Question Answering by Searching Large Corpora with Linguistic Methods

Michael Kaisser, Tilman Becker
Saarland University / DFKI GmbH
{mkaisser,becker}@dfki.de

Abstract

In this paper we describe the QuALiM Question Answering system which uses linguistic analysis of questions as well as candidate sentences in its answer finding process. To this end we have developed a rephrasing algorithm based on linguistic patterns that describe the structure of questions and candidate sentences and where precisely to find the answer in the candidate sentences. With this method and a fall-back strategy, both using the web as their primary data source, we participated in TREC 2004. We present our official results and a follow-up evaluation to elucidate the contribution of the methods used.

1 Introduction

At least since Frege and Wittgenstein we understand how to interpret sentences in natural language as expressing facts. In a sentence, different entities are referred to by linguistic expressions and the structure of the sentence describes the relationship between them. When someone asks a question, he asks not so much for a word or a few words that make up the answer, but rather for a fact that he has some partial knowledge of, but where he is lacking an important entity to describe it completely.

E.g., in the question "When was Franz Kafka born?" the asker already knows that there is a person called Franz Kafka, and that this person must, as all persons, be born sometime. What he lacks, and asks for, is the date of birth. The fact the person looks for can in a davidsonian way be described like this:

born("Franz Kafka",e) AND born("July 3, 1883",e)

The knowledge the question asker has is this:

born("Franz Kafka",e) AND born(X,e)

Sentences containing the facts and entities that are sought can be expressed with a wide variety of linguistic means. However, a simple sentence containing the answer to the question might be:

"Franz Kafka was born on July 3, 1883."

The observation one can make here is that certain words from the question reoccur in the answer sentence. In this case "Franz" "Kafka", "was" and "born". This is the basic observation most IR systems, and therefore also most QA systems, make use of. When searching for an answer to a question like the one described these systems search for the (non-stop) words in the question in a document collection, and if they find as many as possible closely together, they look for a date which is also close to these words and return this as the answer. However, they cannot be sure that the answer they found is the correct one:

"Franz Kafka was not born on July 3, 1884."

What these systems are not taking into account is that natural language has a structure and that it is this structure that determines in which relation certain linguistic entities in a sentence stand to each other. A related aspect is the variety of linguistic means to refer to entities. E.g., the correct answer is also contained in:

"Franz Kafka's birthday was July 3, 1883."

These are the basic observations that lead to the development of QuALiM: "Question Answering with Linguistic Methods".

2 Overview

In this section we will give an short overview of how QuALiM answers questions. We will then in the following sections describe the different modules in more detail. Then, the TREC results and a post-TREC evaluation with its discussion are presented. After the conclusion, an appendix lists the third-party applications that were used in QuALiM.

QuALiM heavily relies on patterns. When it is asked a question it searches for a syntactic description in one of these patterns that matches the question. Most of these patterns also contain syntactic descriptions of potential answer sentences. This information can be used to predict the surface structure of possible answer sentences. Google is used to search for answer sentences and the retrieved sentences are then checked on whether they really match the syntactic structure proposed in the first place. To do this the sentences are parsed and tagged. If a candidate sentence still matches, the system knows which phrase in the sentence is the answer. The answer itself is then checked on its semantic type by using a Named Entity Recognition system.

QuALiM also implements a fallback mechanism, which does not propose reformulations, but instead sends queries created from key words and key phrases in the question to Google. From the returned snippets n-grams are mined, which are also checked on their semantic type.

3 The Rephrasing Algorithm

3.1 Strict Pattern Matching

As described above, the rephrasing part of QuALiM relies on patterns which are used to define linguistic constraints on questions, potential answer sentences to these questions and the answers themselves. In detail a pattern used in QuALiM consists of three parts:

- *Sequences* are used to classify the questions according to their linguistic (mostly syntactic) structure.
- *Targets* are used to describe the linguistic (mostly syntactic) structure of potential answer candidates.
- *AnswerType* elements express semantic constraints on the answers.

Figure 1 gives an example of such a pattern.

