Deduction Based Question Answering and its Application

LogAnswer Embedding AD in QAS

Ulrich Furbach
Björn Pelzer
Ingo Glöckner
Hermann Helbig
Deduction Based Question Answering and its Application

LogAnswer Embedding AD in QAS

Ulrich Furbach
Björn Pelzer

Ingo Glöckner
Hermann Helbig
LogAnswer - a Question-Answering System
Who was the lead singer of Nirvana?

Kurt Cobain
Who was the lead singer of Nirvana?

Kurt Cobain
Why use Logic in QA?

Shallow NLP fails when:
Why use Logic in QA?

Shallow NLP fails when:

• words in the knowledge sources do not match exactly:

The Carthaginian general led his army over the Alps in 218 BC.

When did Hannibal cross the Alps?
Why use Logic in QA?

Shallow NLP fails when:

- words in the knowledge sources do not match exactly:
  
  The Carthaginian general led his army over the Alps in 218 BC.

- words match, but they are too far apart:

  Mongolia is very sparsely populated for its size. The landlocked country between China and Russia has a population of 2.9 million people.
The LogAnswer project combines

<table>
<thead>
<tr>
<th>NLP Components</th>
<th>Deduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>• NLP-components</td>
<td>• Theorem prover E-KRHyper</td>
</tr>
<tr>
<td>• MultiNet knowledge base, based on snapshot of German Wikipedia</td>
<td></td>
</tr>
</tbody>
</table>

[Image of logos: FernUniversität in Hagen and Universität Koblenz - Landau]
MultiNet Knowledge Base

MultiNet:

- Multilayered Extended Semantic Networks
- language independent, but tools mostly German
"Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
MultiNet to First-Order Logic
MultiNet Knowledge Base

- snapshot of German Wikipedia
- formal representations of ~12 million sentences, generated semi-automatically
- plus ~12,000 background knowledge axioms, manually adapted from MultiNet inference rules and from WordNet
Hypertableau

- splitting with purification solves the problem of variables shared between tableau branches:
  - only one branch needs to be worked on at any time
  - emphasis on unit operations
  - proof confluent

**E-hyper tableau calculus:**
- clause tree instead of literal tree
- four extension rules instead of one
- adds term ordering (reduction ordering)
- adds redundancy handling

Jelia 96

Cade 07
The LogAnswer question answering system:

"Who was Ian Fleming?"
The LogAnswer question answering system:

"Who was Ian Fleming?"

parse & transform

word info + FOL query
The LogAnswer question answering system:

"Who was Ian Fleming?"

parse & transform

word info + FOL query

information retrieval & machine learning

KB

Wikipedia-snapshot, 29 million sentences
The LogAnswer question answering system:

"Who was Ian Fleming?"

parse & transform

word info + FOL query

information retrieval & machine learning

200 best KB fragments, "answer candidates"

KB

Wikipedia-snapshot, 29 million sentences
The LogAnswer question answering system:

"Who was Ian Fleming?"

parse & transform

word info + FOL query

information retrieval & machine learning

200 best KB fragments, "answer candidates"

KB

Wikipedia-snapshot, 29 million sentences
The LogAnswer question answering system:

- "Who was Ian Fleming?"
  - parse & transform
  - word info + FOL query
  - information retrieval & machine learning

- KB
  - Wikipedia-snapshot, 29 million sentences

- proofs
  - ATP E-KRHyper
    - FOL query
    - + answer candidate
    - + background knowledge

- 200 best KB fragments, "answer candidates"
The LogAnswer question answering system:

"Who was Ian Fleming?"

parse & transform

word info + FOL query

information retrieval & machine learning

200 best KB fragments, "answer candidates"

"British author"

ML-based proof ranking & NL answer generation

proofs

ATP E-KRHyper

FOL query
+ answer candidate
+ background knowledge

KB

Wikipedia-snapshot, 29 million sentences

Logik KB

Backgr. KB
Retrieval of Answer Candidates

Pre-analysed and indexed text passages allow quick computation of syntactic filtering criteria:
Retrieval of Answer Candidates

Pre-analysed and indexed text passages allow quick computation of syntactic filtering criteria:

- **matchRatio**: relative proportion of lexical concepts and numerals in the question which find a match in the text passage,

- **failedMatch**: number of lexical concepts and numerals in the question which find no match in the text passage,

- **failedNames**: proper names which find no match in the text passage,

- **containsBrackets**: indicates whether passage contains parentheses. “...Sydney (Australia) ...”
Retrieval of Answer Candidates

Pre-analysed and indexed text passages allow quick computation of syntactic filtering criteria:

- **matchRatio**: relative proportion of lexical concepts and numerals in the question which find a match in the text passage,
- **failedMatch**: number of lexical concepts and numerals in the question which find no match in the text passage,
- **failedNames**: proper names which find no match in the text passage,
- **containsBrackets**: indicates whether passage contains parentheses. "...Sydney (Australia) ..."

