The boy wants the girl to believe him

\[ \lambda x.\lambda y.\text{boy}'(x) \land \text{girl}'(y) \land \ldots \]

\[ \ldots \]

Lecture 8: Graph-Based Representations for Semantics

1. Generalised quantifiers
2. Logico-semantic graphs
3. Clause Union
4. Functor, argument and bilinearity
Lecture 8: Graph-Based Representations for Semantics

1. Generalised quantifiers
2. Logico-semantic graphs
3. Clause Union
4. Functor, argument and bilinearity
Generalised Quantifiers
Quantification over individuals/sets

- What is \( [\text{every student smokes}] \)?
  \( \forall x (\text{student}'(x) \rightarrow \text{smoke}'(x)) \)
- What is \( [\text{some students smoke}] \)?
  \( \exists x (\text{student}'(x) \land \text{smoke}'(x)) \)

\( x \)

\( x \)
Quantification over individuals/sets

- What is $[\text{every student smokes}]$?
  $\forall x (\text{student}'(x) \rightarrow \text{smoke}'(x))$

- What is $[\text{some students smoke}]$?
  $\exists x (\text{student}'(x) \land \text{smoke}'(x))$

Diagram:
- Student set
- Smoke set
- Teacher set
Quantification over individuals/sets

- What is \([\text{every student smokes}]\)?
  \[ \forall x (\text{student}'(x) \rightarrow \text{smoke}'(x)) \]

- What is \([\text{some students smoke}]\)?
  \[ \exists x (\text{student}'(x) \land \text{smoke}'(x)) \]
Quantification over individuals/sets

- What is $[\text{every student smokes}]$?
  $\forall x (\text{student}'(x) \rightarrow \text{smoke}'(x))$

- What is $[\text{some students smoke}]$?
  $\exists x (\text{student}'(x) \land \text{smoke}'(x))$

$\lambda P. [\lambda Q. [\forall x (P(x) \rightarrow Q(x))] [\lambda Q. [\exists x (P(x) \land Q(x))]]$
Quantification over individuals/sets

- What is \([\text{every student smokes}]\)?
  \[\forall x (\text{student'}(x) \rightarrow \text{smoke'}(x))\]
- What is \([\text{some students smoke}]\)?
  \[\exists x (\text{student'}(x) \land \text{smoke'}(x))\]

\[\text{every} = \lambda P. [\lambda Q. [\forall x (P(x) \rightarrow Q(x))]]\]
\[\text{some} = \lambda P. [\lambda Q. [\exists x (P(x) \land Q(x))]]\]
Quantification over individuals/sets

- What is \([\textit{every student smokes}]\)?
  \[
  \forall x (\text{student}'(x) \to \text{smoke}'(x))
  \]

- What is \([\textit{some students smoke}]\)?
  \[
  \exists x (\text{student}'(x) \land \text{smoke}'(x))
  \]

what is the type of the NP (\textit{every student})?

Is it \(\langle e, \langle e, t \rangle \rangle\)? Or \(\langle \langle e, t \rangle, t \rangle\)?
Generalised quantifiers

\[ \langle \langle e, t \rangle, t \rangle \]

- Every student smokes.
  the bucket associated with student is the only element in the bucket associated with every student.

- Assume we have two students in our world model:

\[
\begin{bmatrix}
  t & \mapsto & 1 \\
  j & \mapsto & 1 \\
  t & \mapsto & 1 \\
  j & \mapsto & 0 \\
  t & \mapsto & 0 \\
  j & \mapsto & 1 \\
  t & \mapsto & 0 \\
  j & \mapsto & 0
\end{bmatrix} \mapsto \begin{bmatrix} 1 \end{bmatrix}
\]
Generalised quantifiers

- **At least three students** smoke.  
  every bucket in the bucket associated with **at least three students** contains at least three students.
- **nothing, most, many, half** …
- FOPL is not expressive enough.

A convenient notation

- $\forall x (\text{student}'(x) \rightarrow \text{smoke}'(x))$
- $\exists x (\text{student}'(x) \land \text{smoke}'(x))$
- at least three$(x, \text{student}'(x), \text{smoke}'(x))$
- every$(x, \text{student}'(x), \text{smoke}'(x))$
- some$(x, \text{student}'(x), \text{smoke}'(x))$
Truth conditions for generalized determiners

