Towards Logic-Based Question Answering under Time Constraints

Ingo Glöckner*

Abstract—**Using logic for question answering (QA) promises more accurate answers and increased flexibility of querying. However, the computational effort for a deep linguistic analysis of the involved natural language (NL) expressions and the subsequent logical inferencing makes it difficult to leverage this power for real-time QA. The paper attributes this difficulty in part to the popular answer validation paradigm, and it proposes a different solution which avoids parsing of answers at query time. The novel approach, which uses logic for assessing the relevance of extracted passages, allows better concurrency of QA subtasks since logic-based passage filtering can run in parallel with the answer extractors. Moreover the substitutions found in the proofs can guide a direct answer generation. A first implementation of logical passage filtering achieved 43% F-measure for factual questions of CLEF07, with response times of only a few seconds. These results are especially interesting since neither aggregation (for better filtering) nor parallelization (for reducing latency) were used in the prototype yet.**

Keywords: real-time question answering, logic

1 Introduction

Question answering systems which extract precise answers to natural language questions from a given document collection have made rapid progress in recent years. The PowerAnswer system [6], for example, answers 60% of the factual questions in the TREC 2006 evaluation. However, most QA systems still rely on shallow linguistic processing and simplistic meaning representations (e.g. bag-of-words models) which work reliably only when the queried information is redundantly expressed in the document collection (and with the same words). A logic-based approch, by contrast, can exploit the available information more fully due to improved content analysis (by deep linguistic processing), powerful reasoning methods beyond keyword matching, and modeling of background knowledge. In practice, logic is typically used for building components for *answer validation* which check the correctness of extracted answers as part of the final answer selection. PowerAnswer is an example of a QA system which implements this approach. A comparison of answer validators in the answer validation exercise (AVE) 2006 [7] found that systems reported to use logic generally outperformed those without logical reasoning.

*Intelligent Information and Communication Systems Group (IICS), University of Hagen, Germany. Email: iglockner@web.de.
Funded by DFG under contract HE 2847/10-1 (LogAnswer).

The starting point for this paper is the MAVE answer validator [1], which achieved a 93% correct answer selection rate in the CLEF07 AVE task. However, this excellent result was only achieved at the cost of tremendous computational effort and response time of a dozen minutes per question. This did not matter for the AVE07 evaluation, but in the real world, no one would like to wait that long for a QA system to answer a question. Clearly the problem cannot be solved by tuning individual system components only. This brings us to the central question of this paper: *How can the (logic-based) QA process be organized in such a way that the correct answers are also found under time constraints?*

The standard approach for incorporating logic in a QA system, viz logical answer validation, is an obstacle in this respect since it requires the logical representation of the answers (or corresponding logical hypotheses) to be available as a precondition for running the logic-based entailment test. In a real-time QA setting, however, there is simply no time for parsing the vast number of answer candidates proposed by the answer extraction stage.

The paper thus favors a different approach to leveraging logic for QA: Rather than trying to decide if a given answer is logically justified from a supporting text passage, it focuses on the use of logic for *passage filtering*, i.e. deciding if a retrieved text passage contains a correct answer to the question (or equivalently, if the question can be answered from the considered passage). The main benefit of logical passage filtering over validation is that there is no need for parsing answers – the task consists solely in proving the logical representation of the question from that of the supporting passage. There is no dependency on prior answer extraction, so that the logic-based passage validation can start directly after retrieval and run in parallel with extractors. But obviously, the results of passage filtering can be used to improve the relevance judgements for extracted answers candidates, since an answer extracted from a candidate passage can only be correct if the passage actually contains any answer to the question. As an alternative to using separate answer extractors, the results of the question-passage proofs (i.e. bindings for the queried variable) can also be used for the direct extraction or generation of corresponding answers.

The remainder of the paper is organized as follows. Section 2 describes a system prototype built for demonstrating the feasibility of this approach. Section 3 explains

how the prover was optimized for finding most proofs in a 10–20 milliseconds range. Section 4 presents an evaluation of the approach based on factual questions of CLEF 2007. Finally, Section 5 summarizes the lessons learnt and identifies issues for future research.

2 System description

This section describes the experimental system which was implemented for testing the feasibility of logical passage filtering for QA. The goal was that of getting some realistic data on the actual processing times per question and on the quality of results that can be achieved by the proposed approach. The system comprises the following main processing stages.

