Natural Language Learning with Formal Semantics

Wenhu Chen
Outline

1. Introduction
2. Formal Semantics
3. Its Application in NLP
4. My Work & Future Direction
Meaning
Linguists can barely reach an agreement on the definition of “meaning”
Meaning

1. Simple
2. Straightforward

Bag of words
Unigram/Bigram
Meaning

Raw-String

1. Simple
2. Straightforward

Bag of words
Unigram/Bigram

Formal Semantics

1. Compositional
2. Explainable

Lambda Calculus
Montague Semantics
Meaning

**Raw-String**
1. Simple
2. Straightforward

   Bag of words
   Unigram/Bigram

**Formal Semantics**
1. Compositional
2. Explainable

   Lambda Calculus
   Montague Semantics

**Real-valued Vector**
1. Data-Drive
2. Computation Efficient

   Word2Vec/Glove
   BERT
Meaning

What aspects do you need the meaning representation to capture?
What aspects do you need the meaning representation to capture?

- Can it lean meaning or just similarity between meaning?
- Can it lean grounded meaning or learn lexical meaning?
1. **Raw-String Representation**

- Cannot directly link to the external world knowledge.
- Cannot combine pieces of knowledge together to perform inference.

Utterance 1: John has an apple
Utterance 2: John has a banana
?
Utterance: John has an apple and a banana.
1. Raw-String Representation

- Assuming we want to build an NLP system to understand math questions?
  a. Such system cannot leverage the mathematical knowledge.
  b. The sample complexity becomes extremely high.

Utterance: What is the largest prime less than 10?
2. Formal Semantics (Montague)

- We construct the meaning of natural language in terms of variables, predicates, arguments.
2. Formal Semantics (Montague)

- We construct the meaning of natural language in terms of variables, predicates, arguments.
- Formal Semantics aim to use explicit and grounded meaning representation to convey information.
2. Formal Semantics (Montague)

- We construct the meaning of natural language in terms of variables, predicates, arguments.
- Formal Semantics aim to use explicit and grounded meaning representation to convey information.
- The formal semantics can help us understand human language from a mathematical logic perspective.
2. Formal Semantics (Montague)

$\max(\text{primes} \cap (-\infty; 10))$

Utterance: What is the largest prime less than 10?
3. Vector Representation

- Data-driven learning process
- Suitable for deep learning framework
3. Vector Representation

- Data-driven learning process
- Suitable for deep learning framework
3. Vector Representation

We don’t need to worry about the structure of computation in the middle.
3. Vector Representation

Tight coupling between machine learning and representation means there’s always risk that some new semantic phenomenon arises and suddenly our model is useless.
## Evaluation Chart

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
<th>Similarity Meaning</th>
<th>Grounded Meaning</th>
<th>Lexical Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Vector</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Evaluation Chart

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
<th>Similarity Meaning</th>
<th>Grounded Meaning</th>
<th>Lexical Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Vector</td>
<td>?</td>
<td>✓</td>
<td>?</td>
<td>✓</td>
</tr>
<tr>
<td>Formal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Evaluation Chart

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
<th>Similarity Meaning</th>
<th>Grounded Meaning</th>
<th>Lexical Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Vector</td>
<td>?</td>
<td>✓</td>
<td>?</td>
<td>✓</td>
</tr>
<tr>
<td>Formal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Formal Semantics

- For a long time, such an explicit meaning representation has dominated the research of NLP community
  - Reasoning
  - Perception
  - Action
Formal Semantics

- For a long time, such an explicit meaning representation has dominated the research of NLP community
  - Reasoning
  - Perception
  - Action

Utterance → Parser
Formal Semantics

- For a long time, such an explicit meaning representation has dominated the research of NLP community
  - Reasoning
  - Perception
  - Action

Utterance → Parser → Semantic Form1 → Question Answering
Utterance → Parser → Semantic Form2 → Language Navigation
Utterance → Parser → Semantic Form3 → Textual Inference
List of Semantic Forms

- Introduction of formal semantics
  - Lambda Calculus (Turing 1937)
  - Combinatory categorial grammar (CCG) (Steedman et al. 1996, 2000)
  - Weighted linear CCG (Clark & Currant et. al 2007)
  - Dependency-based compositional semantics (DCS) (Liang et al. 2011)
  - Lambda DCS (Liang et al. 2013)
  - Abstract Meaning Representation (Banarescu et al. 2013)
  - Domain Specific Languages (DSL)
    - Functional program semantics (Liang et al. 2017, Dawn et al. 2018)
    - SQL, etc (Zhong et al. 2017, Yu et al. 2018)
Semantic Forms