Each question that the system is asked is checked whether it matches one of the sequences in the pattern files. The sequence which can be seen in figure 1 matches any question that starts with the word “When”, followed by the word “did”, followed by an NP, followed by a verb in its infinitive form, followed by an NP or a PP, followed by a question mark which has in addition to be the last element in the question.

In the TREC 2004 question set, this sequence matched five questions:

- When did Floyd Patterson win the title?
- When did Amtrak begin operations?
- When did Jack Welch become chairman of General Electric?
- When did Jack Welch retire from GE?
- When did the Khmer Rouge come into power?

For the TREC 2004 runs QuALiM used 157 patterns (with 157 sequences) to classify incoming questions. If a question matches a sequence, the targets are used to predict the linguistic structure of potential answer sentences. Two targets are shown in figure 1. For the question “When did Amtrak begin operations?”, they would suggest the following answer sentences (or answer sentence parts):

1. Amtrak began operations in ANSWER
2. In ANSWER (,) Amtrak began operations

The numbers in the *ref* elements are variables that point back to the sequence element with the corresponding *id* attribute. Note, that QuALiM copies the complete linguistic information for the sequence element into the target, not just its surface appearance. Beside the *target* elements that can be seen in the

```
<pattern name="When+did+NP+Verb+NPorPP" level="5">
  <sequence>
    <word id="1">When</word>
    <word id="2">did</word>
    <parse id="3">NP</parse>
    <morph id="4">V INF</morph>
    <parse id="5">NP|PP</parse>
    <final>?</final>
  </sequence>

  <target name="target1">
    <ref>3</ref>
    <ref morph="V PAST">4</ref>
    <ref>5</ref>
    <word>in</word>
    <answer>NP</answer>
  </target>
  <target name="target2">
    <word>in</word>
    <answer>NP</answer>
    <punctuation optional="true">,<punctuation>
    <ref>3</ref>
    <ref morph="V PAST">4</ref>
    <ref>5</ref>
  </target>

  ... more targets ...

   <answerType phrases="NP|PP">
     <built-in weight="2">dateComplete</built-in>
     <namedEntity weight="4">Date</namedEntity>
     <built-in weight="3">year|in_year</built-in>
     <other ignore="true"/>
   </answerType>
 </pattern>
```

Figure 1: Example pattern as used in the current version of the QuALiM system.

example (*ref*, *word*, *punctuation* and *answer*), three others exist: *pos*, *parse* and *unknown*, but the latter two are–up to now–just implemented with dummy functionality and could therefore not be used for the TREC runs.

Although the targets describe linguistic structures they can also be used to propose surface structures of the potential answer sentences. Some target elements can be used for this, namely *ref*, *word* and *punctuation*, while the others cannot. With this information search queries can be created that are send to Google. Unfortunately, as Google ignores punctuation in queries, this information is also not useful when creating the queries. But the *ref* and *word* elements in the targets seen in figure 1 provide enough information to generate the following two queries:

1. "Amtrak began operations in"

2. "In" "Amtrak began operations"

QuALiM will send these queries to Google and harvest the first 40 snippets returned. It will then try to extract sentences from the snippets that contain all words from the search query. For the first query listed above, the first five sentences QuALiM can extract are:

- "Since Amtrak began operations in 1971, federal outlays for intercity rail passenger service have been about $18 billion."
- "Amtrak began operations in 1971."
- "Amtrak of the obligation to operate the basic system of routes that was largely inherited from the private railroads when Amtrak began operations in 1971."
- "Amtrak began operations in 1971, as authorized by the Rail Passenger Service Act of 1970."
- "A comprehensive history of intercity passenger service in Indiana, from the mid-19th century through May 1, 1971, when Amtrak began operations in the state."

QuALiM will now parse and tag the candidate sentences and check weather the linguistic structure described in the target really matches the sentences. If the system finds this structure, it also knows which part of the sentence must be the answer. In the first four examples given above it is "1971", in the last "the state" (which is sorted out in a later processing step, when the system recognizes that "the state" is not an appropriate answer for this type of question).

