Question Generation with Minimal Recursion Semantics

Xuchen Yao

European Masters in Language and Communication Technologies

Supervisors: Prof. Hans Uszkoreit and Dr. Yi Zhang, Saarland University
Co-supervisor: Dr. Gosse Bouma, University of Groningen

28 July 2010, Master Colloquium
Outline

Introduction
Definition
Usage
Template/Syntax/Semantics-based Approaches

Background
MRS/ERG/PET/LKB

System Architecture
Overview
MRS Transformation for Simple Sentences
MRS Decomposition for Complex Sentences
Question Reranking

Evaluation
QGSTEC 2010
Question Generation (QG)

The task of generating reasonable questions from a text.

*Deep QG*: why, why not, what-if, what-if-not, how

*Shallow QG*: who, what, when, where, which, how many/much, yes/no

Jackson was born on August 29, 1958 in Gary, Indiana.

- **Who** was born on August 29, 1958 in Gary, Indiana?
- **Which** artist was born on August 29, 1958 in Gary, Indiana?
- **Where** was Jackson born?
- **When** was Jackson born?
- **Was** Jackson born on August 29, 1958 in Gary, Indiana?
Question Generation (QG)

The task of generating reasonable questions from a text.

*Deep QG*: why, why not, what-if, what-if-not, how

*Shallow QG*: who, what, when, where, which, how many/much, yes/no

Jackson was born on August 29, 1958 in Gary, Indiana.

- **Who** was born on August 29, 1958 in Gary, Indiana?
- **Which** artist was born on August 29, 1958 in Gary, Indiana?
- **Where** was Jackson born?
- **When** was Jackson born?
- **Was** Jackson born on August 29, 1958 in Gary, Indiana?
# Outline

## Introduction
- Definition
- Usage
  - Template/Syntax/Semantics-based Approaches

## Background
- MRS/ERG/PET/LKB

## System Architecture
- Overview
- MRS Transformation for Simple Sentences
- MRS Decomposition for Complex Sentences
- Question Reranking

## Evaluation
- QGSTEC 2010
Usage

- Intelligent tutoring systems
  - QG can ask learners questions based on learning materials in order to check their accomplishment or help them focus on the keystones in study.
  - QG can also help tutors to prepare questions intended for learners or prepare for questions possibly from learners.

- Closed-domain question answering (QA) systems
  - Some closed-domain QA systems use pre-defined (sometimes hand-written) question-answer pairs to provide QA services.
  - By employing a QG approach such systems could expand to other domains with a small effort.
Outline

Introduction
Definition
Usage
Template/Syntax/Semantics-based Approaches

Background
MRS/ERG/PET/LKB

System Architecture
Overview
MRS Transformation for Simple Sentences
MRS Decomposition for Complex Sentences
Question Reranking

Evaluation
QGSTEC 2010
Approaches

- Template-based
  - What did <character> <verb>?

- Syntax-based
  - John plays football. (S V O)
  - John plays what? (S V WHNP)
  - John does play what? (S Aux-V V WHNP)
  - Does John play what? (Aux-V S V WHNP)
  - What does John play? (WHNP Aux-V S V)

- Semantics-based
  - play(John, football)
  - play(who, football)
  - play(John, what) || play(John, what sport)
Approaches

- Template-based
  - *What did <character> <verb>*?

- Syntax-based
  - John plays football. (S V O)
  - John plays what? (S V WHNP)
  - John does play what? (S Aux-V V WHNP)
  - Does John play what? (Aux-V S V WHNP)
  - What does John play? (WHNP Aux-V S V)

- Semantics-based
  - play(John, football)
  - play(*who*, football)
  - play(John, *what*) || play(John, *what sport*)
Outline

Introduction
  Definition
  Usage
  Template/Syntax/Semantics-based Approaches

Background
  MRS/ERG/PET/LKB

System Architecture
  Overview
  MRS Transformation for Simple Sentences
  MRS Decomposition for Complex Sentences
  Question Reranking

Evaluation
  QGSTEC 2010
DELPH-IN (MRS/ERG/PET/LKB)
Deep Linguistic Processing with HPSG: http://www.delph-in.net/

John likes Mary.

like(John, Mary)

Parsing with PET

Generation with LKB

John likes Mary.

