Cross-lingual Semantic Parsing

Part III: Cross-lingual Learning of an Open-domain Semantic Parser

Kilian Evang

University of Düsseldorf
Semantic Parsing: What?

From Words to (Logical) Meaning

She likes to read books

\[
\begin{array}{c}
\text{x1} \ p1 \ e1 \\
\text{female(x1)} \\
\text{x2} \ e2 \\
\text{book.n.01(x2)} \\
\text{read.v.01(e2)} \\
\text{Agent(e2, x1)} \\
\text{Theme(e2, x2)} \\
\text{like.v.02(e1)} \\
\text{Experiencer(e1, x1)} \\
\text{Stimulus(e1, p1)} \\
\end{array}
\]

DRT Kamp (1984)
Semantic Parsing: Why?

Translate to something a computer can “understand”

- commands for robots, e.g., Dukes (2014)
- queries for databases, e.g., Reddy et al. (2014)
- formulas for (probabilistic) reasoners, e.g., Beltagy et al. (2015)
Semantic Parsing: How?

System for English (Curran et al., 2007; Bos, 2015)

She likes to read books

C&C tools

She

NP

likes

(S[dl]/NP)/(S[to]/NP)

NP

to

(S[to]/NP)/(S[b]/NP)

NP

read

(S[b]/NP)/NP

NP

books

NP

She

NP

likes

(S[dl]/NP)/(S[to]/NP)

NP

to

(S[to]/NP)/(S[b]/NP)

NP

read

(S[b]/NP)/NP

NP

books

NP

Boxer

x1 p1 e1

female(x1)

x2 e2

book.n.01(x2)

read.v.01(e2)

Agent(e2, x1)

Theme(e2, x2)

like.v.01(e1)

Experiencer(e1, x1)

Stimulus(e1, p1)
How Boxer Interprets CCG Derivations

She
NP
\( \lambda x_1. (x_1 \in \text{female}(x_1)) \)

likes
(S[cc]\(d\)NP)/(S[cc]\(d\)NP)
\( \lambda y_1. x_2. x_3. (x_2 @ x_3) \)

p1: (\( x_1 @ x_4 \), \( x_5 @ x_4 \)) @ \( x_6 \)

like(e1)
Time(e1, t1)
Stimulus(e1, p1)
Experimenter(e1, v4)
time(t1)
t1 = now

read
(S[cc]\(d\)NP)/(S[cc]\(d\)NP)
\( \lambda y_1. x_2. (x_1 @ x_3) \)

r1 = now

likes to read
S[cc]\(d\)NP
\( \lambda y_1. x_2. (x_1 @ x_3) \)

p1: e2
read(e2)
Agent(e2, v3)

like(e1)
Time(e1, t1)
Stimulus(e1, p1)
Experimenter(e1, v3)
time(t1)
t1 = now

She likes to read
S[cc]\(d\)NP
\( \lambda y_1. (x_1 @ x_3) \)

female(x1)
p1: e2
read(e2)
Agent(e2, x1)

like(e1)
Time(e1, t1)
Stimulus(e1, p1)
Experimenter(e1, v3)
time(t1)
t1 = now
Abbreviated Semantics

<table>
<thead>
<tr>
<th>NP</th>
<th>(S[dcl]\ NP)/(S[to]\ NP)</th>
<th>(S[to]\ NP)/(S[b]\ NP)</th>
<th>(S[b]\ NP)/NP</th>
<th>NP</th>
</tr>
</thead>
<tbody>
<tr>
<td>she'</td>
<td>likes</td>
<td>to</td>
<td>read</td>
<td>books</td>
</tr>
<tr>
<td></td>
<td>(S[dcl]\ NP)/(S[to]\ NP)</td>
<td>(S[to]\ NP)/(S[b]\ NP)</td>
<td>(S[b]\ NP)/NP</td>
<td>NP book'</td>
</tr>
<tr>
<td></td>
<td>like'</td>
<td>to'</td>
<td>read'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S[b]\ NP</td>
<td>book'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>read'@book'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S[to]\ NP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to'@(read'@book')</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S[dcl]\ NP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>like'@(to'@(read'@book'))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S[dcl]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(like'@(to'@(read'@book')))@she'</td>
<td></td>
</tr>
</tbody>
</table>
Learn (rudimentary) semantic parser from nothing but

- existing source language system (C&C+Boxer)
- parallel data
- unsupervised word aligner
- (POS tagger for target language)

