Outline

- Parsing
  - CKY
  - Approximations
  - Some tricks
  - Learning agenda-based parsers
Parsing

We need to compute compute

\[ \text{argmax}_{d \in \mathcal{D}(x)} p_{\theta}(d | x) \]

**Inference:** Find best tree given model
Parsing

We need to compute

\[ \arg\max_{d \in D(x)} p_{\theta}(d \mid x) \]

**Inference:** Find best tree given model

\[ E_{q_{\theta}}(d \mid x) [\phi(x, d)] \]
\[ E_{p_{\theta}}(d \mid x) [\phi(x, d)] \]

**Learning:** Computing gradient

\[ \nabla O(\theta) = E_{q_{\theta}}(d \mid x) [\phi(x, d)] - E_{p_{\theta}}(d \mid x) [\phi(x, d)] \]
Outline

• Parsing
  – CKY
  – Approximations
  – Some tricks
  – Learning agenda-based parsers
Finding best tree

Can we use CKY?

Only if $f(x,y) = \sum_{r \in d} g(x, r)^\top \theta$

$r = (A, B, C, i, j, k)$ is a rule production

Why not? is $g$ sufficient?
Finding best tree

Maybe we can only have "type" features?

Root : \texttt{WonElection} (Type(Country))

Binary : \texttt{WonElection} | Set : Type(Country)

| won |

Root : \texttt{WonAward} (Type(Country))

Binary : \texttt{WonAward} | Set : Type(Country)

| won |

Type features: binary-unary type match
Finding best tree

Maybe we can only have ”type” features?

Root :\textit{WonElection}(\text{Type}(\text{Country}))

Binary :\textit{WonElection} \quad \text{Set :\text{Type}(\text{Country})}

\quad \text{won} \quad \text{country}

Root :\textit{WonAward}(\text{Type}(\text{Country}))

Binary :\textit{WonAward} \quad \text{Set :\text{Type}(\text{Country})}

\quad \text{won} \quad \text{country}

\textbf{Type features: binary-unary type match}

We can get exact decoding if augment state:

Binary-Person-Country, Binary-Person-Award, Set-Country, Set-Award

\textbf{Explosion in grammar size!}

If number of types is $T$, then $N$ grows by at least $T^2$
Finding best tree

In practice many features depend on the logical form itself

- Bridging features

If we could hold the logical form as part of the state we could use dynamic programming, but usually that is not possible
Logical form features

\[ E_{q_\theta}(d|x)[\phi(x, d)] \]
\[ = \sum_{d \in D(x)} \frac{\exp(\phi(x,d)^\top \theta) \cdot R(d)}{\sum_{d'} \exp(\phi(x,d')^\top \theta) \cdot R(d')} \phi(x, d) \]

\( R(d) \) depends on entire logical form - too large!
Logical form features

\[ E_{q \theta}(d|x) [\phi(x, d)] \]
\[ = \sum_{d \in D(x)} \frac{\exp(\phi(x,d)^\top \theta) \cdot R(d)}{\sum_{d'} \exp(\phi(x,d')^\top \theta) \cdot R(d')} \phi(x, d) \]

\( R(d) \) depends on entire logical form - too large!

Do approximate inference!
Outline

• Parsing
  – CKY
  – Approximations
  – Some tricks
  – Learning agenda-based parsers
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

\[
\begin{array}{c}
\text{abraham} \\
\text{lincoln} \\
\text{born} \\
\text{in}
\end{array}
\]
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

![Beam Parsing Diagram](image)
Beam parsing

• Keep top-$K$ derivations in every chart cell

• Combine chart cells in all possible ways
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

```
K
abraham
```

```
K
lincoln
```

```
K
born
```

```
in
```
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

```
  K  K  K  508
abraham lincoln born in
```
Beam parsing

