Improving QA Accuracy by Question Inversion

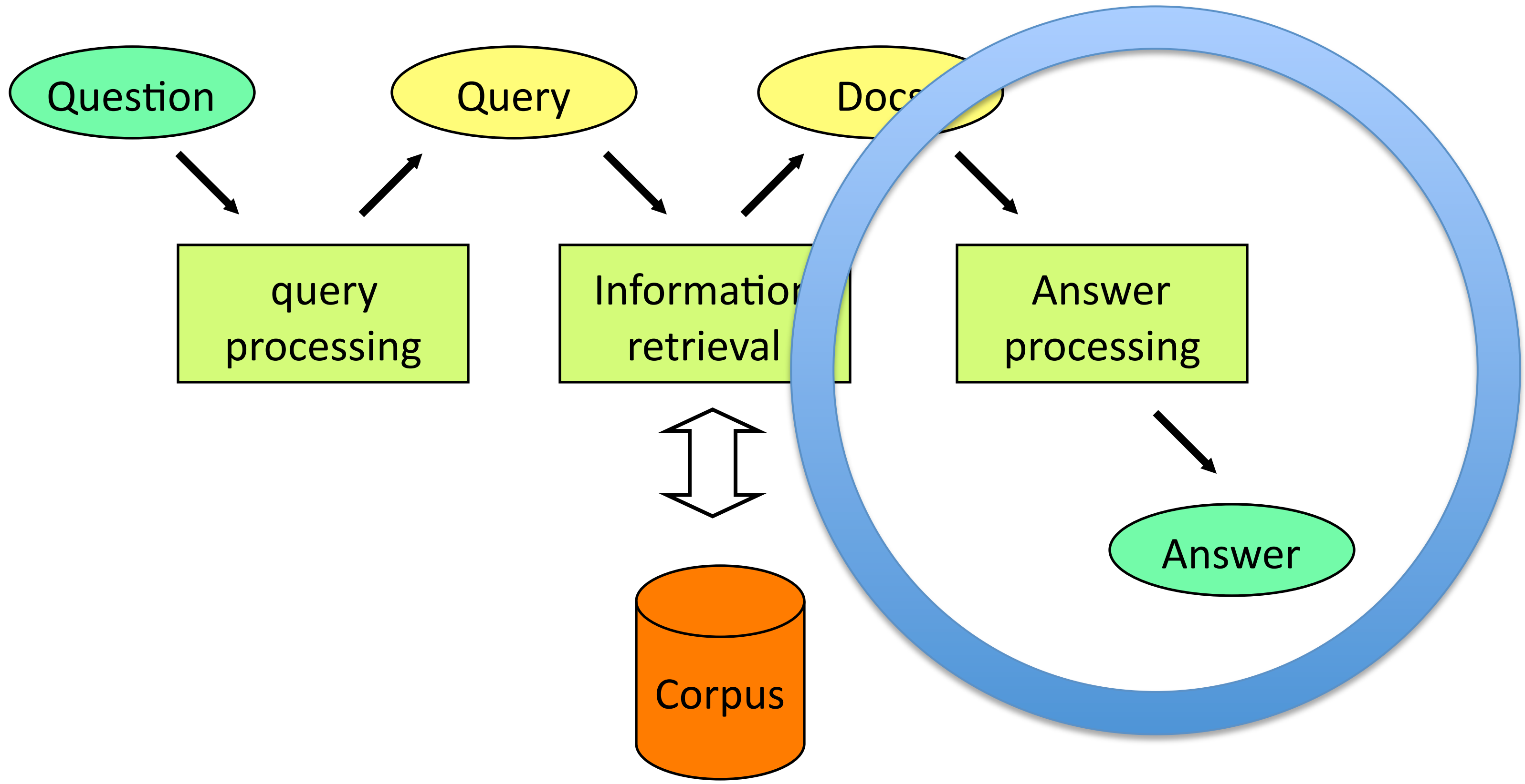
John Prager, Pablo Duboue, Jennifer Chu-Carroll

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



“Inverted Questions”