Q: Can it be done?
Method

1. CCG category projection $\rightarrow$ target-language lexicon
2. CCG derivation projection $\rightarrow$ target-language training data
3. parser training $\rightarrow$ target-language semantic parser
Introduction

Category Projection

Derivation Projection

Translation Divergences

Parsing

Results
Ways to Generate Candidate Lexical Items

1. Zettlemoyer and Collins (2007) and much subsequent work: hand-written lexical templates
2. Kwiatkowksi et al. (2010): recursive splitting of gold-standard meaning representations
3. this work: projection from English parse trees to target-language words
### Parallel Corpus: Tatoeba.org

<table>
<thead>
<tr>
<th>Source</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>She likes to read books.</td>
<td>She liest gern Bücher.</td>
</tr>
<tr>
<td>Ši šatas legi librojn.</td>
<td>Elle aime lire des livres.</td>
</tr>
<tr>
<td>Szeret könyvet olvasni.</td>
<td>Le piace leggere libri.</td>
</tr>
<tr>
<td>Szeret könyveket olvasni.</td>
<td>A lei piace leggere libri.</td>
</tr>
<tr>
<td>彼女は本を読むのが好きだ。</td>
<td>泰语: 她喜歡讀書。</td>
</tr>
<tr>
<td>Ze leest graag boeken.</td>
<td>Ela gosta de ler livros.</td>
</tr>
<tr>
<td>Le gusta leer libros.</td>
<td>O kitap okumayı seviyor.</td>
</tr>
<tr>
<td>هیا اهوازت لکروآ سفریم.</td>
<td></td>
</tr>
</tbody>
</table>
Automatic Annotation of English Sentences

She ∼ NP ∼ (S[dcl]\ NP)/(S[to]\ NP) ∼ like′ ∼ (S[to]\ NP)/(S[b]\ NP) ∼ to′ ∼ (S[b]\ NP)/ NP ∼ read′ ∼ books

NP ∼ NP ∼ S[to]\ NP ∼ to′@(read′@book′) ∼ S[to]\ NP ∼ read′@book′ ∼ S[b]\ NP ∼ >0

NP ∼ S[dcl]\ NP ∼ like′@(to′@(read′@book′)) ∼ S[dcl] ∼ (like′@(to′@(read′@book′))}@she′ ∼ <0
Category Projection: How?

- automatically word-align English and target-language sentence
- assign categories, interpretations to target-language words (1:1, 1:2, 2:1)
- drop category subdistinctions (dcl, b, to...)
- use undirected slashes
Example

She

(S[dcl] \ NP)/(S[to] \ NP)

like'

(S[to] \ NP)/(S[b] \ NP)

to'

(S[b] \ NP)/NP

NP

read'

book'

She likes to read books

Ze leest graag boeken
Example

She (S[dcl] \ NP)/(S[to] \ NP)

likes (S[to] \ NP)/(S[b] \ NP)

to (S[b] \ NP)/ NP

books NP

NP

she'

She

likes
to

read

books

NP

NP

she'

Ze

leest
graag
boeken
Example

```
NP  (S[dcl]\ NP)/(S[to]\ NP)  (S[to]\ NP)/(S[b]\ NP)  (S[b]\ NP)/ NP  NP
she'  like'  to'  read'  book'
She  likes  to  read  books
Ze  leest  graag  boeken
NP  (S|NP)| NP
she'  read'
```
Example

\[
\begin{align*}
\text{She} & \quad \text{likes} & \quad \text{to} & \quad \text{read} \\
\text{She} & \quad \text{likes} & \quad \text{to} & \quad \text{read}
\end{align*}
\]
Example

- **NP**
  - *she'*

- **she**
  - **likes**

- **like'**

- **to'**

- **to**

- **read'**

- **read**

- **books**

- **Ze**
  - **leest**

- **graag**

- **boeken**

- **NP**
  - *she'*

- **read'**

- **λx. (likes'@(to'@x))**
Experiment

- 13K English-Dutch sentence pairs
- C&C/Boxer to make derivations for English sentences
- IBM Model 4 to align sentences, 3-best alignments in both directions
- for each word, cutoff is 0.1
- result: 24K Dutch candidate lexical items
Introduction

Category Projection

Derivation Projection

Translation Divergences

Parsing

Results
Derivation Projection

Need target-language training derivations

- parse target-language sentences
- use target-language candidate lexical items from category projection
- constraint: parse must have same semantics as the English one
Example

