Outline

- Learning
  - Overview
  - Details
  - Example
  - Lexicon learning
  - Supervision signals
Outline

• Learning
  – Overview
  – Details
  – Example
  – Lexicon learning
  – Supervision signals
Supervision in syntactic parsing

Input:

\[
S \\
NP \\
NP \\
ESSLLI 2016 \\
NP \\
the known summer school \\
VP \\
V \\
is \\
VP \\
V \\
located \\
PP \\
in Bolzano \\
\]

Output:

They play football

\[
S \\
NP \\
They \\
VP \\
V \\
play \\
NP \\
football \\
\]
Supervision in semantic parsing

Input:

**Heavy supervision**

*How tall is Lebron James?*
HeightOf.LebronJames

*What is Steph Curry’s daughter called?*
ChildrenOf.StephCurry ⊓ Gender.Female

*Youngest player of the Cavaliers*
arg min(PLAYEROF.Cavaliers, BirthDateOf)

...

**Light supervision**

*How tall is Lebron James?*
203cm

*What is Steph Curry’s daughter called?*
Riley Curry

*Youngest player of the Cavaliers*
Kyrie Irving

...
Supervision in semantic parsing

Input:

**Heavy supervision**

*How tall is Lebron James?*
*HeightOf.LebronJames*

*What is Steph Curry’s daughter called?*
*ChildrenOf.StephCurry ⊢ Gender.Female*

*Youngest player of the Cavaliers*
*arg min(PlayerOf.Cavaliers, BirthDateOf)*
...

**Light supervision**

*How tall is Lebron James?*
*203cm*

*What is Steph Curry’s daughter called?*
*Riley Curry*

*Youngest player of the Cavaliers*
*Kyrie Irving*
...

Output:

*Clay Thompson’s weight*
*WeightOf.ClayThompson*

*Clay Thompson’s weight*
*ClayThompson’s weight*

*Weight of Clay Thompson*
*Weight.ClayThompson*

*205 lbs*
Learning in a nutshell

0. Define model for derivations
Learning in a nutshell

0. Define model for derivations
1. Generate candidate derivations (later)
Learning in a nutshell

0. Define model for derivations
1. Generate candidate derivations (later)
2. Label as correct and incorrect

_utterance_ Parsing Label
Learning in a nutshell

0. Define model for derivations
1. Generate candidate derivations (later)
2. Label as correct and incorrect
3. Update model to favor correct trees

"utterance" Parsing Label Update model
Training intuition

Where did Mozart tupress?

Vienna
Training intuition

Where did Mozart tupress?
PlaceOfBirth.WolfgangMozart
PlaceOfDeath.WolfgangMozart
PlaceOfMarriage.WolfgangMozart

Vienna
Training intuition

Where did Mozart tupress?

PlaceOfBirth.WolfgangMozart ⇒ Salzburg
PlaceOfDeath.WolfgangMozart ⇒ Vienna
PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna
Training intuition

Where did Mozart tupress?

PlaceOfBirth.WolfgangMozart ⇒ Salzburg
PlaceOfDeath.WolfgangMozart ⇒ Vienna
PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna
Training intuition

*Where did Mozart tupress?*

- PlaceOfBirth.WolfgangMozart ⇒ Salzburg
- PlaceOfDeath.WolfgangMozart ⇒ Vienna
- PlaceOfMarriage.WolfgangMozart ⇒ Vienna

**Vienna**

*Where did Hogarth tupress?*
Training intuition

Where did Mozart tupress?

PlaceOfBirth.WolfgangMozart ⇒ Salzburg
PlaceOfDeath.WolfgangMozart ⇒ Vienna
PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna

Where did Hogarth tupress?

PlaceOfBirth.WilliamHogarth
PlaceOfDeath.WilliamHogarth
PlaceOfMarriage.WilliamHogarth

London
Training intuition

Where did Mozart toupess?

\[
\text{PlaceOfBirth.WolfgangMozart} \Rightarrow \text{Salzburg}
\]
\[
\text{PlaceOfDeath.WolfgangMozart} \Rightarrow \text{Vienna}
\]
\[
\text{PlaceOfMarriage.WolfgangMozart} \Rightarrow \text{Vienna}
\]

Vienna

Where did Hogarth toupess?

\[
\text{PlaceOfBirth.WilliamHogarth} \Rightarrow \text{London}
\]
\[
\text{PlaceOfDeath.WilliamHogarth} \Rightarrow \text{London}
\]
\[
\text{PlaceOfMarriage.WilliamHogarth} \Rightarrow \text{Paddington}
\]

London
Training intuition

*Where did Mozart type press?*

- PlaceOfBirth.WolfgangMozart ⇒ Salzburg
- PlaceOfDeath.WolfgangMozart ⇒ Vienna
- PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna

*Where did Hogarth type press?*

- PlaceOfBirth.WilliamHogarth ⇒ London
- PlaceOfDeath.WilliamHogarth ⇒ London
- PlaceOfMarriage.WilliamHogarth ⇒ Paddington

London
Training intuition

Where did Mozart tupress?
PlaceOfBirth.WolfgangMozart ⇒ Salzburg
PlaceOfDeath.WolfgangMozart ⇒ Vienna
PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna

Where did Hogarth tupress?
PlaceOfBirth.WilliamHogarth ⇒ London
PlaceOfDeath.WilliamHogarth ⇒ London
PlaceOfMarriage.WilliamHogarth ⇒ Paddington

London
Outline

• Learning
  – Overview
  – Details
  – Example
  – Lexicon learning
  – Supervision signals
Constructing derivations

Type.Person \( \cap \) PlaceLived.Chicago

\[ \text{intersect} \]

\[ \text{Type.Person} \]
\[ \text{who} \]
\[ \text{PlaceLived.Chicago} \]

\[ \text{join} \]

\[ \text{lexicon} \]

\[ \text{people} \]

\[ \text{PlaceLived} \]
\[ \text{Chicago} \]

\[ \text{lexicon} \]

\[ \text{lived in} \]
\[ \text{Chicago} \]
Many possible derivations!

\[ x = \text{people who have lived in Chicago} \]

set of candidate derivations \( D(x) \)
Feature vector and parameters in $\mathbb{R}^F$:

$$\phi(x, d) \quad \theta \quad \leftarrow\text{learned}$$

- apply join: 1 1.2
- apply intersect: 1 0.6
- apply lexicon: 3 2.1
- $lived$ maps to $\text{PlacesLived}$: 1 3.1
- $lived$ maps to $\text{PlaceOfBirth}$: 0 -0.4
- $born$ maps to $\text{PlaceOfBirth}$: 0 2.7
- ... ... ...

$x$: utterance

$d$: derivation

Type(Person $\cap$ PlaceLived.Chicago)

intersect

Type.Person $\quad who$ PlaceLived.Chicago

join

people

PlaceLived

Chicago

lived in

Chicago

lexicon

lexicon

lexicon
Feature vector and parameters in $\mathbb{R}^F$:

$$\phi(x, d) \quad \theta \quad \leftarrow\text{learned}$$

- apply join: 1, 1.2
- apply intersect: 1, 0.6
- apply lexicon: 3, 2.1

- *lived* maps to *PlacesLived*: 1, 3.1
- *lived* maps to *PlaceOfBirth*: 0, -0.4
- *born* maps to *PlaceOfBirth*: 0, 2.7

$$\text{Score}_\theta(x, d) = \phi(x, d)^\top \theta =$$

$$1.2 \cdot 1 + 0.6 \cdot 1 + 2.1 \cdot 3 + 3.1 \cdot 1 + -0.4 \cdot 0 + 2.7 \cdot 0 + \ldots$$
Deep learning alert!

The feature vector $\phi(x, d)$ is constructed by hand.
Deep learning alert!

The feature vector $\phi(x, d)$ is constructed by hand

Constructing good features is hard
Deep learning alert!

The feature vector $\phi(x, d)$ is constructed by hand.

Constructing good features is hard.

Algorithms are likely to do it better.
Deep learning alert!

The feature vector $\phi(x, d)$ is constructed by hand

Constructing good features is hard

Algorithms are likely to do it better

Perhaps we can train $\phi(x, d)$

$\phi(x, d) = F_\psi(x, d)$, where $\psi$ are the parameters
Log-linear model

Candidate derivations: $D(x)$

Model: distribution over derivations $d$ given utterance $x$

$$p_\theta(d \mid x) = \frac{\exp(\text{Score}_\theta(x,d))}{\sum_{d' \in D(x)} \exp(\text{Score}_\theta(x,d'))}$$
Log-linear model

Candidate derivations: \( D(x) \)

Model: distribution over derivations \( d \) given utterance \( x \)

\[
p_\theta(d \mid x) = \frac{\exp(\text{Score}_\theta(x,d))}{\sum_{d' \in D(x)} \exp(\text{Score}_\theta(x,d'))}
\]

\( \text{score}_\theta(x,d) \) [1, 2, 3, 4]

\[
p_\theta(d \mid x) = \left[ \frac{e}{e+e^2+e^3+e^4}, \frac{e^2}{e+e^2+e^3+e^4}, \frac{e^3}{e+e^2+e^3+e^4}, \frac{e^4}{e+e^2+e^3+e^4} \right]
\]
Log-linear model

Candidate derivations: \( D(x) \)

Model: distribution over derivations \( d \) given utterance \( x \)

\[
p_\theta(d \mid x) = \frac{\exp(\text{Score}_\theta(x, d))}{\sum_{d' \in D(x)} \exp(\text{Score}_\theta(x, d'))}
\]

