A Natural Language Question Answering System as a Participant in Human Q&A Portals

Tiansi Dong
University of Hagen, Germany
tiansi.dong@fernuni-hagen.de

Ulrich Furbach
University Koblenz-Landau, Germany
uli@uni-koblenz.de

Ingo Glöckner
University of Hagen, Germany
ingo.gloeckner@fernuni-hagen.de

Björn Pelzer
University Koblenz-Landau, Germany
bpelzer@uni-koblenz.de

Abstract

LogAnswer is a question answering (QA) system for the German language, aimed at providing concise and correct answers to arbitrary questions. For this purpose LogAnswer is designed as an embedded artificial intelligence system which integrates methods from several fields of AI, namely natural language processing, machine learning, knowledge representation and automated theorem proving. We intend to employ LogAnswer as a virtual user within Internet-based QA forums, where it must be able to identify the questions that it cannot answer correctly, a task that normally receives little attention in QA research compared to the actual answer derivation. The paper presents a machine learning solution to the wrong answer avoidance (WAA) problem, applying a meta classifier to the output of simple term-based classifiers and a rich set of other WAA features. Experiments with a large set of real-world questions from a QA forum show that the proposed method significantly improves the WAA characteristics of our system.

1 Introduction

A question answering (QA) system aims at automatically finding concise answers to arbitrary questions phrased in natural language. It delivers only the requested information, unlike search engines which refer to full documents. For example, given the question *"What was the name of the first German Chancellor?"*[1], ideally a QA system would respond with *"Konrad Adenauer"*. This usage is intuitive, it saves time and allows a satisfactory result presentation even on compact mobile devices. Recently QA has been drawing attention: *True Knowledge* [Tunstall-Pedoe, 2010] is an English language web-based system, and IBM's *Watson* [Ferrucci *et al.*, 2010] has successfully participated in a quiz show.

LogAnswer [Furbach *et al.*, 2010; Glöckner and Pelzer, 2009] is a web-based QA system for the German language. It works with a knowledge base (KB) derived from the entire German Wikipedia, and the answers are produced using a synergistic combination of natural language processing (NLP), machine learning (ML) algorithms and automated theorem proving (ATP). LogAnswer and its results have until now been published mainly in two areas: In linguistics and in particular in the context of the CLEF competition [Glöckner and Pelzer, 2009; 2010], where LogAnswer is a regular participant. In the ResPubliQA track of CLEF 2010, LogAnswer gave correct answers to 52.5% of the questions, the second best performance among non-English QA systems. In the field of automated reasoning, the adaptation of the Tableau-system E-KRHyper as an integral part of LogAnswer has attracted interest [Furbach *et al.*, 2010]. The ATP community seems to admit that embedding a prover where it has to use millions of axioms raises important questions: over 200 LogAnswer reasoning problems have become part of the well-recognized benchmark-suite TPTP [Sutcliffe, 2010].

In order to expand the usage of LogAnswer and to achieve a better real-world evaluation, we are in the process of adapting the system to QA forums. Such Internet forums provide their visitors with a venue for asking and answering each others' questions; examples are *Frag Wikia!*, *COSMiQ*, *WikiAnswers* and the commercial system *JustAnswer*.[2] QA forums in the context of automated QA have been considered before in [Surdeanu *et al.*, 2008], but unlike LogAnswer this system does not derive answers, instead it relies on a stock of archived answers already phrased by forum users.

For our experiments we have chosen the German language QA forum *Frag Wikia!* due to its permissive Creative Commons license. A forum integration not only serves our research, instead it has mutual benefits. Forum visitors who ask questions can receive an automatically derived answer within a few seconds, while the expert users, who typically provide the answers, can be relieved of the task of handling mundane or repeat questions. However, there is a risk of QA inundating forums with incorrect answers. Real-world forum questions are considerably more difficult to handle than competition questions; in an initial test LogAnswer only succeeded at 8.9% of the questions, even when using less stringent conditions than in CLEF. Therefore it is important that the QA sys-

[1] All examples have been translated into English for this paper.

