Incremental Processing

Incremental Semantic Parsing

Referent
Nikolina Koleva

16 June 2011
Motivation

Why incremental \textbf{semantic} parsing?

- more suitable for slot filling
- enables modeling dialogue phenomena
  - \textit{shared utterances, ellipsis, alignment}

\textless; for dialogue systems
Overview

1. RUBISC’s Incremental Chunking
   1.1 Regular Grammar for Semantics
   1.2 Unification
   1.3 Robustness
   1.4 Incrementality
   1.5 End-of-Sentence Detection
   1.6 Evaluation

2. Incremental Parsing, or Incremental Grammar?
   2.1 Parsing vs. Generation
   2.2 Contextual Model
   2.3 Shared Utterances
   2.4 Cross-Speaker Ellipsis
   2.5 Alignment
RUBISC

a Robust Unification-Based Incremental Semantic Chunker

- incremental slot filling
- chunks based on semantic content rather than syntax
- partial or complete semantics during the incoming speech
- evaluated on German speech corpus
RUBISC’s Domain

- Actions
  - turn, flip, move, grasp
- Objects
  - cross, l etc.
- End positions
  - head, back, leg etc.
Semantic Slots & Grammar Rules

@: action @: entity: xpos
@: entity: name @: entity: ypos @: end

action: grasping, end: empty → nimm | nehme
{take}
action: turning → drehe?
{turn}
action: flipping → spig(le | el)
{flip}
action: movement → bewegt
{moved}

entity: name: x → kreuz | plus | ((das | ein) x$)
{cross | plus (the | a) x}
entity: name: w → treppe | ((das | ein) w$)
{staircase | (the | a) w}
end: head → (in | an) den kopf
{(on | in) the head}
end: leg2 → ins? das (hinterbein | hintere bein | rechte bein | zweites bein)
{in the hind leg | back leg | right leg | second leg}
entity: ypos: 1 → der (unteren | zweiten) reihe
{(lower | second) row}
entity: xpos: 1 → das erste
{the first}
entity: xpos: -1 → das letzte
{the last}
end: horizontal, action: flipping → horizontal
{horizontally}
action: grasping, end: empty → nimm | nehme
action: flipping → spieg(le | el)
action: turning → drehe?
action: movement → bewegt

entity: name: x → kreuz | plus | ((das | ein) x$)
entity: name: w → treppe | ((das | ein) w$)
end: head → (in | an) den kopf
end: leg2 → ins? das (hinterbein | hintere bein | rechte bein | zweites bein)

entity: ypos: 1 → der (unteren | zweiten) reihe
entity: xpos: 1 → das erste
entity: xpos: -1 → das letzte
end: horizontal, action: flipping → horizontal
Incremental Chunking

**Grammar**
- action:turning -> turn
- object:xpos:2 -> the second
- object:ypos:-1 -> the upper row
- end:right -> to the right clockwise

**chunks**
- [turn] -> erm
- [erm the piece] -> erm the piece
- [erm the piece erm the] -> erm the piece erm the
- [erm the piece erm the second] -> [erm the piece erm the second]
- [in the upper row] -> to
- [to erm clockwise] -> [to erm clockwise]

**semantics**
- action:turning
- end:-
- object:[name:-] [xpos:-] [ypos:-]
- action:turning
- end:-
- object:[name:-] [xpos:2] [ypos:-]
- action:turning
- end:-
- object:[name:-] [xpos:2] [ypos:-1]
- action:turning
- end:right
- object:[name:-] [xpos:2] [ypos:-1]
RUBISC

• Unification-based
• Robust
• Incremental
schieb das mh... das horizontal mh liegt ins Vorderbein

{move that mh which is horizontal into the front leg}
Robustness

• pronunciation variants considered

spieg(le | el) \{flip\}

• search only for relevant information in a chunk

in uh der zweiten äh Reihe \rightarrow\text{ zweite Reihe is found}
\{in uh the second uh row\} \{second row\}

• mechanism for jumping over words

das merkwürdige Kreuz \rightarrow\text{ das Kreuz is found}
\{the strange cross\} \{the cross\}

➡
merkwürdig \{strange\} is unknown word
Incrementality

- not strictly incremental
- sometimes more than one word required for a chunk

```
in das rechte  waiting...  Bein
{into the right}  {leg}
```

- first slot filled after 3.5 words
- average length of utterance = 12.4 words
End-of-Sentence Detection

- semantic-pragmatic completeness
- certain slots filled
- in RUBISC
  1. action
  2. end
  3. name or (xpos and ypos)
Evaluation

• 500 utterances manual labeled

• run on 400 unseen utterances

• 100 labelled and 50 other utterances for grammar development
  
  ! annotators could influence the grammar or the chunker to perform better ⇒ they weren’t involved in their development