2.1 Question analysis

WOCADI [2], a robust parser for German, is used for a deep linguistic analysis of the given question. The syntactic-semantic analysis results in a semantic representation expressed in the MultiNet formalism [3], a variant of semantic networks specifically suited for natural-language processing. The parser handles intrasentential coreference resolution and provides the necessary information for the subsequent treatment of intersentential anaphora. The question analysis module, which is part of the IRSAW QA framework [4], further extracts query terms, the expected answer type, and other information of interest to subsequent processing stages.

2.2 Passage retrieval

The IRSAW framework also provided the retrieval module used for finding candidate passages based on a regular (bag-of-words) IR approach, assuming a segmentation into single-sentence passages for the moment. Using the WOCADI parser, all documents are linguistically analyzed prior to indexing. In particular, there is no need for parsing any documents at query time since the precomputed logical representation of each passage can be retrieved from the database.

2.3 Query construction

The next step consists in translating the semantic network representation of the question into the actual logical query to be proved from the passage representations. To this end, the analysis of the question is turned into a conjunctive list of query literals. The query construction step also involves a synonym normalization which replaces all lexical concepts with canonical synset representatives. This process is based on 48,991 synsets (synonym sets) for 111,436 lexical constants. For example, the question *Wie hieß der Sänger von Nirvana?*[1] translates into the logical query:

$$\begin{aligned}&\mathrm{val}(X_1, \textit{nirvana.0}), \mathrm{sub}(X_1, \textit{name.1.1}), \mathrm{attr}(X_2, X_1),\\ &\mathrm{attch}(X_2, X_3), \mathrm{sub}(X_3, \textit{gesangsolist.1.1}),\\ &\mathrm{subs}(X_4, \textit{heißen.1.1}), \mathrm{arg1}(X_4, X_3), \mathrm{arg2}(X_4, \mathit{FOCUS})\end{aligned}$$

based on the lexical concepts (word senses) *nirvana.0* (Nirvana), *name.1.1* (name), *gesangssolist.1.1* (singer soloist), and *heißen.1.1* (be named). The example demonstrates the synonym normalization technique since the original lexical concept *sänger.1.1* (singer) is replaced with the canonical *gesangssolist.1.1* (singer soloist). By convention, the FOCUS variable represents the queried information (the name of the person in this case).

2.4 Robust entailment test

The basic idea of the logic-based passage filtering is that the passage contains a correct answer to the question if there is a proof of the question from the passage representation and the background knowledge.[2] However, in order to achieve more robustness against knowledge gaps and errors of semantic analysis, the prover is actually embedded in constraint relaxation loop. To this end, the literals of the query are sorted according to a least-effort heuristics, i.e. literals with least alternatives to be checked (highest likelihood of failure) come first. If a proof fails, the longest provable prefix of the ordered query is determined and the first non-provable literal is removed from the query. By repeating this process, literals are skipped until a proof of the remaining query succeeds. The number of skipped literals is then used as robust indicator of entailment strength. For example, consider the question for the lead singer of Nirvana, and the following supporting text passage (translated from German): *Fans and friends of the lead singer of US-American rock band Nirvana reacted with grief and dismay to the suicide of Kurt Cobain this weekend.* Here linguistic analysis fails to identify Kurt Cobain and the Nirvana lead singer, i.e. there are two distinct entities rather than a single one representing Kurt Cobain, the lead singer of Nirvana. Therefore a perfect proof fails, but the system finds a relaxation proof with two skipped literals $\mathrm{sub}(X_3, \textit{gesangsolist.1.1})$, $\mathrm{attch}(X_2, X_3)$, which binds the FOCUS variable to the value for Kurt Cobain.

The relaxation approach is computationally very expensive: a new proof is tried after each simplification of the query. When a query with n literals cannot be proved from the passage at all, this can waste $n \cdot t$ time where t is the time limit for each attempted proof. It therefore makes sense to restrict the maximal number of iterations of the relaxation loop. However, if the relaxation process is stopped before all query literals are either proved or skipped, then there is uncertainty as to the number of literals that would eventually be proved or skipped in a complete relaxation proof. Thus, one can only specify upper and lower bounds on the number of provable

[1] *What was the name of the singer of Nirvana?*

[2] The prototype uses the same sources of background knowledge as the MAVE system described in [1].

literals, assuming the extreme cases that the remaining literals are either all provable or all not provable.