The system will place all answers it has found in a *weighted sequence bag*, a bag of word sequences, where each word sequence has a weight attached. In the given example the weighted sequence bag will look like this:

```
4: "1971"
1: "the state"
```

3.2 Fuzzy Pattern Matching

Experiments during the construction phase of QuALiM showed that the described constraints are sometimes too strict. It happens that the retrieved sentences contain the correct answer, but not exactly at the position described by the targets. For the second target shown in figure 1 a possible answer sentence received from Google might be:

"In 1971, the railroad company Amtrak began operations."

This sentence does not match the target because no single NP is placed between the word "In" and the NP "Amtrak". In such a case QuALiM will extract the string where it actually expected the NP, here: "1971, the railroad company". This string contains two NPs: "1971" and "the railroad company". With this information another weighted sequence bag is created:

```
1: "1971", "the railroad company"
```

The results form both pattern matching algorithms are then combined, but before this is done the weights from the exact target matching algorithm are multiplied with 5. So the outcome looks like this:

```
21: "1971"
5: "the state"
1: "the railroad company"
```

This method of combining the results proofed to be quite effective. If QuALiM can find many sentences that match the targets exactly, the fuzzy results are of nearly no importance. If there are no or only a few exact matching sentences found, the fuzzy results will become more important.

4 The Google Fallback Mechanism

QuALiM implements a second mechanism to find answers. It constructs three search queries that are combinations from all NPs in the question and all non stop words:

"When was Jim Inhofe first elected to the senate?" becomes

1. **Jim Inhofe senate first elected**
2. **"Jim Inhofe" "the senate"**
3. **"Jim Inhofe" "the senate" first elected**

These queries are sent to Google and from the snippets returned, n-grams are mined. These n-grams are placed in a weighted sequence bag, where the weights initially show how often an n-gram has been found in the snippets. The results form the third query are doubled. The weights are then modified, so that n-grams with more words receive a bonus over shorter n-grams.

The results from the Google fallback algorithm are combined with the results from the rephrasing algorithms. But before this is done the weights of the rephrasing algorithms are multiplied with 4. (So the overall ratio strict:fuzzy:fallback is 20:4:1)

5 Type Checking

What remains to do is to check the combined output of the three answer finding mechanisms on their type. To do this the *answerType* element in the XML structure is used. Several information sources can be combined here: *named entity recognition*, *WordNet* or *built-in named entity recognition features*, which can be used to recognize standard date specifications, year specifications, numbers, number/unit compounds etc.

The textual content of an *answer type* element specifies the condition an item in the sequence bag must fulfill, to match the answer type element. Each element from the sequence bag is matched against all answer type elements. All weights from the matching answer type elements are summed up, and finally the value of the sequence bag element is multiplied with that sum.

6 List and Definition Questions

The approach QuALiM uses to answer list questions is a simple modification of the factoid approach. The system tries to put a question in the singular form and it will then return all answers above a given cut-off level. It is possible to write special list patterns (similar to factoid patterns as seen in figure 1), but only two patterns existed for the TREC runs.

Definition questions are processed as follows: The series' target is send to Google, and as with the Google fallback mechanism a weighted sequence bag is created, with the most frequent words or word sequences occurring in the snippets at the top. The system then looks for sentences (or sentence parts) in the AQUAINT document collection that contain many of the entries in the weighted sequence bag. For each sentence, the weights of the entries in the sequence bag that occur in the sentence lead to a score which will be divided through the character length of the sentence. The sentence with the best score is selected. Then, all values of all entries in the sequence bag that occurred in the selected sentence are divided by three and again all sentences are scored and the best one is selected. This is repeated until the sentences selected so far exceeded a certain character length.

7 Related Work

QuALiM builds on two pillars: Creative use of the web, especially by using a search engine (Google in our case) as an Information Retrieval system and by using linguistic methods to analyze questions and candidate sentences in order to locate the exact answers in the latter.

