The LogAnswer Project at QA4MRE 2011

Ingo Glöckner[1], Björn Pelzer[2], and Tiansi Dong[1]
[1] Intelligent Information and Communication Systems Group (IICS),
University of Hagen, 59084 Hagen, Germany
{ingo.gloeckner,tiansi.dong}@fernuni-hagen.de
[2] Department of Computer Science, Artificial Intelligence Research Group
University of Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz
bpelzer@uni-koblenz.de

Abstract. We present the prototype of a pure logic-based answer validation system that was developed for QA4MRE. While the prototype uses core functionality of the LogAnswer question answering (QA) system available on the web, it had to be designed from scratch in order to meet the demands of the QA4MRE task. Specific improvements include the use of coreference resolution in order to allow knowledge processing on the document level, the integration of a fragment of OpenCyc in order to allow more flexible reasoning, and the extraction of 'provenance information' that explains the preference of the system for the chosen answer candidate. Results show that the new prototype was not mature yet at the time of the participation in the QA4MRE task. However, the system has a solid architecture that allows a powerful logic-based answer validation, and our analysis of the submitted runs clearly identifies the necessary improvements that will unfold the potential of the system.[1]

1 Introduction

LogAnswer [2] is a question answering (QA) system for German that can be tested on the web, using the German Wikipedia as the document collection.[2] The system was repeatedly evaluated in the QA@CLEF and ResPubliQA QA system evaluations [3,4,5]. Recently, we have focused on the use of LogAnswer as a virtual forum participant in Human Q&A portals on the web (such as Yahoo! answers). Experiments using data from the German QA portal FragWikia![3] show that the questions from real users are substantially more difficult to handle than the artificial questions of the earlier QA@CLEF experiments [6,7]. In particular, the relationship between the question and the relevant text passage that answers the question can be rather loose, with few lexical overlap between question and answer phrases. We thus embraced the new QA4MRE challenge of CLEF 2011 that also fosters the development of QA technologies which no longer assume a simple lexical relationship between questions and answer passages.

[1] LogAnswer is supported by the DFG with grants FU 263/12-2 and GL 682/1-2.
[2] see http://www.loganswer.de/
[3] see http://frag.wikia.com/

It was not possible, though, to apply the existing LogAnswer system to the QA4MRE task, because the system is based on some implicit assumptions that no longer hold in the QA4MRE context. First of all, the existing LogAnswer system uses a sentence-level segmentation, i.e. the candidate retrieval delivers single sentences and the answer selection models can operate on single sentences only. In the QA4MRE task, however, the content of a full document must be processed and the restriction to single sentences does not make sense. The evaluation models of LogAnswer further depend on the retrieval score produced by the retrieval system, while there is a fixed reading test document without any retrieval step in QA4MRE. Finally, the training sets for the answer selection models of LogAnswer were obtained from the earlier QA@CLEF competitions, where there was often a close lexical relationship between question and answer sentence. Consequently, it was not possible to reuse the existing validation features and the existing answer selection models of LogAnswer. The switch to a document level validation forced us to construct a new prototype specifically for QA4MRE, using only those basic techniques from LogAnswer that seemed useful for the task. Since a new system had to be built, many advanced aspects already covered by the existing LogAnswer system (use of fallback techniques, sophisticated validation models obtained by machine learning, etc.) were dropped for the time being.

We state the following goals for the QA4MRE participation:

- Develop a core solution for the QA4MRE task that implements a logic-based answer validation for document level segmentation, reusing functionality of the first LogAnswer prototype whenever possible.
- Instead of using the background collection provided by QA4MRE, or external evidence as provided by hit counts of search engines, we decided to focus on strengthening the reasoning capabilities of our system by extending its background knowledge. The specific goal was that of coupling LogAnswer to (a fragment of) OpenCyc, so that its background knowledge will cover more possible paraphrases.
- Another goal for QA4MRE was the addition of functionality that can extract an explanation why the chosen answer was prefered over the alternatives. In our case, such a justification for the best answer includes the concrete logical rules and facts and the relevant sentences necessary to prove the answer.