\( \text{score}_\theta(x, d) \)

\[
p_\theta(d \mid x) = \left[ \frac{e}{e+e^2+e^3+e^4}, \frac{e^2}{e+e^2+e^3+e^4}, \frac{e^3}{e+e^2+e^3+e^4}, \frac{e^4}{e+e^2+e^3+e^4} \right]
\]

Parsing: find the top-\( K \) derivation trees \( D_\theta(x) \)
Features

Dense features:

- intersection=0.67
- ent-popularity:HIGH
- denoation-size:1
Features

Dense features:

• intersection=0.67
• ent-popularity:HIGH
• denoation-size:1

Sparse features:

• bridge-binary:STUDY
• born:PlaceOfBirth
• city:Type.Location
Features

Dense features:

- intersection=0.67
- ent-popularity:HIGH
- denotation-size:1

Sparse features:

- bridge-binary:STUDY
- born:PlaceOfBirth
- city:Type.Location

Syntactic features:

- ent-pos:NNP NNP
- join-pos:V NN
- skip-pos:IN
Features

Dense features:

• intersection=0.67
• ent-popularity:HIGH
• denotation-size:1

Sparse features:

• bridge-binary:STUDY
• born:PlaceOfBirth
• city:Type.Location

Syntactic features:

• ent-pos:NNP NNP
• join-pos:V NN
• skip-pos:IN

Grammar features:

• Binary->Verb
Learning $\theta$: maximum-likelihood

Training data:

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s Bulgaria’s capital?</td>
<td>Sofia</td>
</tr>
<tr>
<td>What movies has Tom Cruise been in?</td>
<td>TopGun, VanillaSky, ...</td>
</tr>
<tr>
<td>What’s Bulgaria’s capital?</td>
<td>CapitalOf.Bulgaria</td>
</tr>
<tr>
<td>What movies has Tom Cruise been in?</td>
<td>Type.Movie $\sqcap$ HasPlayed.TomCruise</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Learning $\theta$: maximum-likelihood

Training data:

<table>
<thead>
<tr>
<th>What's Bulgaria's capital?</th>
<th>Sofia</th>
</tr>
</thead>
</table>
| What movies has Tom Cruise been in? | TopGun, VanillaSky, ...

\[ \arg \max_\theta \sum_{i=1}^{n} \log p_\theta(y^{(i)} | x^{(i)}) = \]

\[ \arg \max_\theta \sum_{i=1}^{n} \log \sum_{d^{(i)}} p_\theta(d^{(i)} | x^{(i)}) R(d^{(i)}) \]
Learning $\theta$: maximum-likelihood

Training data:

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What's Bulgaria's capital?</td>
<td>Sofia</td>
</tr>
<tr>
<td>What movies has Tom Cruise been in?</td>
<td>TopGun, VanillaSky,...</td>
</tr>
</tbody>
</table>

$$
\arg \max_{\theta} \sum_{i=1}^{n} \log p_{\theta}(y^{(i)} | x^{(i)}) =
$$

$$
\arg \max_{\theta} \sum_{i=1}^{n} \log \sum_{d^{(i)}} p_{\theta}(d^{(i)} | x^{(i)}) R(d^{(i)})
$$

$$
R(d) = \begin{cases} 
1 & d.z = z^{(i)} \\
0 & \text{o/w}
\end{cases}
$$

$$
R(d) = \begin{cases} 
1 & [d.z]_K = y^{(i)} \\
0 & \text{o/w}
\end{cases}
$$

$$
R(d) = F_1([d.z]_K, y^{(i)})
$$
Optimization: stochastic gradient descent

For every example:

\[ O(\theta) = \log \sum_d p_\theta(d \mid x) R(d) \]
\[ \nabla O(\theta) = E_{q_\theta(d \mid x)}[\phi(x, d)] - E_{p_\theta(d \mid x)}[\phi(x, d)] \]
\[ p_\theta(d \mid x) \propto \exp(\phi(x, d)^\top \theta) \]
\[ q_\theta(d \mid x) \propto \exp(\phi(x, d)^\top \theta) \cdot R(d) \]
Optimization: stochastic gradient descent

For every example:

\[ O(\theta) = \log \sum_d p_\theta(d \mid x) R(d) \]
\[ \nabla O(\theta) = E_{q_\theta(d \mid x)}[\phi(x, d)] - E_{p_\theta(d \mid x)}[\phi(x, d)] \]
\[ p_\theta(d \mid x) \propto \exp(\phi(x, d)^\top \theta) \]
\[ q_\theta(d \mid x) \propto \exp(\phi(x, d)^\top \theta) \cdot R(d) \]

\[ p_\theta(\mathcal{D}(x)) = [0.2, 0.1, 0.1, 0.6] \]
\[ R(\mathcal{D}(x)) = [1, 0, 0, 1] \]
Optimization: stochastic gradient descent

