The LogAnswer Project at CLEF 2008: Towards Logic-Based Question Answering

Ingo Glöckner[1] and Björn Pelzer[2]

[1] Intelligent Information and Communication Systems Group (IICS), University of Hagen, 59084 Hagen, Germany
ingo.gloeckner@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

LogAnswer is a logic-oriented question answering system jointly developed by the AI research group at the University of Koblenz-Landau and by the IICS at the University of Hagen. The system was designed to address two notorious problems of the logic-based approach: Achieving robustness and acceptable response times. The main innovation of LogAnswer is its use of logic for simultaneously extracting answer bindings and validating the corresponding answers. In this way the inefficiency of the classical answer extraction/answer validation pipeline is avoided. The prototype of the system, which can also be tested on the web, demonstrates response times suitable for real-time querying. Emphasis was also placed on developing techniques for making the logic-based approach more robust against gaps in the background knowledge and against errors of linguistic analysis. To this end, the optimized deductive subsystem is combined with shallow techniques by machine learning. The same background knowledge as in the MAVE validator of the IICS presented at CLEF 2007 was used: 10,000 lexical-semantic relations (e.g. describing nominalizations), 109 logical rules, and a list of synonyms covering more than 111,000 lexical constants which is also utilized for determining the shallow features. Two monolingual runs of LogAnswer for German were submitted to QA@CLEF 2008. The results of 29 correct answers in the best run (accuracy: 0.145) indicate that further development of the current prototype is necessary. An error analysis shows that the linguistic processing and also the coreference resolution generally performed quite well. The rudimentary implementation of answer extraction based on the answer substitution determined by the prover must be improved, though, since extracted answers for appositions and constructions involving a defining verb are not reliable yet.[3]

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search Process, Selection process*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Predicate Logic, Semantic networks*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Experimentation, Measurement, Verification

Keywords

Logical Question Answering, Real-Time Question Answering, Answer Selection, Robust Inference

[3]Funding of this work by the DFG (Deutsche Forschungsgemeinschaft) under contracts FU 263/12-1 and HE 2847/10-1 (LogAnswer) is gratefully acknowledged.

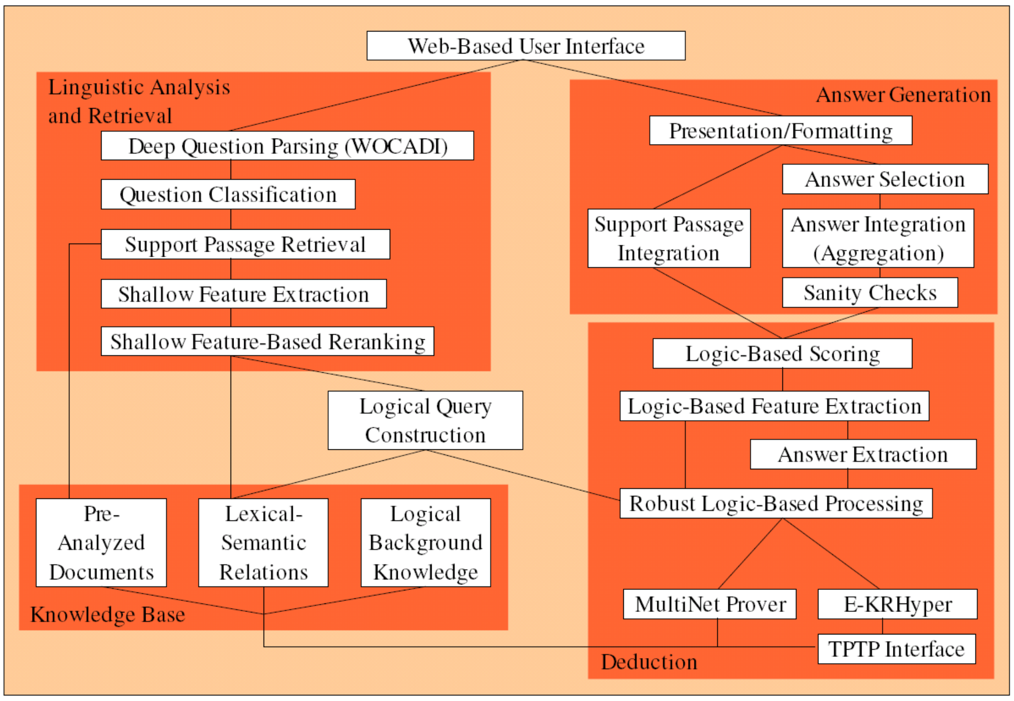
Web-Based User Interface
Linguistic Analysis and Retrieval
Deep Question Parsing (WOCADI)
Question Classification
Support Passage Retrieval
Shallow Feature Extraction
Shallow Feature-Based Reranking
Answer Generation
Presentation/Formatting
Answer Selection
Support Passage Integration
Answer Integration (Aggregation)
Sanity Checks
Logic-Based Scoring
Logic-Based Feature Extraction
Answer Extraction
Logical Query Construction
Robust Logic-Based Processing
Pre-Analyzed Documents
Lexical-Semantic Relations
Logical Background Knowledge
Knowledge Base
MultiNet Prover
E-KRHyper
TPTP Interface
Deduction