2.5 Feature extraction

The classification of passages rests on the following logic-based features which depend on the chosen limit on the number of relaxation steps:

- *skippedLitsLb* Number of literals skipped in the relaxation proof
- *skippedLitsUb* Number of skipped literals, plus literals with unknown status (not in known provable prefix of literal list)
- *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 (not yet skipped) vs. all query literals, i.e. $1 - skippedLitsLb/allLits$
- *boundFocus* Signals that the queried variable was bound in the relaxation proof. To gain robustness, the partial proof of the longest proven prefix of the current query fragment is used for determining the binding when a proof fails in the last relaxation cycle
- *npFocus* Indicates that the queried variable was bound to a constant which corresponds to a nominal phrase (NP) in the text
- *phraseFocus* Signals that extraction of an answer for the binding of the queried variable was successful.

A simplistic solution for answer extraction given the binding of the queried variable was used for computing the *phraseFocus* feature. The method leverages information on word alignment, as provided by the parser for nominal phrases (NPs), in order to find answer strings for the bindings of the queried variable. This is done by cutting verbatim text from the original text passage. Thus, in the above relaxation example, the prover binds the $FOCUS$ variable to a constant, say *c43*. The parse provides the information that an NP which corresponds to *c43* can be found at character positions 321–331 in the text, and the exact answer *Kurt Cobain* is obtained by extracting the specified substring from the passage.

In addition to the logic-based features, three 'shallow' features are used which depend only on morpho-lexical analysis, i.e. these features do not require a deep parse and can be computed without the help of the prover:

- *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* Number of proper names mentioned in question, but not in the passage.

The matching technique used to obtain the values of these shallow features also takes into account lexical-semantic relations (e.g. synonyms and nominalizations), see [1].

2.6 Passage classification using techniques from Machine Learning

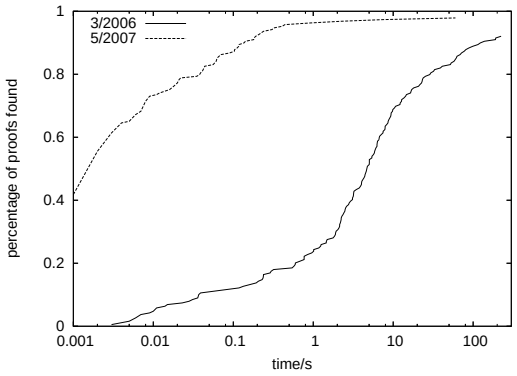
The Weka machine learning toolbench [8] was used for learning the mapping from features of retrieved passages to yes/no decisions concerning containment of a correct answer to the question in the considered passage. The low precision of the original passage retrieval step means a strong disbalance between positive and negative examples in the data sets.[3] In order to emphasize the results of interest (i.e. positive results) and achieve sufficient recall, *cost-sensitive learning* was applied. Thus, false positives were weighted by 0.25 while lost positives (i.e. false negatives) were given a full weight of 1. The Weka Bagging learner was chosen for learning the classifier, using default settings (averaging over results of 10 decision trees computed by the Weka REPTree decision tree learner). It was wrapped in a Weka CostSensitiveClassifier to implement the cost-sensitive learning.

3 Optimizing the prover for the task

Applying logic in a real-time QA setting only makes sense if a very fast theorem prover is available. Experiments with general-purpose provers revealed that a more specialized solution would be needed to approach the desired time frame of only a few milliseconds per proof. Therefore a dedicated prover for MultiNet representations (described in [5]) was used as the starting point for this work. The prover was then systematically optimized both for speed and scalability by utilizing term indexing, caching techniques, lazy computation (i.e. index strucures are only built on demand), by optimizing the heuristics used for determining literal order, and by identifying and removing performance bottlenecks with the help of profiling tools. This process was guided by a realistic benchmark for prover performance in a QA context, which consists of the MultiNet analysis of a 100 sentence text (the German Wikipedia article about aviation pioneer Amelia Earhart), 244 implicational rules to support flexible querying, and a set of 172 example questions. These changes achieved a speedup of prover performance by a factor of 400 within one year of optimizing against the benchmark. As shown by Fig. 1, 70% of the questions can now be proved in 10 ms or less.

[3] In the experiment described in the next section, only 2.3% of the retrieved passages contain a correct answer.