[2] frag.wikia.com, cosmiq.de, de.answers.com, justanswer.de

tem recognizes which questions it cannot answer correctly. The QA community is aware of this problem: The Answer Validation Exercise [Peñas *et al.*, 2006] was part of CLEF from 2006 to 2008, and as of 2009 the ResPubliQA track has been measuring the c@1 score which rewards avoidance of poor answers, but the results show that this strategy is hardly used [Peñas *et al.*, 2009].

For our intended forum integration we have thus put emphasis on detecting unsuitable answers. We will report the results of our first experiments and in particular we will discuss how ML techniques are used to improve the correctness of our results. The paper is divided as follows: Section 2 gives a description of LogAnswer. Section 3 introduces the research question of Wrong Answer Avoidance (WAA). Section 4 presents the ML approach to the WAA problem. Section 5 summarizes our results and outlines some future work.

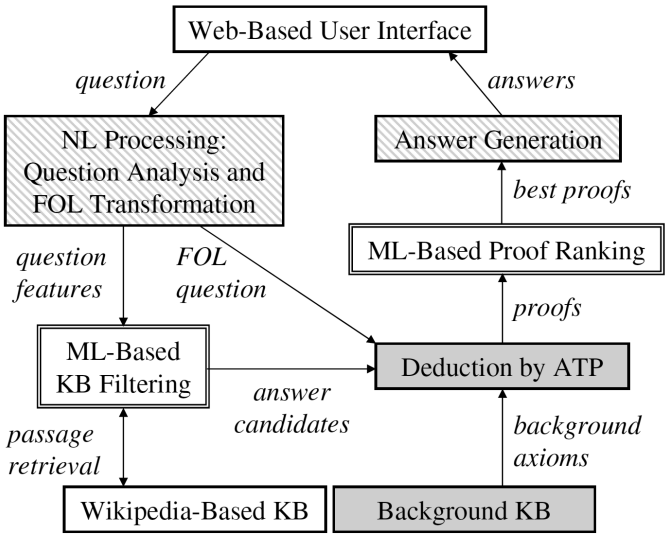

Figure 1: The LogAnswer processing cycle for a question

2 The LogAnswer System

LogAnswer is designed as a German language QA system on the web (`www.loganswer.de`). The user enters a question into the interface, and LogAnswer then presents a list of answers, highlighted in the relevant textual sources to provide a context. The answers are derived from an extensive knowledge base, which has been obtained by translating a snapshot of the entire German Wikipedia into a semantic network representation in the MultiNet (**Multi**layered Extended Semantic **Net**works) formalism [Helbig, 2006]. 29.1 million sentences have been parsed, together with an additional 12,000 logical rules and facts they form the general background knowledge. To make the semantic networks accessible to modern theorem provers, the MultiNet knowledge base is translated into First-Order Logic (FOL) as described in [Furbach *et al.*, 2010]. The gray boxes in Figure 1 comprise the reasoning part of the system: the background KB containing general world knowledge and the automated Hypertableaux [Baumgartner *et al.*, 1996] theorem prover E-KRHyper [Pelzer and Wernhard, 2007], which has been shown to be very suitable for the type of reasoning problems occurring in QA, characterized by their large number of often irrelevant axioms.