Using the web for Question Answering is an idea that became more and more important in recent years. AnswerBus [Ans], BrainBoost [Bra], IONaut [ION] and START[STA] are systems that are online and that search the web for an answer to a question a user has asked. Furthermore quite a few research papers exist on that topic: [DBB+02, KEW01, RFQ+04] to name just a few.

Our way to use the web shows similarities to [DBB+02]. In this paper the authors also present a way to reformulate questions as possible answer sentences, exploiting redundancy in large corpora such as the web. However, their reformulations are not based on the syntactic structure, but rather on sim-

ple string manipulations. Reformulations based on the syntactic structure offer a number of benefits: The reformulations are more exact, more reformulations can be described and the knowledge about the sentence structure can be used to locate the exact answer in candidate sentences.

In general not much work has been done so far on using linguistic knowledge to find candidate sentences, on making use of this knowledge to evaluate their credibility and to locate the part of the sentence that contains the answer.

There are three systems we want to mention here which actually parse candidate sentences and make use of this information, although in quite different ways:

- LCC's PowerAnswer system which incorporates a Logic Prover named Cogex to filter out incorrect answers. [HMC+03]
- MIT's START and especially Omnibase which builds up a database of so-called ternary expressions. [KFY+02, Kat97]
- ITCirst's Diogene systems which shows–in the TREC 2004 version–some similarities to QuALiM. [TKM04]

8 Discussion

We see our syntactic patterns as a generic approach to handle various linguistic phenomena:

- We can combine different variants of the same underlying question in one pattern. (The rephrasing algorithm allows to have patterns with more than one sequence, although this feature was not used in the TREC 2004 runs.)
- We can distinguish a question like "Who is the Queen of England" (Who+is+NP+?) form "Who is Albert Einstein?" (Who+is+Person_Name+?). This can be done because information from named entity recognition can be included in the sequences and it is is useful because both questions require different answer types.
- We can perform reformulations that introduce words which are not in the question into the answer sentence: E.g, we can reformulate "When+was+Person+born+?" to "Person+'s+birthday+is/was+ANSWER".
- Almost all syntactic phenomena the parser can handle can be described with the patterns: appositions, active/passive transformations, dative shift, subordinate clauses, NP-gerunds and many more.

Although we use the complete syntactic structure returned from the parser to determine phrase boundaries, we abstract away from that by using flat descriptions in our patterns. As a result, we cannot describe some linguistic phenomena, for example PP-attachment.

Furthermore, currently all patterns have to be written manually, which takes up a lot of time. This is the reason why we performed the TREC runs with only 157 patterns. We simply had no time to write more of them.

9 Evaluation

9.1 TREC Evaluation

Table 1 shows the results QuALiM received in TREC 2004. The runs differed in parameter settings (e.g. the number of NIL answers returned for factoid questions, the length of the answer list when answering list questions etc.)

	run 1	run 2	run 3	rank
Factoid	**0.343**	0.339	**0.343**	4th
List	0.096	0.111	**0.125**	9th
Other	0.145	0.181	**0.211**	10th
Combined	0.232	0.242	**0.256**	6th

Table 1: TREC 2004 results for the QuALiM system.

9.2 Post-TREC Evaluation

After TREC 2004 we conducted several evaluations here at DFKI. We were mainly interested in the per-

	strict	fuzzy	fallback	combined
# correct	46	58	91	105
# inexact	6	10	9	7
# wrong	20	31	114	106
# answers returned	72	98	214	218
# no answer found	158	132	16	11
accuracy	0.200	0.252	0.395	0.456

Table 2: Results of the QuALiM system in our post-TREC evaluations. See the text for details.

formance and behavior of the different algorithms implemented. For the additional runs performed and evaluated at DFKI, we used the TREC 2004 question set, but we resolved the questions manually. So we changed the Question "When was he born?" in the "Franz Kafka" series to "When was Franz Kafka born?" and told the system to process the questions as they are in the file and not to combine them with the target. Furthermore, we turned the document localization module off. So the system returned just answers, but no AQUAINT documents. This means that there can be no "NIL" answers and no "unsupported" judgments. Naturally–as we disabled two error sources–the results we received are better than the results we received in TREC 2004.[1] We performed four more runs, telling the system it should only answer factoid questions and ignore list and definitional questions.