In the paper, we first present the system description of the prototype built for QA4MRE. We then show the results of the system on the QA4MRE test set for German. Finally we discuss various problems that became visible in a first error analysis of the QA4MRE runs.

2 System Description

2.1 Parsing of Reading Tests, Questions and Candidate Answers

The reading tests, questions and answer candidates were parsed by applying the WOCADI parser [8]. It computes a meaning representation for each sentence in the form of a Multilayered Extended Semantic Network (MultiNet) [9].

2.2 Question Classification

The LogAnswer system uses a question classification in order to determine the category of the question (factoid or definition question) and the expected answer type (e.g. PERSON or DATE). In the QA4MRE context, we lacked training data for exploiting the criterion of an expected answer type check, so we ignored the answer type. Still, the distinction between factoid questions and definition questions was important for our QA4MRE runs, since it affects the amount of background knowledge being used. Specifically, LogAnswer uses a very small synonym system for answering definition questions (it only contains spelling variants and very obvious synonyms that everybody takes for granted). In this way, definitions by stating a synonym are not regarded as trivial answers in the synonym mode. For factoid questions, on the other hand, the full system of all synonyms known to LogAnswer is used.

Let us now consider the additional processing steps applied to the parses of the reading test documents that serve to further elaborate the document representation for subsequent logical processing.

2.3 Coreference Resolution

When parsing the reading tests, which are multi-sentence documents, the coreference resolver CORUDIS [8] is applied. The discourse entities mentioned in the individual sentences are represented by nodes in the corresponding semantic network. CORUDIS finds groups of such nodes that denote the same individual. Among other coreference phenomena, this allows CORUDIS to handle anaphoric pronouns that refer to entities introduced earlier in the text. Note that the coreference resolution, and the identification of all constants that refer to the same individuum, are essential to the success of logic-based processing on the document level. This is because, due to the unique name assumption, the combined information about all mentions of a discourse entity can only be utilized if all mentions of this discourse entity are represented by the same constant.

2.4 Recognition of the Speaker of a Talk

A special task not handled by CORUDIS is the recognition of the speaker of a talk. In the reading test documents, which are manuscripts of talks, the speakers refer to themselves in the first person singular as “ich” (I), and the name of the speaker is only mentioned once at the beginning of the manuscript of the talk. Knowing the speaker is essential for QA4MRE, since many questions refer to the name of the speaker of the talk. The LogAnswer prototype for QA4MRE contains a simplistic solution to identifying the speaker from the manuscript of a talk; the method simply chooses the first name mentioned in the manuscript. All mentions of the first person singular in the text are then identified with the found speaker, so that the corresponding pieces of information become accessible via the speaker’s name.

2.5 Assimilation of Sentence Representations

By *assimilation* we mean the integration of the semantic representations of individual sentences of a text into a semantic representation of the text as a whole. In the case of semantic networks, this means that the graphs are merged, keeping each edge that occurs in several sentence networks only once. Coreference resolution and the special treatment of first person singular ensure that mentions of the same entity in different sentences are finally represented by the same node in the network (modulo errors of coreference resolution, of course). The semantic networks are translated into a logical representation by treating nodes as constants and edges as literals based on the edge symbol as a binary predicate. The logical representation of the text thus becomes a large conjunction of ground literals.

