Improving QA Accuracy by Question Inversion

John Prager, Pablo Duboue, Jennifer Chu-Carroll

Presentation by Sam Cunningham and Martin Wintz
Goal

• Re-rank answers based on additional “inverted questions” in order to improve accuracy.

• Also: Identify pressing issues related to question answering (QA)
QA basic framework

- Question
- Query
- Documents
- Information retrieval
- Query processing
- Answer processing
- Corpus
- Answer
“Inverted Questions”

• “What is the capital of France?”
• Becomes: “Of what country is Paris the capital?”
Related Work

• The LCC system (Moldovan & Rus, 2001)
  – “Logic Prover”

• Clarke et al., 2001; Prager et al. 2004b
  – Assign confidence boost based on redundancy of answer
System Overview
Inverting the question

• Identify term with known type (the pivot term)
  – For example in: “What was the capital of Germany in 1985” Germany is identified as a (COUNTRY).

• Then given a candidate answer <CandAns> formulate the inverted question as:
  – “Of what (COUNTRY) was <CandAns> the capital in 1985”
Difficulties

• Estimated 79% of question in TREC can be inverted meaningfully and those questions are hard to identify.

• Need to have a comprehensive and accurate notion of types

• Some inverted questions have so many answers they’re not useful
Inversion Algorithm

• Not actually formulating inversions in natural language.

• Qframe
  – Keywords
  – Answer type
  – Relationships
Keywords: \{1945, Germany, capital\}
AnswerType: CAPITAL
Relationships: \{(Germany, capital), (capital, CAPITAL), (capital, 1945)\}

Keywords: \{1945, <CANDANS>, capital\}
AnswerType: COUNTRY
Relationships: \{(COUNTRY, capital), (capital, <CANDANS>), (capital, 1945)\}
Using the inversion

• If the answer to an inverted question (validating answer) matches the original question, that question is validated

• Use this notion of validation, along with the scores of the validating answers, to re-rank candidate answers
Using the inversion

- Only concerned with re-ranking top two results
- Learn a decision tree to decide whether to re-rank second result as first one
Decision tree algorithm

1. If $C_1 = \text{nil}$ and $V_2$, return $C_2$
2. If $V_j$ and $A_j > a_1$, return $C_1$
3. If not $V_1$ and not $V_2$ and $\text{type}(T) \in \text{MUSTCONSTRAIN}$, return nil
4. If not $V_1$ and not $V_2$ and $\text{type}(T) \notin \text{SOFTREFUTATION}$, if $S_1 > a_2$, return $C_1$ else nil
5. If not $V_2$, return $C_1$
6. If not $V_1$ and $V_2$ and $A_2 > a_3$ and $P_2 < a_4$ and $S_1 - S_2 < a_5$ and $S_2 > a_6$, return $C_2$
7. If $V_j$ and $V_2$ and $(A_2 - P_2/a_7) > (A_1 - P_1/a_7)$ and $A_1 < a_8$ and $P_1 > a_9$ and $A_2 < a_{10}$ and $P_2 > a_{11}$ and $S_1 - S_2 < a_{12}$ and $(S_2 - P_2/a_7) > a_{13}$, return $C_2$
8. else return $C_1$
Decision tree algorithm

1. If $C_1 = \text{nil}$ and $V_2$, return $C_2$
2. If $V_j$ and $A_1 > a_1$, return $C_1$
3. If not $V_1$ and not $V_2$ and
   $\text{type}(T) \in \text{MUSTCONSTRAIN}$,
   return nil
4. If not $V_1$ and not $V_2$ and
   $\text{type}(T) \notin \text{SOFTREFUTATION}$,
   if $S_1 > a_2$, return $C_1$ else nil
5. If not $V_2$, return $C_1$
6. If not $V_1$ and $V_2$ and
   $A_2 > a_3$ and $P_2 < a_4$ and
   $S_1 - S_2 < a_5$ and $S_2 > a_6$, return $C_2$
7. If $V_j$ and $V_2$ and
   $(A_2 - P_2/a_7) > (A_1 - P_1/a_7)$ and
   $A_1 < a_8$ and $P_1 > a_9$ and
   $A_2 < a_{10}$ and $P_2 > a_{11}$ and
   $S_1 - S_2 < a_{12}$ and $(S_2 - P_2/a_7) > a_{13}$,
   return $C_2$
8. else return $C_1$

Don’t be scared by the variables!