• used baseline (no slot filling)
  
  ▶ always returns the empty frame

<table>
<thead>
<tr>
<th>action:</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>end:</td>
<td>None</td>
</tr>
<tr>
<td>entity:name:</td>
<td>None</td>
</tr>
<tr>
<td>entity:xpos:</td>
<td>None</td>
</tr>
<tr>
<td>entity:ypos:</td>
<td>None</td>
</tr>
<tr>
<td>Evaluation results in %</td>
<td>baseline</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>concept error</td>
<td>83,0</td>
</tr>
<tr>
<td>frame correct</td>
<td>17,0</td>
</tr>
<tr>
<td>slots correct</td>
<td>64,3</td>
</tr>
<tr>
<td>action correct</td>
<td>27,8</td>
</tr>
<tr>
<td>end correct</td>
<td>68,0</td>
</tr>
<tr>
<td>name correct</td>
<td>48,0</td>
</tr>
<tr>
<td>xpos correct</td>
<td>87,5</td>
</tr>
<tr>
<td>ypos correct</td>
<td>89,5</td>
</tr>
</tbody>
</table>

**worst case**

**best case**
<table>
<thead>
<tr>
<th>Evaluation results in %</th>
<th>baseline</th>
<th>RUBISC</th>
<th>without unification</th>
<th>without robustness</th>
<th>inter annotator agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>concept error</td>
<td>83,0</td>
<td>45,5</td>
<td>49,5</td>
<td>73,3</td>
<td>32,3</td>
</tr>
<tr>
<td>frame correct</td>
<td><strong>17,0</strong></td>
<td><strong>54,5</strong></td>
<td><strong>50,3</strong></td>
<td><strong>26,8</strong></td>
<td><strong>67,8</strong></td>
</tr>
<tr>
<td>slots correct</td>
<td>64,3</td>
<td>86,8</td>
<td>84,6</td>
<td>78,8</td>
<td>92,1</td>
</tr>
<tr>
<td>action correct</td>
<td>27,8</td>
<td>90,3</td>
<td>85,8</td>
<td>64,3</td>
<td>89,0</td>
</tr>
<tr>
<td>end correct</td>
<td>68,0</td>
<td>85,8</td>
<td>81,0</td>
<td>73,8</td>
<td>95,8</td>
</tr>
<tr>
<td>name correct</td>
<td>48,0</td>
<td>86,3</td>
<td>84,5</td>
<td>79,0</td>
<td>86,8</td>
</tr>
<tr>
<td>xpos correct</td>
<td>87,5</td>
<td>83,0</td>
<td>83,0</td>
<td>86,5</td>
<td>94,5</td>
</tr>
<tr>
<td>ypos correct</td>
<td>89,5</td>
<td>88,8</td>
<td>88,8</td>
<td>90,3</td>
<td>94,5</td>
</tr>
</tbody>
</table>

worst case

best case
## Evaluation

<table>
<thead>
<tr>
<th>Evaluation results in %</th>
<th>baseline</th>
<th>RUBISC</th>
<th>without unification</th>
<th>without robustness</th>
<th>inter annotator agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>concept error</td>
<td>83,0</td>
<td>45,5</td>
<td>49,5</td>
<td>73,3</td>
<td>32,3</td>
</tr>
<tr>
<td>frame correct</td>
<td>17,0</td>
<td>54,5</td>
<td>50,3</td>
<td>26,8</td>
<td>67,8</td>
</tr>
<tr>
<td>slots correct</td>
<td>64,3</td>
<td>86,8</td>
<td>84,6</td>
<td>78,8</td>
<td>92,1</td>
</tr>
<tr>
<td>action correct</td>
<td><strong>27,8</strong></td>
<td><strong>90,3</strong></td>
<td><strong>85,8</strong></td>
<td><strong>64,3</strong></td>
<td><strong>89,0</strong></td>
</tr>
<tr>
<td>end correct</td>
<td>68,0</td>
<td>85,8</td>
<td>81,0</td>
<td>73,8</td>
<td>95,8</td>
</tr>
<tr>
<td>name correct</td>
<td>48,0</td>
<td>86,3</td>
<td>84,5</td>
<td>79,0</td>
<td>86,8</td>
</tr>
<tr>
<td>xpos correct</td>
<td>87,5</td>
<td>83,0</td>
<td>83,0</td>
<td>86,5</td>
<td>94,5</td>
</tr>
<tr>
<td>ypos correct</td>
<td>89,5</td>
<td>88,8</td>
<td>88,8</td>
<td>90,3</td>
<td>94,5</td>
</tr>
</tbody>
</table>