Here is a short description of the runs:

strict In this run we only used the rephrasing algorithm in its strict version as described in section 3.1.

fuzzy For this runs we used the fuzzy version of the rephrasing algorithm. See section 3.2 (Note that NPs at the exact position still received a weight four times higher as NPs not being at the exact position.)

fallback This run used only the Google fallback mechanism as described in section 4.

combined This run combined the lazy rephrasing algorithm with the Google fallback mechanism.

Table 2 shows the results we obtained.

After the system answered all 230 factoid questions, we sorted the results by their confidence values. (Similar to the TREC 2002 QA task, where all participants had to do this, see [Vor02].) Then we calculated what fraction of the best x answers are correct answers. The results can be seen in figure 2. If you draw for example an imaginary vertical line from 41 on the x axis, it will cut the *strict* curve at 0.94, meaning that 94% of the 41 answers the system was most sure of, were correct. This line cuts the fallback curve at 0.72, so only 72% of the best 41 answers from the fallback mechanism are correct. Figure 3 is based on the same data as figure 2, but this time we computed the fraction of correct and inexact answers. The little diamonds that can be seen on the curves mark the last correct answer the system returned, while the boxes mark the point from where on the system found no more data to process and therefore could only return an "answer unknown" response with a confidence value of 0.

We think that these results are quite interesting:

- Generally, it has to be said that the fallback mechanism performs better than the rephrasing algorithm, at least when the results are measured using accuracy as in TREC 2004. (In the diagrams accuracy can be seen at the rightmost point of the diagram, where the fraction of the top 230 (i.e., of all) questions was computed.

[1] In our additional runs, when all questions finding methods are combined, we receive an accuracy of 0.456. We think that this is quite reasonable. In TREC 2004 our best run returned 79 correct and 20 unsupported answers. As we performed no document localization we can expect an accuracy of roughly 99/230=0.430. The additional 0.026 points result from the fact that we used resolved questions. (We actually would have expected that this results in a larger performance gain.)

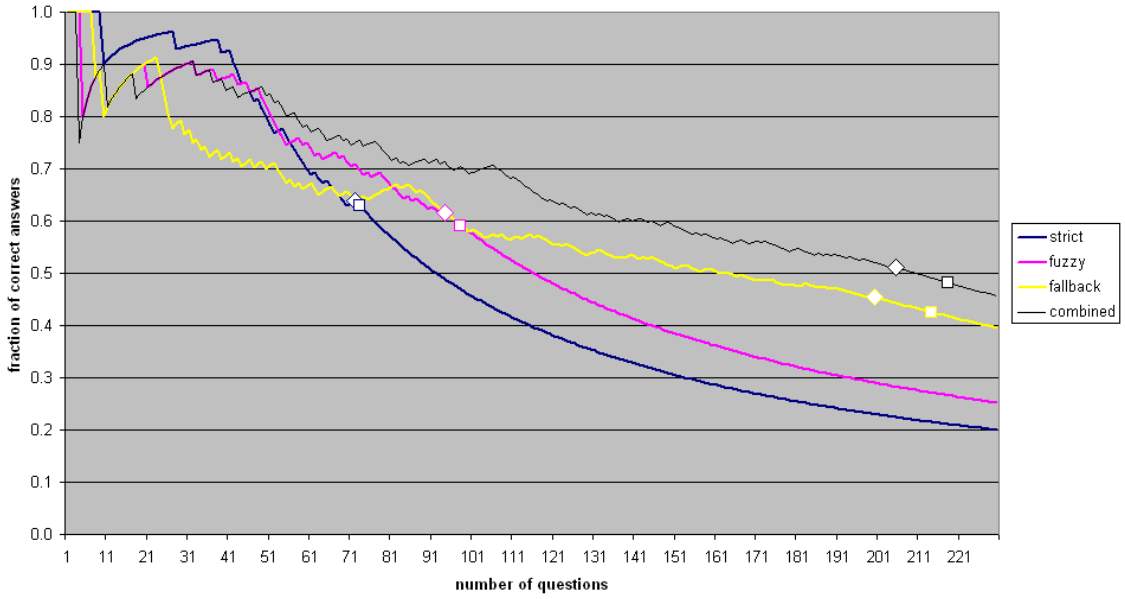

Figure 2: Fraction of correct answers of the top x answers returned, when ordering them by their confidence values.