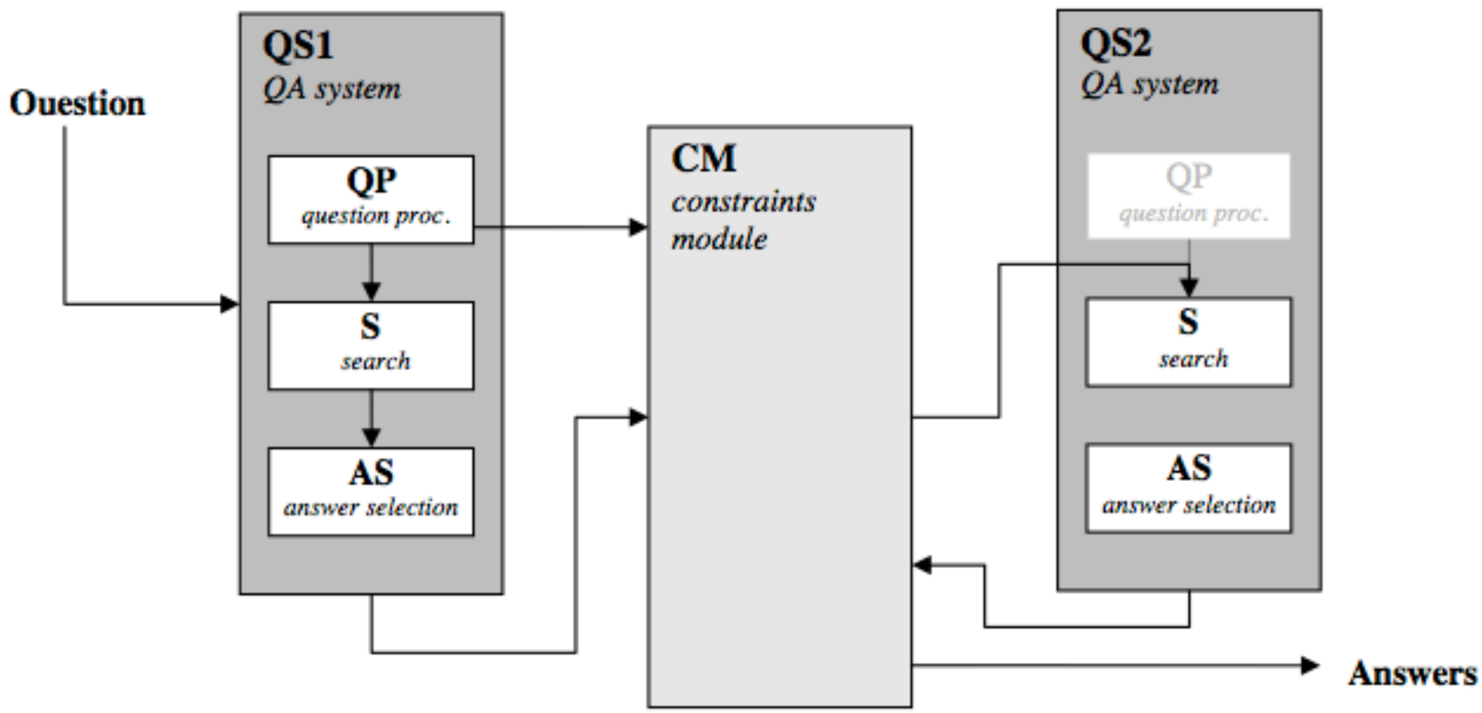
- “What is the capital of France?”
- Becomes: “Of what country is Paris the capital?”

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
  - Answertype
  - 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**