**worst case**

**best case**
### Evaluation

<table>
<thead>
<tr>
<th>Evaluation results in %</th>
<th>baseline</th>
<th>RUBISC</th>
<th>without unification</th>
<th>without robustness</th>
<th>inter annotator agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>concept error</td>
<td>83,0</td>
<td>45,5</td>
<td>49,5</td>
<td>73,3</td>
<td>32,3</td>
</tr>
<tr>
<td>frame correct</td>
<td>17,0</td>
<td>54,5</td>
<td>50,3</td>
<td>26,8</td>
<td>67,8</td>
</tr>
<tr>
<td>slots correct</td>
<td>64,3</td>
<td>86,8</td>
<td>84,6</td>
<td>78,8</td>
<td>92,1</td>
</tr>
<tr>
<td>action correct</td>
<td>27,8</td>
<td>90,3</td>
<td>85,8</td>
<td>64,3</td>
<td>89,0</td>
</tr>
<tr>
<td>end correct</td>
<td>68,0</td>
<td>85,8</td>
<td>81,0</td>
<td>73,8</td>
<td>95,8</td>
</tr>
<tr>
<td>name correct</td>
<td>48,0</td>
<td>86,3</td>
<td>84,5</td>
<td>79,0</td>
<td>86,8</td>
</tr>
<tr>
<td>xpos correct</td>
<td>87,5</td>
<td>83,0</td>
<td>83,0</td>
<td>86,5</td>
<td>94,5</td>
</tr>
<tr>
<td>ypos correct</td>
<td>89,5</td>
<td>88,8</td>
<td>88,8</td>
<td>90,3</td>
<td>94,5</td>
</tr>
</tbody>
</table>

*worst case*
Error Analysis

- 50 utterances

- vocabulary restrictions

- restrictions in regular expressions

in den Rücken

\{into the back\} ✓

so dass es den Rücken berührt

\{such that it touches the back\} ✗

- start positions confusable with end positions
Intermediate Conclusion

• incremental chunking by sense units suitable for spoken dialogue systems

• expressive power limited to extra coding work
  • lack of generalization
  • suitable for restricted domain
Nikolina: Now, I’m going to continue with

Benjamin: The second paper.

Nikolina: the second paper.

Ruth: What did Alex ...

Hugh: Design? A kaleidoscope.
• to model shared utterances, cross-speaker elipses, alignment

• Need for

  1. incrementality

  2. interaction between parsing and generation processes
Background: Dynamic Syntax

- Parsing-directed grammar formalism
- **Semantic** interpretation of an utterance represented by decorated tree structure
- Grammaticality $\equiv$ parsability
  - $\Rightarrow$ Successful incremental construction of a tree structure
    - Logical form
John likes Mary.
John likes Mary.
John likes Mary.
John likes Mary.

### Diagram:

- **John**: 
  - **{john’}**: 
    - **likes**: 
      - **{john’}**: 
        - **{like’}(mary’)(john’), ◇
          - **{like’}**: 
            - **{mary’}**: 

- **Mary**: 
  - **{john’}**: 
    - **{like’}(mary’)**: 
      - **{like’}**: 
        - **{mary’}**: 

John likes Mary.

John

\{john'\}

likes

\{john'\}

\{\}

\{like'\}

\{\}

\{\}

Mary

\{john'\}

\{like'(mary')(john'), \diamond\}

\{like'(mary')\}

\{like'\}

\{mary'\}
Generation with Dynamic Syntax

W1, W2, W3

partial trees

goal tree
Generation with Dynamic Syntax

\[ \begin{align*}
\text{partial tree} & \quad \subseteq \quad \text{goal tree} \\
\{ w_1, w_2, w_3 \} & \quad \subseteq \quad \{ w_1'(w_3')(w_1'), \diamond \} \\
\{ w_1' \} & \quad \subseteq \quad \{ w_2'(w_3') \} \\
\{ w_2' \} & \quad \subseteq \quad \{ w_3' \}
\end{align*} \]
Generation with Dynamic Syntax

\[ \{w_1', w_2', w_3'\} \subset \{w_1'(w_3') (w_1'), \Diamond\} \]

\[ \{w_1'\} \subset \{w_2'(w_3')\} \]

\[ \{w_2'\} \subset \{w_3'\} \]
John likes Mary.

\[
\begin{array}{c}
\{\text{like'}(\text{mary'})(\text{john'}), \Diamond\} \\
\{\text{john'}\} & \{\text{like'}(\text{mary'})\} \\
\{\text{like'}\} & \{\text{mary'}\}
\end{array}
\]
Generation with Dynamic Syntax

John likes Mary.

{like'(mary')(john'), ◇}

{john'}

{like'}

{mary'}

goal tree
John likes Mary.

```
{like'(mary')(john'), ◇}
{john'}
{like'(mary')}
{like'}
{mary'}
```

![goal tree diagram](image)
Generation with Dynamic Syntax

John likes Mary.