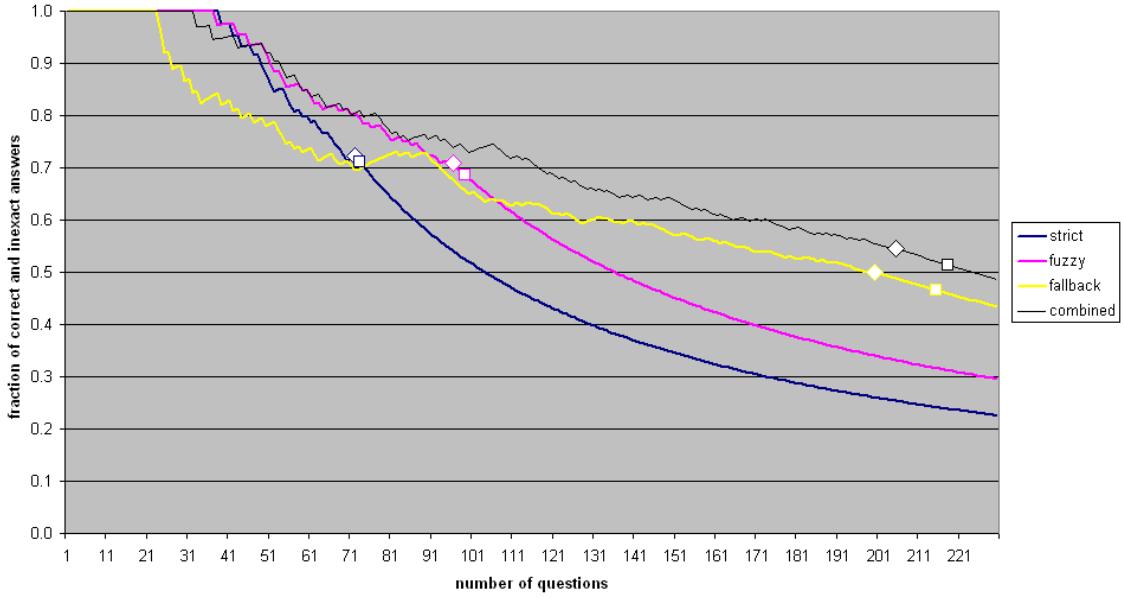

Figure 3: Fraction of correct and inexact answers of the top x answers returned, when ordering them by their confidence values.

- The rephrasing algorithms run out of data very fast. This due to the fact that the TREC 2004 runs were performed with only 157 patterns. For most of the questions this approach did simply not catch. Either, because there was no pattern for this question type, or because the reformulations could not be found on the web. It is left for further work to evaluate how much performance gain can be achieved with more patterns.

- As mentioned, the rephrasing algorithms do not return that many answers, but a large fraction of them is correct. Also, the confidence values the algorithms return seem to be really useful. Answers with a high confidence value are almost always correct. Of the 38 answers the strict rephrasing algorithm was most sure of, 36 were correct, two were inexact and none were wrong. Of the next 35 answers 10 were correct, four were inexact and 21 were wrong.[2] After these 73 questions, the algorithm returned no more answers.

- The confidence values of the fallback mechanism are not that useful. The mechanism returned 215 answers, but it is difficult for the system to determine whether it has found a correct answer or not.

- Especially for the rephrasing algorithms, the first non-correct answers the system returns, are not completely wrong, but rather inexact. (See the difference between figures 2 and 3)

10 Conclusion and Further Directions

Most QA systems use keyword approaches combined with named entity recognition techniques to locate the answer to a question in a given document collection. In this paper we presented a new way to locate answers: Feeding syntactic reformulations into an Information Retrieval system and–more important–using a syntactic analysis of the results in order to find the exact answer to a question.