Ze

NP
she'

(S|NP)|NP
read'

(S|NP)|(S|NP)
λx.(likes'@(to'@x))

boeken

NP
book'
Example

\[
\begin{array}{cccc}
\text{Ze} & \text{leest} & \text{graag} & \text{boeken} \\
\text{NP} & \text{(S|NP)|NP} & \text{(S|NP)|(S|NP)} & \text{NP} \\
\text{she'} & \text{read'} & \lambda x.(\text{likes'}@(to'@x)) & \text{book'} \\
\end{array}
\]
Example

\[
\begin{array}{c c c c}
\text{Ze} & \text{leest} & \text{graag} & \text{boeken} \\
\text{NP} & (S|\text{NP})|\text{NP} & (S|\text{NP})(S|\text{NP}) & \text{NP} \\
\text{she'} & \text{read'} & \lambda x. (\text{likes'}\circ (\text{to'}\circ x)) & \text{book'} \\
\hline
\lambda x. (\text{likes'}\circ (\text{to'}\circ (\text{read'}\circ x))) & (S|\text{NP})|\text{NP} & \text{NP} & \text{NP} \\
\text{S|NP} & \text{S|NP} & \text{likes'}\circ (\text{to'}\circ (\text{read'}\circ \text{book'})) & \text{NP}
\end{array}
\]
### Example

<table>
<thead>
<tr>
<th>Ze</th>
<th>leest</th>
<th>graag</th>
<th>boeken</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>(S</td>
<td>NP)</td>
<td>NP</td>
</tr>
<tr>
<td>she'</td>
<td>read'</td>
<td>λx.(likes'@(to'@x))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;<em>x</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(S</td>
<td>NP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>λx.(likes'@(to'@read'@x))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S</td>
<td>NP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>likes'@(to'@read'@book')</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(likes'@(to'@read'@book'))@she'</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;0</td>
<td></td>
</tr>
</tbody>
</table>
Ze leest graag boeken

\[
\begin{array}{|c|}
\hline
x_1 \ p_1 \ e_1 \\
\text{female}(x_1) \\
\begin{array}{|c|}
\hline
x_2 \ e_2 \\
\text{book.n.01}(x_2) \\
\text{read.v.01}(e_2) \\
\text{Agent}(e_2, \ x_1) \\
\text{Theme}(e_2, \ x_2) \\
\end{array} \\
\text{like.v.02}(e_1) \\
\text{Experiencer}(e_1, \ x_1) \\
\text{Stimulus}(e_1, \ p_1) \\
\hline
\end{array}
\]
Experiment

- trying to project 13K training derivations
- sometimes, no parse found
- sometimes, search space explodes despite pruning
  - abort if agenda > 256
- successful for 8K derivations
Verb + complementizer vs. adverb

She NP : she

(S[dcl] \ NP) / (S[to] \ NP) : like′ (S[to] \ NP) / (S[b] \ NP) : to′ (S[b] \ NP) / NP : read′ NP : book′

She likes to read books

Ze leest graag boeken

NP : she′ (S[NP] | NP : read′) (S[NP] | (S[NP] : \ x.(likes′ @ (to′ @ x))) NP : book′

(S[NP] | NP : read′ @ (read′ @ book′)) S | NP : (likes′ @ (to′ @ (read′ @ book′))) S : (likes′ @ (to′ @ (read′ @ book′))) @ she′
Pro-drop

\[
S[dcl] : (have'@three'@son')@he' \\
\]

\[
S[dcl] \setminus NP : have'@three'@son' \\
\]

\[
NP : three'@son' \\
\]

\[
N : three'@son' \\
\]

\[
NP : he' (S[dcl] \setminus NP)/NP : have' \\
\]

\[
He \quad had \quad three \quad sons \quad Aveva \quad tre \quad figli \\
\]

\[
S[dcl]/NP : \lambda x.(have'@x)@he' \\
\]

\[
N / N : three' \quad N : figli' \quad N : three'@son' \\
\]

\[
N : three'@son' \quad * \\
\]

\[
NP : three'@son' \\
\]

\[
S[dcl] : (have'@three'@son')@he' \]