Scores are aggregated into a quality estimate using decision trees. The 'best' 200 logical text passage representations are answer candidates and will be evaluated deductively.
User question: 
Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate: 
"Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
User question: Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate: "Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
User question: Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate: "Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
Rudy Giuliani war Bürgermeister welcher US-Stadt?

"Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
User question:
Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate:
"Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
Rudy Giuliani war Bürgermeister welcher US-Stadt?

"Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
User question: Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate: "Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
User question: Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate: "Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
User question: Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate: "Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
User question:
Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate:
"Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."
User question:
Rudy Giuliani war Bürgermeister welcher US-Stadt?

Answer candidate:
"Hinter der Anklage stand der spätere Bürgermeister von New York, Rudolph Giuliani."

\[
\begin{align*}
\neg \text{attach}(\text{FOCUS}, X1) & \lor \neg \text{sub}(\text{FOCUS}, \text{us_stadt.1.1}) \\
\lor \neg \text{attr}(X1, X2) & \lor \neg \text{val}(X2, \text{rudy.0}) \lor \neg \text{sub}(X2, \text{vorname.1.1}) \\
\lor \neg \text{attr}(X1, X3) & \lor \neg \text{val}(X3, \text{giuliani.0}) \lor \neg \text{sub}(X3, \text{nachname.1.1}) \\
\lor \neg \text{sub}(X1, \text{bürgermeister.1.1})
\end{align*}
\Rightarrow \text{FOCUS} = c215 \quad (2 \text{ relaxations})
Answer Generation

The FOCUS-variable represents the core object of the question.
The FOCUS-variable represents the core object of the question.
Answer Generation

The FOCUS-variable represents the core object of the question.
The FOCUS-variable represents the core object of the question.
The FOCUS-variable represents the core object of the question.

Answer Generation
Answer Generation

The FOCUS-variable represents the core object of the question.

"New York"
Answer Presentation

Decision tree computes a quality score for each answer, using criteria like:

- **skippedLits**: number of query literals skipped by relaxation,
- **npFocus**: FOCUS-variable was bound to a nominal phrase constant,
- **focusEatMatch**: answer type matches expected answer type,
- **irScore**: the original quality estimate for the text passage
Decision tree computes a quality score for each answer, using criteria like:

- **skippedLits**: number of query literals skipped by relaxation,
- **npFocus**: FOCUS-variable was bound to a nominal phrase constant,
- **focusEatMatch**: answer type matches expected answer type,
- **irScore**: the original quality estimate for the text passage

The top 5 answers are presented to the user, together with the text passages to provide context and links to the documents.
Performance Issues

Many prover runs for a single question:

- 200 candidates with up to 5 relaxations each
- >12,000 clauses input for each run

short response time (~5 sec)
Performance Issues

Many prover runs for a single question:

• 200 candidates with up to 5 relaxations each
• >12,000 clauses input for each run

creating index structures for input can exceed time slot
Performance Issues

Many prover runs for a single question:
- 200 candidates with up to 5 relaxations each
- >12,000 clauses input for each run

Plenty of overlap between prover runs:
- all runs use the same background knowledge base (~97% of the clauses in a run)
- all relaxation runs for a candidate use the same candidate
Use overlap between relaxation runs for one candidate:

- input sets for runs differ only in one (dropped) query literal
- relaxed query clause subsumes the previous query clause

\[
\neg Q_1 \lor \neg Q_2 \lor \neg Q_4 \lor \neg Q_5 \subseteq \neg Q_1 \lor \neg Q_2 \lor \neg Q_3 \lor \neg Q_4 \lor 
\]

- all clauses derived before first derivation *Split* can be reused in next run
Use overlap between relaxation runs for one candidate:

- input sets for runs differ only in one (dropped) query literal
- relaxed query clause subsumes the previous query clause

\[ \neg Q_1 \lor \neg Q_2 \lor \neg Q_4 \lor \neg Q_5 \subseteq \neg Q_1 \lor \neg Q_2 \lor \neg Q_3 \lor \neg Q_4 \lor \]

- all clauses derived before first derivation \textit{Split} can be reused in next run

no rigid variables!
Extensions of Reasoning

- Webservices
- Background knowledge
- Partitioning and Heuristics
Webservices in E-KRHyper
Webservices in E-KRHyper

- **webservice query/answer pair**: \( kb(q, a) \)
  
  where \( q \) and \( a \) may be arbitrarily complex terms

  \( kb(conv(eur, usd, 299.95), 392.87) \)
  
  \( kb(q, a) \) is true iff \( a \) is a webservice-reply to \( q \).

- **webservices representation**: \( KB^{ext} \) of ground unit clauses \( kb(q, a) \)

- use non-ground \( kb \)-literals in clauses to access webservices:

  \[ \text{dollarprice}(x, z) \leftarrow \text{europrice}(x, y) \land kb(conv(eur, usd, y), z). \]
Webservices in E-KRHyper

In theory:
Webservices in E-KRHyper

In theory:

$$\text{kb}(\text{conv(eur, usd, 100.00), 131.13})$$

$$\text{kb}(\text{conv(eur, usd, 299.95), 392.87})$$

$$p(a)$$

$$q(b)$$

$$r(a,b)$$  $$s(b)$$  $$\neg p(a)$$  $$\neg q(b)$$
Webservices in E-KRHyper

In theory:

Infeasible in practice:

$KB^{ext}$ may be infinite, retrieving a $kb(q,a)$ takes time.

$\Rightarrow$ minimize accesses to $KB^{ext}$.
Example:

europrice(coke, 1.20)

dollarprice(x,z) ← europrice(x,y) ∧ kb(conv(eur, usd, y), z).
Example:

europrice(coke, 1.20)

dollarprice(x,z) ← europrice(x,y) ∧ kb(conv(eur, usd, y), z).
Example:

dollarprice(x,z) ← europrice(x,y) \land kb(conv(eur, usd, y), z).

\[ \sigma = \{x/\text{coke}, y/1.20, \ldots \} \]
Example:

dollarprice(x,z) ← europrice(x,y) ∧ kb(conv(eur, usd, y), z).

σ = \{x/coke, y/1.20, kb(conv(eur, usd, 1.20), z)\}
Example:

\[ \text{dollarprice}(x,z) \leftarrow \text{europrice}(x,y) \land \text{kb}!(\text{conv(eur, usd, y), z}). \]

\[ \sigma = \{x/\text{coke}, y/1.20, \text{kb}!(\text{conv(eur, usd, 1.20), z}) \} \]

webservice request to currency converter, convert €1.20 to $
Example:

\[ \text{dollarprice}(x, z) \leftarrow \text{europrice}(x, y) \land \text{kb}(\text{conv(eur, usd, y, z)}). \]

\[ \sigma = \{ x/\text{coke}, y/1.20, \} \]

\[ \text{kb}(\text{conv(eur, usd, 1.20, z)}), 1.57 \]

Webservice request to currency converter, convert €1.20 to $
Example:

\[ \text{dollarprice}(x, z) \leftarrow \text{europrice}(x, y) \land \text{kb}(\text{conv(eur, usd, y), z}). \]

\[ \sigma = \{ x/\text{coke}, y/1.20, z/1.57 \} \]

\[ \text{wb(\text{conv(eur, usd, 1.20), z})} \]

\[ 1.57 \]

webservice request to currency converter, convert €1.20 to $
Example:

\[
dollarprice(x, z) \leftarrow \text{europrice}(x, y) \land \text{kb}(\text{conv}(\text{eur}, \text{usd}, y), z).
\]

\[
\sigma = \{x/\text{coke}, y/1.20, z/1.57\}
\]

webservice request to currency converter, convert €1.20 to $

\text{dollarprice(coke, 1.57)}

\text{¬europrice(coke, 1.20)}

\text{¬kb(conv(eur, usd, 1.20), 1.57)}
proxy always responds immediately:

- "wait" - proxy asks webservice; prover does other inferences and will ask again later
- <result> - webservice has been asked and result is in proxy cache
- "no result" - webservice has been asked but cannot provide a useful result
ws-query 1: "wait"  Continue in current branch with other inferences.
ws-query 1: "wait"  Continue in current branch with other inferences.
ws-query 1: "wait" Continue in current branch with other inferences.
ws-query 1: "wait"  Continue in current branch with other inferences.
ws-query 1: "wait"  Continue in current branch with other inferences.

ws-query 1: <result>
ws-query 1: "wait"  Continue in current branch with other inferences.

ws-query 1: <result>

ws-query 2: "wait"
Continue in current branch with other inferences.
When the current branch is exhausted and a query is still waiting, postpone the branch.

postponed
When the current branch is exhausted and a query is still waiting, postpone the branch.

postponed
Continue in current branch with other inferences.

When the current branch is exhausted and a query is still waiting, postpone the branch.

postponed
When the current branch is exhausted and a query is still waiting, postpone the branch.

Return to postponed branch when all other branches have been closed or postponed.

Waiting query returns...

- `<result>`: continue reasoning
- "no result": done, branch is model
- "wait": depends on configuration:
  - either treat as "no result",
  - or wait indefinitely until some waiting query returns `<result>` or "no result"
Webservices in Use
Webservices in Use

- Yahoo GeoPlanet
- Weather Service
- Currency Converter
- OpenCyc
- DBPedia
- SpassYago -- coming soon?
Webservices for Abductive Relaxation

logical query representation:
false \leftarrow Q_1 \land ... \land Q_n
Webservices for Abductive Relaxation

logical query representation:

\[ \text{false} \leftarrow Q_1 \land \ldots \land Q_n \]

1. add abductive relaxation clause:

\[ \text{relAns}(\text{rel}(c_1, x_1), \ldots, \text{rel}(c_m, x_m)) \leftarrow \\
Q_1' \land \ldots \land Q_n' \land \text{kb}(\text{abduce}(c_1), x_1) \land \ldots \land \text{kb}(\text{abduce}(c_m), x_m) \]

where \( Q_1', \ldots, Q_n' \) result from \( Q_1, \ldots, Q_n \) by replacing each occurrence \( c_i \) of a constant with a fresh variable \( x_i \).
Webservices for Abductive Relaxation

logical query representation:

\[ \text{false} \leftarrow Q_1 \land ... \land Q_n \]

1. add abductive relaxation clause:

\[
\text{relAns}(\text{rel}(c_1, x_1), ..., \text{rel}(c_m, x_m)) \leftarrow Q_1' \land ... \land Q_n' \land \text{kb(abduce}(c_1), x_1) \land ... \land \text{kb(abduce}(c_m), x_m)
\]

where \( Q_1', ..., Q_n' \) result from \( Q_1, ..., Q_n \) by replacing each occurrence \( c_i \) of a constant with a fresh variable \( x_i \).

2. add trivial abduction unit clause:

\[
\text{kb(abduce}(x), x) \leftarrow .
\]
"Who invented Coca-Cola?"

false ← is(x, person) \& invent(x, cocacola).
"Who invented Coca-Cola?"

\[ false \leftarrow is(x, \text{person}) \land invent(x, \text{cocacola}). \]

E-KRHyper adds:

\[ rel\text{Ans}(rel(person,y), rel(cocacola, z)) \leftarrow\]
\[ is(x, y) \land invent(x, z)\]
\[ \land \text{kb}(\text{abduce(person), y}) \land \text{kb}(\text{abduce(cocacola), z}). \]

\[ \text{kb}(\text{abduce}(x), x) \leftarrow. \]
"Who invented Coca-Cola?"

\[
\text{false } \leftarrow \text{is}(x, \text{person}) \land \text{invent}(x, \text{cocacola}).
\]

E-KRHyper adds:

\[
\text{relAns}(\text{rel}(\text{person},y), \text{rel}(\text{cocacola},z)) \leftarrow \\
\text{is}(x, y) \land \text{invent}(x, z) \\
\land \text{kb}(\text{abduce}(\text{person}), y) \land \text{kb}(\text{abduce}(\text{cocacola}), z).
\]

\[
\text{kb}(\text{abduce}(x), x) \leftarrow .
\]