Our experiments suggest that linguistic processing of candidate sentences results in a better knowledge of when an answer is correct or not. If we adopt the terms precision and recall from Information Retrieval (which are not commonly used in Question Answering), we can conclude that making use of a syntactic analysis of candidate sentences results in good precision values. But, because natural language offers so many different ways to express answers, it is not easy to obtain good recall values.

This last point leads directly to our future plans: We want to explore how recall can be improved, without sacrificing the good precision values (or even find ways to improve precision as well). The most obvious way to do this is to increase the size of the pattern set. Because it is very time consuming to write the patterns manually, we are currently thinking about making use of Machine Learning techniques in order to do this.

A Technical Details

QuALiM is written in Java 1.4. It uses the following 3rd party modules:

- ANNIE (a named Entity Recognition system): http://gate.ac.uk/ie/annie.html

- Google (accessed through the GoogleAPI): http://www.google.com/apis/

- Link Parser: http://www.link.cs.cmu.edu/link/

- QTag (a POS tagger): http://web.bham.ac.uk/o.mason/software/tagger/

- WordNet: http://www.cogsci.princeton.edu/~wn/

- XTAG morphology database: http://www.cis.upenn.edu/~xtag/

[2] The confidence values of the rephrasing algorithms reflect the amount of answer sentences that were found on the web and how useful they were judged when being checked on the correct syntax and on the correct answer type. So low confidence values indicate that there is something wrong: Either not much data was found or the syntactic structure or the answer type did not match.

You can find more information about QuALiM here: http://www.dfki.de/~mkaisser/
For example an electronic copy of my Master's Thesis: [Kai04]

References

[Ans] AnswerBus Question Answering System. http://www.answerbus.com.

[Bra] BrainBoost - Question Answering Search Engine. http://www.brainboost.com.

[DBB+02] Susan Dumais, Michele Bankom, Eric Brill, Jimmy Lin, and Andrew Ng. Web Question Answering: Is More Always Better? *Proceedings of UAI 2003*, 2002.

[HMC+03] Sanda Harabagiu, Dan Moldovan, Christine Clark, Mitchell Bowden, John Williams, and Jeremy Bensley. Answer Mining by Combining Extraction Techniques with Abductive Reasoning. In *The Proceedings of the 2003 Edition of the Text REtrieval Conference, TREC 2003*, 2003.

[ION] ionaut - The IO question answering system. http://www.ionaut.com:8400.

[Kai04] Michael Kaisser. Question Answering by Searching Large Corpora with Linguistic Methods. Master's thesis, Saarland University, Germany, 2004.

[Kat97] Boris Katz. Annotating the World Wide Web using Natural Language. In *Proceedings of the 5th RIAO conference on Computer Assisted Information Searching on the Internet (RIAO '97)*, 1997.

[KEW01] Cody C. T. Kwok, Oren Etzioni, and Daniel S. Weld. Scaling Question Answering to the Web. In *Proceedings of the 10th World Wide Web Conference (WWW 2001)*, 2001.

[KFY+02] Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform Access to Heterogeneous Data for Question Answering. In *Proceedings of the 7th International Workshop on Applications Of Natural Language to Information Systems (NLDB2002)*, 2002.

[RFQ+04] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. Probabilistic Question Answering on the Web. *Journal of the American Society for Information Science and Technology (JASIST)*, 2004.

[STA] The START Natural Language Question Answering System. http://www.ai.mit.edu/projects/infolab/.

[TKM04] Hristo Tanev, Milen Kouylekov, and Bernardo Magnini. Combining Linguistic Processing and Web Mining for Question Answering: ITC-irst at TREC-2004. In *The Proceedings of the 2004 Edition of the Text REtrieval Conference, TREC 2004*, 2004.

[Vor02] Ellen M. Vorheese. Overview of the TREC 2002 Question Answering Track. In *The Proceedings of the 2002 Edition of the Text REtrieval Conference, TREC 2002*, 2002.