- Combinatory Categorial Grammar
  - Definition
  - Rules
  - Learning

- Dependency-based compositional semantics
  - Definition
  - Rules
  - Learning

- Neural Symbolic Machine (Functional-Program)
  - Definition
  - Learning

Yoav et. al, Learning Compact Lexicons for CCG Semantic Parsing, 2014
Yoav et. al, Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions, 2013
Luke et. al, Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars, 2005
Steedman et. al, Surface Structure and Interpretation. 1996
Combinatory Categorial Grammar

- Categorial Formalism
  - Compositional
  - Puts information on the words
Combinatory Categorial Grammar

- Categorial Formalism
  - Compositional
  - Puts information on the words
- Transparent interface between syntax and semantics
Combinatory Categorial Grammar

- Categorial Formalism
  - Compositional
  - Puts information on the words
- Transparent interface between syntax and semantics
- Including Predicate-argument structure, quantification and information structure.
Combinatory Categorial Grammar

- Categorial Formalism
  - Compositional
  - Puts information on the words
- Transparent interface between syntax and semantics
- Including Predicate-argument structure, quantification and information structure.
- Same expressive power as lambda calculus.
CCG Categories

- Basic Building Block
- Capture Syntactic and Semantic Information Jointly

\[
\text{ADJ} : \lambda x. \text{func}(x)
\]
CCG Categories

- Basic Building Block
- Capture Syntactic and Semantic Information Jointly

Syntax ADJ : \( \lambda x. \text{func}(x) \) Semantic
## CCG Categories

- Primitive symbols: N, S, NP, ADJ and PP
- Syntactic combination operation (/,\)
- Slashes specify argument order

<table>
<thead>
<tr>
<th>Syntax</th>
<th>ADJ</th>
<th>Semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>(S\NP)/ADJ</td>
<td>$\lambda f. \lambda x. f(x)$</td>
</tr>
<tr>
<td>Syntax</td>
<td>NP</td>
<td>S</td>
</tr>
</tbody>
</table>
CCG Lexical Entries
- Pair words and phrases with meaning
- Meaning captured by a CCG category

<table>
<thead>
<tr>
<th>Natural Language</th>
<th>CCG Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>fun</td>
<td>$\lambda x. \text{fun}(x)$</td>
</tr>
</tbody>
</table>
CCG Lexical Entries

- Pair words and phrases with meaning
- Meaning captured by a CCG category

<table>
<thead>
<tr>
<th>word</th>
<th>I-</th>
<th>ADJ:</th>
<th>(S\NP)/ADJ:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fun</td>
<td>I-</td>
<td>$\lambda x. \text{fun}(x)$</td>
<td>$\lambda f \cdot \lambda x. f(x)$</td>
</tr>
<tr>
<td>is</td>
<td>I-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCG</td>
<td>I-</td>
<td>NP: CCG</td>
<td></td>
</tr>
</tbody>
</table>
CCG Operations

- Equivalent to function application
- Two direction of application

\[
B : g \quad A \backslash B : f \Rightarrow A : f(g) \quad (<) \\
A / B : f \quad B : g \Rightarrow A : f(g) \quad (>)
\]
CCG Operations

- Equivalent to function application
- Two direction of application

\[
\begin{align*}
B : g & \quad \rightarrow & \quad A \setminus B : f & \quad \Rightarrow & \quad A : f(g) \\
A/B : f & \quad \Rightarrow & \quad B : g & \quad \Rightarrow & \quad A : f(g)
\end{align*}
\]
CCG Parsing

- Use Lexicon to match words and phrases with their CCG categories
- Combine categories using operators

<table>
<thead>
<tr>
<th>CCG</th>
<th>is</th>
<th>fun</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>$S\backslash NP/ADJ$</td>
<td>$ADJ$</td>
</tr>
<tr>
<td>CCG</td>
<td>$\lambda f.\lambda x. f(x)$</td>
<td>$\lambda x. fun(x)$</td>
</tr>
</tbody>
</table>
CCG Parsing

- Use Lexicon to match words and phrases with their CCG categories
- Combine categories using operators

\[
\begin{array}{ccc}
\text{CCG} & \text{is} & \text{fun} \\
NP & S\backslash NP/ADJ & ADJ \\
 & \lambda f. \lambda x. f(x) & \lambda x. \text{fun}(x) \\
CCG & S\backslash NP & \\
 & \lambda x. \text{fun}(x) \\
\end{array}
\]
CCG Parsing

- Use Lexicon to match words and phrases with their CCG categories
- Combine categories using operators

\[
\begin{array}{c|c|c}
\text{CCG} & \text{is} & \text{fun} \\
\hline
\text{NP} & S \backslash NP/ADJ & ADJ \\
\text{CCG} & \lambda f. \lambda x. f(x) & \lambda x. \text{fun}(x) \\
\hline
S \backslash NP & \lambda x. \text{fun}(x) \\
\hline
S & \text{fun}(CCG)
\end{array}
\]
Lexicon Problem