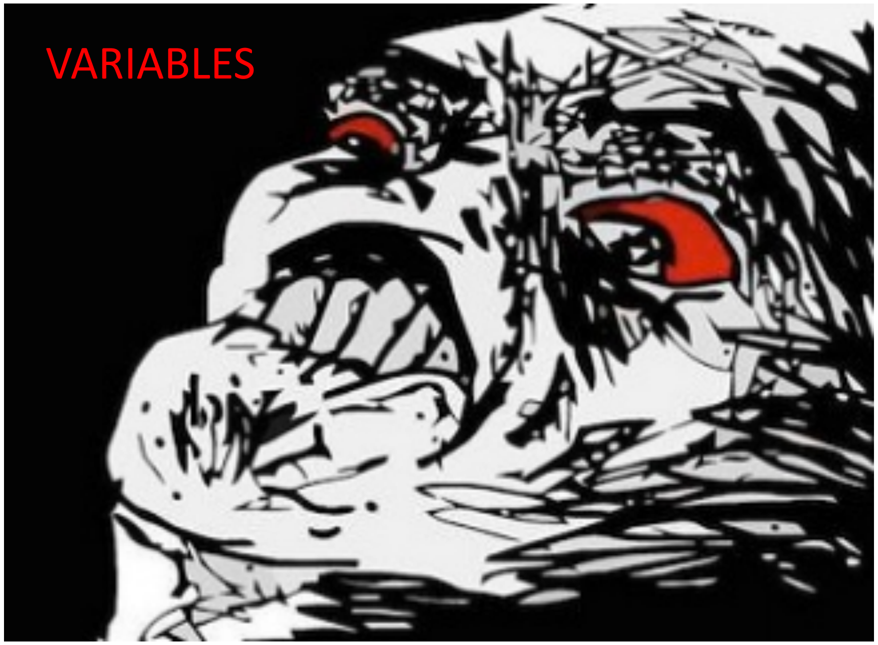
- Use this notion of validation, along with the scores of the validating answers, to re-rank candidate answers

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 **= 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**
   $type(T) \in$ MUSTCONSTRAIN,
   return nil
4. **If not** V_1 **and not** V_2 **and**
   $type(T) \notin$ 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_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



Decision tree algorithm

1. **If** C_1 **= 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** $type(T) \in$ MUSTCONSTRAIN, **return nil**
4. **If not** V_1 **and not** V_2 **and** $type(T) \notin$ 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** $S_1\text{-}S_2 < a_5$ **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** $S_1\text{-}S_2 < a_{12}$ **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. **If** C_1 **= 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**
   $type(T) \in$ MUSTCONSTRAIN,
   return nil
4. **If not** V_1 **and not** V_2 **and**
   $type(T) \notin$ 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**
   $S_1\text{-}S_2 < a_5$ **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**
   $S_1\text{-}S_2 < a_{12}$ **and** $(S_2 - P_2/a_7) > a_{13}$,
   return C_2
8. **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 **= 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**
   $type(T) \in$ MUSTCONSTRAIN,
   return nil
4. **If not** V_1 **and not** V_2 **and**
   $type(T) \notin$ 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**
   $S_1\text{-}S_2 < a_5$ **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**
   $S_1\text{-}S_2 < a_{12}$ **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 **= 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**
   type(T) $\in$ MUSTCONSTRAIN,
   return nil
4. **If not** V_1 **and not** V_2 **and**
   type(T) $\notin$ 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_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 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

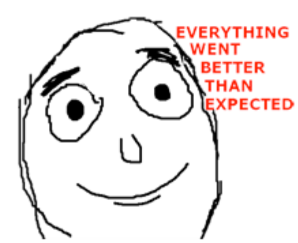
1. **If** C_1 **= nil and** V_2**, return** C_2
2. **If** V_1 **and** $A_1 > a_1$**, return** C_1
3. **If not** V_1 **and not** V_2 **and** $type(T) \in$ MUSTCONSTRAIN, **return nil**
4. **If not** V_1 **and not** V_2 **and** $type(T) \notin$ SOFTREFUTATION, **if** $S_1 > a_2$**,, return** C_1 **else 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_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 **= 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**
   type(T) $\in$ MUSTCONSTRAIN,
   return nil
4. **If not** V_1 **and not** V_2 **and**
   type(T) $\notin$ 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_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 ans



Decision tree algorithm

- Train on TREC11corpus of question-answer sets to learn threshold values (a_k)

Evaluation

- 50 hand-crafted questions of the form "What is the capital of *X*?"
- AQUAINT corpus
  - ~1 million news-wire documents.
- CNS corpus
  - 37,000 documents from the Center for Nonproliferation Studies.

	AQUAINT baseline	AQUAINT w/constraints	CNS baseline	CNS w/constraints
Firsts (non-nil)	39/50	43/50	7/23	4/23
Total nils	0/0	0/0	0/27	16/27
Total firsts	39/50	43/50	7/50	20/50
% correct	78	86	14	40

Table 2. Evaluation on AQUAINT and CNS corpora.

Evaluation II

- Processed 414 factoid questions from TREC12

	Baseline	Constraints
Firsts (non-nil)	105	113
nils	3	5
Total firsts	108	118
% correct	26.1	28.5

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