As the complete knowledge base is too large to be handled by a theorem prover, the initial processing steps use the question to identify a relevant KB fragment. The question is analyzed by linguistic methods and then translated into a MultiNet and FOL representation. The striped boxes depict the linguistic parts of the system in the figure. Our example question from the introduction would be translated into the logical representation $\exists x \exists y(\mathit{pmod}(y, \mathit{first}, \mathit{chancellor}) \wedge \mathit{prop}(x, \mathit{german}) \wedge \mathit{sub}(x, y)$.[3] The Wikipedia contents are matched against the given query, combining retrieval and shallow linguistic methods. They compute lists of features, like the number of lexemes matching between passage and question or the occurrence of proper names in the passage. A ML-based ranking technique uses these features to filter out the most promising text passages, resulting in up to 200 text passages which might be relevant for the query. The ML-based components utilize decision tree learning, they are depicted by double-framed boxes in Figure 1. The features can be computed even for text passages which cannot be parsed completely, for example due to syntactic errors. This makes the system more robust and allows the extraction of answers even from flawed textual sources. As the ranking must compare parsed and unparsed passages, it has been trained using a coupled learning approach which handles both types simultaneously [Furbach *et al.*, 2010].

One passage retrieved for our example question is *"The following schools are named after Konrad Adenauer, the first German Chancellor"*, as it shares several words with the question and it contains a personal name.

The FOL representations (*answer candidates*) of these text passages comprise about 230 ground unit facts each, and they are of course already precomputed and contained in the knowledge base. The candidates are now individually tested by E-KRHyper, each in conjunction with the background KB and the logical query representation. A successful proof instantiates variables in the FOL question with ground terms representing the answer. Query relaxation techniques enable the system to overcome gaps in the knowledge base and increase the likelihood of finding a proof in short time, at the cost of lowering the probability that the answer is relevant for the query. Therefore ML has been used to augment the ATP: All proofs are ranked by a classifier which has been trained on the reliability of relaxed proofs. It is possible that not all 200 candidates can get tested within the 5 seconds usually allowed by the application scenario. Therefore the classifier has also been trained to handle untested candidates, using their initially computed shallow features. Effectively this method is an anytime algorithm whose results improve in quality as the processing time allotment increases [Hartrumpf *et al.*, 2009]. Finally the highest ranked proofs or candidates are translated back into NL answers that are displayed to the user. For our example question the desired answer *"Konrad Adenauer"* is extracted from the passage above and presented in the results.

To summarize, LogAnswer utilizes a synergy of several AI

[3] The identifiers have been simplified to improve legibility.

techniques, harnessing NLP, knowledge representation and ATP to manage different aspects of QA, while ML links the subsystems and alleviates their individual weaknesses.

3 LogAnswer in a QA Portal

Our interest is to let LogAnswer act as a virtual user in a real-world question answering portal, so we have tested the QA system using data from the *Frag Wikia!* forum. We used 3,996 unanswered questions[4] from the portal for this evaluation. Note that the unanswered questions comprise unclear and very specialized questions accumulated over time that no other user felt able to answer. If LogAnswer, equipped with its Wikipedia knowledge base, is able to answer a substantial portion of these questions, then it could become a valuable participant in the forum. For each question, the validation and answer selection methodology of LogAnswer (cf. Figure 1) was applied and the 4 top-ranked answer sentences were determined for each question. We then annotated the resulting 15,737 answer sentences for correctness; a sentence was judged correct if it contains an answer to the question and also justifies the answer. In the following, we call a question answered if a correct answer sentence occurs in the answer list, i.e. on the top 4 ranks. LogAnswer has answered 8.9% of the questions according to this criterion.[5]

An example for one of these correctly answered questions is *"Who said 'Give me the power to issue money in a country and I care not who makes the laws'?"*, where LogAnswer responds that it was a banker from the Rothschild banking house (whereas *Frag Wikia!* now contains an answer from a human user which we do not consider correct).

As no single *Frag Wikia!* user is even close to having answered 8.9% of all posted questions, the system could become a most valuable forum participant, as long as it does not bother the forum users with too many wrong responses for the remaining 91.1% questions. This introduces the research question of Wrong Answer Avoidance (WAA): How can we prevent a QA system from posting wrong results for questions that it cannot answer?

