Translating Natural Language to SPARQL

Xiaoyu Yin

TU Dresden

10 December 2018

Outline

Background

Project Structure

Experiment & Analysis

Future Work

Summary

Outline

Background

- Math Word Problem Generation
- Question Generation

Project Structure

- Preprocessing
- Question Generating
- Postprocessing

Experiment & Analysis

- Implementation
- Experiment
- Results and Analysis

Future Work

Summary

Math Word Problem Generation

Math Word Problem

a word problem is a mathematical exercise where significant background information on the problem is presented as text rather than in mathematical notation.

Example (Math Word Problem)

John has 4 apples. Jack has 5 apples. How many apples do they have in total?

Math Word Problem Generation

Existing methods:

- requirements ⇒ problems meeting requirements
- problem and themes ⇒ problem in new theme
- ...

Our method (ideal):

- problem ⇒ more problems

Question Generation

Example (Math Word Problem)

John has 4 apples. Jack has 5 apples. How many apples do they have in total?

⇓

John and Jack have 9 apples in total. Jack has 5 apples. How many apples does **John** have?

⇓

John and Jack have 9 apples in total. John has 4 apples. How many apples does **Jack** have?

Our goal

Generate a nearly new problem with **no extra input**

Question Generation

Example (Math Word Problem)

John has 4 apples. Jack has 5 apples. How many apples do they have in total?

⇓

John and Jack have 9 apples in total. Jack has 5 apples. How many apples does **John** have?

⇓

John and Jack have 9 apples in total. John has 4 apples. How many apples does **Jack** have?

Our goal

Generate a nearly new problem with **no extra input**

Question Generation

Example (Math Word Problem)

John has 4 apples. Jack has 5 apples. How many apples do they have in total?

⇓

John and Jack have 9 apples in total. Jack has 5 apples. How many apples does **John** have?

⇓

John and Jack have 9 apples in total. John has 4 apples. How many apples does **Jack** have?

Our goal

Generate a nearly new problem with **no extra input**

Question Generation

Example (Math Word Problem)

John has 4 apples. Jack has 5 apples. How many apples do they have in total?

⇓

John and Jack have 9 apples in total. Jack has 5 apples. How many apples does **John** have?

⇓

John and Jack have 9 apples in total. John has 4 apples. How many apples does **Jack** have?

Our goal

Generate a nearly new problem with **no extra input**

Question Generation

General QG

Generating different kinds of questions from a given sentence or paragraph. (like who, when, why, etc.)

In this project

Only a subset of question types is needed.

Example

John has 4 apples.

✗ Who has 4 apples?

✗ Why does John have 4 apples?

✓ How many apples does John have?

Question Generation

General QG

Generating different kinds of questions from a given sentence or paragraph. (like who, when, why, etc.)

In this project

Only a subset of question types is needed.

Example

John has 4 apples.

✗ Who has 4 apples?

✗ Why does John have 4 apples?

✓ How many apples does John have?

Outline

- Background
  - Math Word Problem Generation
  - Question Generation
- Project Structure
  - Preprocessing
  - Question Generating
  - Postprocessing
- Experiment & Analysis
  - Implementation
  - Experiment
  - Results and Analysis
- Future Work
- Summary

Old math word problem

⇓

1. Preprocessing
2. Question Generating
3. Postprocessing

⇓

New math word problem

Preprocessing

Prepare the original math word problem input for the other tasks

Steps

1. Text Simplification
2. Pronouns Removal
3. Sentence Filtering
4. ...

Text simplification

Preprocessing

Keep sentences in simple form, important for question generation

Example

Input: John has 4 apples and Jack has 5 apples.
Output: John has 4 apples. Jack has 5 apples.

Avoid such question:

✗ John and Jack have 9 apples in total. How many apples does John have and How many apples does Jack have?

Method adopted from *[HS10]*

Text simplification

Preprocessing

Keep sentences in simple form, important for question generation

Example

Input: John has 4 apples and Jack has 5 apples.
Output: John has 4 apples. Jack has 5 apples.

Avoid such question:

✗ John and Jack have 9 apples in total. How many apples does John have and How many apples does Jack have?