2.6 OpenCyc-Based Concept Expansion

For QA4MRE, we have aligned our concept system (of HaGenLex [10] concepts for German) with a fragment of OpenCyc.[4] As described in Sect. 2.12, this process resulted in about 20,000 additional redundancy-free subconcept-superconcept relationships between HaGenLex concepts. We felt that integrating all of these subsumption relationships into each reading test would place too much load on the prover. Therefore, a map from lexical concept ids to all known superconcepts (as mediated by the OpenCyc link) is computed in advance. If an entity mentioned in the text belongs to a certain base concept, then it is automatically marked as belonging to all known superconcepts based on this pre-computed map. We thus use OpenCyc (via an alignment to our own concept system for German) in order to enrich the logical representation of the documents. For example, suppose that a document mentions a tsunami. From the OpenCyc link, we know that every tsunami is a "Naturkatastrophe" (natural disaster). The original sentence representation then contains a statement like SUB(*c243*, *tsunami.1.1*), expressing that entity *c243* is an instance of a tsunami (with HaGenLex word sense index *1.1*). The OpenCyc expansion then adds an assertion SUB(*c243*, *naturkatastrophe.1.1*), expressing that *c243* is also an instance of a natural disaster.

2.7 Hypothesis Construction

By a *hypothesis*, we mean a declarative statement formed from the question and the candidate answer. For example, given the question "Was ist Bonos Einstellung zum digitalen Zeitalter" (What is Bono's attitude with respect to the digital age) and the possible answer "Enthusiasmus" (enthusiasm), we may form the hypothesis "Bonos Einstellung zum digitalen Zeitalter ist Enthusiasmus" (Bono's attitude with respect to the digital age is enthusiasm).

[4] We used the OWL version of OpenCyc, `http://sw.opencyc.org/`

The construction of a hypothesis is crucial for answer validation since it is the validity of the hypothesis (given the reading test document and the knowledge of the system) that eventually decides if the answer is justified or not.

The hypothesis can be formed on the textual level, but for our purposes it is only important to know the logical representation of the hypothesis, which can be constructed from the logical analysis of the question and of the answer. The question literals and answer literals are then combined into a conjunction of hypothesis literals as explained in [11].

The benefit of hypothesis construction on the logical level is that it works well for highly inflecting languages like German. In fact, a template-based construction of textual hypotheses followed by another application of the parser is problematic for German due to mismatches of word forms with respect to syntactic case. On the other hand, QA4MRE comes with a diversity of answers expressed as nominal phrases, prepositional phrases, numbers, adjectives, adverbs, (causal or modal) auxiliary sentences, or infinitive constructions. Not all of these answers can be parsed by WOCADI, as required for hypothesis construction. Therefore we decided to implement a traditional, textual hypothesis construction as a fallback solution to be applied when the direct construction of the hypothesis on the logical level fails. Here, part of the question is replaced by the candidate answer, the resulting textual hypothesis is parsed by WOCADI, and a literal pattern for the hypothesis is formed from the parsing result. This fallback technique was only implemented for special question types, such as causal questions, modal questions, "wann" (when) questions, "wie viele" (how many) questions, and "in welchem Jahr" (in which year) questions.

2.8 Robust Logical Processing

While LogAnswer normally combines logic-based and shallow techniques, we opted for a pure logic-based validation approach in QA4MRE (mainly due to lack of training data for a robust multi-criteria approach).

In principle, the logic-based validation is accomplished by trying to prove the logical hypothesis from the logical representation of the reading test document and from the general background knowledge of the system. If such a proof succeeds, then the answer from which the hypothesis was constructed is considered correct, otherwise it is incorrect. However, due to errors of the linguistic analysis, failure of coreference resolution, and gaps in the background knowledge, it is not realistic to demand a perfect logic-based validation based on a complete proof of the hypothesis.

Thus, if a complete proof of the query fails, then the system resorts to query relaxation. We have described the use of relaxation in LogAnswer elsewhere [2]. For the purposes of this paper, it is sufficient to know that the system will remove problematic hypothesis literals one at a time until a proof of the remaining hypothesis fragment succeeds. In practice, we impose a limit on the number of admissible relaxation steps.

In general, the result of this kind of robust logical processing is only a proof of a hypothesis fragment. From this we cannot conclude that the hypothesis as

a whole would be provable assuming ideal knowledge and an error-free linguistic analysis. However, the more literals must be skipped, the less likely it becomes that the hypothesis is indeed correct. Thus, we use the skipped literal count and other indicators related to the relaxation proof as the basis for scoring answers.