4 An Approach to Wrong Answer Avoidance

There are several ways to achieve WAA. For LogAnswer, one option is retraining the answer selection model for *Frag Wikia!* specific questions and establishing a suitable rejection threshold. However, since the answer ranking model of LogAnswer was trained on 58,803 annotated candidate sentences, the effort for establishing a similar training set for *Frag Wikia!* seems prohibitive. Moreover, the focus on WAA and specifics of the QA portals use case suggest the utilization of criteria for detecting problematic questions (like user profiles of forum participants) that should not be part of an answer selection model. For example, if the question is *"Where is the best place to study IT?"*, LogAnswer says *"Dynamic IT"*. Obviously this answer shall be prevented. This can be achieved by taking into account certain features of the question. In this case it contains a superlative and also asks about an opinion (facts in Wikipedia should not contain opinions). If a user is only interested in opinions, it might even be best to block all questions of the user. In order to capture such criteria, we implement wrong answer rejection by a dedicated WAA filter that follows the answer selection stage.

4.1 Baselines for Wrong Answer Avoidance

Filtering User-Assigned Categories *Frag Wikia!* users can assign categories to their questions. Independently of the actual LogAnswer results we blocked 295 categories that are outside the scope of Wikipedia and thus impossible for LogAnswer to handle, for example specialized gaming and help topics. The manual selection is useful both as a filtering baseline and as a knowledge source for training. As a standalone filtering method it is very weak since 66% of the sample questions are uncategorized and admit no such filtering.

Fisher Classifier Baselines In order to achieve WAA for uncategorized questions one must consider their actual content. Fisher's method is a popular choice for spam filters, so we adopt it here as a baseline. Let $F = \{f_1, \ldots, f_k\}$ be a set of features in the question, e.g. the set of question words. Further let $P(y|f_i)$ express the conditional probability that the computed response for a question with the feature f_i is correct. Assuming conditional independence, the test statistic $X^2 = -2\sum_{i=1}^{k} \ln P(y|f_i)$ is χ^2 distributed with $2k$ degrees of freedom, which allows computation of the corresponding p-value (the output of the classifier). We tried several feature sets, T: tokens/word forms transformed to lowercase, L: lemmas (base forms) of words, A: categories as analyzed atoms, S: categories split into words, and combinations TA, LA, TS, LS. We estimate the conditional probabilities from Y_i (number of questions with correct responses with feature f_i) and N_i (number of wrongly answered questions with feature f_i) using Lidstone smoothing, i.e. $P(y|f_i) = (Y_i + w\beta)/(Y_i + N_i + w)$. We tried two choices of parameters, O: add-one smoothing ($w = 2$, $\beta = \frac{1}{2}$), and W: using $w = 1$ and the overall probability of the YES class for the prior, i.e. $\beta = 0.089$. The resulting baseline models are called TAO, TAW, TSO, TSW etc.

4.2 Proposed Solution

The Fisher classifier baselines have several trade-offs. For example, the lemma-based L variants promise a higher rejection rate than the T variants, but the filter might become too unspecific. We thus propose a superordinate filtering model that bases its decisions on the results of all baseline filters. This approach can incorporate a large number of additional criteria for recognizing problematic questions and wrong responses, beyond the simple baseline classifiers.

The model can be determined using any supervised learning technique which can cope with the approx. 200 data at-

[4] All data sets and annotations mentioned in this section are publicly available from `http://www.loganswer.de/resources/loganswer_ijcai2011.tgz`. We use a *Frag Wikia!* dump as of 2010-01-09. Ignoring automatically generated questions marked as 'imported' as well as redirects and other unwanted cases, the question set comprises 17,462 regular questions.

[5] Restricting attention to Wikipedia-oriented questions that LogAnswer has a chance of answering, the answer rate is 21.5%.

tributes and allows redundancy and dependencies in the attribute set. We have chosen established learning techniques that meet these requirements, such as logistic regression. We also try a reweighting of training examples to enhance the few positive items (answered questions) in the training set.