INDEX: e2
RELS: <
[ PROPER_Q_REL<0:4>  [ NAMED_REL<0:4>
  LBL: h3  LBL: h7
  ARG0: x6  ARG0: x6
  RSTR: h5 (PERS: 3 NUM: SG)
  BODY: h4 ]  CARG: "John"
]

[ _like_v_1_rel<5:10>  [ NAMED_REL<11:17>
  LBL: h8  LBL: h13
  ARG0: e2 [ e SF: PROP TENSE: PRES ]
  ARG1: x6  ARG0: x9
  ARG2: x9 (PERS: 3 NUM: SG)
  BODY: h4 ]  CARG: "John"
]

> HCONS: < h5 qeq h7 h12 qeq h13 >

English Resource Grammar
Dependency MRS

like(John, Mary)

Figure: DMRS for “John likes Mary.”
Initial Idea

like(John,Mary) -> like(who,Mary)

Figure: “John likes Mary” → “Who likes Mary?”
Details

(THEORY) MRS: Minimal Recursion Semantics
a meta-level language for describing semantic structures in some underlying object language.

(GRAMMAR) ERG: English Resource Grammar
a general-purpose broad-coverage grammar implementation under the HPSG framework.

(TOOL) LKB: Linguistic Knowledge Builder
a grammar development environment for grammars in typed feature structures and unification-based formalisms.

(TOOL) PET: a platform for experimentation with efficient HPSG processing techniques
a two-stage parsing model with HPSG rules and PCFG models, balancing between precise linguistic interpretation and robust probabilistic coverage.
Outline

Introduction
  Definition
  Usage
  Template/Syntax/Semantics-based Approaches

Background
  MRS/ERG/PET/LKB

System Architecture
  Overview
  MRS Transformation for Simple Sentences
  MRS Decomposition for Complex Sentences
  Question Reranking

Evaluation
  QGSTEC 2010
MrsQG

http://code.google.com/p/mrsqg/

1. Term extraction
2. FSC construction
3. Parsing with PET
4. MRS Decomposition
   - Apposition Decomposer
   - Coordination Decomposer
   - Subclause Decomposer
   - Subordinate Decomposer
   - Why Decomposer
5. MRS Transformation
6. Generation with LKB
7. Output selection
8. Output to console/XML
Term Extraction

- Stanford Named Entity Recognizer
- a regular expression NE tagger
- an Ontology NE tagger

Jackson was born on **August 29, 1958** in **Gary, Indiana**.
Introduction
  Definition
  Usage
  Template/Syntax/Semantics-based Approaches

Background
  MRS/ERG/PET/LKB

System Architecture
  Overview
    MRS Transformation for Simple Sentences
    MRS Decomposition for Complex Sentences
    Question Reranking

Evaluation
  QGSTEC 2010
MRS Transformation

1. Plain text
2. Term extraction
3. FSC construction
4. MRS Decomposition
   - Apposition Decomposer
   - Coordination Decomposer
   - Subclause Decomposer
   - Subordinate Decomposer
   - Why Decomposer
5. MRS Transformation
6. Generation with LKB
7. Output selection
8. Output to console/XML
**WHO**

Figure: “John likes Mary” $\rightarrow$ “Who likes Mary?”
Figure: “Mary sings on Broadway.” → “Where does Mary sing?”
Figure: “Mary sings at 10.” → “When does Mary sing?”
WHY

Figure: “John fights for Mary.” → “Why does John fight?”
Outline

Introduction
  Definition
  Usage
  Template/Syntax/Semantics-based Approaches

Background
  MRS/ERG/PET/LKB

System Architecture
  Overview
  MRS Transformation for Simple Sentences
  MRS Decomposition for Complex Sentences
  Question Reranking

Evaluation
  QGSTEC 2010
Why?

