, such as [8], process documents to produce a set of data structures which is mined to produce answers. In [6], the authors describe a system in which the input text is processed to produce a set of "quasi-logical forms" (QLFs), from which the answer in obtained by resolving the reference for the question variable. And in [2], questions are resolved by Prolog unification against a set of discourse representation structures.

Mixed strategies are also frequently employed, as in the three-stage approach described in [5], which involved: (i) surface patterns learned from the Web; (ii) semantic type extraction based on a predefined taxonomy; and (iii) searching the document set for occurrences of hyponyms of a key word extracted from the question. A system where different strategies were applied to different question types is described in [9].

By contrast, our approach involves logical forms that explicitly capture the *n-ary* relationships among the syntactical components of atomic propositions, not just their binary relationships, so that ancillary data structures are not needed in order to infer higher order relationships. Moreover, question pattern matching is performed not against the surface text itself, but against the logical forms that are extracted from the text, and includes all features of the input question. Finally, our approach uses a network of concept nodes that is unique in that it is used for: (i) indexing the logical forms; (ii) capturing "is-a" relationships among concepts; and (iii) providing the entry points for extending "is-a" relationships into the WordNet [4] lexical database.

## 3 Knowledge Representation

### 3.1 Proposition Tuples

We propose the notion of a "proposition tuple", which we define as a logical proposition consisting of an atomic predicate and its arguments, as the basic unit of knowledge that is mined for question answering. For a simple sentence, the main verb may be taken as the predicate, with the remaining syntactical components as its arguments. Thus, for *"Peter threw the ball,"* we have the predicate *threw(Peter,ball).* This is not generally the case for more complex sentences. Consider, for example, the following sentence from the AQUAINT corpus used in TREC: *"Tourism is one of the major industries in Port Arthur, a town at the southern tip of the island."* The implementation described in Section 6 of this paper successfully splits off the apposition, producing two simpler sentences: *"Port Arthur is a town at the southern tip of the island,"* and *"Tourism is one of the major industries in Port Arthur."* In this form, each of these simpler sentences can now be expressed as a logical proposition from which the predicate and its arguments may be easily discerned. In our implementation, sentence splitting is performed by heuristics for recognizing appositions, subordinate clauses, non-defining relative clauses, coordinations, and similar syntactical constructs, from which separate sentences are constructed, although it should be noted that our algorithms do not depend upon the particular method for performing such splitting.

Given that the input text can be decomposed into distinct propositions through sentence splitting as defined above, we define the proposition tuple for a given proposition as the 6-tuple

*<subject,verb,gerinf,modifiers,indirect,direct>*

where "subject" refers to the noun phrase representing the subject of the sentence, "verb" refers to the main verb phrase, "gerinf" represents any gerund or infinitive form, "modifiers" refers to adverbials and adverbial complements, typically prepositional phrases, "indirect" refers to the indirect object, if any, and "direct" refers to the direct object, if any. The purpose of separating the gerund/infinitive from the main verb is to facilitate question answering.

Proposition tuples capture the n-ary relationships among all of the non-null syntactical components of the input proposition. Thus, a single tuple may encode the binary relationships between subject and verb, object and verb, subject and object, and between any of the modifiers and other sentence components or even other modifiers. It also encodes higher order relationships, such as a ternary relationship between a given subject, object, and verb, and so on. As such an encoding mechanism, the tuple represents an efficient means of representing all such relationships.

As an example, consider the following sentence, which is also taken from the AQUAINT corpus: *"As a professor at the University of Chicago in the early 1940s, Fermi designed and built the first nuclear reactor that later was put into use for research into nuclear weapons."* There are two propositions contained in this sentence, so that there are, correspondingly, two proposition tuples, the first of which is:

Proposition tuple #1:

| | |
|---|---|
| subject: | Fermi, |
| verb: | designed |
| gerinf: | null |
| modifiers: | as a professor; at the University of Chicago; in the early 1940s |
| indirect: | null |
| direct: | the first nuclear reactor that later was put into use for research into nuclear weapons |

where the second proposition tuple is identical to the first, except that the verb is "built". The three verb modifiers in this example, all of which are prepositional phrases, are separated by semicolons in the modifier field, indicating that they are separate and distinct objects. This supports question answering using any combination of these modifiers, from none to all, in any order. Also noteworthy is the capture of an entire phrasal concept as the direct object. Phrasal concepts form the basis of the concept network discussed in the next subsection.