2.9 Computation of Scoring Metrics

Due to the lack of a training set for QA4MRE, we decided to use a primitive ad-hoc scoring formula. This formula omits many of the answer selection features normally used by LogAnswer, because we felt unable to integrate all these features in a hand-coded scoring metric.

In order to be able to calculate the scoring metrics, the result of the last relaxation proof based on the hypothesis for the answer of interest is needed. The relaxation result consists of the list of hypothesis literals that the system managed to prove in the last proof attempt, the list of failed literals that were skipped by query relaxation, and a list of literals with unknown status. These are the literals that the prover was not able to prove in the last proof attempt of the relaxation loop, but that are also not yet classified as skipped literals.[5] Using the same criteria, we can not only split the hypothesis literals in *proved-h-lits*, *skipped-h-lits* and *unknown-status-h-lits*. We can also restrict attention to the hypothesis literals that stem from the question, obtaining *proved-q-lits*, *skipped-q-lits* and *unknown-status-q-lits*. Similarly, we can restrict attention to the hypothesis literals that stem from the answer, yielding literal sets *proved-a-lits*, *skipped-a-lits* and *unknown-status-a-lits*.

The basic scoring metric ϱ for an answer candidate is then defined as the arithmetic mean of the following six scoring criteria:

- ϱ_1: based on the number of skipped literals and a parameter $\alpha = 0.7$:

$$\varrho_1 = \alpha^{\#\textit{skipped-lits}}$$

- ϱ_2: additionally based on the number of literals with unknown status and a parameter $\beta = 0.8$:

$$\varrho_2 = \alpha^{\#\textit{skipped-lits}} \cdot \beta^{\#\textit{unknown-status-lits}}$$

- ϱ_3: optimistic proportion of provable question literals:

$$\varrho_3 = 1 - \frac{\#\textit{skipped-q-lits}}{\#\textit{all-q-lits}}$$

- ϱ_4: pessimistic proportion of proved question literals:

$$\varrho_4 = \frac{\#\textit{proved-q-lits}}{\#\textit{all-q-lits}}$$

[5] They can be provable given the remaining query fragment or not, but we do not know since the relaxation cycle has already been stopped.

– ϱ_5: optimistic proportion of provable answer literals:

$$\varrho_5 = 1 - \frac{\#\textit{skipped-a-lits}}{\#\textit{all-a-lits}}$$

– ϱ_6: pessimistic proportion of proved answer literals:

$$\varrho_6 = \frac{\#\textit{proved-a-lits}}{\#\textit{all-a-lits}}$$

The basic score ϱ as described so far is combined with a second score that tries to capture the coherence of the evidence for the answer, judging from the witness sentences that justify the answer (see the Sect. 2.11 on provenance information for a description of how this list of sentences that justify the answer is computed).

Given this set of sentences that were needed for the proof, the sentences are now bundled into sequences of directly adjacent sentences without intermittent gaps. For example, if the proof of the hypothesis from the representation of the test document involves sentences number 1, 2, 4, 5, and 7 from the text, then we have three groups of adjacent sentences, $\{1, 2\}$, $\{4, 5\}$, $\{7\}$. We then iterate over these blocks in the natural order of the text. If one of the sentences in the current block involves a discourse entity (expressed by a constant) introduced in a block that was already considered, then the current block is viewed as connected to the earlier block, otherwise it is considered unconnected. We count the number $u = u(a)$ of all unconnected blocks of witness sentences for the given answer a.

The *final answer selection score* σ is computed from the basic score ϱ and the unconnected sentence score u as $\sigma = \varrho \cdot \gamma^{u-1}$, where $\gamma = 0.7$ in our experiments.