Method adopted from *[HS10]*

Pronouns removal

Preprocessing

Example

Input: John has 4 apples. **He** took 3 apples from Jack.
Output: John has 4 apples. **John** took 3 apples from Jack.

Disambiguate the generated question

✗ How many apples did **he** take from Jack?
✓ How many apples did **John** take from Jack?

Acheived by using *Coreference Resolution*

Pronouns removal

Preprocessing

Example

Input: John has 4 apples. **He** took 3 apples from Jack.
Output: John has 4 apples. **John** took 3 apples from Jack.

Disambiguate the generated question

✗ How many apples did **he** take from Jack?
✓ How many apples did **John** take from Jack?

Acheived by using *Coreference Resolution*

Sentence Filtering

Preprocessing

Example

John had 4 apples. John gave Jack some of his apples. John has 1 apple now. John gave Jack 3 apples.

Find all the sentences that do not have numbers.

Achieved by using *Named Entity Recogniton*

Preprocessing

Other Steps

- Sentence Splitting
  not trival as it looks!
- Question Answering
  - Replaced with manual answering

Example

John spent 5.20 dollars.

Preprocessing

Other Steps

- Sentence Splitting
  not trival as it looks!
- Question Answering
  - Replaced with manual answering

Example

John has 4 apples. Jack has 5 apples.
How many apples do they have in total?

Question Generating

Detect components of a sentence and raise a question about the numeric part.

Rule-based detection and template-based question generation adopted from [DRMD16]

- Clause detection rule
  - find **subject phrases** and **verb phrases**
  - find **auxiliary word** for the template
- Numeric Phrase detection rule
  - find **numerical phrases**
- Question template of *How-many*

Achieved by using *Part-Of-Speech tags pattern matching*

Question Generating

Detect components of a sentence and raise a question about the numeric part.

Rule-based detection and template-based question generation adopted from [DRMD16]

- Clause detection rule
  - find **subject phrases** and **verb phrases**
  - find **auxiliary word** for the template
- Numeric Phrase detection rule
  - find **numerical phrases**
- Question template of *How-many*

Achieved by using *Part-Of-Speech tags pattern matching*

Question Generating

- Clause detection rule (a *POS tag pattern*)
  $\{< DT >? < JJ.? > * < NN.?|PRP|PRP\$|POS|IN|DT|CC|VBG|VBN > + < RB.?|VB.?|MD|RP > +\}$

Example

John(**NNP**) has(**VBZ**) 4(CD) apples(NNS) .

Clause John has

Subject John

Verb has

Auxiliary word does

Numeric 4 apples

Question How many apples does John have?

Question Generating

- Numeric Phrase detection rule (a *POS tag pattern*) $\{<DT>?<CD>+<RB>?<JJ|JJR|JJS>?<NN|NNS|NNP|NNPS|VBG>+\}$

Example

John(NNP) has(VBZ) 4(**CD**) apples(**NNS**) .

Clause John has

Subject John

Verb has

Auxiliary word does

Numeric 4 apples

Question How many apples does John have?

Question Generating

- Question template of *How-many*
  - "How many" + part of the numeric phrase excluding cardinal number + auxiliary word + subject part + remaining words of verb phrase + remaining part of the sentence as it is + "?"

Example

John(NNP) has(VBZ) 4(CD) apples(NNS) .

Clause John has

Subject John

Verb has

Auxiliary word does

Numeric 4 apples

Question How many apples does John have?

Postprocessing

- Combine questions and sentences to form a new math word problem
- Reorder sentences if necessary (pronouns removal failed)

Example

Input: He(**?**) gave Jack some of his apples. John had 4 apples. He has 1 apple now. How many apples did he give to Jack?

Move the underlined sentence to the first position

Output: John had 4 apples. He(**John**) gave Jack some of his apples. He has 1 apple now. How many apples did he give to Jack?

