Mapping Natural Language to Description Logic

Bikash Gyawali  Anastasia Shimorina  Claire Gardent  Samuel Cruz-Lara  Mariem Mahfoudh

CNRS/LORIA, Nancy, France

May 31, 2017
A reversible approach that:

- Maps Natural Language (NL) Sentences to Description Logic (DL) Axioms
- Generates Text to describe DL Axioms.
A reversible approach that:

- Maps Natural Language (NL) Sentences to Description Logic (DL) Axioms
- Generates Text to describe DL Axioms.
A reversible approach that:

- Maps Natural Language (NL) Sentences to Description Logic (DL) Axioms
- Generates Text to describe DL Axioms.
Map System Installation Design Principle (SIDP) text to Ontology Axioms.

- Airbus Industry.

**SIDP Text**

Ex: Pipes shall be identified by Labels.

**Ontology Axiom**

\[
\text{Pipe} \sqsubseteq \exists \text{identificationArg}_2 \cdot (\text{Identification} \sqcap \exists \text{identificationArg}_3 \cdot \text{Label})
\]
Motivation

From Text to Model (Semantic Parsing):

- Semantic Reasoning on text.
- Knowledge Base (KB) enrichment.
- Manual mapping is difficult: time-consuming, expertise needed.
- Text keeps evolving: Consistency of newly updated SIDPs with existing ones.

From Model to Text (Generation):

- Easy comprehension of complex axioms.
- Verification of Parsing results.
Outline of this Talk

▶ Related Work

▶ Contributions

▶ Approach Overview – Resources and Methodologies

▶ Experiments – Results and Evaluation

▶ Conclusion
Related Work

Semantic Parsing:


Ontology Learning:

  - Identify new concepts, instances and taxonomy of concepts.
  - Identify new properties and their values for instances.
  - No processing of sentence level axioms.
Related Work

Generation:


Contributions

- Derive complex DL Axioms from Natural Language Sentences.
- Regeneration as a measure of Semantic Parse accuracy.
- Ontology Enrichment using derived Axioms
- Reversible (Semantic Parser – Generator) and Robust Framework.
Approach Overview

Input
SIDPs

Semantic Parser

Grammar (Manual)
Lexicon (Automatic)

OWL Axioms

Generated SIDPs

BLEU Scoring

Generator

Ontology Enrichment
Resources (Grammar and Lexicon)

Handcrafted Grammar: FB-LTAG with Unification Based Semantics.

```
S
  NP↓  VP
    NP  AUX  AUX  V
        Pipes  shall  be  used

AUX
  ADV  AUX
    not
```

L6: \( \text{Pipe}(X) \)  \( \text{L0: subset}(Y,L1) \)
L2: \( \exists \text{useArg2inv},L3 \)
L3: \( \text{Use}(Z) \)
L4: \( \neg(L5) \)
Handcrafted Grammar : FB-LTAG with Unification Based Semantics.

\[
\begin{align*}
S &\quad NP \downarrow \quad VP \\
\quad NP &\quad AUX \quad AUX \quad V \\
\quad NP_{L6} &\quad Pipes \\
\quad AUX &\quad ADV \quad AUX \\
\quad ADV &\quad not
\end{align*}
\]

\[
\begin{align*}
S &\quad NP_{Y} \quad VP \\
\quad NP_{L6} &\quad AUX \quad AUX_{L1} \quad V \\
\quad Pipes &\quad shall \quad be \quad used \\
\quad L6: \pipe(X) &\quad L0: \subset(Y,L1) \\
\quad L2: \exists(useArg2inv,L3) &\quad L3: Use(Z) \\
\quad AUX_{L4} &\quad ADV \quad AUX_{L5} \\
\quad ADV &\quad not \\
\quad L4: \neg(L5)
\end{align*}
\]
Three main steps

1. Select grammar trees based on input (words or semantic literals)

2. Combine selected trees using adjunction and substitution

3. Extract solutions (semantic representations or sentence)
**Parsing and Generation**

**Parsing**: 
pipes shall not be used. 
pipes shall be used.
Pipes shall not be used.

Pipes shall be used.
 Parsing and Generation

Parsing:

Pipes shall not be used.  
L6: Pipe(X)  
L0: subset(L6, L1)  
L5: exists(useArg2inv, L3)  
L3: Use(Z)

Pipes shall be used.
Pipes shall not be used.