2.10 Selection of the Best Answer, or NOA Decision

According to the QA4MRE guidelines, systems can either commit to one of the answer candidates for a given question, or they can refuse to answer the question, leaving it unanswered. Since the LogAnswer prototype developed for QA4MRE uses a pure logical validation approach, a no answer (NOA) response will always be generated if question processing fails (no full parse of question or empty question literal pattern). Moreover, the answer candidates for a question can only be evaluated if they admit the construction of a logical hypothesis (either directly from the answer parse, or indirectly by parsing a textual hypothesis), and if a non-empty set of answer literals is part of the generated hypothesis. Finally, the answer must come with valid provenance information (see Sect. 2.11), since the QA4MRE guidelines do not allow empty provenance data. If no answer candidate for a question fulfills these requirements, then a NOA decision is made. Otherwise the answer with maximum score will be chosen, even if the score is so low that it indicates a poor quality of the validation, and even if there is no clear preference for one answer over another (i.e. even if there is no clear winner judging from the validation scores). We did not introduce a NOA threshold for validation scores due to lack of development data.

2.11 Provenance Generation

The QA4MRE result specification requires systems to provide one or more so-called provenance elements for each answered question. These provenance elements are expected to provide an explanation that justifies the chosen answer. They can include sentences from the reading test, from documents in the QA4MRE background collection, or knowledge specific to the QA system.

The computation of provenance information is based on the results of the prover in the last proof of the relaxation cycle. The prover outputs the list of used axioms (i.e. used implicative rules) and the answer substitution (for the proved fragment of the query). The names of the used axioms are directly turned into provenance elements. Since these names are usually chosen in a meaningful way, showing the names of the axioms should be informative to users.

Apart from the axiom names, it is important to reconstruct the used facts from the background knowledge and the relevant sentences of the reading test document itself. To this end, we first determine the list of all used facts (in our system, these facts are always variable-free, i.e. ground facts). We start with the literals from the instantiated hypothesis pattern and add all literals that occur in the premise of instantiated used axioms. From this set we remove all literals that occur in the conclusion of any instantiated used axiom. The remaining literals are ground facts that occur either in the background knowledge of LogAnswer or in the logical representation of the reading test document.

Those facts that stem from the background knowledge of LogAnswer are easily identified since they are not contained in the representation of the reading test document. If a used fact is identified as stemming from the background knowledge, then it is directly turned into a provenance element. Since the arguments of such a fact are lexical concept identifiers and the predicate usually describes a lexical-semantic relationship (e.g. relationship between a verb and its nominalization), the meaning of such a fact is self-explanatory.

For those used facts that are not background facts, the corresponding sentence from the reading test document should be included as provenance information, rather than the logical fact itself that was extracted from the sentence. For example, the fact SUB(*c243*, *tsunami.1.1*) represents a concrete tsunami mentioned in the text. Since a single sentence can cover a large number of such facts, it is preferable to present the sentence instead of the individual facts. In some cases there may be indeterminacy in the process of determining matching sentences, i.e. there can be several sentences whose meaning representation contains a considered fact.

In order to find good configurations of witness sentences, we iterate over all used facts, starting with those facts with the smallest number of available witness sentences and proceeding to facts with more possible witness sentences.

For the considered fact, we first look if the fact is contained in one or more of the already chosen witness sentences. If yes, then the fact is already covered and no new witness sentence must be added. Otherwise we must choose a new witness sentence. To this end, we consider all witness sentences containing the considered fact. We then choose that sentence that covers the highest number of

other used facts not covered by the witness sentences already chosen. In case of ties, we prefer the sentence that also has a higher number of covered used facts (without the restriction to those used facts that still need justification).

As the result of this process, we know a selection of witness sentences from the reading test document that together cover all considered used facts. We thus add provenance elements for all of these sentences, in the order in which they occur in the text. The chosen witness sentences also affect the validation score of the considered answer, see Sect. 2.9.

The extraction of provenance information may fail in rare cases, resulting in empty provenance data. Since the task specification of QA4MRE requires at least one provenance element to be included for each non-NOA decision, the LogAnswer system will drop the best answer and resort to a NOA response instead if the provenance information is empty.

