Combining Logic and Machine Learning for Answering Questions

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, 56070 Koblenz
bpelzer@uni-koblenz.de

Abstract. LogAnswer is a logic-oriented question answering system developed by the AI research group at the University of Koblenz-Landau and by the IICS at the University of Hagen. The system addresses two notorious problems of the logic-based approach: Achieving robustness and acceptable response times. Its main innovation is the 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 be tested on the web, demonstrates response times suitable for real-time querying. Robustness to gaps in the background knowledge and errors of linguistic analysis is achieved by combining the optimized deductive subsystem with shallow techniques by machine learning.

1 Introduction

The LogAnswer project (funded by the DFG - Deutsche Forschungsgemeinschaft - under contracts FU 263/12-1 and HE 2847/10-1) is aimed at investigating the opportunities of a logic-based approach for question answering (QA). Special emphasis is placed on two problems that still obstruct the successful application of logic in practical QA systems: achieving robustness (i.e., how can a logic-based QA system find useful results given that its background knowledge is necessarily incomplete?), and efficiency (i.e., how can answers be generated within a few seconds, given the computational effort of deductive reasoning?) The paper explains the design of the LogAnswer prototype that tries to overcome these problems by combining logic and machine learning. Based on an analysis of the results of the system in QA@CLEF 2008, the main shortcomings of the first prototype are identified. The results of this error analysis are instructive since they illustrate some general issues for the logic-based approach to question answering.

2 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.

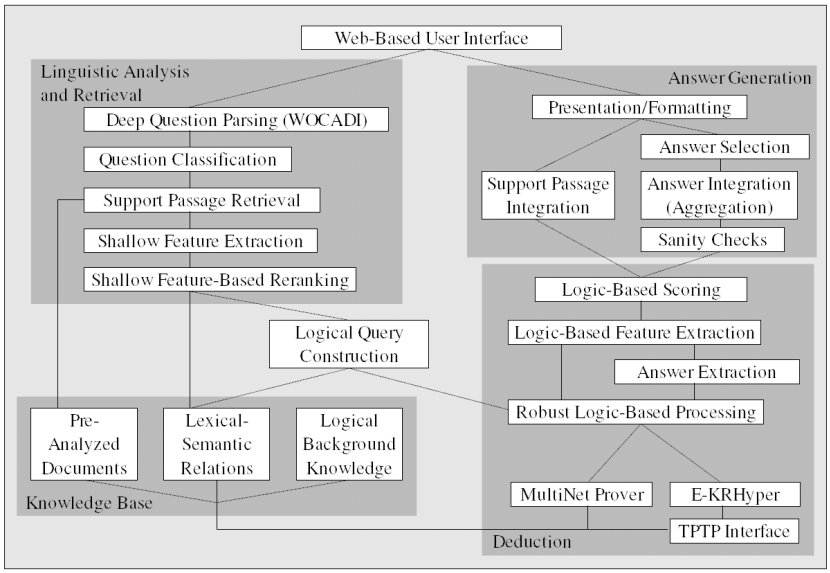

Fig. 1. System architecture of the LogAnswer prototype

Question Input In normal operation, questions are entered into the LogAnswer web search box.[3] For QA@CLEF, a batch querying option was added.

Deep Linguistic Processing of the Question The question is analyzed by the WOCADI parser [1], which generates a semantic representation in the MultiNet formalism [2]. The standard coreference resolution module of WOCADI is used for treating follow-up questions involving pronouns and nominal anaphora.

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

Support Passage Retrieval The document collection of LogAnswer comprises the 11/2006 snapshot of the German Wikipedia; for QA@CLEF, the news collection of CLEF was added. All texts are parsed by WOCADI at indexing time. The resulting MultiNet representations are segmented into passages and stored in a Lucene-based retrieval module.[4] The following kinds of information are indexed:

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

[4] 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) rather than word forms or word stems for indexing. However, there is no word sense disambiguation at the indexing level, i.e. all possible word senses for each word are indexed.
- Synonymy relationships are utilized for replacing all possible synonym variants by a canonical representation.[5] For example, *attacke.1.1* (attack) is replaced by the canonical *angriff.1.1* during indexing. Since all word senses of a word are normalized in this way, occurrences of other words that can have one of these meanings are also covered. A similar normalization at query time ensures that all synonym variants can be used for retrieval.
- Nominalizations are indexed. Thus, if the text contains *erfindung.1.1* (invention), then *erfinden.1.1* (invent) is also added to the index, and vice versa.
- Compound decompositions are indexed – *verteidigungsminister.1.1* (minister of defence) results in *minister.1.1* to be indexed as well.
- Adjective-attribute relationships are expanded. If the text contains *hoch.1.1* (high), then *höhe.1.1* (height) is indexed as well.