<table>
<thead>
<tr>
<th>Determiner</th>
<th>Truth conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[every] (P)(Q)</code></td>
<td>$P \subseteq Q$</td>
</tr>
<tr>
<td><code>[some] (P)(Q)</code></td>
<td>$P \cap Q \neq \emptyset$</td>
</tr>
<tr>
<td><code>[no] (P)(Q)</code></td>
<td>$P \cap Q = \emptyset$</td>
</tr>
<tr>
<td><code>[three] (P)(Q)</code></td>
<td>$|P \cap Q| = 3$</td>
</tr>
<tr>
<td><code>[less than three] (P)(Q)</code></td>
<td>$|P \cap Q| &lt; 3$</td>
</tr>
<tr>
<td><code>[at least three] (P)(Q)</code></td>
<td>$|P \cap Q| \geq 3$</td>
</tr>
<tr>
<td><code>[most] (P)(Q)</code></td>
<td>$|P \cap Q| \geq |P - Q|$</td>
</tr>
<tr>
<td><code>[few] (P)(Q)</code></td>
<td>$|P \cap Q| \ll |P - Q|$</td>
</tr>
</tbody>
</table>
Truth conditions for generalized determiners

<table>
<thead>
<tr>
<th>Determiner</th>
<th>Truth conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>[every] (P)(Q)</td>
<td>( P \subseteq Q )</td>
</tr>
<tr>
<td>[some] (P)(Q)</td>
<td>( P \cap Q \neq \emptyset )</td>
</tr>
<tr>
<td>[no] (P)(Q)</td>
<td>( P \cap Q = \emptyset )</td>
</tr>
<tr>
<td>[three] (P)(Q)</td>
<td>( | P \cap Q | = 3 )</td>
</tr>
<tr>
<td>[less than three] (P)(Q)</td>
<td>( | P \cap Q | &lt; 3 )</td>
</tr>
<tr>
<td>[at least three] (P)(Q)</td>
<td>( | P \cap Q | \geq 3 )</td>
</tr>
<tr>
<td>[most] (P)(Q)</td>
<td>( | P \cap Q | \geq | P - Q | )</td>
</tr>
<tr>
<td>[few] (P)(Q)</td>
<td>( | P \cap Q | \ll | P - Q | )</td>
</tr>
</tbody>
</table>

\( [\text{the}] \)
Logico-Semantic Graphs
Abstract Meaning Representation

There are several projects working on developing “conceptual graphs” as comprehensive meaning representations. We introduce Abstract Meaning Representation and English Resource Semantics.

• Basic units are “concepts” as well as asymmetric “links/dependency” between such concepts.

\[
\text{be-located-at-91} \\
\text{ARG0} \quad \text{every} \\
\text{ARG1} \quad \text{desk} \\
\text{quant} \quad \text{computer}
\]

\text{a computer is on every desk.}
Abstract Meaning Representation

- AMR is a semantic representation aimed at large-scale human annotation in order to build a giant semantics bank.
- We do a practical, replicable amount of abstraction (limited canonicalization).
- Capture many aspects of meaning in a single simple data structure.
- AMR annotations are not tied to individual words or any syntactic derivation

PENMAN notation

*The dog is eating a bone*

(e / eat-01
 :ARG0 (d / dog)
 :ARG1 (b / bone))

slide from https://github.com/nschneid/amr-tutorial
Abstract Meaning Representation

- AMR is a semantic representation aimed at large-scale human annotation in order to build a giant semantics bank.
- We do a practical, replicable amount of abstraction (limited canonicalization).
- Capture many aspects of meaning in a single simple data structure.
- AMR annotations are not tied to individual words or any syntactic derivation.

PENMAN notation

*The dog is eating a bone*

(e / eat-01
  :ARG0 (d / dog)
  :ARG1 (b / bone))

Inter-annotator agreement: 70–80% SMATCH
There is nothing as practical as a good theory.
Different representations of logical forms

- **Every desk has a computer**
- \(\forall x (\text{desk}'(x) \rightarrow (\exists y (\text{computer}'(y) \land \text{have}'(e, x, y))))\)
- every'\((x, \text{desk}'(x), a'(y, \text{computer}'(y), \text{have}'(e, x, y)))\)
Different representations of logical forms

- *Every desk has a computer*
- \( \forall x (\text{desk}'(x) \rightarrow (\exists y (\text{computer}'(y) \land \text{have}'(e, x, y)))) \)
- \( \text{every}'(x, \text{desk}'(x), \text{a}'(y, \text{computer}'(y), \text{have}'(e, x, y))) \)

ARG0: which word “introduces” a variable.
Different representations of logical forms

- *Every desk has a computer*
- $\forall x (\text{desk}'(x) \rightarrow (\exists y (\text{computer}'(y) \land \text{have}'(e, x, y))))$
- $\text{every}' (x, \text{desk'}(x), a'(y, \text{computer'}(y), \text{have}'(e, x, y)))$