- $a_k$: Learned parameters
- $C_i$: top two candidate answers
- $S_i$: Scores of candidate answers
- $V_i$: whether $C_i$ is validated
- $P_i$: rank of validating answer
- $A_i$: Score of validating answer
Decision tree algorithm

1. \textbf{If \( C_1 = \text{nil} \) and \( V_2 \), return \( C_2 \)}
2. \textbf{If \( V_1 \) and \( A_1 > a_1 \), return \( C_1 \)}
3. \textbf{If not \( V_1 \) and not \( V_2 \) and}
   \( \text{type}(T) \in \text{MUSTCONSTRAIN}, \)
   return \( \text{nil} \)
4. \textbf{If not \( V_1 \) and not \( V_2 \) and}
   \( \text{type}(T) \not\in \text{SOFTREFUTATION}, \)
   \textbf{if} \( S_1 > a_2 \), return \( C_1 \) \textbf{else} \( \text{nil} \)
5. \textbf{If not \( V_2 \), return \( C_1 \)}
6. \textbf{If not \( V_1 \) and \( V_2 \) and}
   \( A_2 > a_3 \) \textbf{and} \( P_2 < a_4 \) \textbf{and}
   \( S_1 - S_2 < a_5 \) \textbf{and} \( S_2 > a_6 \), return \( C_2 \)
7. \textbf{If \( V_1 \) and \( V_2 \) and}
   \( (A_2 - P_2/a_7) > (A_1 - P_1/a_7) \) \textbf{and}
   \( A_1 < a_8 \) \textbf{and} \( P_1 > a_9 \) \textbf{and}
   \( A_2 < a_{10} \) \textbf{and} \( P_2 > a_{11} \) \textbf{and}
   \( S_1 - S_2 < a_{12} \) \textbf{and} \( (S_2 - P_2/a_7) > a_{13}, \)
   \textbf{return} \( C_2 \)
8. \textbf{else return} \( C_1 \)

If there is no first answer and the second answer has been validated return the second answer.
Decision tree algorithm

1. If $C_1 = \text{nil}$ and $V_2$, return $C_2$
2. **If $V_1$ and $A_1 > a_1$, return $C_1$**
3. If not $V_1$ and not $V_2$ and
   
   
   
   \[\text{type}(T) \in \text{MUSTCONSTRAIN},\]
   
   return nil
4. If not $V_1$ and not $V_2$ and
   
   \[\text{type}(T) \not\in \text{SOFTREFUTATION},\]
   
   if $S_1 > a_2$, return $C_1$ else nil
5. If not $V_2$, return $C_1$
6. If not $V_1$ and $V_2$ and
   
   $A_2 > a_3$ and $P_2 < a_4$ and
   
   $S_2 - S_2 < a_5$ and $S_2 > a_6$, return $C_2$
7. If $V_1$ and $V_2$ and
   
   $(A_2 - P_2/a_7) > (A_1 - P_1/a_7)$ and
   
   $A_1 < a_8$ and $P_1 > a_9$ and
   
   $A_2 < a_{10}$ and $P_2 > a_{11}$ and
   
   $S_2 - S_2 < a_{12}$ and $(S_2 - P_2/a_7) > a_{13}$,
   
   return $C_2$
8. else return $C_1$

If the first answer is validated with a score above a given threshold return the first answer.
Decision tree algorithm

1. If $C_1 = \text{nil}$ and $V_2$, return $C_2$
2. If $V_1$ and $A_1 > a_1$, return $C_1$
3. \textbf{If not $V_1$ and not $V_2$ and}
   \quad $type(T) \in \text{MUSTCONSTRAIN}$,
   \quad return nil
4. \textbf{If not $V_1$ and not $V_2$ and}
   \quad $type(T) \notin \text{SOFTREFUTATION}$,
   \quad if $S_1 > a_2$, return $C_1$ else nil
5. If not $V_2$, return $C_1$
6. \textbf{If not $V_1$ and $V_2$ and}
   \quad $A_2 > a_3$ and $P_2 < a_4$
   \quad and $S_1 - S_2 < a_5$ and $S_2 > a_6$, return $C_2$
7. \textbf{If $V_1$ and $V_2$ and}
   \quad $(A_2 - P_2/a_7) > (A_1 - P_1/a_7)$
   \quad and $A_1 < a_8$ and $P_1 > a_9$
   \quad and $A_2 < a_{10}$ and $P_2 > a_{11}$
   \quad and $S_1 - S_2 < a_{12}$ and $(S_2 - P_2/a_7) > a_{13}$,
   \quad return $C_2$
8. else return $C_1$