L6: Pipe(X)  L0: subset(L6,L1)  L5: exists(useArg2inv,L3)  L3: Use(Z)  L1: not(L5)

Pipes shall be used.
Parsing and Generation

Parsing:

Pipes shall not be used.

L6: Pipe(X)
L0: subset(L6, L1)
L5: exists(useArg2inv, L3)
L3: Use(Z)
L1: not(L5)

Pipes shall be used.

L6: Pipe(X)
L0: subset(L6, L5)
L5: exists(useArg2inv, L3)
L3: Use(Z)
Parsing and Generation

**Full Parse**

\[
\begin{align*}
S & \quad NP_{L6} \quad VP \\
& \quad NP_{L6} \quad AUX \quad AUX_{L1} \quad V \\
& \quad AUX \quad be \quad used \\
& \quad ADV \quad AUX_{L5} \\
& \quad L6: \text{Pipe}(X) \quad L0: \text{subset}(Y,L1) \\
& \quad L2: \text{exists}(useArg2inv,L3) \\
& \quad L3: \text{Use}(Z) \\
& \quad L4: \text{not}(L5) \\
\end{align*}
\]

**Parsing:**

Pipes shall not be used.  
\[
L6:\text{Pipe}(X) \quad L0:\text{subset}(L6,L1) \quad L5:\text{exists}(useArg2inv,L3) \quad L3:\text{Use}(Z) \quad L1:\text{not}(L5)
\]

Pipes shall be used.  
\[
L6:\text{Pipe}(X) \quad L0:\text{subset}(L6,L5) \quad L5:\text{exists}(useArg2inv,L3) \quad L3:\text{Use}(Z)
\]
Parsing and Generation

Parsing:

Pipes shall not be used.

L6: Pipe(X) L0: subset(Y,L1) L5: exists(useArg2inv,L3) L3: Use(Z) L1: not(L5)

Pipes shall be used.

L6: Pipe(X) L0: subset(L6,L5) L5: exists(useArg2inv,L3) L3: Use(Z)

Pipes shall not be used.
Parsing and Generation

Parsing:

Pipes shall not be used.

L6: Pipe(X)  L0: subset(Y, L1)  L5: exists(useArg2inv, L3)  L3: Use(Z)  L1: not(L5)

Pipes shall be used.

L6: Pipe(X)  L0: subset(L6, L5)  L5: exists(useArg2inv, L3)  L3: Use(Z)

Pipes shall not be used.
**Parsing and Generation**

Parsing:

Pipes shall not be used.  
L6: Pipe(X)  L0: subset(Y,L1)  L5: exists(useArg2inv,L3)  L3: Use(Z)  L1: not(L5)

Pipes shall be used.  
L6: Pipe(X)  L0: subset(L6,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)

Pipes shall not be used.
 Parsing :

Pipes shall not be used.  L6: Pipe(X)  L0: subset(Y,L1)  L5: exists(useArg2inv,L3)  L3: Use(Z)  L1: not(L5)
Pipes shall be used.  L6: Pipe(X)  L0: subset(Y,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)
Pipes shall not be used.  L6: Pipe(X)  L0: subset(Y,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)
### Parsing and Generation

**Parsing:**

Pipes shall not be used.

Pipes shall be used.