• Keep top-$K$ derivations in every chart cell

• Combine chart cells in all possible ways
Beam parsing

• Keep top-$K$ derivations in every chart cell

• Combine chart cells in all possible ways

abraham lincoln born in

\[ K^2 \]

\[ \begin{array}{cccc}
K & K & K & K \\
\text{abraham} & \text{lincoln} & \text{born} & \text{in} \\
\end{array} \]
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

```
abraham lincoln born in
```

```2
K
K
K
K
abraham lincoln born in
```
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

\begin{center}
\begin{tikzpicture}
\draw (0,0) -- (2,2) -- (0,4) -- cycle;
\draw (1,1) -- (2,2) -- (0,3) -- cycle;
\draw (0,1) -- (2,2) -- (1,3) -- cycle;
\draw (0,2) -- (2,2) -- (1,3) -- cycle;
\node at (0,0) {$K$}; \node at (1,0) {$K$}; \node at (2,0) {$K$}; \node at (3,0) {$K$}; \node at (0,1) {$K$}; \node at (1,1) {$K$}; \node at (2,1) {$K^2$}; \node at (3,1) {$K$}; \node at (0,2) {$K$}; \node at (1,2) {$K$}; \node at (2,2) {$K^2$}; \node at (3,2) {$K$}; \node at (0,3) {$K$}; \node at (1,3) {$K$}; \node at (2,3) {$K$}; \node at (3,3) {$K$}; \node at (0,4) {$K$}; \node at (1,4) {$K$}; \node at (2,4) {$K$}; \node at (3,4) {$K$};
\node at (0.5,0.5) {$abraham$}; \node at (1.5,0.5) {$lincoln$}; \node at (2.5,0.5) {$born$}; \node at (3.5,0.5) {$in$};
\end{tikzpicture}
\end{center}
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

\[
\begin{array}{cccc}
  & K & K & K \\
K & & K & K \\
\text{abraham} & \text{lincoln} & \text{born} & \text{in}
\end{array}
\]
Beam parsing

• Keep top-$K$ derivations in every chart cell
• Combine chart cells in all possible ways

```
abraham lincoln born in
K K K K
K K K 2
8
```
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

```
abraham lincoln born in
K K K K
K K K
8
```
Beam parsing

- Keep top-\(K\) derivations in every chart cell
- Combine chart cells in all possible ways

\[
\begin{array}{cccc}
  & K^2 & K & K \\
 K & K & K & K \\
 abraham & lincoln & born & in \\
\end{array}
\]
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways
Beam parsing

• Keep top-$K$ derivations in every chart cell
• Combine chart cells in all possible ways
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

abraham lincoln born in

Diagram: 

```
  K   K
K   K   K
  K   K
K   K   K
abraham lincoln born in
```
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

\[
\begin{array}{cccc}
K & K & K & K^2 \\
K & K & K & \\
K & K & & \\
K & & & \\
abraham & lincoln & born & in
\end{array}
\]
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

```
K
K  K
K  K  K
K  K  K  K
abraham  lincoln  born  in
```
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

Building a chart cell is now $O(K^2 + L)$ and not $O(1)$
Beam parsing

- Keep top-$K$ derivations in every chart cell
- Combine chart cells in all possible ways

Building a chart cell is now $O(K^2 + L)$ and not $O(1)$

Assume root beam $\hat{D}(x) \sim D(x)$

$$E_{p_\theta} [\phi(x, d)] = \sum_{d \in \hat{D}(x)} p_\theta(d) \phi(x, d)$$

$$E_{q_\theta} [\phi(x, d)] = \sum_{d \in \hat{D}(x)} q_\theta(d) \phi(x, d)$$
Beam Parsing variants

• Keep top-$K$ for every span $(i, j)$ and category $A$

• Keep top-$K$ for every span $(i, j)$
Beam Parsing variants

- Keep top-\(K\) for every span \((i, j)\) and category \(A\)

- Keep top-\(K\) for every span \((i, j)\)

Prune cell with probability ratio
Beam Parsing variants

- Keep top-$K$ for every span $(i, j)$ and category $A$  
- Keep top-$K$ for every span $(i, j)$

Prune cell with probability ratio

for $\alpha = 10$:

$$[0.6, 0.2, 0.1, 0.05, 0.03, 0.02]$$
Beam Parsing variants

- Keep top-$K$ for every span $(i, j)$ and category $A$
- Keep top-$K$ for every span $(i, j)$

Prune cell with probability ratio for $\alpha = 10$:

$$[0.6, 0.2, 0.1]$$
Beam Parsing variants

- Keep top-\(K\) for every span \((i, j)\) and category \(A\)
- Keep top-\(K\) for every span \((i, j)\)

Prune cell with probability ratio

for \(\alpha = 10:\)

\[0.6, 0.2, 0.1\]

Learn a classifier that predict \(K\) for every chart cell, [Bodenstab et al., 2011]
Outline

• Parsing
  – CKY
  – Approximations
  – Some tricks
  – Learning agenda-based parsers
Coarse pruning