When introducing a detection criterion it is often clear in advance whether it provides positive or negative evidence for the filtering decision. For example, a baseline model judging a response wrong should strengthen the evidence that the response is indeed wrong. Such qualitative knowledge about the effect of attributes on the outcome cannot be incorporated by standard learning techniques. Therefore we also consider rank-optimizing decision trees, a method for inducing decision trees that optimizes a ranking metric on the data and that supports this kind of apriori information [Glöckner, 2009].

4.3 A Feature Set for Wrong Answer Avoidance

We developed over 200 features for WAA. As these cannot be described in detail here, we provide an overview to motivate the different classes of features, beginning with more general features not specific to the QA forum use case.

Answer Selection Features: The ML-computed quality scores of the 4 top-ranked answer sentences (see Sect. 2) serve as features for the WAA filter.

Features Derived from the Question Type: LogAnswer uses a question classification to answer questions of different types appropriately, for example questions asking for definitions or questions asking for personal names. Around 100 binary features identifying the individual question types and abstractions thereof were added for the WAA filtering.

Features Capturing Linguistic Characteristics: Linguistic defects in a question are also a hint that an attempt to answer the question automatically may fail. We have annotated the 3,996 questions in order to assess the impact of specific defects. As shown in Table 1, 2,027 questions (50.7%) contain problematic constructions, with lower answer rates for all problem classes. Features for unknown words, out-of-vocabulary words with respect to Wikipedia, and missing capitalization were added to identify such defects.

We continue with features tailored to QA forum questions.

Age of Question: Easy questions are answered quickly by forum users, while difficult or unclear ones may never be answered. Thus, the age of an unanswered question is included as one of the features related to question difficulty.

Blocked Category Features: The manually compiled list of categories not covered by Wikipedia (see Sect. 4.1) is utilized in the form of a feature expressing whether a question belongs to a blocked topic. An auxiliary classifier has been trained on the user-categorized questions; based only on the question words it identifies the affiliation with unsuitable topics even for questions with no user-assigned category.

Baseline Classifier Features: The confidence in the YES decision determined by each of the baseline classifiers (see Sect. 4.1) is also used as a feature, turning the learned model into a meta classifier. When using cross validation, the baseline classifier features are determined only from questions in the current training folds. Since fitting of the baseline classifiers to the question being trained would spoil the training data, the feature values for each question are generated from baseline feature statistics without the question's own data.

Problem (P)	$\frac{\lvert P \cap A \rvert}{\lvert P \rvert}$	$\frac{\lvert P^c \cap A \rvert}{\lvert P^c \rvert}$	$\lvert P \rvert$	p-value
any problem	0.065	0.114	2,027	4.81e-8
sp	0.025	0.098	514	5.81e-8
quote	0.007	0.092	150	3.11e-4
cap	0.069	0.100	1,430	7.56e-4
coll	0.019	0.092	155	1.93e-3
grammar	0.047	0.092	277	0.011
punct	0.035	0.091	171	0.012
hyph	0.031	0.091	130	0.018
multisent	0.020	0.090	49	0.059†
ref	0.034	0.090	59	0.135
blank	0.055	0.090	127	0.175
markup	0.000	0.089	15	0.247†
noq	0.066	0.090	152	0.309
link	0.040	0.089	25	0.335†
lang	0.077	0.089	13	0.676†
(all questions)	0.089		3,996	

Table 1: Effect of linguistic problems on results, sorted by the p-value of the χ^2 test (or Fisher's exact test for small sample sizes, marked by '†'). P is the set of questions in the problem class, A: set of answered questions at rank 4. Problem classes: *sp* (spelling error), *quote* (missing quotation marks), *cap* (capitalization error), *coll* (colloquial language), *grammar* (wrong grammar), *punct* (punctuation error), *hyph* (missing hyphen), *multisent* (multi-sentence question), *ref* (unclear reference), *blank* (missing or wrong blanks), *markup* (presence of markup in the text), *noq* (not a question, just unspecific phrase or keywords), *link* (question contains URL or domain name), *lang* (question language other than German).