Figure 1: Effect of performance improvements of the prover in the Amelia benchmark



A second benchmark served to adjust the depth limit for iterative deepening, and the time limit for each attempted proof. It comprises 1,805 provable hypothesis fragments which occurred in relaxation proofs for the second AVE07 run of the MAVE validator (see [1]). These problems are better suited for fine-tuning of parameters than the Amelia benchmark since they involve passage-size snippets and exactly the same background knowledge as in the present experiment. In order to determine a suitable depth limit for iterative deepening, the distribution of actual proof depths for the AVE problems was computed, see Table 1. A maximum depth limit of $d = 2$ was chosen for the later experiments; this means only about 1.5% loss of possible proofs judging from the test data. Based on the actual proof times in the AVE benchmark, it was further decided to use a maximum time of 100 ms per proof for the present experiment, which corresponds to about 7% loss of possible proofs due to the time limit.

4 Evaluation

This section presents an experimental evaluation of the proposed model for logic-based passage filtering.

4.1 Test data for the filtering experiment

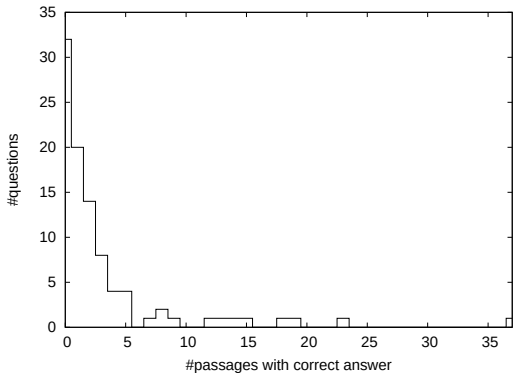
The questions of the CLEF07 QA track for German were considered a suitable starting point for this experiment, since they target at the document collections currently supported by the IRSAW IR module (CLEF News and Wikipedia).[4] From the full set of 200 questions, a further selection had to be made. First of all, the questions are organized into topic groups, and follow-up questions can contain anaphoric references to mentions in earlier questions of a group. These follow-up questions were eliminated since discourse processing is not of relevance here. Moreover all definition questions were deleted from the remaining list of 116 topic-start questions, since knowing the logical correctness of an answer is not sufficient for deciding if the answer is also suitable as a definition. The

[4]See `http://www.clef-campaign.org/2007.html`

d	count	p-avg	p-max	l-avg	l-max
0	512	1.9 ms	34 ms	0.34 ms	9.6 ms
1	1072	12.4 ms	761 ms	1.7 ms	69.2 ms
2	193	104.4 ms	1391 ms	25.3 ms	158 ms
3	27	230.0 ms	1615 ms	45.6 ms	202 ms
4	1	177.0 ms	177 ms	14.8 ms	14.8 ms

Table 1: Exact proof depths for AVE problems. Abbreviations: d (depth), p-avg (average time per problem), p-max (maximum time for a problem), l-avg (average time per literal), l-max (maximum time for proving a literal)

Figure 2: Number of questions with a given number of valid supporting passages



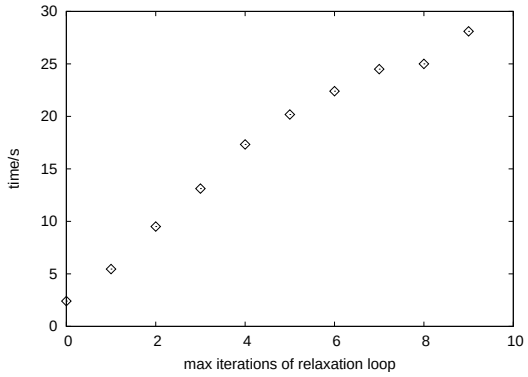
remaining 96 factual questions were checked for parsing quality and two questions[5] for which construction of a logical query failed were discarded. In the end, there were 94 questions left for the passage filtering experiment.

For each question, IRSAW retrieved up to 200 one-sentence snippets from the pre-analyzed corpora, resulting in a total of 18,700 candidate passages (199 per question). From the full set of candidate passages, all snippets with a missing parse or chunk parse were eliminated. These passages can be ignored since the logical filtering approach is only applicable to passages with a full syntactic-semantic analysis. The remaining 12,524 passages with a complete parse (133 per query) were annotated for containment of a correct answer to the question, starting from known CLEF07 annotations.

This annotation revealed very low accuracy of the original passage retrieval. In fact, only 291 of the parseable passages contain a correct answer (i.e. there are 3.1 valid passages per query on average). Figure 2 visualizes the number of questions with a given number of valid supporting passages in the test set. For 62 of the 94 questions, there is at least one correct supporting passage, i.e.