- prover will run until proof is found or timeout
- if timeout, return any derived relAns-facts instead
- let user decide if abductive relaxations were acceptable(?)
false ← is(x, person) \land invent(x, cocacola).
relAns(rel(person,y), rel(cocacola,z)) ←
    is(x, y) \land invent(x, z) \land kb(abduce(person), y) \land kb(abduce(cocacola), z).
kb(abduce(x), x) ←.
false ← is(x, person) ∧ invent(x, cocacola).
relAns(rel(person,y), rel(cocacola,z)) ←
    is(x, y) ∧ invent(x, z) ∧ kb(abduce(person), y) ∧ kb(abduce(cocacola), z).
k(b(abduce(x), x) ←.
false ← is(x, person) ∧ invent(x, cocacola).

cbor(relAns(rel(person, y), rel(cocacola, z)) ←
  is(x, y) ∧ invent(x, z) ∧ kb(abduce(person), y) ∧ kb(abduce(cocacola), z).

kb(abduce(x), x) ← .

kb(abduce(x), x)

invent(j_j_berzelius, softdrink)

is(j_j_berzelius, person)
false ← is(x, person) ∧ invent(x, cocacola).

relAns(rel(person,y), rel(cocacola,z)) ←
  is(x, y) ∧ invent(x, z) ∧ kb(abduce(person), y) ∧ kb(abduce(cocacola), z).

kb(abduce(x), x) ←.

 kb(abduce(x), x)
  |                 |
 invent(j_j_berzelius, softdrink)
  |                 |
 is(j_j_berzelius, person)

σ = {x/j_j_berzelius,
y/person,
false ← is(x, person) \land invent(x, cocacola).

relAns(rel(person,y), rel(cocacola,z)) ←
    is(x, y) \land invent(x,z) \land kb(abduce(person), y) \land kb(abduce(cocacola), z).

kb(abduce(x), x) ← .

σ = \{ x/j\_j\_berzelius,
y/person,
z/softdrink \}
false ← is(x, person) ∧ invent(x, cocacola).

relAns(rel(person,y), rel(cocacola,z)) ←
   is(x, y) ∧ invent(x, z) ∧ kb(abduce(person), y) ∧ kb(abduce(cocacola), z).

kb(abduce(x), x) ←.

σ = {x/j_j_berzelius, y/person, z/softdrink}
false ← is(x, person) ∧ invent(x, cocacola).

relAns(rel(person,y), rel(cocacola,z)) ←
    is(x, y) ∧ invent(x,z) ∧ kb(abduce(person),y) ∧
    kb(abduce(cocacola), z)

kb(abduce(x), x) ←.

σ = {x/j_j_berzelius, y/person, z/softdrink}
false ← \( \text{is}(x, \text{person}) \land \text{invent}(x, \text{cocacola}). \)

\[
\text{relAns}(\text{rel}(\text{person}, y), \text{rel}(\text{cocacola}, z)) \leftarrow \\
\quad \text{is}(x, y) \land \text{invent}(x, z) \land \text{kb}(\text{abduce}(\text{person}), y) \land \text{kb}(\text{abduce}(\text{cocacola}), z)
\]

\[
\text{kb}(\text{abduce}(x), x) \leftarrow .
\]

\[
\sigma = \{ x/y/j\_j\_berzelius, \\
y/p\_person, \\
z/z/\text{softdrink} \}
\]

webservice request to ontology browser, superclasses of \text{cocacola}?
false ← is(x, person) ∧ invent(x, cocacola).

relAns(rel(person,y), rel(cocacola,z)) ←
    is(x, y) ∧ invent(x, z) ∧ kb(abduce(person),y) ∧ kb(abduce(cocacola), z)

kb(abduce(x), x) ←.

web service request to ontology browser, superclasses of cocacola?
response: kb(abduce (cocacola), softdrink)

σ = {x/j_j_berzelius, y/person, z/softdrink}
false ← is(x, person) ∧ invent(x, cocacola).

relAns(rel(person,y), rel(cocacola,z)) ←
    is(x, y) ∧ invent(x,z) ∧ kb(abduce(person),y) ∧ kb(abduce(cocacola), z)

kb(abduce(x), x) ← .

webservice request to ontology browser,
superclasses of cocacola?
response: kb(abduce (cocacola),softdrink)

σ = {x/j_j_berzelius, y/person, z/softdrink}
false ← is(x, person) ∧ invent(x, cocacola).