2.12 Resources Used: OpenCyc Integration

Concerning the general background knowledge of LogAnswer and the resources used, we refer to [2]. In order to allow additional inferences, the existing background knowledge of LogAnswer was enhanced by knowledge from OpenCyc. To this end, we had to couple (part of) our concept system for German with the concept identifiers used in OpenCyc. We established this connection in two ways: First of all, we know corresponding English words for ca. 6,000 of our German lexemes, due to the evolving English lexicon HaEnLex that is fully aligned with the German HaGenLex [10]. If an OpenCyc concept id was formed from the same word, or if it was linked to an English Wikipedia topic with the same name as the word of interest, then the OpenCyc concept was considered as an alignment candidate for the original German lexeme. Alternatively, we used the following chain for generating alignment hypothesis: Starting from the German lexeme, we looked for matching topics or redirects in the German Wikipedia. Then, the DBpedia[6] map from German to English topics was used, and from there on the OpenCyc-Wikipedia link was used, again resulting in alignment candidates. Several ten thousands of these alignment candidates were manually checked, and all subconcept-superconcept relationships were extracted (and mapped back into our German concept world) based on these manually verified alignments. After eliminating redundancy and inconsistencies, we ended up with ca. 20,000 subconcept-superconcept relationships and around 1,500 new synonymy relationships between German lexemes.

3 Experimental Results

3.1 System Configuration in the QA4MRE Runs

Two runs were submitted for the QA4MRE task for German, based on different configurations of the LogAnswer prototype. The *loga1101dede* run was subject

[6] see `http://dbpedia.org`

to the restrictions of the QA4MRE guidelines on admissible resources for the first submitted run. Therefore, the synonym system used by LogAnswer in the first run was deliberately restricted to a very limited set of synonyms that only covers spelling variants but no synonyms proper.[7] The restriction was imposed since some of the synonyms stem from external resources (such as GermaNet), and external resources or ontologies may not be used in the first submitted run according to the guidelines.

The rule-based background knowledge of LogAnswer was also filtered, by eliminating all lexical-semantic relations (such as subconcept-superconcept relationships), and by eliminating any rules expressing domain knowledge. We only kept general logical rules that specify the behaviour of the expressive means of MultiNet, and rules for exploiting simple temporal and local regularities. This filtering was made because rule bases and ontologies may not be used in the first run. We checked back with the QA4MRE organizers and they confirmed that the general rules that were kept belong to the core functionality of the QA system, so that they were admissible in the first run.

The second run, *loga1102dede*, is not subject to the special restrictions on resources that the QA4MRE guidelines impose on the first run. So, we have activated the full knowledge used by LogAnswer, including the full set of synonyms and all other lexical-semantic facts and logical rules available in the system. Moreover, the OpenCyc integration was activated in the second run, resulting in about 20,000 additional subconcept-superconcept relationships to be utilized.

The two runs were generated with the following parameters: The initial proof depth for iterative deepening for the prover was set to 1 and the maximum proof depth to 2. Each single proof attempt in the relaxation process was allowed a 2 seconds time. Up to 5 relaxation steps were allowed for a given answer candidate.

3.2 Results Achieved

The results of the two LogAnswer runs were different looking at individual choices, but as a whole exactly the same scores were achieved: In both runs, LogAnswer committed to one of the answers for 88 out of the 120 questions, leaving 32 questions unanswered. Considering the answered questions, the chosen answer was correct in 21 cases. Thus, the overall accuracy was 0.18, and the accuracy for those questions that LogAnswer decided to answer was 0.24. The resulting c@1 score was 0.22. Compared to other systems, we find that LogAnswer scored slightly better than the average c@1 score of 0.21 considering all participants. However, the result of LogAnswer cannot be satisfactory since it is very close to random guessing (which would yield an expected c@1 score of 0.2), and the margin to the best system with a c@1 score of 0.57 is quite large.

[7] That is, the synonyms normally used for definition questions are used for all questions in the first run.