Most of parsing time is in building logical forms and extracting features

- Extracting lexical items
- Executing logical forms
- Suggesting binary predicates for bridging

Do 2-pass parsing. First only recognizing chart cells (without feature extraction), and then again pruning unnecessary chat cells.
Coarse pruning

1. Build derivations without logical forms and features (one derivation per rule application) - fast!
Coarse pruning

1. Build derivations without logical forms and features (one derivation per rule application) - fast!

2. Record cell reachability for every rule application: \((A, i, j) \rightarrow (B, i, k), (C, k, j)\)

Coarse pruning

1. Build derivations without logical forms and features (one derivation per rule application) - fast!

2. Record cell reachability for every rule application: \((A, i, j) \rightarrow (B, i, k), (C, k, j)\)

3. Collect all chart cells that are reachable from \((S, 1, n)\)
Coarse pruning

1. Build derivations without logical forms and features (one derivation per rule application) - fast!

2. Record cell reachability for every rule application: \((A, i, j) \rightarrow (B, i, k), (C, k, j)\)

3. Collect all chart cells that are reachable from \((S, 1, n)\)

4. Parse again ignoring unreachable states
Coarse pruning

1. Build derivations without logical forms and features (one derivation per rule application) - fast!

2. Record cell reachability for every rule application: \((A, i, j) \rightarrow (B, i, k), (C, k, j)\)

3. Collect all chart cells that are reachable from \((S, 1, n)\)

4. Parse again ignoring unreachable states

We know unreachable states do not contribute to root derivations over sentence
Coarse parsing

won
country

Root 0.0273
Root 0.007
Root 0.162
Binary 0.9
Set 0.7
Binary 0.1
Set 0.3
Mod 1.0
Conj 1.0
Coarse parsing

\[ \pi(0, 2, \text{Root}) \rightarrow (\pi(0, 1, \text{Binary}), \pi(1, 2, \text{Set})) \]
\[ (\pi(0, 1, \text{Set}), \pi(1, 2, \text{Binary})) \]
\[ (\pi(0, 1, \text{Set}), \pi(1, 2, \text{Mod})) \]

\[ \pi(0, 1, \text{Binary}) \rightarrow \]
\[ \pi(0, 1, \text{Set}) \rightarrow \]
\[ \pi(1, 2, \text{Binary}) \rightarrow \]
\[ \pi(1, 2, \text{Set}) \rightarrow \]
\[ \pi(1, 2, \text{Mod}) \rightarrow \]
\[ \pi(1, 2, \text{Conj}) \rightarrow \]
Unary rules in CKY

Unary rules are convenient when designing grammars

- Set $\rightarrow$ Entity
- Set $\rightarrow$ Unary
Unary rules in CKY

Unary rules are convenient when designing grammars

- Set $\rightarrow$ Entity
- Set $\rightarrow$ Unary

Assume:

rules $A \rightarrow B$ and $B \rightarrow C$

A derivation $d_{i:j}^C$

Computing $\pi(i, j, A)$ before $\pi(i, j, B)$

- Why this fails?
Unary rules

Solution:

Construct an acyclic (!) directed graph $G = (V, E)$:

$V = \mathcal{N}$

$E$ are unary rules

\[ A \rightarrow B \rightarrow C \rightarrow D \rightarrow A \]
Unary rules

Solution:

Construct an acyclic (!) directed graph $G = (V, E)$:

$V = \mathcal{N}$

$E$ are unary rules

Sort topologically edges and apply rules from end to start after binary rules

- Guaranteed to apply $B \rightarrow C$ before $A \rightarrow B$
Unary rules

Solution:

Construct an acyclic (!) directed graph $G = (V, E)$:

$V = N$

$E$ are unary rules

Sort topologically edges and apply rules from end to start after binary rules

- Guaranteed to apply $B \rightarrow C$ before $A \rightarrow B$

Need to verify grammar has no unary cycles
Outline

- Parsing
  - CKY
  - Approximations
  - Some tricks
  - **Learning agenda-based parsers**
Learning to search

We have a globally-normalized model, but our search procedure is inexact and slow

Perhaps we can make it faster with transition-style methods?

Let’s try to change the learning problem so that we explicitly learn to search
Fixed-order beam parsing

abraham lincoln born in
Fixed-order beam parsing

(lexicon) Set $\Rightarrow$ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
Fixed-order beam parsing

\( K \)

\( \text{abraham lincoln born in} \)

(lexicon) Set ⇒ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
Fixed-order beam parsing

(lexicon) Set $\Rightarrow$ Phrase

Abraham, AbeLincoln, AbrahamFilm, ...