relAns(rel(person, y), rel(cocacola, z)) ←
  is(x, y) ∧ invent(x, z) ∧ kb(abduce(person), y) ∧ kb(abduce(cocacola), z).

kb(abduce(x), x) ←.

relAns(rel(person, person), rel(cocacola, softdrink))

invent(j_j_berzelius, softdrink)

is(j_j_berzelius, person)

σ = {x/j_j_berzelius, y/person, z/softdrink}

webservice request to ontology browser, superclasses of cocacola?
response: kb(abduce (cocacola), softdrink)
false ← is(x, person) ∧ invent(x, cocacola).

relAns(rel(person,y), rel(cocacola,z)) ←
  is(x, y) ∧ invent(x, z) ∧ kb(abduce(person), y) ∧ kb(abduce(cocacola), z).

relAns(rel(person,person), rel(cocacola,softdrink))

σ = {x/j_j_berzelius, y/person, z/softdrink}

"Who invented Coca-Cola?"
- "J.J. Berzelius, if 'soft drink' can replace 'Coca-Cola'." (No!)
Partitioning of Clause set

Compute dependency graphs in pre-processing

Test:

309 multi-literal clauses

10,061 units

Reduction of 20% of clauses and 10% of time
Issues for AD

- Time-constraints
- Relaxing queries
- Abduction
- Webservices
- Confidence Measures
QA Forums

Forum users ask and answer questions.
Users can grade answers or answerers.
QA-systems like LogAnser usually give the best answer they can find – even if it is poor.
QA-systems like LogAnser usually give the best answer they can find – even if it is poor.

If 18% of questions are answered correctly, then almost 82% get wrong answers.

High risk that forum users perceive LogAnswer as a nuisance!
QA-systems like LogAnser usually give the best answer they can find – even if it is poor.

If 18% of questions are answered correctly, then almost 82% get wrong answers.

High risk that forum users perceive LogAnswer as a nuisance!

Improve answer derivation to increase number of correct answers.
QA-systems like LogAnser usually give the best answer they can find – even if it is poor.

If 18% of questions are answered correctly, then almost 82% get wrong answers.

High risk that forum users perceive LogAnswer as a nuisance!

Improve answer derivation to increase number of correct answers.

Improve detection of unanswerable questions and bad answers – if "best" answer is irrelevant or wrong, keep quiet instead of posting.
QA-systems like LogAnser usually give the best answer they can find – even if it is poor.

If 18% of questions are answered correctly, then almost 82% get wrong answers.

High risk that forum users perceive LogAnswer as a nuisance!

- Improve answer derivation to increase number of correct answers.
- Improve detection of unanswerable questions and bad answers – if "best" answer is irrelevant or wrong, keep quiet instead of posting.
Block question categories:
Some categories contain mostly unanswerable questions.

• celebrities: "What is the phone number of Justin Bieber?"
• video games: "How do you unlock the Bowser Level in Super Mario Sunshine?"
• computer help: "When I create a file in Corel (Windows), how do I..."
• ...

Avoiding “Impossible” Questions

Block question categories:
Some categories contain mostly unanswerable questions.

• celebrities: "What is the phone number of Justin Bieber?"
• video games: "How do you unlock the Bowser Level in Super Mario Sunshine?"
• computer help: "When I create a file in Corel (Windows), how do I..."
• ...

Blocking selected categories is quite safe (2.2% false positives), but 66% of questions have no category at all.
Avoiding "Impossible" Questions

Block question categories:
Some categories contain mostly unanswerable questions.

- celebrities: "What is the phone number of Justin Bieber?"
- video games: "How do you unlock the Bowser Level in Super Mario Sunshine?"
- computer help: "When I create a file in Corel (Windows), how do I..."
- ...

Blocking selected categories is quite safe (2.2% false positives), but 66% of questions have no category at all.

Block question types:
Some questions ask for opinions, recognizable by keywords.

- "What do you think of...?"
- "What is the best/worst/most beautiful...?"
External Ontologies

- Cyc is too large for ATP:
  3.3 million formulas in the TPTP-version of OpenCyc, requires 15 minutes to load and 7GB RAM in E-KRHyper.
  - Use query-based axiom selection methods.
  - No equational reasoning for Cyc-axioms.
  - Only use some parts of Cyc (i.e. subclass-assertions).