### 3.2 Concept Network

Given that a proposition tuple represents the interrelationships among the non-null syntactical components of the associated proposition, the tuple is relevant for answering questions that pertain to any of the discourse entities contained in any of these components. Accordingly, as an indexing mechanism, we create "concept nodes" for all such entities and associate the tuple with each such node. For these purposes, a discourse entity is taken to be each noun phrase contained in a subject, indirect object, and direct object, and the noun phrase objects of prepositional modifiers. Direct quotes, subordinate clauses, and other phrasal concepts are not further decomposed at this stage. For example, given the sentence *"Peter enjoyed reading the book that Mary recommended,"* we generate the simple proposition tuple *<'Pe-*

*ter','enjoyed','reading',null,null,'the book that Mary recommended'>*, from which we create concept nodes for 'Peter', 'Mary', and 'the book that Mary recommended'.

To construct a concept set that will support complex question answering, we consider, for example, a document containing the sentences:

*The Russian submarine Kursk sank in the Barents Sea.*
*The Russian submarine Kursk sank in deep water.*

If the question were *"Where did the submarine sink?"* both *"in the Barents Sea"* and *"in deep water"* could arguably be acceptable answers. However, if the question were *"In what sea did the submarine sink?"* then only the first would be acceptable. Since the propositions represented by these sentences are syntactically identical, we distinguish them by postulating a mechanism by which we recognize *"the Barents Sea"* as an instance of the *"sea"* category contained in the question.

The method for distinguishing these cases is to encode all "is-a" relationships as explicit parent-child relationships among the corresponding concept nodes. Thus, if the document also contained a sentence stating, essentially, that the Barents Sea is a sea, this would be sufficient. However, we observe anecdotally that neither ordinary conversation nor newswire text typically contains such explicit categorizations. Therefore, as a corollary to the rule above, we postulate the need for deriving parent-child node relationships from individual concept nodes so that, in the example above, *"the Barents Sea"* is a *"sea"* because the lowercase head noun *"sea"* can be found to be a common noun, of which *"the Barents Sea"* is therefore an instance.

Table 1 list a number of parent-child derivations that the system described in Section 6 has implemented. The arrows in the "Example" column of the table run from the child concept to the parent concept. No doubt additional derivations may be found; however, the set presented serves to illustrate the mechanism.

**Table 1.** Parent-child concept derivations.

| Derivation | Example |
|---|---|
| Common noun-proper noun | "space shuttle Atlantis"-->"space shuttle" |
| Common noun-common noun | "oil tanker" --> "tanker" |
| Proper noun with preposition | "King of England" --> "king" |
| Common noun parent of proper | "Nobel Prize" --> "prize" |
| Proper noun-common noun | "Cadillac sedan" --> "Cadillac", "sedan" |
| Adjective-common noun | "Russian submarine" --> "submarine" |
| Multiple proper names | "Peter and Paul" --> "Peter", "Paul" |
| NP coordination | "fish and chips" --> "fish", "chips" |
| NP following preposition | "jar of beans" --> "jar" |
| NP following adjective/adverb | "fast cars" --> "cars" |
| NP following prep in proper noun | "Nobel Prize for Physics">"Nobel Prize" |
| Possessive form | "Mary's car" --> "car" |
| Business entity suffix | "IBM Corp." --> "IBM" |
| Comma-separated location | "Normandy, France" --> "France" |
| Concept prefixed by ordinal | "49$^{th}$ pageant" --> "pageant" |
| Title before proper name | "Miss Lara Dutta" --> "Lara Dutta" |
| Cardinal number prefix | "five coins" --> "coins" |
| Concept begins with dollar sign | "$200 Million" --> "dollar" |
| Subordinate clause without "that" | "the book Mary read" --> "book" |

To support these capabilities, we define a "concept node" to be the 4-tuple:

<name,{parents},{children},{tuples}>, where "name" refers to the noun phrase for the discourse entity for which the node is constructed, "{parents}" and "{children}" refer to the (possibly empty) sets of parent and children nodes for the concept, and "{tuples}" refers to the tuples in which the concept (discourse entity) appears.