Question Transformation does not Work Well without Sentence Simplification

Input Sentence:
ASC takes a character as input, and returns the integer giving the ASCII code of the input character.

Desired question:
(a) What does ASC take as input?
(b) What does ASC return?

Actual questions that could have been generated from MRS transformation:
(c) What does ASC take as input and returns the integer giving the ASCII code of the input character?
(d) ASC takes a character as input and returns what giving the ASCII code of the input character?
Why?

Question Transformation does not Work Well without Sentence Simplification

Input Sentence:
ASC takes a character as input, and returns the integer giving the ASCII code of the input character.

Desired question:
(a) What does ASC take as input?
(b) What does ASC return?

Actual questions that could have been generated from MRS transformation:
(c) What does ASC take as input and returns the integer giving the ASCII code of the input character?
(d) ASC takes a character as input and returns what giving the ASCII code of the input character?
MRS Decomposition

Complex Sentences -> Simple Sentences

1. Plain text
2. Term extraction
3. FSC construction
4. MRS Decomposition
   - Apposition Decomposer
   - Coordination Decomposer
   - Subclause Decomposer
   - Subordinate Decomposer
   - Why Decomposer
5. MRS Transformation
6. Generation with LKB
7. Output selection
8. Output to console/XML
Coordination Decomposer

“John likes cats very much but hates dogs a lot.”
Coordination Decomposer

left: “John likes cats very much.”  right:  "John hates dogs a lot."

```
_very+much_a_1
  ARG1/EQ
  _like_v_1
  ARG2/NEQ
  _cat_n_1
    RSTR/H
    undef_q

_hate_v_1
  ARG1/NEQ
  named(”John”)
  ARG2/NEQ
  _dog_n_1
    RSTR/H
    udef_q
```

```
_a+lot_a_1
  ARG1/EQ

[arg1/eq]
  _hate_v_1
  ARG1/NEQ
  named(”John”)
  ARG2/NEQ
  _dog_n_1
    RSTR/H
    udef_q
```

```
[arg1/eq]
  _like_v_1
  ARG1/NEQ
  _cat_n_1
    RSTR/H
    undef_q
```

Subclause Decomposer
identifies the verb, extracts its arguments and reconstructs MRS

(a): Bart is the cat that chases the dog.

(b): Bart is the cat.

(c): The cat chases the dog.
Connected Dependency MRS Graph

**Connected DMRS Graph**

A Connected DMRS Graph is a tuple $G = (N, E, L, S_{pre}, S_{post})$ of:

- a set $N$, whose elements are called *nodes*;
- a set $E$ of connected pairs of vertices, called *edges*;
- a function $L$ that returns the associated label for edges in $E$;
- a set $S_{pre}$ of pre-slash labels and a set $S_{post}$ of post-slash labels.