Moreover all answer types contained in a sentence are indexed. For answering definition questions, 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 processing requires parsed sentences. The system was configured to retrieve 100 supporting snippets per question.

Shallow Feature Extraction and Reranking In order to save processing time, LogAnswer normally restricts logical processing to a small number of most promising passages. To this end, the passages are reranked using shallow features that can be computed quickly without 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 $defLevel = 2$ indicates that the snippet contains a defining verb or apposition, and $defLevel = 1$ indicates a relative clause. Finally, the *irScore* feature 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 [3, 4]. It was implemented using the Weka toolbench [5]. The training data consisted of 17,350 annotated snippets retrieved in a run of LogAnswer on the QA@CLEF 2007 questions.

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

Logical Query Construction The parse of 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):

$$\text{modp}(X_1, FOCUS, \textit{hoch.1.1}), \text{sub}(X_2, \textit{berg.1.1}), \text{prop}(X_2, X_1), \text{attr}(X_2, X_3),$$
$$\text{prop}(X_2, \textit{chilenisch.1.1}), \text{val}(X_3, \textit{la\_silla.0}), \text{sub}(X_3, \textit{name.1.1}).$$

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).

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 (currently expressed in Horn logic).[6] 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. The resulting skip count indicates (non-)entailment [6, 4]. 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 depends on factors like internal literal order of the prover which are arbitrary to some degree. Combining relaxation results of different provers can alleviate this problem. 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.[7] 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 [7].
- E-KRHyper [8] 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* [9], which integrates a superposition-based handling of equality into the hyper tableau calculus [10]. 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 operates on the axioms and the negated query literals, with a refutation result indicating a successful answer and providing the binding for the queried

[6] 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) [6].

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

variable. If the reasoning is interrupted due to exceeding the time limit, then partial results can be retrieved for guiding the relaxation process [12].

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 Based on the results of the relaxation proof and on the extracted answer, LogAnswer 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. $1 - skippedLitsUb/allLits$); *litRatioUb* (relative proportion of potentially provable literals vs. all query literals, i.e. $1 - skippedLitsLb/allLits$); *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 $focusDefLevel = 2$ indicates that the answer binding found by the prover corresponds to an apposition, and $focusDefLevel = 1$ 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 shallow feature-based ranking, and the logical processing is stopped after a pre-defined time limit. For QA@CLEF, no time limit was imposed, so every retrieved passage was subjected to 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. The compatibility of expected and found answer type is treated by answer-type related features passed to the machine learning method.

Table 1. Results of LogAnswer in QA@CLEF 2008

Run	#Right	#Unsupported	#Inexact	#Wrong	Accuracy	CWS	MRR
loga081dede	29	1	11	159	0.145	0.032	0.194
loga082dede	27	1	9	163	0.135	0.029	0.179

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 [3]. The $k = 3$ distinct answers with the highest aggregated scores were chosen for the QA@CLEF runs. For each answer, the supporting passage with the highest score was selected as a justification.

3 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 the 'OPT' combination of the MultiNet prover and the E-KRHyper prover described in [4]. The motivation for using more than one prover is that following several relaxation paths by applying multiple provers might increase the chance of discovering a good provable query fragment. While the combination of the provers worked well in earlier experiments [4], results in the QA@CLEF 2008 task were slightly worse for the combined method compared to the first run which used only one prover.

4 Error Analysis and Discussion

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. This means that, assuming perfect answer extraction and validation, the system can theoretically answer 119 non-NIL questions correctly. In order to extend this limit, the retrieval stage should be optimized. The following improvements are likely the most urgent:

– The retrieval module was configured to return only 100 sentences per question. Increasing this number will improve recall but incur more processing effort. A good trade-off for these factors should be assessed experimentally.

– The restriction to single-sentence snippets must be dropped. The analysis of the texts should be improved by resolving coreference. Deictic temporal expressions (like *'yesterday'*) should be resolved based on the document date.
– The system only indexes sentences with a full 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 such answers should be added.