_goal tree_

{like'(mary')(john'), ◇}
{john'} {like'(mary')}
{like'} {mary'}
Generation with Dynamic Syntax

John likes Mary.

Goal tree:

\[
\{\text{like}'(\text{mary}')(\text{john}')\}, \diamond
\]

\[
\{\text{like}'\}\{\text{mary}'\}
\]

\[
\{\text{john}'\}\{\text{like}'(\text{mary}')\}
\]

\[
\{\text{like}'\}\{\text{mary}'\}
\]
Contextual Model

1. semantic trees

2. sequences of words

3. associated lexical actions

aremos available to parsing and generation \( \Rightarrow \) enables modeling of complex dialogue phenomena
Parsing in Context

Parser state consists of set of triples \( \langle T, W, A \rangle \)

- \( T \) is (partial) semantic tree
- \( W \) is the sequence of words
- \( A \) is the set of actions used to create \( T \) from \( W \)

initial parser state :=\{\langle T_a, \emptyset, \emptyset \rangle \} /T_a \) is the basic axiom/

any tree: context :=\{\langle T_0, W_0, A_0 \rangle, \langle T, W, A \rangle \}

parse completed (disambiguation resolved)

context :=\{\langle T_1, W_1, A_1 \rangle, \langle T, W, A \rangle \}
Generation in Context

Generator state consists of pair $\langle T_g, X \rangle$

- $T_g$ is the goal semantic tree
- $X$ is set of pairs $(S, P)$
  - $S$ is candidate partial string
  - $P$ is the analogous parser state

Initial generator state $:= \langle T_g, \{(\emptyset, P_0)\} \rangle$, where $P_0 := \{\langle T_a, \emptyset, \emptyset \rangle\}$

∀ any tree: context $:= \{\langle T_0, W_0, A_0 \rangle, \langle T, W, A \rangle\}$
Shared Utterances

• necessarily same intermediate representation shared

• parsing and generation can start from any state

\[
\langle T_g, \{(\emptyset, P_0)\} \rangle \Rightarrow \langle T_g, \{(\emptyset, P_t)\} \rangle
\]

**Hearer to Speaker**

Requirement: \( \exists T_a \in P_t \land T_a \subseteq T_g \)

\[
\langle S_t, P'_t \rangle
\]

**Speaker to Hearer**

initial speaker needs \( P'_t \)

Requirement: \( \forall T_a \in P'_t \land T_a \subseteq T_g \rightarrow \text{keep}(T_a) \)
What did Alex ... / ...design?

\[ P_t = \langle \{+Q\}, \{\text{what, did, alex}\}, \{a_1, a_2, a_3\} \rangle \]

\[ \{\text{WH}\} \quad \{?T_y(e \rightarrow t), \Diamond\} \]

\[ G_t = \left( \begin{array}{c}
\{\text{alex'}\} \\
\{\text{design'(WH)}\}
\end{array} \right), (\emptyset, P_t) \]
Shared Utterances: Hearer to Speaker

What did Alex ... / ...design?

\[
G_t = \left( \left\{ \text{design'('WH')('alex')}, \Diamond \right\} \right)
\]

\[
G_1 = \left( \left\{ \text{design'('WH')('alex')}, \Diamond \right\} \right)
\]
Cross-Speaker Ellipsis

A: What did Alex design?

B: A kaleidoscope.

{+Q, design’(WH)(alex’), ◇}  {design’(kaleidoscope’)(alex’), ◇}

{alex’}  {design’(WH)}  {kaleidoscope’}  {design’}

{WH}  {design’}

↑

{kaleidoscope’}
Cross-Speaker Ellipsis

A: What did Alex design?

B: A kaleidoscope.

\{+Q, design'(WH)(alex'), ◇\}
\{design'(WH)\}
\{alex\}
\{design'\}
\{WH\}
\{design'\}
\{kaleidoscope'\}
\{design'\}
\{kaleidoscope'\}
\{alex\}
\{design'(kaleidoscope')\}
\{kaleidoscope'\}
\{design'\}
VP Ellipsis

A: A policeman who arrested Bill read him his rights.

B: The policeman who arrested Tom did too.

reuse sequence of actions from the context
Alignment

• parsing and generation search for suitable entries in the lexicon

• expensive process

minimising effort via re-use sequence of selected actions from the context
Summary

- semantic parsing
  - suitable for dialogue systems
  - influenced by the current context
  - provides straightforward model of complex dialogue phenomena
References


• http://www.kcl.ac.uk/research/groups/ds/

• http://godel.stanford.edu/mpurver/ds/
Thank you for your attention!
• Is robustness always an advantage?

• Is it a disadvantage that the semantic of the first two words isn’t put together?