Specifically,

- $N$ is a set of all Elementary Predications (EPs) defined in a grammar;
- $S_{pre}$ contains all pre-slash labels, namely $\{\text{ARG}^*, \text{RSTR}, \text{L-INDEX}, \text{R-INDEX}, \text{L-HNDL}, \text{R-HNDL}, \text{NULL}\}$;
- $S_{post}$ contains all post-slash labels, namely $\{\text{EQ}, \text{NEQ}, \text{H}, \text{HEQ}, \text{NULL}\}$;
- $L$ is defined as: $L(x, y) = [\text{pre}/\text{post}, \ldots]$. For every node $x, y \in N$, $L$ returns a list of pairs $\text{pre}/\text{post}$ that $\text{pre} \in S_{pre}, \text{post} \in S_{post}$. If $\text{pre} \neq \text{NULL}$, then the edge between $(x, y)$ is directed: $x$ is the governor, $y$ is the dependant; otherwise the edge between $x$ and $y$ is not directed. If $\text{post} = \text{NULL}$, then $y = \text{NULL}$, $x$ has no dependant by a $\text{pre}$ relation.
## Generic Decomposing Algorithm

```plaintext
function decompose(rEPS, eEPS, relaxEQ = 1, keepEQ = 1)

parameters:
- rEPS: a set of eps for which we want to find related eps.
- eEPS: a set of exception eps.
- relaxEQ: a boolean value of whether to relax the post-slash value from EQ to NEQ for verbs and prepositions (optional, default:1).
- keepEQ: a boolean value of whether to keep verbs and prepositions with a post-slash EQ value (optional, default:1).

returns:
- a set of eps that are related to rEPS

; assuming concurrent modification of a set is permitted in a for loop

aEPS ← the set of all eps in the DMRS graph
retEPS ← ∅ ;; initialize an empty set

for tEP ∈ rEPS and tEP /∈ eEPS do
    for ep ∈ aEPS and ep /∈ eEPS and ep /∈ rEPS do
        pre/post ← L(tEP, ep) ;; ep is the dependant of tEP
        if pre ≠ NULL then ;; ep exists
            if relaxEQ and post = EQ and (tEP is a verb ep or (tEP is a preposition ep and pre = ARG2)) then
                assign ep a new label and change its qeq relation accordingly
            end if
            retEPS.add(ep)  
aEPS.remove(ep)
        end if

        pre/post ← L(ep, tEP) ;; ep is the governor of tEP
        if pre ≠ NULL then ;; ep exists
            if keepEQ = 0 and ep is a (verb ep or preposition ep) and post = EQ and ep has no empty ARG* then
                continue ;; continue the loop without going further below
            end if
            retEPS.add(ep)  
aEPS.remove(ep)
        end if
    end for
end for

if retEPS ≠ ∅ then
    return rEPS ∪ decompose(retEPS, eEPS, relaxEQ = 0) ;; the union of two
else
    return rEPS
end if
```

---

**Generic Decomposing Algorithm**

```plaintext
function decompose(rEPS, eEPS, relaxEQ = 1, keepEQ = 1)

parameters:
- rEPS: a set of eps for which we want to find related eps.
- eEPS: a set of exception eps.
- relaxEQ: a boolean value of whether to relax the post-slash value from EQ to NEQ for verbs and prepositions (optional, default:1).
- keepEQ: a boolean value of whether to keep verbs and prepositions with a post-slash EQ value (optional, default:1).

returns:
- a set of eps that are related to rEPS

; assuming concurrent modification of a set is permitted in a for loop

aEPS ← the set of all eps in the DMRS graph
retEPS ← ∅ ;; initialize an empty set

for tEP ∈ rEPS and tEP /∈ eEPS do
    for ep ∈ aEPS and ep /∈ eEPS and ep /∈ rEPS do
        pre/post ← L(tEP, ep) ;; ep is the dependant of tEP
        if pre ≠ NULL then ;; ep exists
            if relaxEQ and post = EQ and (tEP is a verb ep or (tEP is a preposition ep and pre = ARG2)) then
                assign ep a new label and change its qeq relation accordingly
            end if
            retEPS.add(ep)  
aEPS.remove(ep)
        end if

        pre/post ← L(ep, tEP) ;; ep is the governor of tEP
        if pre ≠ NULL then ;; ep exists
            if keepEQ = 0 and ep is a (verb ep or preposition ep) and post = EQ and ep has no empty ARG* then
                continue ;; continue the loop without going further below
            end if
            retEPS.add(ep)  
aEPS.remove(ep)
        end if
    end for
end for

if retEPS ≠ ∅ then
    return rEPS ∪ decompose(retEPS, eEPS, relaxEQ = 0) ;; the union of two
else
    return rEPS
end if
```
English Sentence Structure
and corresponding decomposers