- Key component of CCG
- Same words often paired with many different categories
- Difficult to learn from limited data
Lexicon Problem

- Key component of CCG
- Same words often paired with many different categories
- Difficult to learn from limited data

The **house** dog

The dog of the **house**
Factored Lexicons

\[ \text{N: } \lambda x. \text{of}(x, ly. \text{house}(y)) \]

\[ \text{ADJ: } \lambda x. \text{house}(x) \]

\[ \text{N: } \lambda x. \text{of}(x, ly. \text{garden}(y)) \]
Factored Lexicons

\[
\text{house} \vdash \text{N: } \lambda x. \text{of}(x, ly. \text{house}(y))
\]

\[
\text{house} \vdash \text{ADJ: } \lambda x. \text{house}(x)
\]

\[
\text{garden} \vdash \text{N: } \lambda x. \text{of}(x, ly. \text{garder}(y))
\]

Lexemes

(garden, \{garden\})

(house, \{house\})

Templates

\[\lambda(\omega, v_i). \]

[\[\omega \rightarrow \text{ADJ: } \lambda x. \text{of}(x, ly. v_1(y))\]]

\[\lambda(\omega, v_i). \]

[\[\omega \rightarrow \text{N: } \lambda x. v_1(x)\]]
Factored Lexicons

Original Lexicon

flight $\vdash S|NP : \lambda x.\text{flight}(x)$
flight $\vdash S|NP/(S|NP) : \lambda f.\lambda x.\text{flight}(x) \land f(x)$
flight $\vdash S|NP\backslash(S|NP) : \lambda f.\lambda x.\text{flight}(x) \land f(x)$
ground transport $\vdash S|NP : \lambda x.\text{trans}(x)$
ground transport $\vdash S|NP/(S|NP) : \lambda f.\lambda x.\text{trans}(x) \land f(x)$
ground transport $\vdash S|NP\backslash(S|NP) : \lambda f.\lambda x.\text{trans}(x) \land f(x)$

Factored Lexicon

$(\text{flight, } \{\text{flight}\})$
$(\text{ground transport, } \{\text{trans}\})$

$\lambda(\omega, \{v_i\}_{1}^{n}).[\omega \vdash S|NP : \lambda x.\text{v}_1(x)]$
$\lambda(\omega, \{v_i\}_{1}^{n}).[\omega \vdash S|NP/(S|NP) : \lambda f.\lambda x.\text{v}_1(x) \land f(x)]$
$\lambda(\omega, \{v_i\}_{1}^{n}).[\omega \vdash S|NP\backslash(S|NP) : \lambda f.\lambda x.\text{v}_1(x) \land f(x)]$
Factored Lexicons

- Capture systematic variations in word usage
- Each variation can then be applied to compact units in lexical matching
- Abstracts the compositional nature of the word
Learning (Data)

- (Natural Language, Lambda Form)

Show me flights to Boston
\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{BOSTON}) \]

I need a flight from baltimore to seattle
\[ \lambda x. \text{flight}(x) \land \text{from}(x, \text{BALTIMORE}) \land \text{to}(x, \text{SEATTLE}) \]

what ground transportation is available in san francisco
\[ \lambda x. \text{ground_transport}(x) \land \text{to_city}(x, \text{SF}) \]
Learning (Structure Prediction)

\[
\begin{align*}
\text{show} & \quad \text{me} & \quad \text{flights} & \quad \text{to} & \quad \text{Boston} \\
S/N & \lambda f \cdot f & N & \lambda x. \text{flight}(x) & \frac{PP/NP}{\lambda y. \lambda x. \text{to}(x, y)} & \frac{NP}{BOSTON} \\
\end{align*}
\[
\begin{align*}
\frac{PP}{\lambda x. \text{to}(x, BOSTON)} \\
\frac{N\backslash N}{\lambda f. \lambda x. f(x) \land \text{to}(x, BOSTON)} \\
\frac{N}{\lambda x. \text{flight}(x) \land \text{to}(x, BOSTON)} \\
\frac{S}{\lambda x. \text{flight}(x) \land \text{to}(x, BOSTON)}
\end{align*}
\]
Learning (Log-Linear Scorer)

\[
x 
\begin{array}{c}
\text{CCG} \\
NP \\
CCG \\
\frac{S\backslash NP/ADJ}{\lambda f.\lambda x.f(x)} \\
\frac{ADJ}{\lambda x.\text{fun}(x)} \\
\frac{S\backslash NP}{\lambda x.\text{fun}(x)} \\
\frac{\text{fun}(CCG)}{<}
\end{array}
\]