These derivations are repeatedly applied to a concept extracted from a tuple component until it cannot be decomposed further. Thus, for the concept "space shuttle Discovery", application of the last derivation followed by the one above it, produces the links that establish that "space shuttle Discovery " is-a "space shuttle" is-a "shuttle", which has the beneficial effect of rooting a set of concepts in a common noun which, in this case, can be found in WordNet. This supports using synonyms when answering queries concerning the nodes so rooted, as described in Section 5.

A node may have more than one parent. For example, a node for "John Hancock" may have parent links to both "insurance company" and "revolutionary war hero." Similarly, a node may have more than one child. For example, the concept "submarine" may have children "Russian submarine" and "attack submarine". Thus, concept nodes constructed in this manner are organized into a network of one or more disjoint sub-networks, each containing one or more related concept nodes.

When concepts are indexed in the network, "nicknames" are generated for proper nouns so that only one node is created for each proper noun concept, and each of its nicknames is mapped as an alias to the concept. Thus, for example, the phrase "John Kennedy" will be mapped to the concept node "John F. Kennedy". The concept addition logic is so constructed that for such proper noun associations, the longer concept is maintained as the concept node. If a shorter concept is encountered first, then when the related longer concept phrase is added to the network, it subsumes the shorter one by adjusting all necessary parent and child links and adding an alias for the shorter one linking it to the master concept node.

## 4 Question Analysis

Questions are decomposed into proposition tuples in the same manner as text, with the addition of free variables for the desired answer. These variables may take the form of a general directive, such as "*who", "*what", *when", "*where", and "*why", or a target preposition type, such as "*in", when the answer is expected to be modified by a preposition. We also define the answer variable "*ans" to serve as a referent to a candidate answer obtained in response to a previous tuple, so that we can construct question patterns as boolean combinations of separate tuple patterns.

For simple questions, a single tuple pattern may suffice. For example, for the question *"What did Peter eat?"* the corresponding question tuple is <'Peter','did eat',_,_,_,"*what">, where we have indicated the free variable with a leading asterisk and null values with underscores for clarity of display.

More complex questions require Boolean combinations of question pattern tuples. For example, the question *"What kind of car does Peter drive?"* the question decomposes into the conjunction:

<'Peter','does drive',_,_,_,"*what"> <and> <'*ans','is',_,_,_,"car">

Peter may drive his mother crazy or a nail with a hammer, but neither is a "car" and will not be returned as an answer because the second pattern would not match.

In a similar fashion, some questions require disjunctive combinations of question pattern tuples to accommodate alternative syntactic realizations of the expected answer. For the question *"What is the title of the book?"* the question decomposes into:

<'*what','is',_,_,_,'the title of the book'> <or>
<'the title of the book','is',_,_,_,'*what'>

Still other questions may require both disjunctive and conjunctive patterns. For example, the question *"What kind of book did Peter give to Mary?"* the question decomposes into: ( <'Peter','did give',__,'to Mary,',__,*what'> <or>
'Peter','did give',__,_,Mary,'*what'> ) <and>
<'*ans','is',_,_,_,'book'>

which is needed to find answers where the document base contains sentences of the form *"Peter gave Mary the novel,"* and *"Peter gave the novel to Mary."*

The final step in question analysis involves creating question tuples for the passive forms of active question constructions, and vice versa, and for possessive forms. Patterns produced at this step are added disjunctively to the existing question tuple set.

## 5 Question Answering

Question answering is a three-stage process in which: (a) the question is analyzed as described in the preceding section; (b) the tuples of interest are retrieved from the concept network; and (c) the tuples retrieved then examined to search for candidate answers. The second and third stages are discussed separately in this section.