Outline

Background

- Math Word Problem Generation
- Question Generation

Project Structure

- Preprocessing
- Question Generating
- Postprocessing

Experiment & Analysis

- Implementation
- Experiment
- Results and Analysis

Future Work

Summary

Implementation[1]

- Kotlin and Java programming language
- Stanford CoreNLP v3.8.0
  - Stanford Named Entity Recognizer
  - Stanford Coreference Resolution
  - Stanford POS Tagger
- Intellij Idea IDE
- macOS High Sierra 10.13.3

[1] source code available at: *this link*

Experiment

Data

20 math word problems were selected from MAWPS[1]

Selection criteria:

- 10 simple problems + 10 complex problems
- equation contains only addition and subtraction
- only problems with how-many questions

The question in each problem was manually transformed into answer form

Simple problem

average number of words in each sentence ≤ 8

[1] available at: http://lang.ee.washington.edu/MAWPS/

Experiment

Example (Original data)

simple	Keith has 20 books . Jason has 21 books . How many books do they have together ? — ["41"] *(average 6 tokens per sentence)*
complex	Mary picked 122 oranges and Jason picked 105 oranges from the orange tree . How many oranges were picked in total ? — ["227"] *(average 11 tokens per sentence)*

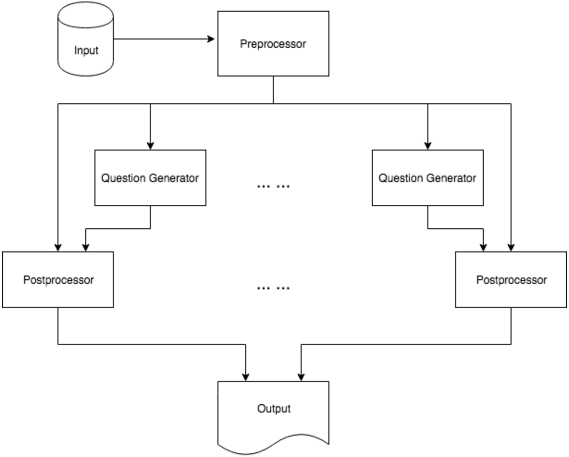
Experiment

Example (After manual transformation)

simple Keith has 20 books. Jason has 21 books. **They have 41 books together.**

complex Mary picked 122 oranges and Jason picked 105 oranges from the orange tree. **227 oranges were picked in total.**

Experiment


Input
Preprocessor
Question Generator
... ...
Question Generator
Postprocessor
... ...
Postprocessor
Output

Results and Analysis I

	Failure	Success	Problems generated
Simple problems	0	10	22
Complex problems	3	7	14

- The system performs worse when facing with more complex problems

Results and Analysis II

Known limitation

- Support only one question type
  - Generate similiar questions (some of them with grammar errors)
  - Unable to deal with multiple numbers.
    e.g. John spent 5 dollars on 4 apples.
- Performance was limited by how far the library can do
  i.e. If the library is not able to split the sentences, detect the tags, or resolve the coreference correctly, then the system would fail

Results and Analysis III

Potential limitation

- Can't identify **distraction**
  e.g. irrelevant numerical information

Outline

Background

- Math Word Problem Generation
- Question Generation

Project Structure

- Preprocessing
- Question Generating
- Postprocessing

Experiment & Analysis

- Implementation
- Experiment
- Results and Analysis

Future Work

Summary

Future Work

- Implement **Question Answering** in preprocessing to reduce the human efforts testing large data
- Extend the **Rules and Templates** in question generation phase
  Enhance the ability to deal with variation of sentences
- Add **Understanding module** to avoid distractions

Outline

- Background
  - Math Word Problem Generation
  - Question Generation
- Project Structure
  - Preprocessing
  - Question Generating
  - Postprocessing
- Experiment & Analysis
  - Implementation
  - Experiment
  - Results and Analysis
- Future Work
- Summary

Summary

- Related fields
    - Math word problem generation
    - Question generation
- Three primary tasks
    - Preprocessing
    - Question Generating
    - Postprocessing
- Implementation and Experiment
    - NLP techniques used
    - 20 problems test
    - Limitations
- Future Work
    - Question Answering
    - More rules and templates

Reference

- Rubel Das, Antariksha Ray, Souvik Mondal, and Dipankar Das, *A rule based question generation framework to deal with simple and complex sentences*, 2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016 (2016), 542–548.

- M Heilman and N. Smith, *Extracting simplified statements for factual question generation*, . . . the 3rd Workshop on Question Generation (2010), 11–20.