Lexical Ambiguity + Many parsing decisions \rightarrow Many potential trees and LFs
Learning (Log-Linear Scorer)

$X$

$y$

- Given a weighted linear model:
  - CCG lexicon $\Lambda$
  - Feature function $f : X \times Y \rightarrow \mathbb{R}^m$
  - Weights $w \in \mathbb{R}^m$
- The best parse is:
  $$y^* = \arg \max_y w \cdot f(x, y)$$
- We consider all possible parses $y$ for sentence $x$ given the lexicon $\Lambda$
**GENLEX**

\[
S \quad \lambda x.\text{flight}(x) \land \text{to}(x, \text{BOSTON}) \\
\downarrow \\
\text{GENLEX}(x, \mathcal{V}; \Lambda, \theta)
\]

<table>
<thead>
<tr>
<th>show</th>
<th>me</th>
<th>flights</th>
<th>to</th>
<th>Boston</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S/N)</td>
<td>(\lambda f.f)</td>
<td>(N)</td>
<td>(\lambda x.\text{flight}(x))</td>
<td>(\lambda y.\lambda x.\text{to}(x, y))</td>
</tr>
</tbody>
</table>
**GENLEX**

Given:
- Sentence $x$
- Validation Function $\mathcal{V}$
- Log-Linear Model $\theta$
- Lexicon $\Lambda_0$

$\text{GENLEX}(x, \mathcal{V}; \Lambda, \theta)$
**GENLEX**

\[ \text{GENLEX}(x, \mathcal{V}; \Lambda, \theta) \]

**Given:**
- Sentence \( x \)
- Validation Function \( \mathcal{V} \)
- Log-Linear Model \( \theta \)
- Lexicon \( \Lambda_0 \)

**Lexical Generation Procedure**

**Lexemes**
- (garden, \{garden\})
- (house, \{house\})

**Templates**
\[ \lambda(\omega, v_i). \quad [\omega \rightarrow \text{ADJ} : \lambda x. \omega f(x, ly. v_1(y))] \]

**Lexicon**
\( \Lambda_0 \)

**Overly Generate**
**GENLEX**

**Lexical Generation Procedure**

\[ \text{GENLEX}(x, \mathcal{V}; \Lambda, \theta) \]

Given:
- Sentence \( x \)
- Validation Function \( \mathcal{V} \)
- Log-Linear Model \( \theta \)
- Lexicon \( \Lambda_0 \)
Unified Learning Algorithm

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$ :

**Step 1:** (Lexical generation)

**Step 2:** (Update parameters)

**Output:** Parameters $\theta$ and lexicon $\Lambda$

- Online
- Input:
  $\{(x_i, V_i) : i = 1 \ldots n\}$
- 2 steps:
  - Lexical generation
  - Parameter update
Unified Learning Algorithm

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T$, $i = 1 \ldots n$:

Step 1: (Lexical generation)
Step 2: (Update parameters)

Output: Parameters $\theta$ and lexicon $\Lambda$
Lexical generation

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$ :

**Step 1:** (Lexical generation)

a. Set $\lambda_G \leftarrow \text{GENLEX}(x_i, \mathcal{V}_i; \Lambda, \theta)$, $\lambda \leftarrow \Lambda \cup \lambda_G$

b. Let $Y$ be the $k$ highest scoring parses from $\text{GEN}(x_i; \lambda)$

c. Select lexical entries from the highest scoring valid parses:
   $$\lambda_i \leftarrow \bigcup_{y \in \text{MAX}_i(Y; \theta)} \text{LEX}(y)$$

d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

**Step 2:** (Update parameters)

**Output:** Parameters $\theta$ and lexicon $\Lambda$

---

Generate a large set of potential lexical entries

- $\theta$ weights
- $x_i$ sentence
- $\mathcal{V}_i$ validation function
- $\text{GENLEX}(x_i, \mathcal{V}_i; \Lambda, \theta)$ lexical generation function
Lexical generation

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$:

**Step 1:** (Lexical generation)

a. Set $\lambda_G \leftarrow GENLEX(x_i, \mathcal{V}_i; \Lambda, \theta)$,
   
   $\lambda \leftarrow \Lambda \cup \lambda_G$

b. Let $Y$ be the $k$ highest scoring parses from $GEN(x_i; \lambda)$

c. Select lexical entries from the highest scoring valid parses:
   
   $\lambda_i \leftarrow \bigcup_{y \in \text{MAX}_i(Y; \theta)} LEX(y)$

d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

**Step 2:** (Update parameters)

**Output:** Parameters $\theta$ and lexicon $\Lambda$
Lexical generation

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$:

**Step 1:** (Lexical generation)

a. Set $\lambda_G \leftarrow GENLEX(x_i, \nu_i; \Lambda, \theta)$,
   $\lambda \leftarrow \Lambda \cup \lambda_G$

b. Let $Y$ be the $k$ highest scoring parses from $GEN(x_i; \lambda)$

c. Select lexical entries from the highest scoring valid parses:
   $\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$

d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

**Step 2:** (Update parameters)

**Output:** Parameters $\theta$ and lexicon $\Lambda$
Lexical generation

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T$, $i = 1 \ldots n$:

**Step 1:** (Lexical generation)

a. Set $\lambda_G \leftarrow GENLEX(x_i, V_i; \Lambda, \theta)$,
   
   $\lambda \leftarrow \Lambda \cup \lambda_G$

b. Let $Y$ be the $k$ highest scoring parses from $GEN(x_i; \lambda)$

c. Select lexical entries from the highest scoring valid parses:

   $\lambda_i \leftarrow \bigcup_{y \in MAXV_i(Y; \theta)} LEX(y)$

d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

**Step 2:** (Update parameters)

**Output:** Parameters $\theta$ and lexicon $\Lambda$
Update parse scorer

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$:

**Step 1:** (Lexical generation)

**Step 2:** (Update parameters)

a. Set $G_i \leftarrow MAXV_i(GEN(x_i; \Lambda); \theta)$
and $B_i \leftarrow \{e | e \in GEN(x_i; \Lambda) \land \neg \mathcal{V}_i(y)\}$

b. Construct sets of margin violating good and bad parses:
   - $R_i \leftarrow \{g | g \in G_i \land \exists b \in B_i$
     s.t. $\langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$
   - $E_i \leftarrow \{b | b \in B_i \land \exists g \in G_i$
     s.t. $\langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$

c. Apply the additive update:
   $\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r)$
   $\frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$

**Output:** Parameters $\theta$ and lexicon $\Lambda$
Update parse scorer

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T, i = 1 \ldots n$:

**Step 1:** (Lexical generation)

**Step 2:** (Update parameters)

a. Set $G_i \leftarrow MAXV_i(\text{GEN}(x_i; \Lambda); \theta)$ and $B_i \leftarrow \{e | e \in \text{GEN}(x_i; \Lambda) \land \neg V_i(y)\}$

b. Construct sets of margin violating good and bad parses:

   $R_i \leftarrow \{g | g \in G_i \land \exists b \in B_i$

   \hspace{1cm} s.t. $\langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$

   $E_i \leftarrow \{b | b \in B_i \land \exists g \in G_i$

   \hspace{1cm} s.t. $\langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$

c. Apply the additive update:

   $\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r)$

   $- \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$

**Output:** Parameters $\theta$ and lexicon $\Lambda$
Update parse scorer

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T$, $i = 1 \ldots n$ :

**Step 1:** (Lexical generation)

**Step 2:** (Update parameters)

a. Set $G_i \leftarrow \text{MAXV}_i(\text{GEN}(x_i; \Lambda); \theta)$
   and $B_i \leftarrow \{e | e \in \text{GEN}(x_i; \Lambda) \land \neg \nu_i(y)\}$

b. Construct sets of margin violating good and bad parses:
   $R_i \leftarrow \{g | g \in G_i \land \exists b \in B_i$
   $s.t. \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$
   $E_i \leftarrow \{b | b \in B_i \land \exists g \in G_i$
   $s.t. \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$

c. Apply the additive update:
   $\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r)$
   $- \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$

**Output:** Parameters $\theta$ and lexicon $\Lambda$

Update towards ‘good’ parses and against ‘bad’ parses
Update parse scorer

Initialize $\theta$ using $\Lambda_0$, $\Lambda \leftarrow \Lambda_0$

For $t = 1 \ldots T$, $i = 1 \ldots n$:

**Step 1:** (Lexical generation)

**Step 2:** (Update parameters)

a. Set $G_i \leftarrow \text{MAXV}_i(\text{GEN}(x_i; \Lambda); \theta)$
   and $B_i \leftarrow \{e | e \in \text{GEN}(x_i; \Lambda) \land \neg \nu_i(y)\}$

b. Construct sets of margin violating good and bad parses:
   $R_i \leftarrow \{g | g \in G_i \land \exists b \in B_i$
   $\quad \text{s.t.} \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$
   $E_i \leftarrow \{b | b \in B_i \land \exists g \in G_i$
   $\quad \text{s.t.} \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$

c. Apply the additive update:
   $\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r)$
   $- \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$