Figure 1: System architecture of the LogAnswer prototype

1 System Description

The system architecture of the LogAnswer QA system is shown in Fig. 1. In the following we describe the processing stages of the system.

Question Input In normal operation of the system, the natural language question of the user is entered into the LogAnswer web search box.[4] For QA@CLEF, a batch querying option was added.

Deep Linguistic Processing of the Question The question is analyzed by the WOCADI parser [7], which generates a semantic representation of the question in the MultiNet formalism [8]. The standard coreference resolution module of WOCADI is used for treating follow-up questions involving pronouns and nominal anaphora.

Question Classification The question classification of LogAnswer, based on 127 recognition rules, identifies the category (*factual* vs. *definition*) and the expected answer type (e.g. *PERSON*) of the question. The descriptive core of the question is also identified. Consider *Nennen Sie einige einfache Elementarteilchen!* (*'Name some elementary particles!'*). Then *nenne* (*'name'*) is not treated as part of the query content since it only specifies what the system should do but does not describe the correct answers.

Support Passage Retrieval The document collection of LogAnswer comprises the November 2006 snapshot of the German Wikipedia; for QA@CLEF, the news collection of CLEF was also added. In order to avoid parsing of documents at query time, all documents are pre-analyzed by the WOCADI parser. The resulting MultiNet representations are segmented into passages and stored in a Lucene-based retrieval module.[5] An interesting aspect of LogAnswer is the kind of information stored in the index:

[4]The system is available online at `www.loganswer.de`.

[5]Notice that at present, only single-sentence snippets are considered, but an extension to larger passages is planned for the future.

- The system uses lexical concepts (word senses from the lexicon) rather than word forms or word stems for indexing. At the moment, all possible word senses for each word are indexed.
- Synonymy relationships are utilized for replacing all possible synonym variants by a canonical representation.[6] For example, the lexical concept *attacke.1.1* (attack) is replaced by the canonical *angriff.1.1* during indexing. A similar normalization at query time ensures that all synonym variants can be used for retrieval.
- Nominalizations are utilized for indexing: if the text contains *erfindung.1.1* (invention), for example, then *erfinden.1.1* (word sense of *'to invent'*) is also added to the index, and vice versa.
- Compound decompositions are added to the index. For example, if the text contains an occurrence of *verteidigungsminister.1.1* (minister of defence), then *minister.1.1* is also indexed.
- Adjective-attribute relationships are expanded. Thus, an occurrence of *hoch.1.1* (high) results in *höhe.1.1* (height) to be indexed as well.