[5]qa07_012: *Welchen US-Präsidenten versuchte Francisco Duran zu töten?* and qa07_013: *Der Sohn welches ehemaligen US-Präsidenten wurde 1994 zum Gouverneur von Texas gewählt?"*

Figure 3: Relaxation steps vs. response time



n	tq/s	td/ms	prec	recall	f-meas
0	2.4	18.1	0.37	0.41	0.39
1	5.5	41.0	0.44	0.39	0.41
2	9.5	71.5	0.41	0.41	0.41
3	13.1	98.6	0.42	0.43	0.43
4	17.3	130.2	0.41	0.40	0.40
5	20.2	151.6	0.40	0.41	0.41
6	22.4	168.3	0.39	0.42	0.41
7	24.5	184.2	0.38	0.41	0.39
8	25.0	187.8	0.42	0.41	0.42
9	28.1	211.0	0.40	0.39	0.39
*	0.3	2.1	0.35	0.38	0.36

Table 2: Quality of passage filtering as a function of allowable relaxation steps n. Abbreviations: tq (time per question), td (time per document), prec (precision), f-meas (F-measure), * (shallow features only)

recall of the approach cannot exceed 0.66. The figure reveals that one question has an unusually high number of 37 correct supporting passages.[6] It was eliminated from the test set since it might otherwise dominate results.

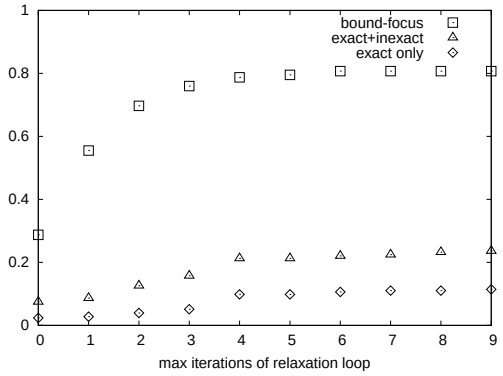
4.2 Experimental results

The experiment is supposed to provide some data on the effect of relaxation settings on processing times and the achieved quality of passage filtering.[7] As to observed response times, Fig. 3 reveals a sublinear dependency on the number of allowable relaxation steps, flattening out for larger relaxation limits which exceed the number of failed literals in most relaxation proofs. The average processing times for questions and documents are also listed in Table 2, along with the precision, recall, and F-measure for each choice of the relaxation parameter. These results were obtained by the cost-sensitive learning approach described in Section 2.6, using 10-fold cross validation.

[6] qa07_160: *Welcher Partei gehört Angela Merkel an?*

[7] All experiments were run on a Linux system using an Intel E6600 CPU at 2,4 GHz clock rate and 2 GB RAM.

Figure 4: Allowable relaxation steps vs. answer extraction results: 'bound-focus' (proportion of correct passages for which an answer binding is found), 'exact+inexact' (useful answers found by the simple extractor), 'exact' (perfect answers found by the extractor)



The table reveals a clear benefit of using logic: All results obtained by using logic-based features are substantially better than the baseline using only shallow features. The best results were obtained for a limit of $n = 3$ relaxation steps. Allowing additional relaxation steps ($n > 3$) did not further improve quality. In general, results were not very sensitive to the relaxation settings. This suggests using small values for n, especially in the scenario where answer extraction is done externally, so that the logic-based information is *only* needed for judging the relevance of passages. However, the proofs might also be of interest as a source of answer substitutions for subsequent NL generation (or direct answer extraction as in the prototype). In this case using more relaxation cycles is potentially useful, even if doing so will not improve precision/recall scores: Since the number of proved literals increases with the number of relaxation steps, more relaxation cycles mean a higher likelihood that the queried variable will eventually be bound, and thus provide a pointer to the actual answer. Fig. 4 illustrates this dependency between the number of relaxation steps and the percentage of correct passages for which the prover finds a binding of the queried variable. For exact proofs, such a binding (and thus potentially an answer) is found for only 28.7% of correct passages, for $n = 3$ relaxation steps the number increases to 76.0% of all passages containing an answer, and for $n = 4$ even to 78.7%.