**Output:** Parameters $\theta$ and lexicon $\Lambda$
Semantic Forms

- Combinatory Categorial Grammar
  - Definition
  - Rules
  - Learning
- Dependency-based compositional semantics
  - Definition
  - Rules
  - Learning
- Neural Symbolic Machine (Functional-Program)
  - Definition
  - Learning

Liang et. al, Learning dependency-based compositional semantics, 2011
Liang et. al, Lambda dependency-based compositional semantics, 2013
Jonathan et. al, Semantic parsing on Freebase from question-answer pairs, 2013
Jonathan et. al, Semantic parsing via paraphrasing, 2014
Dependency-based Compositional Semantics

- Weakness of CCG Parse
  - Too much hand-coded rules for lexicon construction.
  - Learning algorithm is complicated.
  - The grammar rules are too strict.
  - Learning requires annotating logic form.
Dependency-Based Compositional Semantics (DCS)

\[ \lambda c \exists m \exists \ell \exists s . \]
\[ \text{city}(c) \land \text{major}(m) \land \text{loc}(\ell) \land \text{CA}(s) \land c_1 = m_1 \land c_1 = \ell_1 \land \ell_2 = s_1 \]

major cities in CA
Dependency-Based Compositional Semantics (DCS)

\[ \lambda c \exists m \exists \ell \exists s . \]
\[ \text{city}(c) \land \text{major}(m) \land \text{loc}(\ell) \land \text{CA}(s) \land c_1 = m_1 \land c_1 = \ell_1 \land \ell_2 = s_1 \]

major cities in CA
Dependency-Based Compositional Semantics (DCS)

\[ \lambda c \exists m \exists \ell \exists s. \\
\text{city}(c) \land \text{major}(m) \land \\
\text{loc}(\ell) \land \text{CA}(s) \land \\
c_1 = m_1 \land c_1 = \ell_1 \land \ell_2 = s_1 \]

major cities in CA
Dependency-Based Compositional Semantics (DCS)
Strong vs. Weak Supervision

**Detailed Supervision (current)**
- doesn’t scale up
- representation-dependent

What is the largest city in California?

\[
\text{expert} \rightarrow \arg\max\{c: \text{city}(c) \land \text{loc}(c, \text{CA}), \text{population}\}
\]
Strong vs. Weak Supervision

**Detailed Supervision (current)**
- doesn’t scale up
- representation-dependent

**What is the largest city in California?**

\[
\text{argmax}\{c: \text{city}(c) \land \text{loc}(c, \text{CA})\}, \text{population}\]

**Natural Supervision (new)**
- scales up
- representation-independent

**What is the largest city in California?**

non-expert

Los Angeles
Lexicon vs. Triggers

\[
\frac{S \backslash NP/ADJ}{\lambda f.\lambda x. f(x)}
\]

requires strict grammar rule during parsing and harsh type constraint
Lexicon vs. Triggers

**CCG**

\[
\frac{S \vdash NP/ADJ}{\lambda f. \lambda x. f(x)}
\]

requires strict grammar rule during parsing and harsh type constraint

**DCS**

\{(city, city), (city, state), (city, river), \ldots \}

requires only trigger words for predicates without grammar rule
Syntactic & Semantic Alignment

Utterance: major cities in CA
Syntactic & Semantic Alignment

Utterance: major cities in CA
Parsing: Lexical Triggers

What is the most populous city in CA?
## Parsing: Lexical Triggers

<table>
<thead>
<tr>
<th>city</th>
<th>state</th>
<th>river</th>
<th>city</th>
<th>state</th>
<th>river</th>
</tr>
</thead>
<tbody>
<tr>
<td>argmax</td>
<td>population</td>
<td>population</td>
<td>CA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What is the most populous city in CA?

### Lexical Triggers:

1. String match
   - *CA* \(\Rightarrow\) *CA*

2. Function words (20 words)
   - *most* \(\Rightarrow\) *argmax*

3. Nouns/adjectives
   - *city* \(\Rightarrow\) *city state river population*
Parsing: Predicates to DCS Trees (DP)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]
Parsing: Predicates to DCS Trees (DP)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]
$C_{i,j} = \text{set of DCS trees for span } [i, j]$
Denotation

Assuming we have our world knowledge containing 1) unary predicates 2) binary predicates in a database.
Denotation

How can we assign values to different nodes?
Constraint 1: the value in the each node should come from its corresponding predicate table.
Denotation

Constraint 2: the assignment of adjacent nodes need to meet the requirement in the edge.
A DCS tree assignment becomes a constraint satisfaction problem (CSP)
Constraint Satisfaction Problem