### 5.1 Tuples of Interest

Since the concept network is, by construction, indexed by the concept names it includes, it is not necessary to search the network for candidate propositions. Thus, the network supports incremental growth as a body of knowledge is acquired, with minimal performance penalty for such growth. The tuples of interest are retrieved as follows. First, noun phrases are extracted from the various components of the question tuple or tuples, and for each such phrase, the concept network is checked for the presence of a concept node by that phrase, or the presence of a node to which such phrase is mapped as an alias. When a relevant concept node is found, all tuples associated with that node are included in the return set. Thus, for a question inquiring about "John Kennedy", the tuples indexed by the "John F. Kennedy" node would be returned. The child nodes of the concept node are also examined recursively, and any new tuples found are also added to the return set. Once all noun phrases are checked against the network, the resultant tuples are returned as the set of tuples of interest. By construction, every tuple in this set contains one or more of the phrases of interest from the question, and no tuples in the concept network that contain one of the desired phrases are omitted.

### 5.2 Question Pattern Matching

Question pattern matching proceeds by a straightforward unification algorithm in which the entire boolean combination of question tuples is applied to each tuple until an answer.

The examination of a single proposition tuple proceeds by checking its tuple components against the non-null components of the question pattern. For each such component that does not involve a free (answer) variable, a matching algorithm is executed. If any such component fails to match, the proposition tuple is rejected and examination proceeds to the next tuple in line. For subjects and direct objects, an examination set is created consisting of all system aliases for a proper noun phrase, if any, otherwise the common noun phrase itself, augmented by all aliases for the target if the phrase contains a member of the target synset. For example, if the target is "Russian submarine Kursk" and the proposition tuple contains the word "submarine", then all target aliases, including the word "Kursk" are included, so that a match will be found against a question pattern in which "Kursk" is specified.

For verbs, the main verb from the proposition tuple is extracted and WordNet methods are used to obtain all verb roots. Each root is then compared against the WordNet root of the main verb in the question tuple and a match is found if any root of one is found in any synset of the WordNet hypernym expansion of the other. This algorithm will find a match between the verbs "give" and "transfer", for example, since "transfer" appears in the hypernym tree for "give."

Gerunds and infinitives are matched in the same manner as main verbs, except that the infinitive "to do" in the question pattern is considered to match any infinitive that is present in the proposition tuple.

Indirect objects receive special treatment, since a nominal indirect is semantically equivalent to a similarly structured sentence in which the same noun phrase occurs as the object of a "to" or "for" preposition. Accordingly, the proposition tuple's nominal indirect is checked, if any, and if none, then its modifiers commencing with "in" or "for", if any, are examined.

Modifiers also receive special treatment since there can be more than one in either the question pattern or the proposition tuple. Where the question pattern contains more than one modifier, a match is recorded only if all modifiers are matched independently. However, so long as all question modifiers are matched, it does not matter that a proposition tuple may have additional, unmatched modifiers, since these are, by construction, irrelevant to the question. We seek to achieve robustness in matching through examinations of WordNet synonyms for a noun phrase head, whenever possible, which obtains for most common nouns and some proper nouns.

Now, where the question pattern component contains the answer variable "*ans", an answer retrieval algorithm is executed according to the component type. For subjects and direct objects, the algorithm returns the corresponding component of the proposition tuple if the expected answer type is confirmed through WordNet. For example, a location type is confirmed if "structure" or "location" appears in the head noun's hypernym tree. Similarly, verbs and other tuple components are retrieved after appropriate type checks by algorithms that parallel their corresponding match methods.

If a match is not obtained for a given proposition tuple, then the reciprocal predicate is checked if the subject and direct object are both nonempty, there is no gerund/infinitive, and the main verb is not copular, not the verb "do", and possesses an

antonym in WordNet. This construction increases recall by supporting a match for the question “What did Mary receive?” where the database contains, for example, “Peter gave Mary the book.” Once the reciprocal tuple is formed, it is matched in the same manner described above for the direct pattern.