Moreover all answer types contained in a sentence are indexed. In order to improve retrieval results for definition question, information about the containment of appositions, relative clauses, copula constructions, and defining verbs like *stehen für* (*'stand for'*), is also stored in the index. Notice that only sentences with a successful parse were indexed since the subsequent logic-based answer extraction requires the semantic representation constructed by the parser. For generating the submitted runs, the system was configured to retrieve 100 supporting snippets per question.

Shallow Feature Extraction and Reranking In order to avoid logical processing of all retrieved passages, LogAnswer tries to identify the most promising cases by reranking passages using shallow features which can be computed very quickly without the help of the prover. These features comprise: *failedMatch* (number of lexical concepts and numerals in the question which cannot be matched with the candidate document); *matchRatio* (relative proportion of lexical concepts and numerals in the question which find a match in the candidate document); *failedNames* (proper names mentioned in the question, but not in the passage); *containsBrackets* (the passage contains a pair of parentheses); *knownEat* (the expected answer type is known); *testableEat* (the expected answer type is fully supported by the current implementation of the answer type check); *eatFound* (an occurrence of the expected answer type has been found in the snippet); *isDefQuestion* (the question is a definition question). The *defLevel* feature is useful for definition questions. A value of $\mathit{defLevel} = 2$ indicates that the snippet contains a defining verb or apposition, and $\mathit{defLevel} = 1$ indicates a relative clause. Finally, there is an *irScore* feature which provides the retrieval score determined by Lucene. The machine learning approach used for reranking the retrieved snippets based on the shallow features is the same as in [5, 6]. The Weka toolbench [12] was used for implementation. The training data consisted of 17,350 annotated snippets retrieved in a run of LogAnswer on the QA@CLEF 2007 questions.

Logical Query Construction The semantic network for the question is turned into a conjunctive list of query literals. For example, *Wie hoch ist der chilenische Berg La Silla?* (*'How high is the Chilean mountain La Silla?'*) translates into the following logical query (with the $FOCUS$ variable representing the queried information):

$$\begin{gathered}\mathrm{modp}(X_1, FOCUS, \mathit{hoch.1.1}), \mathrm{sub}(X_2, \mathit{berg.1.1}), \mathrm{prop}(X_2, X_1), \mathrm{attr}(X_2, X_3),\\ \mathrm{prop}(X_2, \mathit{chilenisch.1.1}), \mathrm{val}(X_3, \mathit{la\_silla.0}), \mathrm{sub}(X_3, \mathit{name.1.1}).\end{gathered}$$

During query construction, concept identifiers of synonyms are normalized by replacing the original concept identifiers with canonical synset representatives (however, no replacement occurs in the example).

[6]The system uses 48,991 synsets (synonym sets) for 111,436 lexical constants.

Robust Logic-Based Processing LogAnswer uses logic for simultaneously extracting and validating answers. To this end, the system tries to prove the logical representation of the question from the representation of the passage and the background knowledge.[7] Robustness is gained by using relaxation: if a proof is not found within a time limit, then query literals are skipped until a proof of the remaining query succeeds, and the skip count indicates (non-)entailment [3, 6]. For efficiency reasons, relaxation is stopped before all literals are proved or skipped; a maximum of 3 relaxation cycles was chosen for the QA@CLEF runs. Notice that relaxation does not necessarily find the largest provable query fragment, since it only inspects a single sequence of simplification steps. Moreover the choice of skipped literals usually depends on factors like internal literal order of the prover which are arbitrary to some degree. It therefore makes sense to abstract from such idiosyncratic aspects, by combining relaxation results of different provers. LogAnswer has interfaces to two provers in order to permit such a combination:

- The system includes a native prover for MultiNet representations, which is part of the MWR+ toolbench.[8] The MultiNet prover is very limited in expressive power (it only supports inferences over range-restricted horn formulas), but its specialization to the task ensures high efficiency [4].
- E-KRHyper [10] is the latest version in the KRHyper-series of theorem provers and model generation systems for first-order logic with equality developed at the University Koblenz-Landau. It is an implementation of the E-*hyper tableau calculus* [2], which integrates a superposition-based handling of equality into the hyper tableau calculus [1]. E-KRHyper is capable of handling large sets of uniformly structured input facts, and it can rapidly switch and retract input clause sets for an efficient usage as a reasoning server. Embedded in the LogAnswer system, E-KRHyper is supplied with the MultiNet axioms transformed into first-order TPTP syntax [11]. The inference process then operates on the axioms and the negated query literals, with a refutation result indicating a successful answer and providing the binding for the queried variable. If the reasoning is interrupted due to exceeding the time limit, then partial results can be retrieved that can guide in the relaxation process [9].