For every example:

\[ O(\theta) = \log \sum_d p_\theta(d \mid x) R(d) \]
\[ \nabla O(\theta) = E_{q_\theta(d \mid x)}[\phi(x, d)] - E_{p_\theta(d \mid x)}[\phi(x, d)] \]
\[ p_\theta(d \mid x) \propto \exp(\phi(x, d)^\top \theta) \]
\[ q_\theta(d \mid x) \propto \exp(\phi(x, d)^\top \theta) \cdot R(d) \]

\[ p_\theta(\mathcal{D}(x)) = [0.2, 0.1, 0.1, 0.6] \]
\[ R(\mathcal{D}(x)) = [1, 0, 0, 1] \]
\[ q_\theta(\mathcal{D}(x)) = [0.25, 0, 0, 0.75] \]
\[ q_\theta = \frac{p_\theta}{p_\theta R} \]
Optimization: stochastic gradient descent

For every example:

\[
O(\theta) = \log \sum_d p_\theta(d \mid x) R(d)
\]
\[
\nabla O(\theta) = E_{q_\theta(d \mid x)}[\phi(x, d)] - E_{p_\theta(d \mid x)}[\phi(x, d)]
\]
\[
p_\theta(d \mid x) \propto \exp(\phi(x, d)^\top \theta)
\]
\[
q_\theta(d \mid x) \propto \exp(\phi(x, d)^\top \theta) \cdot R(d)
\]

\[
p_\theta(\mathcal{D}(x)) = [0.2, 0.1, 0.1, 0.6]
\]
\[
R(\mathcal{D}(x)) = [1, 0, 0, 1]
\]
\[
q_\theta(\mathcal{D}(x)) = [0.25, 0, 0, 0.75]
\]
\[
q_\theta = \frac{p_\theta}{p_\theta R}
\]

Gradient:

\[
0.05 \cdot \phi(x, d_1) - 0.1 \cdot \phi(x, d_2) - 0.1 \cdot \phi(x, d_3) + 0.15 \cdot \phi(x, d_4)
\]
Training

Input: \( \{x_i, y_i\}_{i=1}^{n} \)

Output: \( \theta \)
Training

Input: $\{x_i, y_i\}_{i=1}^n$

Output: $\theta$

$\theta \leftarrow 0$
Training

Input: $\{x_i, y_i\}_{i=1}^{n}$

Output: $\theta$

$\theta \leftarrow 0$

for iteration $\tau$ and example $i$

$$
\mathcal{D}(x_i) \leftarrow \text{arg max}^K (p_\theta(d \mid x_i))
$$
Training

Input: \( \{x_i, y_i\}_{i=1}^{n} \)

Output: \( \theta \)

\( \theta \leftarrow 0 \)

for iteration \( \tau \) and example \( i \)

\[
D(x_i) \leftarrow \arg\max_K \left( p_\theta(d \mid x_i) \right)
\]

\[
\theta \leftarrow \theta + \eta_{\tau,i} \left( E_{q_\theta(d \mid x_i)}[\phi(x_i, d)] - E_{p_\theta(d \mid x_i)}[\phi(x_i, d)] \right)
\]
Training

Input: \( \{x_i, y_i\}_{i=1}^{n} \)

Output: \( \theta \)

\[ \theta \leftarrow 0 \]

for iteration \( \tau \) and example \( i \)

\[ \mathcal{D}(x_i) \leftarrow \text{arg max}^K (p_\theta(d | x_i)) \]

\[ \theta \leftarrow \theta + \eta_{\tau,i} (E_{q_\theta(d|x_i)}[\phi(x_i, d)] - E_{p_\theta(d|x_i)}[\phi(x_i, d)]) \]

\( \eta_{\tau,i} \): learning rate

Regularization often added (L2, L1, ...)

15
Training (structured perceptron)

Input: \( \{x_i, y_i\}_{i=1}^n \)

Output: \( \theta \)
Training (structured perceptron)

Input: \( \{x_i, y_i\}_{i=1}^{n} \)

Output: \( \theta \)

\( \theta \leftarrow 0 \)
Training (structured perceptron)

Input: \( \{x_i, y_i\}_{i=1}^n \)

Output: \( \theta \)

\( \theta \leftarrow 0 \)

for iteration \( \tau \) and example \( i \)

\( \hat{d} \leftarrow \arg \max(p_\theta(d | x_i)) \)

\( d^* \leftarrow \arg \max(q_\theta(d | x_i)) \)
Training (structured perceptron)

Input: \(\{x_i, y_i\}_{i=1}^{n}\)