A DCS tree assignment becomes a constraint satisfaction problem (CSP)
Dynamic Programming => time complexity = $O(#\text{node})$
Learning

\[ z: \text{city} \quad \text{loc} \quad \text{CA} \]
\[ x: \quad \text{city} \quad \text{in} \quad \text{California} \]

\[
\text{features}(x, z) = \begin{pmatrix}
\text{in} & \text{loc} & 1 \\
\text{city} & \text{1} & \text{1} \\
\text{loc} & 1 \\
\ldots
\end{pmatrix} \in \mathbb{R}^d
\]

\[
\text{score}(x, z) = \text{features}(x, z) \cdot \theta
\]

\[
p(z \mid x, \theta) = \frac{e^{\text{score}(x, z)}}{\sum_{z' \in Z(x)} e^{\text{score}(x, z')}}
\]
Learning

Objective Function:

$$\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)$$

Interpretation  Semantic parsing
Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y \mid z, w) p(z \mid x, \theta)$$

Interpretation  Semantic parsing

EM-like Algorithm:

parameters $\theta$

(0.2, -1.3, ..., 0.7)

enumerate/score DCS trees

numerical optimization (L-BFGS)

$k$-best list

- tree3 ✓
- tree8 ✓
- tree6 x
- tree2 x
- tree4 x
Semantic Forms

- Combinatory Categorial Grammar
  - Definition
  - Rules
  - Learning
- Dependency-based compositional semantics
  - Definition
  - Rules
  - Learning
- Neural Symbolic Machine (Functional-Program)
  - Definition
  - Learning

Liang, Chen et al. Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision. 2017
Liang, Chen et al. Memory Augmented Policy Optimization for Program Synthesis and Semantic Parsing. 2018
Agarwal et al. Learning to Generalize from Sparse and Underspecified Rewards. 2019
Neural Symbolic Machine

- Weakness of traditional Semantic Parsing
  - The previous methods heavily rely on the predicate mapping, the mapping procedure is based on rules.
  - The coverage is problematic when dealing with more complex utterance.
  - The previous frameworks are dominated by the composition rules enforced as prior, only few parameters to learn.
  - The symbolic executor and the ranking model are separated without interaction.
Neural Symbolic Machine
Neural Symbolic Machine

- Simplified Domain Specific Language
- Almost no rules/triggers required
- Interacted Learning & Search
- Learning-driven framework
Domain Specific Language

GO (Hop M1 ParentOf) (Argmin M2 BornIn) RETURN

(Argmin M2 BornIn)

= \{ e_1 \mid e_1 \in M2, \\
\exists e_2: (e_1, \text{BornIn}, e_2) \in \mathcal{K}, \\
\forall e \in M2, e_3: (e, \text{BornIn}, e_3) \in \mathcal{K} \Rightarrow e_2 \leq e_3 \} \\
= \{ \text{m.Malia} \}
1. CCG/DCS are both general-purpose semantic form.
2. Evolve gradually towards more domain-specific languages.
3. Sacrifice some generalization, but greatly simplify the induction procedure.
Function definitions

- \((\text{Hop } v \ p)\): entities reached from \(v\) using relation \(p\)
- \((\text{ArgHop } v1 \ v2 \ p)\): entities in \(v1\) from which \(v2\) can be reached using relation \(p\)
- \((\text{ArgMin } v \ n)\): entities in \(v\) which have the lowest number in field \(n\)
- \((\text{ArgMax } v \ n)\): entities in \(v\) which have the highest number in field \(n\)
- \(...\)
- \((\text{Count } v)\): number of entities in \(v\)
Compositionality

x: Largest city in the US => y: NYC
Memory-aided Sequence

\[ x: \text{Largest city in the US} \Rightarrow y: \text{NYC} \]

The graph structure is flattened as sequence
Large Search Space

( Argmax → v1 → Population → )

Hop → v0 → Size → Elevation

Count
Compiler-aided prune

```
( Argmax v1 Population )
   ^  ^  ^
   Hop v0 Size
      Count Elevation
```

Code assistance
Semantic Parsing as Seq-to-Seq

<table>
<thead>
<tr>
<th>Key</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>R1(m.USA)</td>
</tr>
</tbody>
</table>

Entity Resolver

Largest, city, in, US
Semantic Parsing as Seq-to-Seq
Semantic Parsing as Seq-to-Seq
Semantic Parsing as Seq-to-Seq
Memory Enc-Dec Model

- Overall model is conditional distribution over all token sequences.

\[
P_\theta(z|x) = P_\theta(z_1, ..., z_T|x_1, ..., x_N) = P_\theta(z_{1:T}|x_{1:N})
\]

\[
= P_\theta(z_1|x_{1:N}) \cdot P_\theta(z_2|x_{1:N}, z_1) \cdot P_\theta(z_3|x_{1:N}, z_1, z_2) \cdot ... \cdot P_\theta(z_T|x_{1:N}, z_{1:T-1})
\]
Memory Enc-Dec Model