Complex
- dependent clause + independent clause

Compound
- coordination of sentences

Simple
- independent & simple clause

Subordinate Clause
- Causal | Non-causal

Relative Clause

Coordination of phrases

Apposition

Others

Decomposer Pool

Decomposed Sentence
Outline

Introduction
- Definition
- Usage
- Template/Syntax/Semantics-based Approaches

Background
- MRS/ERG/PET/LKB

System Architecture
- Overview
- MRS Transformation for Simple Sentences
- MRS Decomposition for Complex Sentences
- Question Reranking

Evaluation
- QGSTEC 2010
The Problem

Will the wedding be held next Monday?

Unranked Realizations from LKB for "Will the wedding be held next Monday?"

Next Monday the wedding will be held?
Next Monday will the wedding be held?
Next Monday, the wedding will be held?
Next Monday, will the wedding be held?
The wedding will be held next Monday?
Will the wedding be held next Monday?
## Question Ranking

### Will the wedding be held next Monday?

<table>
<thead>
<tr>
<th>MaxEnt Model for Declaratives</th>
<th>Language Model for Interrogatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.31</td>
<td>Next Monday will the wedding be held?</td>
</tr>
<tr>
<td>1.63</td>
<td>Will the wedding be held next Monday?</td>
</tr>
<tr>
<td>1.35</td>
<td>Next Monday the wedding will be held?</td>
</tr>
<tr>
<td>1.14</td>
<td>Will the wedding be held next Monday?</td>
</tr>
<tr>
<td>0.77</td>
<td>Next Monday, the wedding will be held?</td>
</tr>
<tr>
<td>0.51</td>
<td>Next Monday will the wedding be held?</td>
</tr>
<tr>
<td>0.29</td>
<td>Next Monday, will the wedding be held?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Combined Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.78</td>
</tr>
<tr>
<td>1.64</td>
</tr>
<tr>
<td>1.44</td>
</tr>
<tr>
<td>1.11</td>
</tr>
<tr>
<td>0.81</td>
</tr>
<tr>
<td>0.76</td>
</tr>
<tr>
<td>0.48</td>
</tr>
</tbody>
</table>
Review of System Architecture

1. Term extraction
2. FSC construction
3. Parsing with PET
4. MRS Decomposition
   - Apposition Decomposer
   - Coordination Decomposer
   - Subclause Decomposer
   - Subordinate Decomposer
   - Why Decomposer
5. MRS Transformation
   - MRS to XML
6. Generation with LKB
7. Output selection
8. Output to console/XML
Outline

Introduction
Definition
Usage
Template/Syntax/Semantics-based Approaches

Background
MRS/ERG/PET/LKB

System Architecture
Overview
MRS Transformation for Simple Sentences
MRS Decomposition for Complex Sentences
Question Reranking

Evaluation
QGSTEC 2010
QGSTEC2010
The Question Generation Shared Task and Evaluation Challenge (QGSTEC) 2010

Task B: QG from Sentences.
Participants are given one complete sentence from which their system must generate questions.

1. **Relevance.** Questions should be relevant to the input sentence.
2. **Question type.** Questions should be of the specified target question type.
3. **Syntactic correctness and fluency.** The syntactic correctness is rated to ensure systems can generate sensible output.
4. **Ambiguity.** The question should make sense when asked more or less out of the blue.
5. **Variety.** Pairs of questions in answer to a single input are evaluated on how different they are from each other.
QGSTEC2010
The Question Generation Shared Task and Evaluation Challenge (QGSTEC) 2010

Task B: QG from Sentences.
Participants are given one complete sentence from which their system must generate questions.