If neither answers have been validated, either reject both answers or possibly return the first one depending on the type of the pivot term.
Decision tree algorithm

1. If $C_1 = \text{nil}$ and $V_2$, return $C_2$
2. If $V_1$ and $A_1 > a_1$, return $C_1$
3. If not $V_1$ and not $V_2$ and
   
   \[
   \text{type}(T) \in \text{MUST CONSTRAIN},
   \]
   
   return nil
4. If not $V_1$ and not $V_2$ and
   
   \[
   \text{type}(T) \notin \text{SOFT REFUTATION},
   \]
   
   if $S_1 > a_2$, return $C_1$ else nil
5. If not $V_2$, return $C_1$
6. \textbf{If not $V_1$ and $V_2$ and}
   
   $A_2 > a_3$ and $P_2 < a_4$ and
   
   $S_1 - S_2 < a_5$ and $S_2 > a_6$, return $C_2$
7. If $V_1$ and $V_2$ and
   
   $(A_2 - P_2/a_7) > (A_1 - P_1/a_7)$ and
   
   $A_1 < a_8$ and $P_1 > a_9$ and
   
   $A_2 < a_{10}$ and $P_2 > a_{11}$ and
   
   $S_1 - S_2 < a_{12}$ and $(S_2 - P_2/a_7) > a_{13}$,
   
   return $C_2$
8. else return $C_1$

If only the second answer is validated then compare the score of both the answer and the validating answer.
Decision tree algorithm

1. If $C_1 = \text{nil}$ and $V_2$, return $C_2$
2. If $V_1$ and $A_1 > a_1$, return $C_1$
3. If not $V_1$ and not $V_2$ and
   
   $\text{type}(T) \in \text{MUSTCONSTRAIN}$,
   
   return nil
4. If not $V_1$ and not $V_2$ and
   
   $\text{type}(T) \notin \text{SOFTREFUTATION}$,
   
   if $S_1 > a_2$, return $C_1$ else nil
5. If not $V_2$, return $C_1$
6. If not $V_1$ and $V_2$ and
   
   $A_2 > a_3$ and $P_2 < a_4$ and
   
   $S_2 - S_1 < a_5$ and $S_2 > a_6$, return $C_2$
7. If $V_1$ and $V_2$ and
   
   $(A_2 - P_2 / a_7) > (A_1 - P_1 / a_7)$ and
   
   $A_1 < a_8$ and $P_1 > a_9$ and
   
   $A_2 < a_{10}$ and $P_2 > a_{11}$ and
   
   $S_1 - S_2 < a_{12}$ and $(S_2 - P_2 / a_7) > a_{13}$,
   
   return $C_2$
8. else return $C_1$

If both answers are validated compare the scores of both the candidate answers and the validating an:
Decision tree algorithm

• Train on TREC11 corpus of question-answer sets to learn threshold values ($a_k$)
Evaluation

• 50 hand-crafted questions of the form “What is the capital of X?”

• AQUAINTE corpus
  – ~1 million news-wire documents.

• CNS corpus
  – 37,000 documents from the Center for Nonproliferation Studies.
<table>
<thead>
<tr>
<th></th>
<th>AQUAIINT baseline</th>
<th>AQUAIINT w/constraints</th>
<th>CNS baseline</th>
<th>CNS w/constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firsts (non-nil)</td>
<td>39/50</td>
<td>43/50</td>
<td>7/23</td>
<td>4/23</td>
</tr>
<tr>
<td>Total nils</td>
<td>0/0</td>
<td>0/0</td>
<td>0/27</td>
<td>16/27</td>
</tr>
<tr>
<td>Total firsts</td>
<td>39/50</td>
<td>43/50</td>
<td>7/50</td>
<td>20/50</td>
</tr>
<tr>
<td>% correct</td>
<td>78</td>
<td>86</td>
<td>14</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 2. Evaluation on AQUAIINT and CNS corpora.
Evaluation II

• Processed 414 factoid questions from TREC12
<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firsts (non-nil)</td>
<td>105</td>
<td>113</td>
</tr>
<tr>
<td>nils</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Total firsts</td>
<td>108</td>
<td>118</td>
</tr>
<tr>
<td>% correct</td>
<td>26.1</td>
<td>28.5</td>
</tr>
</tbody>
</table>

Table 3. Evaluation on TREC12 Factoids.
Conclusion

• Slight improvement

• Computationally expensive

• Lacks robust notion of term equivalence
Discussion

- Probability-based scores
- Better confidences
- Better NER
- Establishing term equivalence