Particle vs. Simplex Verb

\[
\begin{align*}
S[b] \setminus NP & : \text{not'}(d'o')( (mess'@it')@up') \\
\frac{(S[b] \setminus NP)/(S[b] \setminus NP) : \lambda x. \text{not'}(d'o'x)}{(S[b] \setminus NP) \setminus (S \setminus NP) : \text{not'}} <_{x}^{1} & \frac{(S[b] \setminus NP)/(S[b] \setminus NP) : \text{do'}}{(S \setminus NP) \setminus (S \setminus NP) : \text{not'}} >_{0}^{0} \\
\frac{(S[b] \setminus NP) \setminus (S[b] \setminus NP) : \lambda x. \text{not'}(d'o'x)}{Versetzung} & \frac{(S[b] \setminus NP) \setminus (S[b] \setminus NP) : \lambda x. \text{not'}(d'o'x)}{es} <_{x}^{0} \\
S[b] \setminus NP & : \text{not'}(d'o'( (mess'@it')@up')) \\
\frac{S[b] \setminus NP : (mess'@it')@up'}{(S[b] \setminus NP) \setminus PR : mess'@it'} >_{0}^{0} & \frac{PR : mess'@it'}{it} >_{0}^{0} \frac{NP : mess'}{PR : up'} \\
\frac{NP : mess'}{it} \frac{NP : mess'}{it} \frac{PR : mess'}{it} & \frac{PR : mess'}{it} \frac{PR : mess'}{it} \frac{PR : mess'}{it}
\end{align*}
\]
Shift-reduce Parsing

- Based on English CCG parser of Zhang and Clark (2011)
- Action types: shift, combine, unary, skip, finish
- Allows fragmentary parses
Parser Training

- Training data: 8K Dutch derivations obtained in derivation projection
- Features: Zhang and Clark (2011)
- Averaged perceptron with early update and beam search ($b = 16$, $T = 10$)
Dealing with OOV Words at Test Time

Pick schematic lexical entries for POS extracted from lexicon, e.g.,

\[
\begin{array}{c}
N \\
f \\
\text{getuige/nounsg}
\end{array} \quad \text{where } f = \lambda x. \quad \boxed{\text{__UNKNOWN__}(x)}
\]
Introduction

Category Projection

Derivation Projection

Translation Divergences

Parsing

Results
Test Data

- 150 English-Dutch sentence pairs
- Discourse Representation Structures automatically produced by C&C/Boxer
- manually corrected by 2 annotators, then adjudicated
Evaluation: Graph Match Measure

Le and Zuidema (2012)

\[
\begin{align*}
x_1 \quad \text{name}(x_1, \text{jones}, \text{nam}) \\
x_2 & \quad \text{ball}(x_2) \\
& \quad \text{see}(e_3) \\
& \quad \text{agent}(e_3, x_1) \\
& \quad \text{patient}(e_3, x_2) \\
\Rightarrow \\
dx_1 \quad \text{name}(x_1, \text{jones}, \text{nam}) \\
& \quad \text{kick}(e_4) \\
& \quad \text{agent}(e_4, x_1) \\
& \quad \text{patient}(e_4, x_2) \\
\lor \\
x_4 & \quad \text{cake}(x_4) \\
& \quad \text{see}(e_5) \\
& \quad \text{agent}(e_5, x_1) \\
& \quad \text{patient}(e_5, x_4)
\end{align*}
\]
Evaluation: Baseline and Upper Bound

- baseline: most frequent lexical entry/schema for each word, all unconnected
- upper bound: silver standard, unseen symbols replaced with __UNKNOWN__
Where It Goes Well
Where It Doesn’t
### Results

<table>
<thead>
<tr>
<th>Language System</th>
<th>English C&amp;C/Boxer</th>
<th>Dutch Our system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>78.88</td>
<td>69.22</td>
</tr>
<tr>
<td>Structure+concepts</td>
<td>68.10</td>
<td>48.35</td>
</tr>
<tr>
<td>Structure+relations</td>
<td>69.06</td>
<td>60.23</td>
</tr>
<tr>
<td>Structure+concepts+relations</td>
<td>58.40</td>
<td>42.99</td>
</tr>
</tbody>
</table>
Conclusions

• Learning an open-domain semantic parser cross-lingually: it can be done, but...
  • Need larger vocabulary
  • Need to study category/derivation projection in more detail
• CCG handles translation divergences well
2018 Update

- new experiments on Dutch, German, Italian
- also learning slash directions
- only syntax, for now
- promising results
- projected parses used to bootstrap the Parallel Meaning Bank (http://pmb.let.rug.nl)
References I


References II