User Identity Features: *Frag Wikia!* has a small group of very active users who account for a substantial portion of the unanswered questions: 35.2% of our sample set were asked by those 40 users with the most unanswered questions, and we have added 40 features to identify these users. Not belonging to this group is also expressed as a feature.

User Profile Features: The *Frag Wikia!* dump contains user activity data which allows computing user profiles in the form of user-related features (e.g. unanswered question rate).

4.4 Experimental Results

In order to evaluate the proposed approach to WAA, the values of all presented features were computed for the 3,996 unanswered question sample, yielding the final data set used in the experiments.[6] Before showing the actual evaluation results, let us briefly explain our choice of evaluation metrics and the chosen result presentation.

Evaluation method As opposed to the situation in IR or normal QA, where the users actively direct their questions to the computer and expect it to find the requested informa-

[6]Note that the features were discretized into 10 equal-sized bins per numeric feature. Features that represent probability estimates were discretized into 200 bins after a logarithmic transform.

Model	AUC	R@.15	R@.2	R@.25	R@.3	R@.35
TSW	0.705	**0.611**	**0.431**	**0.259**	**0.144***	**0.056***
LSW	0.705	**0.628**	0.397	0.158	0.056	**0.048**
LAW	0.701	**0.637**	**0.406**	0.158	0.054	**0.042**
TAW	0.701	**0.617**	**0.437***	**0.265***	**0.144***	**0.051**
TW	0.692	**0.639***	**0.428**	0.192	0.048	**0.042**
LW	0.692	**0.623**	0.394	0.144	0.039	0.034
TSO	0.664	0.572	0.014	0.011	0.011	0.008
LSO	0.660	0.470	0.006	0.006	0.006	0.006
TAO	0.650	0.456	0.014	0.014	0.008	0.008
LAO	0.645	0.383	0.008	0.008	0.006	0.000
TO	0.640	0.454	0.014	0.014	0.011	0.011
LO	0.634	0.268	0.008	0.008	0.008	0.008

Table 2: Results of baselines, sorted by AUC. Results not significantly worse than the best result at a 0.05 significance level using the permutation test are shown in bold, the best result is marked by an asterisk.

tion, users in a QA forum address all participants without expecting a reaction of any specific user. Thus, the users will not miss a response of the QA system if it does not generate one. The system will only be noticed when it generates a response, and in this case it should show an acceptable answer quality—so precision of the submitted responses is the major factor that decides upon acceptance of the virtual participant in the QA forum. Therefore we treat precision as the independent variable that can be freely chosen. Recall becomes the dependent variable and is only of interest in relation to the desired precision level. We hence show recall scores achieved by the learned models for fixed precision levels. Let R@p denote the maximum recall level at which the model yields a precision $\geq$ p. We will consider R@p for p $\in \{0.15, 0.2, 0.25, 0.3, 0.35\}$. Smaller values are irrelevant since returning the unfiltered data already yields 8.9% precision, and higher precision cannot be achieved by the models at acceptable recall levels yet, as we shall see.

The precision and recall scores for all trained models were determined by 10-fold cross validation.

Results of Baseline Methods First we consider the effectiveness of a filtering by manually assigned excluded categories. This blocking eliminates 534 questions. 526 of these do indeed have incorrect answers, resulting in a rejection precision of 97.8% (14.4% of all 3,641 failed cases), while only the remaining 8 (2.2%) are false positives (lost correct answers). The rejection rate is insufficient, though: Since most wrong responses pass the filter, the precision of filtered responses increases from 8.9% to no more than 10.0%.

Results of the Fisher classifiers that serve as the automatic filtering baseline are shown in Table 2, using the naming scheme of Sect. 4.1. The table is sorted by the AUC (area under the ROC curve). The 'W' smoothing parameters worked better than add-one smoothing ('O' in model names). For the precision levels of interest, TSW and TAW performed best.