AbeLincoln, LincolTown, LincolnFilm, ...
Fixed-order beam parsing

\( \text{lexicon} \) Set \( \Rightarrow \) Phrase \( \text{Abraham, AbeLincoln, AbrahamFilm, ...} \)

\( \text{lexicon} \) Set \( \Rightarrow \) Phrase \( \text{AbeLincoln, LincolTown, LincolnFilm, ...} \)
Fixed-order beam parsing

\[
\begin{array}{ccc}
K & K & 391 \\
abraham & lincoln & born in
\end{array}
\]

(lexicon) Set $\Rightarrow$ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set $\Rightarrow$ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary $\Rightarrow$ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set $\Rightarrow$ Phrase BornBelgium, BornThisWay, BornAlbum, ...
Fixed-order beam parsing

(lexicon) Set $\Rightarrow$ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set $\Rightarrow$ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary $\Rightarrow$ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set $\Rightarrow$ Phrase BornBelgium, BornThisWay, BornAlbum, ...
Fixed-order beam parsing

\( K \) \( K \) \( K \) \( 508 \)

\textbf{abraham} \hspace{1cm} \textbf{lincoln} \hspace{1cm} \textbf{born} \hspace{1cm} \textbf{in}

(lexicon) Set \Rightarrow Phrase \hspace{1cm} Abraham, AbeLincoln, AbrahamFilm, ...

(lexicon) Set \Rightarrow Phrase \hspace{1cm} AbeLincoln, LincolTown, LincolnFilm, ...

(lexicon) Binary \Rightarrow Phrase \hspace{1cm} BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...

(lexicon) Set \Rightarrow Phrase \hspace{1cm} BornBelgium, BornThisWay, BornAlbum, ...
Fixed-order beam parsing

(lexicon) Set ⇒ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set ⇒ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary ⇒ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set ⇒ Phrase BornBelgium, BornThisWay, BornAlbum, ...
Fixed-order beam parsing

\[
K^2
\begin{array}{cccc}
K & K & K & K \\
abraham & lincoln & born & in \\
\end{array}
\]

(lexicon) Set ⇒ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set ⇒ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary ⇒ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set ⇒ Phrase BornBelgium, BornThisWay, BornAlbum, ...
(inters) Set ⇒ Set Set AbeLincoln ⊓ AbeLincoln, ...
Fixed-order beam parsing

(lexicon) Set ⇒ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set ⇒ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary ⇒ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set ⇒ Phrase BornBelgium, BornThisWay, BornAlbum, ...
(inters) Set ⇒ Set Set AbeLincoln \(\sqcap\) AbeLincoln, ...
Fixed-order beam parsing

\[
\begin{array}{c c c c c}
K & K^2 & K & K & K \\
K & & & & \\
 \text{abraham} & \text{lincoln} & \text{born} & \text{in} & \\
\end{array}
\]

(lexicon) Set $\Rightarrow$ Phrase

\begin{align*}
&\text{Abraham, AbeLincoln, AbrahamFilm, ...} \\
&\text{AbeLincoln, LincolTown, LincolnFilm, ...} \\
&\text{BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...} \\
&\text{BornBelgium, BornThisWay, BornAlbum, ...} \\
&\text{AbeLincoln, AbeLincoln, ...} \\
&\text{BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...}
\end{align*}
Fixed-order beam parsing

\[
\begin{array}{ccc}
K & K & K \\
K & K & K \\
\text{abraham} & \text{lincoln} & \text{born} & \text{in} \\
\end{array}
\]

(lexicon) Set $\Rightarrow$ Phrase Abraham, AbeLincoln, AbrahamFilm, ...

(lexicon) Set $\Rightarrow$ Phrase AbeLincoln, LincolTown, LincolnFilm, ...

(lexicon) Binary $\Rightarrow$ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...

(lexicon) Set $\Rightarrow$ Phrase BornBelgium, BornThisWay, BornAlbum, ...

(inters) Set $\Rightarrow$ Set Set AbeLincoln $\sqcap$ AbeLincoln, ...