Different processing is performed for question patterns that seek to confirm “is-a” relationships for previously found candidate answers. Where, for example, we seek to confirm whether a candidate answer for the location of the sinking of the submarine Kursk is in fact an instance of the “sea” class, the parents of the candidate answer are searched recursively to the head of their equivalence network for the desired class. If none is found, then the WordNet hypernym tree for each root of the network is checked. We note that there can be more than one such root, since a concept node can have more than one parent. If the desired class name is found in the hypernym tree for any node examined, then the candidate concept is deemed a member of the class and a match is recorded.

## 6 A Test Implementation

The algorithms described above were implemented in our Semantic Extractor (SEMEX) tool, a Java application for testing semantic extraction algorithms. SEMEX incorporates the Brill tagger [3], the Cass partial parser [1], and a comprehensive set of empirically derived grouping and sentence splitting heuristics, plus rudimentary pronomial coreference resolution, to provides a graphical user interface for viewing intermediate results at each stage of processing, from POS tagging, through parsing, sentence splitting, syntactic role assignment and resolution, and concept extraction.

SEMEX was configured to exercise the TREC 2005 QA questions against the top-50 documents for each target returned by NIST's generic IR engine from the AQUAINT newswire collection. SEMEX output for the first 200 factoid questions was analyzed in detail and scored manually using the TREC-furnished factoid answer patterns, adjusted to eliminate the limitations of the tool and document set. Thus, a question was discarded from the analysis if: (a) its answer was not in the top-50 documents; (b) the document containing the answer was not readable by SEMEX; (c) the answer was in a direct quote, since SEMEX was not configured to look within direct quotes; (d) the answer was in the dateline or headline; (e) the question form was not one of the types that was implemented in SEMEX; or (f) the answer pattern provided by TREC was a wrong answer, or no answer, to the question. Similarly, an answer was considered found successfully if only a minor manual correction to the parsed output of text or question using the SEMEX GUI was sufficient to allow SEMEX to run on the question, as this were considered a near-term programming improvement for the tool. Importantly, however, questions whose answers required reasoning over the concept set or more than minor syntactical parsing adjustments were retained as failures, as these indicated areas of further inquiry.

On this basis, SEMEX was able to answer correctly 66 out of the 135 non-discarded questions, for an adjusted score of 48%. Of the 66 correct answers, 22 were found without manual intervention, for a raw score of 16%. Both of these scores compare favorably with other systems, as [10] reports factoid answer accuracies in a range from 21.3% to 77.0% for the top 10 runs submitted, with only the top 3 systems achieving factoid accuracy scores greater than 35%. And with sixty-three runs sub-

mitted to the QA track, it is clear that most of the systems achieved factoid accuracies below the lower bound of this range.

Of the 65 discards, 29 involved questions that had no answers in the documents, 14 involved the headline or dateline, 9 involved documents that could not be read, 6 involved question types that were not implemented, and 2 had incorrect answers.

Generally, the newswire articles that comprised the document sets provided challenges to our implementation, as they were characterized by numerous spurious HTML codes and meta-data within the text, such as identification of sources, sports scores, and multiple datelines within the text. Where acceptable parses were obtained, however, the algorithms proposed operated well. Algorithm success and limitations are well illustrated in a separate test where for the target *"Russian submarine Kursk sinks"* TREC question 66.7 asked the list question, *"Which U.S. submarines were reportedly in the area?"* For this question, SEMEX generated the question pattern:

<or> [S:*which][V:were][GI:][M:reportedly;in the area][IO:][DO:]
<and> [S:*ans][V:is][GI:][M:][IO:][DO:U.S. submarine]

and returned the answer "the Toledo". What is significant about this result is that the document text consisted of three sentences, the first two of which were:

*<P> The second U.S. submarine in the Barents Sea when the Kursk sank was the Toledo, a Russian news agency reported Thursday. </P> <P> The agency, Interfax, said the Toledo was in the area along with another U.S. submarine, the Memphis, during the Russian naval exercises in mid-August, when the Kursk sank, with the loss of 118 lives. </P>*

In this example, the fact that the Toledo was a U.S. submarine is established in the first sentence of the document, while the fact that it was reportedly in the area when the Kursk sank was furnished by the second sentence. The connection between these two facts is provided by the concept network, in which the "Toledo" concept has a parent, "second U.S. submarine in the Barents Sea when the Kursk", which by operation of the derivations of Table 1 in Section 3.3, has a parent, "second U.S. submarine" which, by a further derivation has a parent, "U.S. submarine", which is the desired class. We note that SEMEX was unable to find the second valid answer, "the Memphis", since the SEMEX did not understand that vessel to have been in the area, although the network structure shows that it was recognized as a U.S. submarine.