Pipes shall not be used.

```
L6: Pipe(X)  L0: subset(Y,L1)  L5: exists(useArg2inv,L3)  L3: Use(Z)  L1: not(L5)
L6: Pipe(X)  L0: subset(L6,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)
L6: Pipe(X)  L0: subset(L6,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)  Partial Parse
```
Parsing and Generation

Parsing:

Pipes shall not be used.

L6: Pipe(X)  L0: subset(L6,L1)  L5: exists(useArg2inv,L3)  L3: Use(Z)  L1: not(L5)

Pipes shall be used.

L6: Pipe(X)  L0: subset(L6,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)

Pipes shall not be used.

L6: Pipe(X)  L0: subset(L6,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)

Generation:

L6: Pipe(X)  L0: subset(L6,L1)  L5: exists(useArg2inv,L3)  L3: Use(Z)  L1: not(L5)
Pipes shall not be used.

L6: Pipe(X)  L0: subset(Y,L1)  L5: exists(useArg2inv,L3)  L3: Use(Z)  L1: not(L5)

Pipes shall be used.

L6: Pipe(X)  L0: subset(L6,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)

Pipes shall not be used.

L6: Pipe(X)  L0: subset(L6,L5)  L5: exists(useArg2inv,L3)  L3: Use(Z)
Parsing and Generation

 Parsing:

Pipes shall not be used.  L6:Pipe(X)  L0:subset(L6,L1)  L5:exists(useArg2inv,L3)  L3:Use(Z)  L1:not(L5)
Pipes shall be used.    L6:Pipe(X)  L0:subset(L6,L5)  L5:exists(useArg2inv,L3)  L3:Use(Z)
Pipes shall not be used.  L6:Pipe(X)  L0:subset(L6,L5)  L5:exists(useArg2inv,L3)  L3:Use(Z)

Generation:

L6:Pipe(X)  L0:subset(L6,L1)  L5:exists(useArg2inv,L3)  L3:Use(Z)  L1:not(L5)
Parsing:

Pipes shall not be used.  
L6: Pipe(X) L0: subset(Y,L1) L5: exists(useArg2inv,L3) L3: Use(Z) L1: not(L5)

Pipes shall be used.  
L6: Pipe(X) L0: subset(L6,L5) L5: exists(useArg2inv,L3) L3: Use(Z)

Pipes shall not be used.  
L6: Pipe(X) L0: subset(L6,L1) L5: exists(useArg2inv,L3) L3: Use(Z) L1: not(L5)

Generation:

L6:Pipe(X) L0:subset(L6,L1) L5:exists(useArg2inv,L3) L3:Use(Z) L1:not(L5)
Conversion Rules

To transform a Flat Semantics formula, $\phi$, to its equivalent DL formula, $\tau(\phi)$:

$$
\tau(\phi) = \begin{cases} 
\text{ObjectSomeValuesFrom}(\text{:R } \tau(C)) & \text{if } \phi = l_i : \exists(R, l_j) \  l_j : C \\
\text{SubClassOf}(\tau(C_1) \ \tau(C_2)) & \text{if } \phi = l_i : \text{subset}(l_j, l_k) \  l_j : C_1 \ l_k : C_2 \\
\text{ObjectIntersectionOf}(\tau(C_1) \ \tau(C_2)) & \text{if } \phi = l_i : \text{and}(l_j, l_k) \  l_j : C_1 \ l_k : C_2 \\
(\tau(C_1) \ \cap \ \tau(C_2)) & \text{if } \phi = l_i : \text{and}(l_j, l_k) \  l_j : C_1 \ l_k : C_2 \\
(\tau(C_1) \ \cup \ \tau(C_2)) & \text{if } \phi = l_i : \text{or}(l_j, l_k) \  l_j : C_1 \ l_k : C_2 \\
\text{not}(\tau(C)) & \text{if } \phi = l_i : \text{not}(l_j) \  l_j : C \\
\text{R}^{-} & \text{if } \phi = R_{\text{inv}} \\
C & \text{if } \phi = l_i : C(x)
\end{cases}
$$

where

- $l_i$ are labels
- $C_i$ are arbitrarily complex DL concepts
- $R$ are DL roles
960 SIDP sentences split into 2 categories:

- **456 Simple SIDPs**: Main clause only.
  