(join) Set $\Rightarrow$ Set Binary BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...
Fixed-order beam parsing

(lexicon) Set \Rightarrow Phrase  Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set \Rightarrow Phrase  AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary \Rightarrow Phrase  BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set \Rightarrow Phrase  BornBelgium, BornThisWay, BornAlbum, ...
(inters) Set \Rightarrow Set Set  AbeLincoln \cap AbeLincoln, ...
(join) Set \Rightarrow Set Binary  BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...
Fixed-order beam parsing

(lexicon) Set $\Rightarrow$ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set $\Rightarrow$ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary $\Rightarrow$ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set $\Rightarrow$ Phrase BornBelgium, BornThisWay, BornAlbum, ...
(inters) Set $\Rightarrow$ Set Set AbeLincoln $\cap$ AbeLincoln, ...
(join) Set $\Rightarrow$ Set Binary BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...
Fixed-order beam parsing

(lexicon) Set ⇒ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set ⇒ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary ⇒ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set ⇒ Phrase BornBelgium, BornThisWay, BornAlbum, ...
(inters) Set ⇒ Set Set AbeLincoln ⊓ AbeLincoln, ...
(join) Set ⇒ Set Binary BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...
Fixed-order beam parsing

\[
\begin{array}{c}
\text{abraham} & \text{lincoln} & \text{born} & \text{in} \\
\hline
\text{K} & \text{K} & \text{K} & \text{K} \\
\text{K} & \text{K} & \text{K} & \text{K} \\
\hline
\end{array}
\]

(lexicon) Set $\Rightarrow$ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set $\Rightarrow$ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary $\Rightarrow$ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set $\Rightarrow$ Phrase BornBelgium, BornThisWay, BornAlbum, ...
(inters) Set $\Rightarrow$ Set Set AbeLincoln $\cap$ AbeLincoln, ...
(join) Set $\Rightarrow$ Set Binary BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...
Fixed-order beam parsing

(lexicon) Set $\Rightarrow$ Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set $\Rightarrow$ Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary $\Rightarrow$ Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set $\Rightarrow$ Phrase BornBelgium, BornThisWay, BornAlbum, ...
(inters) Set $\Rightarrow$ Set Set AbeLincoln $\cap$ AbeLincoln, ...
(join) Set $\Rightarrow$ Set Binary BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...
Fixed-order beam parsing

\[
\begin{array}{cccc}
K & K & K \\
K & K & K \\
K & K & K \\
\end{array}
\]

\(\text{abraham} \quad \text{lincoln} \quad \text{born} \quad \text{in}\)

(lexicon) Set \(\Rightarrow\) Phrase
\(\text{Abraham}, \text{AbeLincoln}, \text{AbrahamFilm}, \ldots\)

(lexicon) Set \(\Rightarrow\) Phrase
\(\text{AbeLincoln}, \text{LincolTown}, \text{LincolnFilm}, \ldots\)

(lexicon) Binary \(\Rightarrow\) Phrase
\(\text{BirthPlaceOf}, \text{BirthDateOf}, \text{ReleaseDateOf}, \ldots\)

(lexicon) Set \(\Rightarrow\) Phrase
\(\text{BornBelgium}, \text{BornThisWay}, \text{BornAlbum}, \ldots\)

(inters) Set \(\Rightarrow\) Set Set
\(\text{AbeLincoln} \cap \text{AbeLincoln}, \ldots\)

(join) Set \(\Rightarrow\) Set Binary
\(\text{BirthPlaceOf.AbeLincoln}, \text{ReleaseDateOf.LincolnFilm}, \ldots\)
Fixed-order beam parsing

(lexicon) Set $\Rightarrow$ Phrase
Abraham, AbeLincoln, AbrahamFilm, ...

(lexicon) Set $\Rightarrow$ Phrase
AbeLincoln, LincolTown, LincolnFilm, ...

(lexicon) Binary $\Rightarrow$ Phrase
BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...

(lexicon) Set $\Rightarrow$ Phrase
BornBelgium, BornThisWay, BornAlbum, ...

(inters) Set $\Rightarrow$ Set Set
AbeLincoln $\cap$ AbeLincoln, ...