ARG0: which word “introduces” a variable.
Different representations of logical forms

- *Every desk has a computer*
- $\forall x (\text{desk}'(x) \rightarrow (\exists y (\text{computer}'(y) \land \text{have}'(e, x, y))))$
- *every’*( $x$, desk’($x$), a’($y$, computer’($y$), have’($e, x, y$)))

ARG0: which word “introduces” a variable.
Different representations of logical forms

- **Every desk has a computer**
- \( \forall x (\text{desk}'(x) \rightarrow (\exists y (\text{computer}'(y) \land \text{have}'(e, x, y)))) \)
- every'(x, desk'(x), a'(y, computer'(y), have'(e, x, y)))

ARG0: which word "introduces" a variable.

Every desk has a computer
Bi-lexical semantic dependency graphs

- Projecting “concept nodes” to “words”.
- Relations between “concepts” \( \Rightarrow \) bi-lexical semantic dependencies
- Reasonably good though not as expressive as conceptual graphs.
Bi-lexical semantic dependency graphs

- Projecting "concept nodes" to "words".
- Relations between "concepts" ⇒ bi-lexical semantic dependencies
- Reasonably good though not as expressive as conceptual graphs.

Discussion on weakness of bi-lexical semantic dependency graphs
Bi-lexical semantic dependency graphs

- Projecting “concept nodes” to “words”.
- Relations between “concepts” \( \Rightarrow \) bi-lexical semantic dependencies
- Reasonably good though not as expressive as conceptual graphs.

Discussion on weakness of bi-lexical semantic dependency graphs

What are the triggers of concepts?

- MWE:

  \[ \text{Cambridge} \quad \text{University} \]
Bi-lexical semantic dependency graphs

- Projecting “concept nodes” to “words”.
- Relations between “concepts” $ \Rightarrow $ bi-lexical semantic dependencies
- Reasonably good though not as expressive as conceptual graphs.

Discussion on weakness of bi-lexical semantic dependency graphs

What are the triggers of concepts?

- MWE:
  
  ![MWE Diagram]

- Construction: *The emails won’t reply themselves.*
Bi-lexical semantic dependency graphs

- Projecting “concept nodes” to “words”.
- Relations between “concepts” ⇒ bi-lexical semantic dependencies
- Reasonably good though not as expressive as conceptual graphs.

Discussion on weakness of bi-lexical semantic dependency graphs

What are the triggers of concepts?

- MWE: Cambridge University
- Construction: The emails won’t reply themselves.

Modification
SemBanking in Natural Language Processing

- Manually-annotated
  - PropBank (Kingsbury & Palmer, 2002)
  - FrameNet (Baker et al., 1998)
- Grammar-based
  - English Resource Semantics (Oepen et al., 2004)
  - Groningen Meaning Bank (Basile et al., 2012)
- Abstract Meaning Representation (Banarescu et al., 2013)
  - QA-SRL (He et al., 2015)

English Resource Semantics (ERS)

**LinGO English Resource Grammar**  
(Flickinger, 2000; Flickinger et al., 2017)

- Hand-designed computational grammar for English based on Head-driven Phrase Structure Grammar;
- declarative, unification-based: parsing and realization; multiple engines;
- $25^+$ person years; coverage of 85–95\% of running text across domains;
- underspecified meaning representation in Minimal Recursion Semantics

**LinGO Redwoods Treebank**  
(Oepen & Lønning, 2006; Flickinger et al., 2012)

- Grammar-based annotation: select rather than generate ‘correct’ analysis
- version 1214: some 85,000 annotated sentences, six$^+$ different domains;
- including Sections 00–21 from the venerable WSJ Corpus; sub-set of Brown Corpus; Wikipedia; tourism; ecommerce; transcribed speech;
- MRS plus various graph-based formats.
- inter-annotator agreement of 0.94 EDM (elementary dependency match);
Online demo

- https://delph-in.github.io/delphin-viz/demo/
- http://erg.delph-in.net/
What is the greatest prime number below 2015?