3.3 Error Analysis and Discussion

We start by pointing out a few problems related to the QA4MRE test set for German, then turning to the issues of the LogAnswer prototype developed for the QA4MRE task.

Poor Document Quality (Missing Blanks etc.) The documents in the German reading tests were of low encoding/formatting quality: There were many missing blanks (probably due to a systematic error when generating the test set), resulting in adjacent words to be wrongly merged into single tokens. In one case even the name of the speaker was wrongly merged with the next word, resulting in this essential piece of information for answering the questions to be completely unavailable. Moreover, there was no structuring by blank lines at all, so that headlines were not clearly separated from the following body of regular text. The result of all this was a loss of useful information in the documents, a poor parsing rate, and total failure of the coreference resolver (see below).

Non-Parseable Questions WOCADI has found a full parse for 106 out of the 120 questions, which is quite satisfactory. We noticed that 3 out of the 14 problematic questions cannot be parsed due to spelling errors or wrong punctuation; these problems could have been avoided by a correction of the test set.[8]

Parsing Rate for Answers We have already remarked in Sect. 2.7 that the syntactic diversity of the answers in the task poses problems to parsing and to hypothesis construction. LogAnswer is flexible in that it can construct logical hypotheses either directly from the question and answer parse; or alternatively from the parse of a textual hypothesis constructed from question and answer. However, it frequently happened that a non-parseable answer also resulted in the constructed textual hypothesis to parse poorly. These difficulties resulted in the generation of 31 NOA responses (another NOA response was due to failed provenance generation).

Let us now turn to the main issues of the system prototype that were revealed by the QA4MRE participation.

Failure of Coreference Resolution Due to the missing blank issue of the documents, but also due to their length, the coreference resolver CORUDIS failed completely. It was not able to find a regular coreference resolution result (a so-called 'coreference partition') for *any* of the twelve reading test documents. While CORUDIS provided some fallback information, it was so sparse that it was practically useless. The failure of the coreference resolver means that no information at all is merged beyond the sentence level. Thus, a document-level knowledge processing became virtually impossible.

[8] A comma is missing in question 9 for reading test 3, and in question 8 for reading test 4. Moreover "das" in question 3 of reading test 7 is wrongly spelled "dass".

Extraction of Provenance Information The extraction of provenance information worked correctly, although the results were often confusing because the struggling logical processing (and extreme relaxation) produced confusing results. Only in one case, a correct answer of the LogAnswer system had to be dropped because the system was not able to extract any provenance information.

Missing Integration of OpenCyc Synonyms in the Second Run In general, several lexical concepts in our German concept space can be associated with the same OpenCyc concept; in this case, one can deduce a synonym relationship between the German lexemes. As mentioned in Sect. 2.12, this mechanism resulted the detection of about 1,500 new synonyms. Unfortunately, they were not included in the second LogAnswer run. So by accident, about 1,500 possible links between German lexemes, and also the connection of these lexemes to the superconcepts mediated by OpenCyc, were lost.

4 Conclusions and Future Work

We have built a new LogAnswer prototype for the QA4MRE task. The system implements the main goals defined in the introduction, but it was clearly not yet mature at the time of the QA4MRE participation.

Due to substantial problems with the quality of the reading test documents, and most importantly due to the total failure of the coreference resolver for these documents, it was impossible to successfully demonstrate the utility of the logic-orientied approach to answer validation in the task. In particular, the extension of the background knowledge of LogAnswer by additional subconcept-superconcept relationships from OpenCyc had no positive effect at all, since the qualitative requirements for using logical reasoning on the document level were obviously not met.

We did not try falling back to shallow linguistic methods (such as a lexical overlap measure) when logical processing fails. One reason for that was the lack of training data for QA4MRE. Moreover the QA4MRE guidelines discourage the use of such lexical methods, pointing out that examples are chosen such that there is no close word-to-word relationship between hypothesis and answer passage.