Answer Extraction If a proof of the question from a passage succeeds, then LogAnswer obtains an answer binding which represents the queried information. In order to find more answers, LogAnswer also tries to determine a substitution when a strict proof of the query fails. The system then resorts to the intermediate substitution of the prover for the largest proven fragment of the query. LogAnswer uses word alignment information provided by WOCADI for extracting the corresponding answer string from the supporting text passage.

Logic-Based Feature Extraction In the following it will be assumed that the answer extraction was successful, i.e. a binding to the queried variable was found and the system has managed to determine the corresponding answer string. Based on the results of the relaxation proof and on the extracted answer, LogAnswer then determines the following logic-oriented features: *skippedLitsLb* (number of literals skipped in the relaxation proof); *skippedLitsUb* (number of skipped literals, plus literals with unknown status); *litRatioLb* (relative proportion of actually proved literals compared to the total number of query literals, i.e. $1 - \mathit{skippedLitsUb}/\mathit{allLits}$); *litRatioUb* (relative proportion of potentially provable literals vs. all query literals, i.e. $1 - \mathit{skippedLitsLb}/\mathit{allLits}$); *npFocus* (the queried variable was bound to a constant which corresponds to a nominal phrase in the text); *focusEatMatch* (the answer type of the answer binding found by the prover matches the expected answer type). The *focusDefLevel* feature is relevant for definition questions. A value of $\mathit{focusDefLevel} = 2$ indicates that the answer binding found by the prover corresponds to an apposition, and $\mathit{focusDefLevel} = 1$ occurs if the answer binding corresponds to a noun phrase involving a relative clause.

Logic-Based Scoring The logic-based answer scores are computed by the same ML approach also used for the shallow reranking. However, the shallow and logic-based features are now combined for better precision. In regular operation of LogAnswer, passages are considered in the order determined by the

[7]The background knowledge of LogAnswer comprises 10,000 lexical-semantic facts (e.g. for nominalizations) and 109 logical rules, which define main characteristics of MultiNet relations and also handle meta verbs like ‘stattfinden’ (take place) [3].

[8]See `http://pi7.fernuni-hagen.de/research/mwrplus`

Table 1: Results of LogAnswer in QA@CLEF 2008

Run	#Right	#Unsupported	#Inexact	#Wrong	CWS	MRR
loga081dede	29	1	11	159	0.032	0.194
loga082dede	27	1	9	163	0.029	0.179

shallow feature-based ranking, and the logical processing is stopped after a pre-defined time limit. For QA@CLEF, this mechanism was switched off, i.e. all passages were considered for deep processing and answer extraction.

Support Passage Selection Depending on user preferences, the system answers the question either by presenting supporting text passages only, or alternatively, by presenting exact answers together with the supporting passage. For QA@CLEF, only the precise answer mode was relevant.

Sanity Checks Two sanity checks are applied in order to eliminate false positives: A triviality check eliminates answers which only repeat contents of the question. For the question *'Who is Virginia Kelley?'*, this test rejects trivial answers like *'Virginia'* or *'Virginia Kelley'*. A special sanity check also rejects incomplete answers to definition questions. For example, *'the mother of Bill Clinton'* is a correct answer to the above question, but *'the mother'* must be rejected as incomplete. Notice that the compatibility of expected and found answer type is not treated as a strict sanity check but rather encoded by answer-type related features passed to the machine learning method.