\[
\langle h_1, \\
h_4:\text{thing}(\text{ARG0 } x_5), \\
h_6:\text{which}_q(\text{ARG0 } x_5, \text{RSTR } h_7, \text{BODY } h_8), \\
h_2:\text{be}_v\text{id}(\text{ARG0 } e_3, \text{ARG1 } x_9, \text{ARG2 } x_5), \\
h_{10}:\text{the}_q(\text{ARG0 } x_9, \text{RSTR } h_{12}, \text{BODY } h_{11}), \\
h_{13}:\text{great}_a\text{_for}(\text{ARG0 } e_{14}, \text{ARG1 } x_9), \\
h_{13}:\text{superl}(\text{ARG0 } e_{15}, \text{ARG1 } e_{14}), \\
h_{13}:\text{compound}(\text{ARG0 } e_{17}, \text{ARG1 } x_9, \text{ARG2 } x_{16}\{\}), \\
h_{18}:\text{undef}_q(\text{ARG0 } x_{16}, \text{RSTR } h_{19}, \text{BODY } h_{20}), \\
h_{21}:\text{prime}_n\text{_1}(\text{ARG0 } x_{16}), \\
h_{13}:\text{number}_n\text{_of}(\text{ARG0 } x_9), \\
h_{13}:\text{below}_p(\text{ARG0 } e_{22}, \text{ARG1 } x_9, \text{ARG2 } x_{23}), \\
h_{24}:\text{number}_q(\text{ARG0 } x_{23}, \text{RSTR } h_{25}, \text{BODY } h_{26}), \\
h_{27}:\text{card}(\text{ARG0 } x_{23}, \text{ARG1 } i_{28}, \text{CARG } 2015) \\
{ h_{25} = q h_{27}, h_{19} = q h_{21}, h_{12} = q h_{13}, h_7 = q h_4, h_1 = q h_2 } \rangle
\]
What is the greatest prime number below 2015?

\[
\langle h_1, h_4: \text{thing}(\text{ARG0 } x_5), h_6: \text{which}_q(\text{ARG0 } x_5, \text{RSTR } h_7, \text{BODY } h_8), h_2: \text{be}_v \text{id}(\text{ARG0 } e_3, \text{ARG1 } x_9, \text{ARG2 } x_5), h_{10}: \text{the}_q(\text{ARG0 } x_9, \text{RSTR } h_{12}, \text{BODY } h_{11}), h_{13}: \text{great}_a \_\text{for}(\text{ARG0 } e_{14}, \text{ARG1 } x_9), h_{13}: \text{superl}(\text{ARG0 } e_{15}, \text{ARG1 } e_{14}), h_{13}: \text{compound}(\text{ARG0 } e_{17}, \text{ARG1 } x_9, \text{ARG2 } x_{16}\{}\{}), h_{18}: \text{udef}_q(\text{ARG0 } x_{16}, \text{RSTR } h_{19}, \text{BODY } h_{20}), h_{21}: \text{prime}_n \_\text{1}(\text{ARG0 } x_{16}), h_{13}: \text{number}_n \_\text{of}(\text{ARG0 } x_9), h_{13}: \text{below}_p(\text{ARG0 } e_{22}, \text{ARG1 } x_9, \text{ARG2 } x_{23}), h_{24}: \text{number}_q(\text{ARG0 } x_{23}, \text{RSTR } h_{25}, \text{BODY } h_{26}), h_{27}: \text{card}(\text{ARG0 } x_{23}, \text{ARG1 } i_{28}, \text{CARG } 2015) \rangle
\]

\[
\{ h_{25} = q \ h_{27}, h_{19} = q \ h_{21}, h_{12} = q \ h_{13}, h_7 = q \ h_4, h_1 = q \ h_2 \}
\]
Clause Union
Aladdin (1992 Disney film)

Three wishes

• to be a prince
• to be saved from drowning underwater
• to free the Genie

Fun with linguistics

• Coordination
  *to be a prince and to be saved*
• Subordination
  *to be a prince who is saved*
• Presupposition
  *to see my mother – Queen Elizabeth*
Subordination

(1) a. David complained that Chris smoked.
   b. David wondered who smoked.
   c. David couldn’t believe how big the house was.
The visitor can’t afford to wait.

- who afford?
- who wait?
- who can’t?
Discussion

The visitor can’t afford to wait.

- who afford?
- who wait?
- who can’t?

▷ afford and wait share an argument
Raising and control

Raising

[[ Kim to be happy] seems]
⇓
[Kim [seems to be happy]]

Control

[Sandy wants [Sandy to go]]
⇓
[Sandy wants [PRO to go]]

• Embedded clause is missing its subject
• Subject or object (or PP-obj) of matrix clause (controller) is interpreted as subject of embedded clause.
Small clause

A small clause is a frequently occurring construction that has the semantic subject–predicate characteristics of a clause, but that lacks the tense of a finite clause and appears to lack the status of a constituent.