Output: \(\theta\)

\[\theta \leftarrow 0\]

for iteration \(\tau\) and example \(i\)

\[\hat{d} \leftarrow \arg\max(p_\theta(d | x_i))\]

\[d^* \leftarrow \arg\max(q_\theta(d | x_i))\]

if \([d^*]_\mathcal{K} \neq [\hat{d}]_\mathcal{K}\)

\[\theta \leftarrow \theta + \phi(x_i, d^*) - \phi(x_i, \hat{d})\]
Training (structured perceptron)

Input: \( \{x_i, y_i\}_{i=1}^{n} \)

Output: \( \theta \)

\( \theta \leftarrow 0 \)

for iteration \( \tau \) and example \( i \)

\( \hat{d} \leftarrow \arg \max (p_{\theta}(d \mid x_i)) \)

\( d^* \leftarrow \arg \max (q_{\theta}(d \mid x_i)) \)

if \([d^*]_K \neq [\hat{d}]_K\)

\( \theta \leftarrow \theta + \phi(x_i, d^*) - \phi(x_i, \hat{d}) \)

Regularization often added with weight averaging
Other simple variants exist:

- E.g., cost-sensitive max-margin training
- That is, find pairs of good and bad derivations that look different but have similar scores and update on those
Outline

• Learning
  – Overview
  – Details
  – **Example**
  – Lexicon learning
  – Supervision signals
Example

./run @mode=simple-lambdadcs \\
-Grammar.inPaths esslli_2016/class3_demo.grammar \\
-SimpleLexicon.inPaths esslli_2016/class3_demo.lexicon \\
(loadgraph geo880/geo880.kg)
size of california
size capital california
size of capital of california
california size
Exercise

Find a pair of natural language utterances that cannot be distinguished using the current feature representation

- The utterances can be not fully grammatical in English
- You can ignore the denotation feature if that helps
- Verify this in sempre (ask me how to disable features)
- Design a feature that will solve this problem
Outline

• Learning
  – Overview
  – Details
  – Example
  – Lexicon learning
  – Supervision signals
The lexicon problem

How is the lexicon generated?

- Annotation
- Exhaustive search
- String matching
- Supervised alignment
- Unsupervised alignment
- Learning
Training

Input: \( \{x_i, y_i\}_{i=1}^{n} \)

Output: \( \theta \)

\( \theta \leftarrow 0 \)

for iteration \( \tau \) and example \( i \)

**Add lexicon entries**

\[
\mathcal{D}(x_i) \leftarrow \arg \max_{K} (p_{\theta}(d \mid x_i))
\]

\[
\theta \leftarrow \theta + \eta_{\tau,i} (E_{q_{\theta}(d \mid x_i)}[\phi(x_i, d)] - E_{p_{\theta}(d \mid x_i)}[\phi(x_i, d)])
\]

\( \eta_{\tau,i} \): learning rate

Regularization often added (L2, L1, ...)

23
Adding lexicon entries

Input: training example \((x_i, y_i)\), current lexicon \(\Lambda\), model \(\theta\)

\[
\Lambda_{\text{temp}} \leftarrow \Lambda \cup \text{GENLEX}(x_i, y_i)
\]

Create expanded temporary lexicon

[Adapted from semantic parsing tutorial, Artzi et al.]
Adding lexicon entries

Input: training example \((x_i, y_i)\), current lexicon \(\Lambda\), model \(\theta\)

\[
\Lambda_{\text{temp}} \leftarrow \Lambda \cup \text{GENLEX}(x_i, y_i) \quad \text{Create expanded temporary lexicon}
\]

\[
\mathcal{D}(x_i) \leftarrow \arg \max_K (p_{\theta, \Lambda_{\text{temp}}}(d \mid x_i)) \quad \text{Parse with temporary lexicon}
\]
Adding lexicon entries

Input: training example \((x_i, y_i)\), current lexicon \(\Lambda\), model \(\theta\)

\[
\Lambda_{\text{temp}} \leftarrow \Lambda \cup \text{GENLEX}(x_i, y_i) \quad \text{Create expanded temporary lexicon}
\]

\[
\mathcal{D}(x_i) \leftarrow \arg \max_K (p_{\theta, \Lambda_{\text{temp}}}(d \mid x_i)) \quad \text{Parse with temporary lexicon}
\]

\[
\Lambda = \Lambda \cup \{l \mid l \in \hat{d}, \hat{d} \in \mathcal{D}(x_i), R(d) = 1\} \quad \text{Add entries from correct trees}
\]
Adding lexicon entries

Input: training example \((x_i, y_i)\), current lexicon \(\Lambda\), model \(\theta\)

\[
\Lambda_{\text{temp}} \leftarrow \Lambda \cup \text{GENLEX}(x_i, y_i) \quad \text{Create expanded temporary lexicon}
\]

\[
D(x_i) \leftarrow \arg \max^K (p_{\theta,\Lambda_{\text{temp}}}(d \mid x_i)) \quad \text{Parse with temporary lexicon}
\]

\[
\Lambda = \Lambda \cup \{l \mid l \in \hat{d}, \hat{d} \in D(x_i), R(d) = 1\} \quad \text{Add entries from correct trees}
\]