Another significant source of errors is answer extraction: LogAnswer found 26 correct non-NIL answers in the *loga081dede* run. However, 46 of the supporting snippets for the top-1 answers actually contain a correct answer. Thus the success rate of answer extraction for sentences at top-1 position is 56.5%. For the top-3 results, the achieved MRR for 120 questions with multiple answers was 0.1944, compared to 0.3222 for perfect extraction. These problems of answer extraction are due to 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 – 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 sentences like *'X is Y'* or *'X means Y'*, 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 shown by the relatively large number of 11 inexact answers of LogAnswer in the *loga081dede* run.

The chosen ML technique was also not very effective, which became clear when experimenting with refinements. While retaining bagging of decision trees as the basic method for probability estimation, the present version of LogAnswer no longer needs reweighting of training items.[8] This was made possible by changing the splitting criterion for decision tree induction in such a way that in each induction step, the selected split maximizes a generalized MRR metric over the training items, compared to all other nodes that currently await splitting:

$$\mathrm{k^*MRR} = \frac{1}{Q}\sum_{q=1}^{Q}\sum_{i=1}^{k_q^*}\frac{w_{i,k_q^*}}{\mathrm{rank}_{q,i}-i+1} \quad \text{where} \quad w_{i,k_q^*} = \frac{2(k_q^*-i+1)}{k_q^*\,(k_q^*+1)},$$

$k_q^* = \min\{k, \mathrm{yes}_q\}$, yes_q is the number of correct results for question q, $\mathrm{rank}_{q,i}$ is the rank of the ith correct result for question q, Q is the number of questions, and k is the window size ($k = 3$ in the current system). The use of k*MRR-maximizing splits has a strong positive effect since the criterion is sensitive to the grouping of training items by question. Another refinement was permitting only those splits which fulfill *monotonicity constraints* on the estimated probabilities that can now be specified for each feature. For example, it makes sense to require

[8] Reweighting was necessary due to the low number of positives in the training set [7].

that (everything else being equal), the estimated correctness probability for a support passage must increase with the number of matched lexical concepts of the query. With these improvements, LogAnswer now finds 39 exact non-NIL answers, which means a 50% gain compared to the *loga081dede* run.

5 Conclusion

With LogAnswer, we have developed a prototype of a logic-based QA system suitable for real-time question answering. While the system works well when used for finding answer sentences [4, 7], the naive solution for extracting exact answers that was added for QA@CLEF 2008 had problems with constructions involving appositions, copula, and defining verbs. 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 vast number of answer candidates from which the few correct ones must be selected by extensive validation. An intrinsic problem of logic-based answer extraction is the restriction to snippets with a full parse. Therefore the logic-based extraction should be complemented with a fallback extraction technique which covers sentences with a failed parse as well.

References

1. Hartrumpf, S.: Hybrid Disambiguation in Natural Language Analysis. Der Andere Verlag, Osnabrück, Germany (2003)
2. Helbig, H.: Knowledge Representation and the Semantics of Natural Language. Springer (2006)
3. Glöckner, I.: University of Hagen at QA@CLEF 2008: Answer validation exercise. In: Working notes for the CLEF 2008 workshop, Århus, Denmark (2008)
4. Glöckner, I., Pelzer, B.: Exploring robustness enhancements for logic-based passage filtering. In: Knowledge Based Intelligent Information and Engineering Systems (Proc. of KES 2008, Part I). LNAI 5117, Springer (2008) 606–614
5. Witten, I.H., Frank, E.: Data Mining. Practical Machine Learning Tools and Techniques. Morgan Kaufmann (2005)
6. Glöckner, I.: University of Hagen at QA@CLEF 2007: Answer validation exercise. In: Working Notes for the CLEF 2007 Workshop, Budapest (2007)
7. Glöckner, I.: Towards logic-based question answering under time constraints. In: Proc. of ICAIA-08, Hong Kong (2008) 13–18
8. Pelzer, B., Wernhard, C.: System Description: E-KRHyper. In: Automated Deduction - CADE-21, Proceedings. (2007) 508–513
9. Baumgartner, P., Furbach, U., Pelzer, B.: Hyper Tableaux with Equality. In: Automated Deduction - CADE-21, Proceedings. (2007)
10. Baumgartner, P., Furbach, U., Niemelä, I.: Hyper Tableaux. In: JELIA'96, Proceedings. (1996) 1–17
11. Sutcliffe, G., Suttner, C.: The TPTP Problem Library: CNF Release v1.2.1. Journal of Automated Reasoning **21**(2) (1998) 177–203
12. Pelzer, B., Glöckner, I.: Combining theorem proving with natural language processing. In: Proc. of the First Int. Workshop on Practical Aspects of Automated Reasoning (PAAR-2008), CEUR Workshop Proceedings (2008) 71–80