- Overall model is conditional distribution over all token sequences.

\[ P_\theta(z|x) = P_\theta(z_1, ..., z_T|x_1, ..., x_N) = P_\theta(z_{1:T}|x_{1:N}) \]

\[ = P_\theta(z_1|x_{1:N}) \cdot P_\theta(z_2|x_{1:N}, z_1) \cdot P_\theta(z_3|x_{1:N}, z_1, z_2) \cdot ... \cdot P_\theta(z_T|x_{1:N}, z_{1:T-1}) \]

- NSM models the probability at each time step \( P_\theta(z_2|x_{1:N}, z_1) \) conditioned on previous output tokens and utterance
Memory Enc-Dec Model

- Overall model is conditional distribution over all token sequences.

\[ P_\theta(z|x) = P_\theta(z_1, \ldots, z_T|x_1, \ldots, x_N) = P_\theta(z_{1:T}|x_{1:N}) \]

\[ = P_\theta(z_1|x_{1:N}) \cdot P_\theta(z_2|x_{1:N}, z_1) \cdot P_\theta(z_3|x_{1:N}, z_1, z_2) \cdot \ldots \cdot P_\theta(z_T|x_{1:N}, z_{1:T-1}) \]

- NSM models the probability at each time step \( P_\theta(z_2|x_{1:N}, z_1) \) conditioned on previous output tokens and utterance

- NSM uses interpreter to aid the decoder by removing impossible tokens at each step to dramatically shrink search space.
Memory Enc-Dec Model

\[ g : h_t^D \rightarrow (s_1, \ldots, s_{|E|}) \rightarrow P_t = \text{normalize}(s) \]

\[ (p_1, \ldots, p_{|E|}) = P_t \]

RETURN

\[
\begin{array}{ccc}
E_1 & s_1 \\
E_2 & s_2 \\
E_3 & s_3 \\
\vdots & \vdots \\
E_8 & s_8 \\
E_9 & s_9 \\
E_{10} & s_{10} \\
\vdots & \vdots \\
\end{array}
\]

GO  ( Hop M1 ParOf )  ( Amin M2 BornIn ) RETURN
Syntactic Constraint

\[ g : h_t^D \rightarrow (s_1, \ldots, s_{|E|}) \rightarrow P_t = \text{normalize}(s) \]

\[ (p_1, \ldots, p_{|E|}) = P_t \]
Semantic Constraint

\[ g : h_t^D \rightarrow (s_1, ..., s_{|E|}) \rightarrow P_t = \text{normalize}(s) \]

\[ (p_1, ..., p_{|E|}) = P_t \]
Augmented REINFORCE

- REINFORCE (explore)
- Iterative Maximum Likelihood (exploit)

\[ V_\theta J_{x}^{RL}(\theta) = \mathbb{E}_{z \sim P_\theta(\cdot | x)}[V_\theta \log P_\theta(z | x) \cdot F_1(y^z, y^*)] \]

\[ \approx 0.9 \cdot \frac{1}{\lambda} \sum_{n=1}^{k} P_\theta(z_n | x) \cdot V_\theta \log P_\theta(z_n | x) \cdot F_1(y^{z_n}, y^*) + \\
0.1 \cdot P_\theta(z_{best} | x) \cdot V_\theta \log P_\theta(z_{best} | x) \cdot F_1(y^{z_{best}}, y^*) \]

\[ \{z_n\} = k \cdot \arg\max_z P_\theta(z | x), \quad \lambda = \Sigma_i P_\theta(z_n | x) \]
Wrap up

- Formal Semantics does have many advantages like strong compositionality, explainability.
- However, the annotation effort or rigid structure makes it hard to scale up to large-scale domains or more realistic datasets.
- The vectorized representation in Deep Learning is still the favorable by the community.
- How to neuralize the formal semantics and apply it to modern architecture remains an open problem.
My Work

1. Natural Language Understanding on Structured Data
   a. Relation Question Answering on Knowledge Graph (Chen et al. NAACL 18)
   b. Table-based Fact Verification on semi-structured table (Chen et al. Arxiv)
   c. Multi-model Question Answering on structured scene graph (Chen et al. Arxiv)

2. Natural Language Generation on Structured Data
   a. Semantically conditioned dialog generation on structured semantic form (Chen et al. ACL 19)
Open Question

- Philosophical problem

Meaning serves for the success of downstream tasks
Open Question

- Philosophical problem

If we can already achieve high accuracy, why do we still need to care about meaning?