We conclude that it would have been best to complement answer selection by a criterion which does not assume a full semantic analysis of the documents, and which is also not based on lexical overlap. Examples are popularity or frequency criteria, for example based on hits in the QA4MRE background collection, or based on an integration of a search engine (such as google) for estimating hit counts.

The attempt to work without the background collection for the moment can also be regarded as futile. However, it is possible that using an external collection (such as the Wikipedia) will achieve similar effects as using the background collection. Moreover the best way of utilizing the QA4MRE background collection is not obvious.

Looking at concrete questions, it is clear that question decomposition would have been helpful in many cases. Besides that, a decomposition of conjunctive answers (which express a conjunction of two or more subanswers) would also have been useful.[9] An example of a conjunctive answer from the test set is "Edinburgh und Oslo" (Edinburgh and Oslo).

Knowing basic biographic data about a person would also have been useful, especially name variants (e.g. full name of a person, stage name or pen name). We will account for this kind of information in the further development of LogAnswer. Our specific goal is a coupling with the DBPedia, which could often have provided the required information in the QA4MRE task.

The success of LogAnswer in the new task is important to us, since QA4MRE (as opposed to earlier QA@CLEF tasks in our opinion), contains questions of realistic difficulty, where the way in which the question and text are phrased are not artificially similar. Therefore we hope that the envisioned changes to LogAnswer will also help us to improve our success in the application domain of our choice (i.e., providing QA support in human question answering portals).

Apart from the mentioned deficiencies of implementation and methods, the lack of a development set for German was also a problem. Without a development set, it was not possible to train validation models using techniques from machine learning (and due to the very different characteristic of QA4MRE compared to the earlier ResPubliQA oder QA@CLEF tasks, a reuse of existing models seemed pointless). Finally, it was not possible to establish an optimal threshold for the NOA decision, so that no threshold at all was applied in our submitted runs. The basis for solving these problems is much better now, since experimental data from the first round of QA4MRE are available.

References

1. Peters, C., Deselaers, T., Ferro, N., Gonzalo, J., Jones, G., Kurimo, M., Mandl, T., Peñas, A., Petras, V., eds.: Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, Revised Selected Papers, Heidelberg, Springer (2009)
2. Furbach, U., Glöckner, I., Pelzer, B.: An application of automated reasoning in natural language question answering. AI Communications **23**(2-3) (2010) 241–265
3. Glöckner, I., Pelzer, B.: Combining logic and machine learning for answering questions. [1]
4. Glöckner, I., Pelzer, B.: The LogAnswer project at CLEF 2009. In: Working Notes for the CLEF 2009 Workshop, Corfu, Greece (September 2009)
5. Glöckner, I., Pelzer, B.: The LogAnswer project at ResPubliQA 2010. In: CLEF 2010 Working Notes. (September 2010)
6. Pelzer, B., Glöckner, I., Dong, T.: Loganswer in question answering forums. In: 3rd International Conference on Agents and Artificial Intelligence (ICAART 2011), SciTePress (2011) 492–497

[9] This should be much easier to achieve compared to question decomposition. The idea is validating the individual conjuncts of a conjunctive answer separately. Then, we use the minimum score of the conjuncts as the score of the total answer. Moreover, all provenance elements of the individual answers have to be merged.

7. Dong, T., Furbach, U., Glöckner, I., Pelzer, B.: A natural language question answering system as a participant in human Q&A portals. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-2011), Barcelona, Spain (July 2011) 2430–2435
8. Hartrumpf, S.: Hybrid Disambiguation in Natural Language Analysis. Der Andere Verlag, Osnabrück, Germany (2003)
9. Helbig, H.: Knowledge Representation and the Semantics of Natural Language. Springer, Berlin (2006)
10. Hartrumpf, S., Helbig, H., Osswald, R.: The semantically based computer lexicon HaGenLex. Traitement automatique des langues **44**(2) (2003) 81–105
11. Glöckner, I.: University of Hagen at QA@CLEF 2007: Answer validation exercise. In: Working Notes for the CLEF 2007 Workshop, Budapest (2007)