The same tendency is visible in the success rates for logic-based answer extraction (also shown in Fig. 4) using the basic extraction method of Section 2.5. The results refer to perfect extracted answers (e.g. *Bill Gates* for *Who is the chairman of Microsoft?*) and essentially correct but inconcise extractions (*Bill Gates, the chairman of Microsoft*), compared to the number of all passages containing an answer. Most of the useful but imperfect answers are complex NPs which contain the requested information as a smaller NP but also some extra information.

n	0	1	2	3	4	5	6	7	8	9
a)	2.4	5.5	9.5	13.1	17.3	20.2	22.4	24.5	25.0	28.1
b)	1.7	3.7	5.4	6.6	7.4	8.0	8.4	8.8	9.0	9.0
c)	0.0	1.2	2.8	2.8	2.4	2.4	2.0	2.4	1.6	2.8

Table 3: Effect of shallow pre-filter for n relaxation steps: a) time per question in seconds (no pre-filter); b) response time with pre-filter; c) percent points loss in recall

The speed and surprisingly good quality of the baseline classifier which uses only shallow features suggests that it might be useful for a pre-filtering of passages prior to application of the deep, logic-based passage filtering. To test this, the shallow features were used for training a pre-filter classifier, again using bagging of decision trees but setting the cost for false positives to 0.01 in order to obtain a highly recall-preserving classifier. The resulting shallow classifier achieves a recall of 85% while eliminating 7,333 of the 12,377 annotated passages. Since only 41% of the original passages pass the shallow pre-test, cascading the fast pre-filter and the slower but more accurate logic-based filter promises a considerable cut in response times. The effect of combining the shallow pre-filter with subsequent logic-based classifier is shown in Table 3. Applying the pre-filter is indeed very effective, and the observed processing times drop to a 1.7–9.0 seconds range (depending on relaxation settings). Compared to using the logic-based filter alone, cascading the shallow and deep filters slightly reduced recall, but only by about 2 percent points. This indicates a strong overlap between the lost positives of the logic-based approach and the pre-test based on shallow features. Precision is hardly affected by adding the pre-filter, with a slight decrease in the order of 1 percent point.

5 Conclusions and future work

The paper has argued that utilizing logic for QA needs not contradict the requirement of answering questions in real time. The proposed approach based on logical passage filtering is superior to logical answer validation in this respect: It does not require any answers to be parsed and can even run in parallel with answer extraction. The paper has presented a prototypical implementation of this approach and an evaluation based on factual questions from CLEF07 in which the method for passage filtering achieved up to 43% F-measure.[8] The experiment revealed a clear benefit of using logic. Relaxation had no strong effect on filtering quality but proved to be important for finding answer bindings. In the future, other relaxation techniques and query decomposition will also be tested. Moreover the retrieval stage will be reconsidered in order to learn about the effect of passage size (sentences, paragraphs, whole documents) on retrieval quality.

[8] Result for $n = 3$ relaxation cycles. On average the system then accepts 2.7 passages per question, of which 1.2 are actually correct.

The prototype uses logical answer extraction in order to determine the actual answer strings from computed bindings of the queried variable. However, the alternative of using external answer extractors (which might use pattern matching, expected answer types, etc.) is also worth trying, and additional experiments are needed to find out which approach works best in practice.

Average response times of the prototype are currently in the range of 1.7–9.0 seconds (depending on relaxation settings), which is already close to the intended time frame of perhaps 1 second or less. Distributing the prover workload by parallelization and improvement of the quick pre-filter are obvious next steps to further reduce average latency. However it is also important to fit response times of logic-based filtering into a defined time slot with a *guaranteed maximum latency*. A solution consists in first sorting the passages according to the probability of relevance determined from the shallow features only, and subsequently applying the slower, logic-based classifier in order of decreasing estimated relevance. The best results considered so far can then be fetched at any time, in particular when processing exceeds the time limit.

References

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

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

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

[4] Johannes Leveling. IRSAW – towards semantic annotation of documents for question answering. In *CNI Spring 2007 Task Force Meeting*. CNI, Phoenix, Arizona, April 2007.

[5] Steffen Marthen. Untersuchungen zur Assimilation größerer Wissensbestände aus textueller Information. Master's thesis, FernUniversität in Hagen, Hagen, Germany, 2002.

[6] Dan Moldovan, Mitchell Bowden, and Marta Tatu. A temporally-enhanced PowerAnswer in TREC 2006. In *Proc. of the 15th Text REtrieval Conference (TREC 2006)*, Gaithersburg, MD, 2006.

[7] Anselmo Peñas, Álvaro Rodrigo, Valentin Sama, and Felisa Verdejo. Overview of the answer validation exercise 2006. In *Working Notes for the CLEF 2006 Workshop*, Alicante, Spain, 2006.

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