Similarly, TREC question 87.3 asked the factoid question, "What Nobel Prize was Fermi awarded in 1938?" for which SEMEX correctly returned "the Nobel Prize for Physics", which was made possible by the parent-child relationship present in the concept network between "Nobel Prize" and the answer returned. Indeed, a separate test has shown that SEMEX would still return the same result if the question had asked what "prize" was Fermi awarded, again as a consequence of the concept network, and also if the question had asked about an "award", which though not in the network was nevertheless a WordNet hypernym of "Nobel Prize" found by SEMEX.

## 7 Conclusions

The test implementation of the algorithms proposed demonstrated the value of a concept network of parent-child concept relationships and indexing proposition tuples that encode the relationships among the syntactical components of atomic sentences. The results also establish the robustness that may be achieved through the use of

WordNet. We note, however, that the current implementation does not explicitly disambiguate words but relies instead on the constellation of syntactical components in a proposition tuple to serve as selectional restrictions when answering questions. It remains a question for further investigation whether a separate disambiguation module is necessary. Further research is also needed to establish the extent to which a concept network of the type proposed can support reasoning and advanced question answering.

## References


1. Abney, S.: Partial Parsing via Finite-State Cascades. In: Proceedings of Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information. Prague, Czech Republic (1996) 8-15.
2. Ahn, K., Bos, J., Clark, S., Curran, J., Dalmas, T., Leidner, J., Smillie, M., Webber, B.: Question Answering with QED and Wee at TREC-2004. In: Voorhees, E.M., Buckland, L.P. (eds.): Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004), NIST Special Publication 500-261. Gaithersburg, MD (2005)
3. Brill, E.: Some Advances in Part of Speech Tagging. In: Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94). Seattle, Washington (1994)
4. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. The MIT Press, Cambridge London (1998)
5. Gaisauskas, R., Greenwood, M., Hepple, M., Roberts, I., Saggion, H.: The University of Sheffield's TREC 2004 Q&A Experiments. In: Voorhees, E.M., Buckland, L.P. (eds.): Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004), NIST Special Publication 500-261. Gaithersburg, MD (2005)
6. Greenwood, M., Roberts, I., Gaizauskas, R.: The University of Sheffield TREC 2002 Q&A System. n: Voorhees, E.M., Buckland, L.P. (eds.): Proceedings of the Eleventh Text REtrieval Conference (TREC 2002), NIST Special Publication 500-251. Gaithersburg, MD (2003)
7. Harabagiu, S., Moldovan, D., Clark, C., Bowden, M., Williams, J., Bensley, J.: Answer Mining by Combining Extraction Techniques with Abductive Reasoning. In: Voorhees, E.M., Buckland, L.P. (eds.): Proceedings of the Twelfth Text REtrieval Conference (TREC 2003), NIST Special Publication 500-255. Gaithersburg, MD (2004) 375-382
8. Litkowski, K.: Question Answering Using XMS-Tagged Documents. In: Voorhees, E.M., Buckland, L.P. (eds.): Proceedings of the Eleventh Text REtrieval Conference (TREC 2002), NIST Special Publication 500-251. Gaithersburg, MD (2003) 156-165
9. Schone, P., Ciany, G., McNamee, P., Mayfield, J., Bassi, T. Kulman, A.: Question Answering with QACTIS at TREC-2004. In: Voorhees, E.M., Buckland, L.P. (eds.): Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004), NIST Special Publication 500-261. Gaithersburg, MD (2005)
10. Voorhees, E.M.: Overview of the TREC 2004 Question Answering Track. In: Voorhees, E.M., Buckland, L.P. (eds.): Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004), NIST Special Publication 500-261. Gaithersburg, MD (2005)