Overgenerate lexical entries and add promising ones
Lexicon generation

Logical form supervision:

$Largest\ state\ bordering\ California$

$\arg\max\ (Type(State) \sqcap \text{Border(California)}, \text{Area})$
Lexicon generation

Logical form supervision:

\[
\text{argmax}(\text{Type(State)} \sqcap \text{Border(California)}, \text{Area})
\]

Enumerate spans

\[
\text{Largest state bordering California}
\]

Use rules to extract sub-formulas

- Largest
  - state
- California
- Border(California)
- \( \lambda f.f(\text{California}) \)
- Area
- \( \lambda x.\text{argmax}(x, \text{Area}) \)
- \( \cdots \)

\[\text{[Zettlemoyer and Collins, 2005]}\]
Lexicon generation

Logical form supervision:

\[ \text{Largest state bordering California} \]
\[ \text{argmax} (\text{Type(State) } \sqcap \text{Border(California)}, \text{Area}) \]

Enumerate spans  Use rules to extract sub-forumulas

\begin{align*}
\text{Largest} & \quad \text{California} \\
\text{state} & \quad \text{Border(California)} \\
\text{bordering} & \quad \lambda f. f(\text{California}) \\
\text{California} & \quad \text{Area} \\
\text{Largest state} & \quad \lambda x. \text{argmax}(x, \text{Area}) \\
& \quad \ldots \\
\end{align*}

Add cross-product to lexicon
Lexicon generation

Denotation supervision:

Largest state bordering California
Arizona
Lexicon generation

Denotation supervision:

Largest state bordering California
Arizona

Enumerate spans    Generate sub-formulas from KB

Largest  California
state     Border(California)
bordering Traverse
California  
Traverse Type.Mountain

Largest state  \( \lambda x. \arg \max (x, \text{Elevation}) \)

Restrict candidates with alignment, string matching, ...
Lexicon generation

Denotation supervision:

\textit{Largest state bordering California}

Arizona

Enumerate spans \hspace{1cm} Generate sub-formulas from KB
\textit{Largest state} \hspace{1cm} California
\textit{bordering} \hspace{1cm} \textit{Border(\text{California})}
\textit{California} \hspace{1cm} \textit{ Traverse}
\textit{Largest state} \hspace{1cm} \textit{Type\text{.Mountain}}
\ldots \hspace{1cm} \ldots

Restrict candidates with alignment, string matching, \ldots

Fancier methods exist (coarse-to-fine)
Unification

Logical form supervision:
Unification

Logical form supervision:

Initialize lexicon with \((x_i, z_i)\):

- \textit{States bordering California}
- \textit{Type(State) \sqcap Border(California)}

Split lexical entry in all possible ways:
Unification

Logical form supervision:

Initialize lexicon with \((x_i, z_i)\):

\[
\text{States bordering California}
\]

\[
\text{Type(State)} \sqcap \text{Border(California)}
\]

Split lexical entry in all possible ways:

Enumerate spans

\[
\begin{align*}
\text{(states, bordering california)} \\
\text{(states bordering, california)}
\end{align*}
\]

Generate sub-formulas from KB

\[
\begin{align*}
\text{(Type(State), } \lambda x. x \sqcap \text{Border(California))} \\
\text{(} \lambda x. \text{Type(State)} \sqcap x, \text{Border(California))} \\
\text{(} \lambda f. \text{Type(State)} \sqcap f(\text{California}), \text{California) }
\end{align*}
\]

\[
\ldots
\]
Unification

For example $x_i, z_i$: [Kwiatkowski et al, 2010]
Unification

For example $x_i, z_i$:

Find highest scoring correct parse $d^*$
Unification

For example $x_i, z_i$:

Find highest scoring correct parse $d^*$

Split all lexical entries in $d^*$ in all possible ways
Unification

For example $x_i, z_i$:

1. Find highest scoring correct parse $d^*$
2. Split all lexical entries in $d^*$ in all possible ways
3. Add to lexicon lexical entry that improves parse score best

[Kwiatkowski et al, 2010]
Do we need a lexicon?