Tested ML Approaches We used Weka 3.4.1 [Witten and Frank, 2005] as a source of standard ML techniques to be applied to the proposed feature set. After preparatory tests on a different subset of 200 *Frag Wikia!* questions, the following learners were chosen for the experiment: Logistic Regression (LR), LogitBoost (LB), and ADTrees (AD). Naive Bayes (NB) was also chosen as a learner baseline though its independence assumptions are clearly violated by our feature set. All learners were run with default settings. We also tried variants LR.3, LB.3 and AD.3 with a reweighting of negative examples by a factor of 0.3 in order to enhance the positive cases in the training data. This reweighting was implemented using the CostSensitiveClassifier of Weka.

Apart from these well-known models, we used bagging of rank-optimizing decision trees [Glöckner, 2009]. By choosing the number of positive examples Y in the training set for the parameter k and assuming uniform weights, the kMRR metric optimized by the tree induction becomes $\sum_{i=1}^{Y} \frac{1}{Y(\mathrm{rank}_i - i + 1)}$, where rank_i is the position of the i-th best positive training instance in the current ranking determined by the probability of the YES class at the leaf nodes. The metric yields 1 if all positive instances are ranked higher than all negative instances. In order to profit from the capability of the method to incorporate qualitative knowledge on the effect of features on the outcome of classification, we reduced the feature set to 35 features with a clear positive or negative effect on response quality. The tree induction was configured with a depth limit of 30 and a minimum permitted leaf size of 20, 30, or 40 training items, yielding models RO20', RO30' and RO40' (the prime ' signals use of the reduced feature set). Due to the large leaf size, class probabilities are estimated without smoothing. The actual classifiers were constructed by stratified bagging of 10 decision trees whose estimated class probabilities are then averaged. For comparison, the remaining learners were also trained on the reduced feature set, yielding additional models LR', LB', etc.

Results of ML Approaches Results for the chosen learners are shown in Table 3. Logistic regression in the reduced feature space with reweighting (LR.3') was the clear winner, followed by bags of rank-optimizing decision trees, Logit Boost, and AD Trees. Naive Bayes only worked well on the reduced feature set and failed completely at higher precision levels. The remaining classifiers outperform the baseline models.

Let us consider a concrete example. Suppose we want a 30% precision for questions in the unanswered question sample. By applying the LR.3' filter with the best R@.3 score, the number of wrong responses that would be posted in the forum is cut from 3,641 to 343, so 90.6% of all wrong responses are eliminated. Still, we lose only about half of the correct answers, with a final answer rate of $8.9\% \times 41.4\% \approx 3.7\%$.

While a 3.7% answer rate appears low and we will work on improving it, one must keep in mind that the unanswered question set contains unusually hard cases that do not reflect the typical difficulty of questions in *Frag Wikia!*. For a random sample of 200 *Frag Wikia!* questions (including answered ones), the answer rate was 18% rather than 8.9% before filtering. Moreover the system could still contribute more correct answers than any human forum participant – especially in a large forum with thousands of questions per day.

Model	AUC	R@.15	R@.2	R@.25	R@.3	R@.35
LR.3'	0.777	**0.862***	**0.710***	**0.544**	**0.414***	0.031
LR'	0.774	**0.851**	0.673	**0.521**	**0.392**	0.034
RO30'	0.770	**0.842**	**0.690**	**0.552**	0.239	0.031
RO40'	0.768	**0.828**	**0.679**	**0.558***	0.166	0.118
RO20'	0.768	**0.839**	0.665	**0.527**	0.273	0.039
LR.3	0.750	0.817	0.634	**0.549**	0.045	0.023
LR	0.747	0.823	0.600	**0.546**	0.070	0.037
NB'	0.739	0.761	0.611	0.366	0.000	0.000
LB'	0.738	0.783	**0.673**	0.448	0.270	0.107
LB.3'	0.734	0.772	**0.687**	0.490	0.285	**0.228***
LB	0.731	0.741	0.634	0.470	0.254	0.065
LB.3	0.729	0.749	**0.670**	0.479	0.304	0.127
NB	0.729	0.780	0.594	0.000	0.000	0.000
AD	0.711	0.746	0.611	0.335	0.166	0.011
AD.3'	0.707	0.752	0.597	0.369	0.192	0.042
AD.3	0.706	0.744	0.608	0.307	0.180	0.011
AD'	0.704	0.744	0.558	0.310	0.166	0.113