1. **Relevance.** Questions should be relevant to the input sentence.
2. **Question type.** Questions should be of the specified target question type.
3. **Syntactic correctness and fluency.** The syntactic correctness is rated to ensure systems can generate sensible output.
4. **Ambiguity.** The question should make sense when asked more or less out of the blue.
5. **Variety.** Pairs of questions in answer to a single input are evaluated on how different they are from each other.
Test Set

- 360 questions were required to be generated from 90 sentences
- 8 question types: yes/no, which, what, when, how many, where, why and who.

<table>
<thead>
<tr>
<th></th>
<th>Wikipedia</th>
<th>OpenLearn</th>
<th>YahooAnswers</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>sentence count</td>
<td>27</td>
<td>28</td>
<td>35</td>
<td>90</td>
</tr>
<tr>
<td>average length</td>
<td>22.11</td>
<td>20.43</td>
<td>15.97</td>
<td>19.20</td>
</tr>
<tr>
<td>question count</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>360</td>
</tr>
</tbody>
</table>
Participants

- Lethbridge, syntax-based, University of Lethbridge, Canada
- MrsQG, semantics-based, Saarland University, Germany
- JUQGG, rule-based, Jadavpur University, India.
- WLV, syntax-based, University of Wolverhampton, UK
Evaluation Grades

Results per criterion without penalty on missing questions

- Relevance
- Question Type
- Correctness
- Ambiguity
- Variety

Results for:
- WLV
- MrsQG
- JUQGG
- Lethbridge
- Worst
Generation Coverage

Coverages on input and output
(generating 360 questions from 90 sentences)

MrsQG: 100.00%
WLV: 90.00%
JUQGG: 80.00%
Lethbridge: 70.00%

sentences: blue
questions: red
### Evaluation Grades

with penalty on missing questions

<table>
<thead>
<tr>
<th></th>
<th>MrsQG</th>
<th>WLV</th>
<th>JUQGG</th>
<th>Lethbridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevance</td>
<td>3.35</td>
<td>3.9</td>
<td>3.85</td>
<td>3.85</td>
</tr>
<tr>
<td>Question Type</td>
<td>2.75</td>
<td>2.6</td>
<td>2.85</td>
<td>2.85</td>
</tr>
<tr>
<td>Correctness</td>
<td>3.9</td>
<td>3.9</td>
<td>3.9</td>
<td>3.85</td>
</tr>
<tr>
<td>Ambiguity</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.75</td>
</tr>
<tr>
<td>Variety</td>
<td>2.75</td>
<td>2.7</td>
<td>2.75</td>
<td>2.75</td>
</tr>
</tbody>
</table>

results per criterion with penalty on missing questions

- **MrsQG**
- **WLV**
- **JUQGG**
- **Lethbridge**
- **Worst**
Evaluation Grades per Question Type

Performance of MrsQG per Question Type

- Relevance
- Question Type
- Correctness
- Ambiguity
- Variety

Legend:
- yes/no(28)
- which(42)
- what(116)
- when(36)
- how many(44)
- where(28)
- why(30)
- who(30)
- worst(354)
Outline

Introduction
  Definition
  Usage
  Template/Syntax/Semantics-based Approaches

Background
  MRS/ERG/PET/LKB

System Architecture
  Overview
  MRS Transformation for Simple Sentences
  MRS Decomposition for Complex Sentences
  Question Reranking

Evaluation
  QGSTEC 2010
Back to our one-line abstract

John plays football \(\rightarrow\) play(John, football) \(\rightarrow\) play(who, which sports) \(\rightarrow\)
Who plays which sports?

Question Generation

NLU

Natural Language Text

\(\textbf{Simplification}\)

Symbol Representation for Text

NLG

\(\textbf{Ranking}\)

Natural Language Questions

\(\textbf{Transformation}\)

Symbol Representation for Questions
Conclusion

- semantics-based (easy in theory, difficult in practice)
  - multi-linguality
  - cross-domain
- deep grammar (worry less, wait more)
  - generation \(<-\>\) grammaticality
  - heavy machinery
Demo?

and never forget
Fourier Transform:

\[
X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}
\]