Type(State) □ Border(California)

Type(State)  Border(California)

Type State Border California

California neighbors
Do we need a lexicon?

\[
\text{Type(State)} \sqcap \text{Border(California)}
\]

Floating parse tree: generalization of bridging

**California neighbors**
Do we need a lexicon?

Type(State) ⊓ Border(California)

Type(State)    Border(California)

Type  State  Border  California

California neighbors

Floating parse tree: generalization of bridging

Perhaps with better learning and search not necessary?
Outline

• Learning
  – Overview
  – Details
  – Example
  – Lexicon learning
  – Supervision signals
Supervision signals

We discussed training from logical forms and denotations
Supervision signals

We discussed training from logical forms and denotations

Other forms of supervision have been proposed:

• Demonstrations
• Distant supervision
• Conversations
• Unsupervised
• Paraphrasing
Training from demonstrations

Input: \((x_i, s_i, t_i)\)

- \(x_i\): utterance
- \(s_i\): start state
- \(t_i\): end state

move forward until you reach the intersection

[Artzi and Zettlemoyer, 2013]
Training from demonstrations

Input: \((x_i, s_i, t_i)\)

\(x_i\): utterance
\(s_i\): start state
\(t_i\): end state

move forward until you reach the intersection

\(\lambda a. \text{move}(a) \land \text{dir}(a.\text{forward})\) \ldots

[Artzi and Zettlemoyer, 2013]
Training from demonstrations

Input: \((x_i, s_i, t_i)\)

- \(x_i\): utterance
- \(s_i\): start state
- \(t_i\): end state

Move forward until you reach the intersection

\[ \lambda a. \text{move}(a) \land \text{dir}(a.\text{forward}) \ldots \]

An instance of learning from denotations

[Artzi and Zettlemoyer, 2013]
Distant supervision

Data generation:

- Decompose declarative text to questions and answers

James Cameron is the director of Titanic

Q: X is the director of Titanic  A: James Cameron
Distant supervision

Data generation:

- Decompose declarative text to questions and answers

\[\text{James Cameron is the director of Titanic}\]

\[Q: \text{X is the director of Titanic} \quad A: \text{James Cameron}\]

\[\text{Declarative text is cheap!}\]
Distant supervision

Training:

- Use existing non-executable semantic parsers

X is the director of Titanic

[Reddy et al., 2014]
Distant supervision

Training:

- Use existing non-executable semantic parsers

\[ \lambda x. \text{director}(x) \land \text{director.of.arg1}(e, x) \land \text{director.of.arg2}(e, \text{Titanic}) \]
Distant supervision

Training:

- Use existing non-executable semantic parsers

\[ X \text{ is the director of Titanic} \]

\[
\lambda x. \text{director}(x) \land \text{director.of.arg1}(e, x) \land \text{director.of.arg2}(e, \text{Titanic})
\]

\[
\lambda x. \text{Director}(x) \land \text{FilmDirectedBy}(e, x) \land \text{FileDirected}(e, \text{Titanic})
\]

\[
\lambda x. \text{Producer}(x) \land \text{FilmProducedBy}(e, x) \land \text{FileProduced}(e, \text{Titanic})
\]

[Reddy et al., 2014]
Distant supervision

Training:

- Use existing non-executable semantic parsers

\[ X \text{ is the director of } \text{Titanic} \]

\[ \lambda x. \text{director}(x) \land \text{director.of.arg1}(e, x) \land \text{director.of.arg2}(e, \text{Titanic}) \]

\[ \lambda x. \text{Director}(x) \land \text{FilmDirectedBy}(e, x) \land \text{FilmDirected}(e, \text{Titanic}) \]

\[ \lambda x. \text{Producer}(x) \land \text{FilmProducedBy}(e, x) \land \text{FileProduced}(e, \text{Titanic}) \]

James Cameron \quad true

James Cameron, Jon Landau \quad false

[Reddy et al., 2014]
Training from conversations

**System:** how can i help you? (OPEN_TASK)

**User:** i would like to fly from atlanta georgia to london england on september twenty fourth in the early evening i would like to return on october first departing from london in the late morning

**System:** leaving what city? (ASK: λx.from(fl, x))

**User:** atlanta georgia

**System:** leaving atlanta. (CONFIRM: from(fl, ATL)) going to which city? (ASK: λx.to(fl, x))

**User:** london

[conversation continues]
Training from conversations

\[ z_1 : \text{From}(\text{Atlanta}) \sqcap \text{To}(\text{London}) \]
\[ z_2 : \text{From}(\text{Atlanta}) \sqcap \text{From}(\text{London}) \]
\[ z_3 : \text{To}(\text{Atlanta}) \sqcap \text{To}(\text{London}) \]
\[ z_4 : \text{To}(\text{Atlanta}) \sqcap \text{From}(\text{London}) \]
Training from conversations

\[ \begin{align*}
&z_1 : \text{From(Atlanta)} \sqcap \text{To(London)} \\
&z_2 : \text{From(Atlanta)} \sqcap \text{From(London)} \\
&z_3 : \text{To(Atlanta)} \sqcap \text{To(London)} \\
&z_4 : \text{To(Atlanta)} \sqcap \text{From(London)}
\end{align*} \]

Define loss:

- Does \( z \) align with conversation?
- Does \( z \) obey domain constraints?
Unsupervised learning

Intuition: assume repeating patterns are correct
Unsupervised learning

Intuition: assume repeating patterns are correct

Input: \( \{ x_i \}_{i=1}^n \)

Output: \( \theta \)

[Goldwasser et al., 2011]
Unsupervised learning

Intuition: assume repeating patterns are correct

Input: $\{x_i\}_{i=1}^n$

Output: $\theta$

$\theta$ initialized manually, $S = \phi$

[Goldwasser et al., 2011]
Unsupervised learning

Intuition: assume repeating patterns are correct

Input: \( \{x_i\}_{i=1}^n \)

Output: \( \theta \)

\( \theta \) initialized manually, \( S = \phi \)

Until stopping criterion met

for example \( x_i \)

\[ S = S \cup (x_i, \arg \max p_\theta(d \mid x_i)) \]
Unsupervised learning

Intuition: assume repeating patterns are correct

Input: \( \{x_i\}_{i=1}^n \)

Output: \( \theta \)

\( \theta \) initialized manually, \( S = \emptyset \)

Until stopping criterion met
  for example \( x_i \)
    \( S = S \cup (x_i, \text{arg max } p_\theta(d | x_i)) \)

Compute statistics of \( S \)

\[\text{Goldwasser et al., 2011}\]
Unsupervised learning

Intuition: assume repeating patterns are correct

Input: \( \{ x_i \}_{i=1}^{n} \)

Output: \( \theta \)

\( \theta \) initialized manually, \( S = \emptyset \)

Until stopping criterion met

- for example \( x_i \)
  
  \[
  S = S \cup (x_i, \text{arg max } p_\theta(d \mid x_i))
  \]

Compute statistics of \( S \)

\( S_{\text{conf}} \leftarrow \) find confident subset
Unsupervised learning

Intuition: assume repeating patterns are correct

Input: \( \{x_i\}_{i=1}^n \)

Output: \( \theta \)

\( \theta \) initialized manually, \( S = \phi \)

Until stopping criterion met
  
  for example \( x_i \)
  
  \[ S = S \cup (x_i, \arg \max p_\theta(d | x_i)) \]

Compute statistics of \( S \)

\( S_{\text{conf}} \leftarrow \text{find confident subset} \)

Train on \( S_{\text{conf}} \)
Unsupervised learning

Intuition: assume repeating patterns are correct

Input: \( \{x_i\}_{i=1}^{n} \)

Output: \( \theta \)

\( \theta \) initialized manually, \( S = \emptyset \)

Until stopping criterion met

for example \( x_i \)

\[ S = S \cup (x_i, \text{arg max } p_\theta(d \mid x_i)) \]

Compute statistics of \( S \)

\( S_{\text{conf}} \leftarrow \text{find confident subset} \)

Train on \( S_{\text{conf}} \)

Substantially lower performance

[Goldwasser et al., 2011]
Paraphrasing

What languages do people in Brazil use?
Paraphrasing

What languages do people in Brazil use?

Type.HumanLanguage ⊓ LanguagesSpoken.Brazil ... CapitalOf.Brazil

B. and Liang, 2014
Paraphrasing

What languages do people in Brazil use?

What language is the language of Brazil? ... What city is the capital of Brazil?

Type.HumanLanguage ⊓ LanguagesSpoken.Brazil ... CapitalOf.Brazil
Paraphrasing

What languages do people in Brazil use?

What language is the language of Brazil? ... What city is the capital of Brazil?

Type.HumanLanguage ⊓ LanguagesSpoken.Brazil ... CapitalOf.Brazil
Paraphrasing

What languages do people in Brazil use?

paraphrase model

What language is the language of Brazil? ... What city is the capital of Brazil?

Type.HumanLanguage \cap LanguagesSpoken.Brazil ... CapitalOf.Brazil

Portuguese, ...
Paraphrasing

What languages do people in Brazil use?

What language is the language of Brazil?

What city is the capital of Brazil?

Type.HumanLanguage ∩ LanguagesSpoken.Brazil

Portuguese, ...

Model: $p_\theta(c, z \mid x) = p(z \mid x) \times p_\theta(c \mid x)$

Idea: train a large paraphrase model $p_\theta(c \mid x)$
Paraphrasing

What languages do people in Brazil use?

What language is the language of Brazil?

What city is the capital of Brazil?

Type.HumanLanguage \cap LanguagesSpoken.Brazil

Portuguese,

Model: \( p_\theta(c, z \mid x) = p(z \mid x) \times p_\theta(c \mid x) \)

Idea: train a large paraphrase model \( p_\theta(c \mid x) \)

More later
Summary

We saw how to train from denotations and logical forms

We see methods for inducing lexicons during training

We reviewed work on how to use even weaker forms of supervision

Still an open problem