Aggregation and Answer Selection The answer integration module computes a global score for each answer, based on the local score for each passage from which the answer was extracted. The aggregation method is detailed in [5]. The $k = 3$ distinct answers with the highest aggregated scores were selected for the submitted QA@CLEF runs. For each answer, the supporting passage with the highest score was selected as the justification for the considered answer.

2 Results on the QA@CLEF Test Set for German

The results of LogAnswer in the QA@CLEF 2008 task are shown in Table 1. The first run, *loga081dede*, used only the native prover of the MultiNet toolkit for logical processing. The second run, *loga082dede*, used a combination of the MultiNet prover and of the E-KRHyper prover based on the 'OPT' method described in [6]. The motivation for using more than one prover is that following several relaxation paths by applying several provers might increase the chance of discovering a good provable query fragment. While the combination of both provers worked well in earlier experiments based on cross-validation reported in [6], results in the QA@CLEF 2008 task were slightly worse for the combined method compared to the first run which used only one prover.

3 Error Analysis

An error analysis was made for the *loga081dede* run in order to identify the main deficits of the subsystems of LogAnswer. Concerning the linguistic processing stage, it was found that parsing of the question failed for 4 out of the 200 questions. Moreover the coreference resolution produced useless results (like unresolved pronouns) for 5 questions. Thus, the linguistic processing of the question was successful for 191 out of 200 questions in the QA@CLEF test set for German. Turning to the passage retrieval stage, the 19,064 retrieved supporting sentences (95.32 per question) were assessed for containment of a correct answer. The annotation revealed that for 119 of the questions, at least one passage which provides a correct answer was retrieved (see Fig. 2 for more details on the performance of the passage retrieval module). This means that, assuming perfect answer extraction and validation, the system can theoretically answer 119

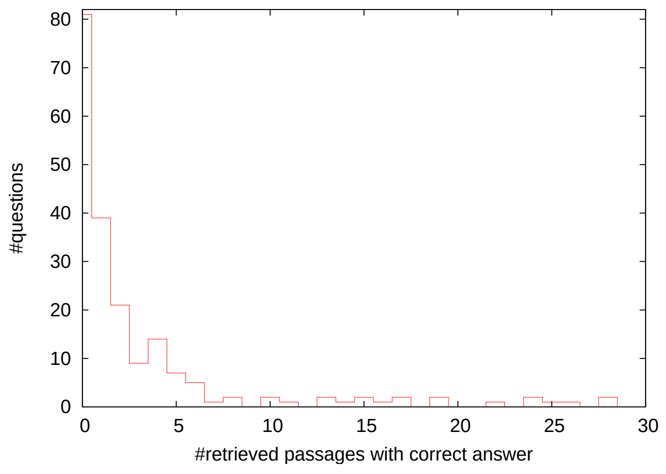

Figure 2: Number of questions with a given number of retrieved passages that answer the question

non-NIL questions correctly. In order to improve this number, the retrieval stage should be optimized. The following improvements are likely the most urgent:

- The retrieval module of LogAnswer is configured to return only 100 support sentences per question. Increasing this number will improve recall at the cost of more processing effort. Experiments should be made in order to find a good trade-off for these factors.
- The restriction to single-sentence snippets must be dropped. The system should use coreference resolution for elaborating the content analysis of the text. Moreover the document date should be taken into account for resolving deictic temporal expressions (like *'yesterday'*).
- At present, the system only indexes sentences with a perfect parse. This means that only about 60% of all sentences in the corpus are visible to LogAnswer. In order to improve recall, non-parseable answers should be indexed as well and a fallback method for answer extraction from answers without a parse must be added.