  Eg: *Pipes shall be identified by labels.*

- **504 Complex SIDPs**: More than one clause.
  
  Eg: *Object shall be qualified for continuous fuel immersion when installed inside fuel tank.*
Experiment and Results

- Coverage and Robustness of the Semantic Parsing Module.

- Correctness of the derived DL formulae
  - Syntactic Correctness
  - Semantic Correctness

- Impact on the Ontology Learning Task
% of Sentences in each category (Simple, Complex and All) that could be parsed.

- **Simple SIDPS**
  - Full: 63.60%
  - Partial: 34.00%
  - Failure: 2.40%

- **Complex SIDPS**
  - Full: 87.90%
  - Partial: 9.52%
  - Failure: 2.58%

- **All SIDPS**
  - Full: 76.35%
  - Partial: 21.15%
  - Failure: 2.50%
% of Parse outputs in each category (Full, Partial and All) that are valid DL formula.

- Full Parse
  - Well-Formed: 100%
  - Ill-Formed: 0%

- Partial Parse
  - Well-Formed: 94.8%
  - Ill-Formed: 5.2%

- All Parse
  - Well-Formed: 96%
  - Ill-Formed: 4%
Semantic Correctness (Full Parses)

% of Regenerated Sentence classified into BLEU categories:

- **Low**: BLEU ≤ 32%
- **Medium**: 33% ≥ BLEU ≤ 66%
- **High**: BLEU ≥ 67%

![Graph showing BLEU scores for Simple-SIDPs and Complex-SIDPs]
Semantic correctness (Partial parses)

- Low: 66.82%
- Medium: 42.07%
- High: 21.72%

**BLEU Scores**

- Simple-SIDPs: 36.21%
- Complex-SIDPs: 23.02%

Simple-SIDPs
Complex-SIDPs
Impact on Ontology Learning

2 key steps for each axiom we derive:

- Add new Concepts and Relations found in the axiom.
- Ensure Consistency and Satisfiability before adding that axiom to the ontology.

Observations:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>New classes</td>
<td>935</td>
</tr>
<tr>
<td>Existing classes</td>
<td>89</td>
</tr>
<tr>
<td>New object properties</td>
<td>84</td>
</tr>
<tr>
<td>Existing object properties</td>
<td>0</td>
</tr>
<tr>
<td>superclasses found</td>
<td>498</td>
</tr>
<tr>
<td>RDFS-label matches found</td>
<td>7</td>
</tr>
<tr>
<td>new added SIDP formulae</td>
<td>798 (85.3%)</td>
</tr>
<tr>
<td>rejected SIDP formulae due to syntax errors</td>
<td>38 (4.0%)</td>
</tr>
<tr>
<td>rejected SIDP formulae due to redundancy</td>
<td>91 (9.7%)</td>
</tr>
<tr>
<td>rejected SIDP formulae due to inconsistency</td>
<td>9 (1.0%)</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

In Summary:

- Bridge between text and model.
- Reversability – Generation as a means of verifying Parsing.

Future Work:

- Use Reversability to build larger training corpus for Machine Learning.
- Learn the text-DL mapping using Deep Learning techniques (cf. Petrucci et al., 2016)
Thank You!
Grammar consists of Tree Schemas rather than the trees.
Lexical information is separately stored in a Lexicon.
Lexicon is automatically extracted.

Semantics:

\[ Rel = Use \]
\[ A2 = useArgument2\text{inv} \]
Tree: nx0V
Anchor: used
Coanchor: Aux1 \( \rightarrow \) shall/AUX
Coanchor: Aux2 \( \rightarrow \) be/AUX