Table 3: Results of ML models for proposed features (same conventions for bold/starred results as in Table 2)

5 Conclusions

We have presented the use case of embedding the QA system LogAnswer within the framework of a QA forum. By participating into QA forums, LogAnswer will benefit both forum users and QA research. However, we have identified obstacles which necessitate system adaptations. For this we have developed and demonstrated an effective ML-based filtering method which operates on manually selected and automatically generated features of questions and answers. Our filtering suppresses a substantial amount of unwanted answers while causing few false positives by erroneously discarding correct answers. For the future we will investigate the use of LogAnswer for the semantic comparison of forum questions, enabling the system to refer users to equivalent and already answered questions. We are also considering to shift to a larger commercial forum which includes a grading mechanism for answers, a feature that is currently missing in *Frag Wikia!*, and which would be helpful for a large scale user evaluation of automated QA in the context of QA forums. Going beyond the forum use case we will integrate existing ontologies like OpenCyc into the knowledge base of LogAnswer.

References

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

[Ferrucci *et al.*, 2010] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

[Furbach *et al.*, 2010] Ulrich Furbach, Ingo Glöckner, and Björn Pelzer. An application of automated reasoning in natural language question answering. *AI Communications*, 23(2-3):241–265, 2010. PAAR Special Issue.

[Glöckner and Pelzer, 2009] Ingo Glöckner and Björn Pelzer. The LogAnswer project at CLEF 2009. In *Working Notes for the CLEF 2009 Workshop*, Corfu, Greece, September 2009.

[Glöckner and Pelzer, 2010] Ingo Glöckner and Björn Pelzer. The LogAnswer project at ResPubliQA 2010. In *CLEF 2010 Working Notes*, September 2010.

[Glöckner, 2009] Ingo Glöckner. Finding answer passages with rank optimizing decision trees. In *Proc. of the Eighth International Conference on Machine Learning and Applications (ICMLA-09)*, pages 208–214. IEEE Press, 2009.

[Hartrumpf *et al.*, 2009] Sven Hartrumpf, Ingo Glöckner, and Johannes Leveling. Efficient question answering with question decomposition and multiple answer streams. In Peters et al. [2009].

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

[Peñas *et al.*, 2009] Anselmo Peñas, Pamela Forner, Richard Sutcliffe, Álvaro Rodrigo, Corina Forăscu, Iñaki Alegria, Danilo Giampiccolo, Nicolas Moreau, and Petya Osenova. Overview of ResPubliQA 2009: Question answering evaluation over European legislation. In *Proceedings of the 10th Cross-Language Evaluation Forum Conference on Multilingual Information Access Evaluation*, CLEF'09, pages 174–196, Berlin, Heidelberg, 2009. Springer.

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

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

[Peters *et al.*, 2009] Carol Peters, Thomas Deselaers, Nicola Ferro, Julio Gonzalo, Gareth J.F. Jones, Mikko Kurimo, Thomas Mandl, Anselmo Peñas, and Vivien Petras, editors. *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, Revised Selected Papers*, Heidelberg, 2009. Springer.

[Surdeanu *et al.*, 2008] Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. Learning to rank answers on large online QA collections. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL 2008)*, 2008.

[Sutcliffe, 2010] Geoff Sutcliffe. The CADE-22 automated theorem proving system competition - CASC-22. *AI Communications*, 23(1):47–59, 2010.

[Tunstall-Pedoe, 2010] William Tunstall-Pedoe. True knowledge: Open-domain question answering using structured knowledge and inference. *AI Magazine*, 31(3):80–92, 2010.

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