Another significant source of errors is answer extraction. LogAnswer found 26 correct non-NIL answers. However, 46 of the supporting snippets for the top-1 answers actually contain a correct answer. Assuming perfect selection, the system would therefore be able to correctly answer 46 non-NIL questions. This means that the success rate of answer extraction for sentences at top-1 position is 56.5%. Considering all top-3 results, we find that the achieved MRR for 120 questions with multiple answers was 0.1944, while for perfect extraction, an MRR of 0.3222 would have been possible in the *loga081dede* run. In practice, the ranking is not fixed though, but rather depends on the results of answer extraction. Therefore the overall success rate of the answer extraction stage for arbitrary snippets is even lower. The reason of these problems are two phenomena not adequately treated in LogAnswer yet:

- The answer is often expressed by an apposition, as in *Albert Einstein, der Begründer der Relativitätstheorie* (*'Albert Einstein, the founder of the theory of relativity'*). In this case, the answer extractor must not return the full noun phrase which corresponds to the answer binding of the queried variable. By contrast, it is necessary to split the extracted noun phrase and identify the relevant part.
- Copula constructions and constructions involving defining verbs also pose problems for the logic-based extraction method. If the sentence has a form such as *'X is Y'* or *'X means Y'*, then the logic-based answer extraction will often extract X even though the question targets at Y.

These problems result in wrong or inexact extractions, as witnessed by the relatively large number of 11 inexact answers of LogAnswer in the *loga081dede run*.

4 Conclusion

With LogAnswer, we have developed a logic-based QA system suitable for real-time question answering. Earlier experiments on the QA@CLEF 2007 questions for German confirm that the system works well when used for finding supporting sentences that contain an answer to the question [4, 6]. However, the naive solution for extracting exact answers that was added for generating runs for QA@CLEF 2008 is not yet reliable enough. Especially appositions, copula constructions and defining verbs pose problems to the current implementation of logic-based answer extraction. Nevertheless, the simultaneous extraction of answer bindings and validation features from a relaxation proof of the question from the supporting snippet should be investigated further, since it avoids the extraction of a large number of answer candidatates from which the few correct ones must then be selected by extensive validation. An intrinsic problem of logic-based answer extraction is that the method only works for snippets with a full parse. For the remaining sentences, there is no meaning representation of the snippet on the logical level and the prover cannot be applied. In order to overcome this limitation, the logic-based extraction should be complemented with a shallow QA technique which can be used for finding answers in snippets with a failed parse.

References

[1] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper Tableaux. In *JELIA'96, Proceedings*, pages 1–17, 1996.

[2] Peter Baumgartner, Ulrich Furbach, and Björn Pelzer. Hyper Tableaux with Equality. In *Automated Deduction - CADE-21, Proceedings*, 2007.

[3] Ingo Glöckner. University of Hagen at QA@CLEF 2007: Answer validation exercise. In *Working Notes for the CLEF 2007 Workshop*, Budapest, 2007.

[4] Ingo Glöckner. Towards logic-based question answering under time constraints. In *Proc. of the 2008 IAENG Int. Conf. on Artificial Intelligence and Applications (ICAIA-08)*, Hong Kong, 2008.

[5] Ingo Glöckner. University of Hagen at QA@CLEF 2008: Answer validation exercise. In *Working notes for the CLEF 2008 workshop*, Århus, Denmark, 2008.

[6] Ingo Glöckner and Björn Pelzer. Exploring robustness enhancements for logic-based passage filtering. In *Proc. of KES-2008*, Lecture Notes in Computer Science. Springer, 2008 (to appear).

[7] Sven Hartrumpf. *Hybrid Disambiguation in Natural Language Analysis*. Der Andere Verlag, Osnabrück, Germany, 2003.

[8] Hermann Helbig. *Knowledge Representation and the Semantics of Natural Language*. Springer, 2006.

[9] Björn Pelzer and Ingo Glöckner. Combining theorem proving with natural language processing. In *Workshop on Practical Aspects of Automated Reasoning (PAAR-08)*, Sydney, Australia, August 2008.

[10] Björn Pelzer and Christoph Wernhard. System Description: E-KRHyper. In *Automated Deduction - CADE-21, Proceedings*, pages 508–513, 2007.

[11] Geoff Sutcliffe and Christian Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[12] Ian H. Witten and Eibe Frank. *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.