(2) a. Jim called me a liar.
   b. They named him Pedro.
   c. Fred wiped the table clean.
   d. Larry pounded the nail flat.
   e. Tracy proved the theorem false.
   f. Bo considered Lou a friend.
   g. We saw Fred leave.
   h. Did you hear them arrive?
   i. Dana preferred for Pat to get the job.
   j. Leslie wanted Chris to go.
   k. Lee believed Dominique to have made a mistake.
(3) Stretching his arms, David yawned.
Functor, Argument and Bilinearity
the drug was introduced in West Germany this year
String-to-graph parsing

_The drug was introduced in West Germany this year_

_task_0: Concept-to-word Alignment
_task_1: Concept Identification
_task_2: Relation Detection
the drug was introduced in West Germany this year
String-to-graph parsing

The drug was introduced in West Germany this year

Task 1: Concept Identification
String-to-graph parsing

_the_ drug was introduced in West Germany this year

Task 1: Concept Identification
String-to-graph parsing

Task 1: Concept Identification
String-to-graph parsing

*the* drug was introduced in West Germany this year

**Task 1:** Concept Identification
String-to-graph parsing

The drug was introduced in West Germany this year

_The_q  _introduce_v_to  _year_n_1  _this_q_dem

_drug_n_1  _in_p  loc_nonsp  named("Germany")

proper_q  named("West")  compound  proper_q

Task 1: Concept Identification
String-to-graph parsing

The drug was introduced in West Germany this year.

Task 0: Concept-to-word Alignment

Task 1: Concept Identification
String-to-graph parsing

the drug was introduced in West Germany this year

Task 0: Concept-to-word Alignment

Task 1: Concept Identification
String-to-graph parsing

The drug was introduced in West Germany this year.

Task 0: Concept-to-word Alignment

Task 1: Concept Identification
String-to-graph parsing

*the drug was introduced in West Germany this year*

**Task 0:** Concept-to-word Alignment

**Task 1:** Concept Identification
String-to-graph parsing

The drug was introduced in West Germany this year.

Task 0: Concept-to-word Alignment
Task 1: Concept Identification
Task 2: Relation Detection
String-to-graph parsing

The drug was introduced in West Germany this year

_task 0_: Concept-to-word Alignment
_task 1_: Concept Identification
_task 2_: Relation Detection
String-to-graph parsing

*the* drug was *introduced in* West Germany *this year*

**Task 0:** Concept-to-word Alignment

**Task 1:** Concept Identification

**Task 2:** Relation Detection
String-to-graph parsing

the drug was introduced in West Germany this year

Task 0: Concept-to-word Alignment
Task 1: Concept Identification
Task 2: Relation Detection
Relation detection

Functor–argument relation

*Did you hear *them* arrive?*

- arrive: functor?
- arrive: argument?
Relation detection

**word2vec**: define $p(w_{t+j}|w_t)$ as

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{|V|} \exp(u_w^T v_c)}$$

**Biaffine parsing**

- dot product $\Rightarrow$ inner product
  
  inner product: a positive-definite symmetric bilinear function

  bilinear function:
  
  - $f(\alpha_1 + \alpha_2, \beta) = f(\alpha_1, \beta) + f(\alpha_2, \beta)$, $f(k\alpha, \beta) = kf(\alpha, \beta)$
  - $f(\alpha, \beta_1 + \beta_2) = f(\alpha, \beta_1) + f(\alpha, \beta_2)$, $f(\alpha, k\beta) = kf(\alpha, \beta)$
  - If $\{e_1, e_2, ... e_n\}$ is a basis, then $f(e_i, e_j) (\forall i, j : 1 \leq i, j \leq n)$ identifies $f$.

- bilinear $\Rightarrow$ biaffine: adding a prior
- $u_i/v_i$ $\Rightarrow$ as functor/argument
- $+j$ (fixed window) $\Rightarrow$ the whole sentence
Representational: directly evaluate the target structure

Maximum Subgraph Parsing

Start from a directed graph $G = (V, E)$ and a score function that evaluates the goodness of a graph.

Search for a subgraph $G' = (V, E' \subseteq E)$ that maximizes:

$$G' = \arg \max_{G^* = (V, E^* \subseteq E)} \text{SCORE}(G^*)$$

First-order factorization

$$G' = \arg \max_{G^* = (V, E^* \subseteq E)} \sum_{e \in E^*} \text{SCORE}_{\text{PART}}(e)$$
Reading and exercise

- Pre-lecture 9 exercise: annotating bi-lexical semantic graphs for the following sentences:
  - His words came after Ukraine’s president urged calm, saying the biggest enemy was panic.
  - Moscow, with more than 100,000 troops near the border, has denied it plans to invade.
References I