(join) Set $\Rightarrow$ Set Binary
BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...
Fixed-order beam parsing

(lexicon) Set \implies Phrase Abraham, AbeLincoln, AbrahamFilm, ...
(lexicon) Set \implies Phrase AbeLincoln, LincolTown, LincolnFilm, ...
(lexicon) Binary \implies Phrase BirthPlaceOf, BirthDateOf, ReleaseDateOf, ...
(lexicon) Set \implies Phrase BornBelgium, BornThisWay, BornAlbum, ...
(inters) Set \implies Set Set AbeLincoln \sqcap AbeLincoln, ...
(join) Set \implies Set Binary BirthPlaceOf.AbeLincoln, ReleaseDateOf.LincolnFilm, ...
Fixed-order beam parsing

Many improbable derivations generated
Goal: control parsing order

abraham lincoln born in
Goal: control parsing order

(lexicon) Set $\Rightarrow$ Phrase

$\text{abraham lincoln} \Rightarrow \text{AbeLincoln}$
Goal: control parsing order

\[
\begin{array}{cccc}
abraham & lincoln & born & in \\
\end{array}
\]

(lexicon) Set $\Rightarrow$ Phrase $abraham lincoln \Rightarrow$ AbeLincoln
(lexicon) Set $\Rightarrow$ Phrase $abraham lincon \Rightarrow$ LincolnMonument
Goal: control parsing order

(lexicon) Set $\Rightarrow$ Phrase  
\[
\begin{array}{c}
\text{abraham lincoln} \\
\text{born} \\
\text{in}
\end{array}
\]

\[
\begin{array}{c}
\text{abraham lincoln} \Rightarrow \text{AbeLincoln} \\
\text{abraham lincon} \Rightarrow \text{LincolnMonument} \\
\text{lincon} \Rightarrow \text{AbeLincoln}
\end{array}
\]
Goal: control parsing order

\[
\begin{array}{ccc}
\text{abraham} & \text{lincoln} & \text{born} \\
1 & 1 & \\
\text{in} & & \\
\end{array}
\]

(lexicon) Set ⇒ Phrase \( abraham \ lincoln \) ⇒ AbeLincoln
(lexicon) Set ⇒ Phrase \( abraham \ lincon \) ⇒ LincolnMonument
(lexicon) Set ⇒ Phrase \( lincon \) ⇒ AbeLincoln
(lexicon) Binary ⇒ Phrase \( born \) ⇒ BirthPlaceOf
Goal: control parsing order

\[
\begin{array}{ccc}
& 2 & \\
abraham & lincoln & born \\
& 1 & 2 \\
\end{array}
\]

- (lexicon) Set $\Rightarrow$ Phrase $abraham lincoln$ $\Rightarrow$ AbeLincoln
- (lexicon) Set $\Rightarrow$ Phrase $abraham lincon$ $\Rightarrow$ LincolnMonument
- (lexicon) Set $\Rightarrow$ Phrase $lincon$ $\Rightarrow$ AbeLincoln
- (lexicon) Binary $\Rightarrow$ Phrase $born$ $\Rightarrow$ BirthPlaceOf
- (lexicon) Binary $\Rightarrow$ Phrase $born$ $\Rightarrow$ BirthDateOf
Goal: control parsing order

2 1 3

abraham lincoln born in

(lexicon) Set ⇒ Phrase \textit{abraham lincoln} ⇒ AbeLincoln
(lexicon) Set ⇒ Phrase \textit{abraham lincon} ⇒ LincolnMonument
(lexicon) Set ⇒ Phrase \textit{lincon} ⇒ AbeLincoln
(lexicon) Binary ⇒ Phrase \textit{born} ⇒ BirthPlaceOf
(lexicon) Binary ⇒ Phrase \textit{born} ⇒ BirthDateOf
(lexicon) Binary ⇒ Phrase \textit{born} ⇒ OrganizationFounded
Goal: control parsing order

<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>abraham</td>
<td>lincoln</td>
<td>born</td>
</tr>
</tbody>
</table>

(lexicon) Set ⇒ Phrase  
(lexicon) Set ⇒ Phrase  
(lexicon) Set ⇒ Phrase  
(lexicon) Binary ⇒ Phrase  
(lexicon) Binary ⇒ Phrase  
(lexicon) Binary ⇒ Phrase  
(skip) Binary ⇒ Binary Token

abraham lincoln ⇒ AbeLincoln
abraham lincon ⇒ LincolnMonument
lincon ⇒ AbeLincoln
born ⇒ BirthPlaceOf
born ⇒ BirthDateOf
born ⇒ OrganizationFounded
born in ⇒ BirthPlaceOf
Goal: control parsing order

abraham lincoln born in 13 2

(lexicon) Set $\Rightarrow$ Phrase

abraham lincoln $\Rightarrow$ AbeLincoln

abraham lincon $\Rightarrow$ LincolnMonument

lincon $\Rightarrow$ AbeLincoln

Binary $\Rightarrow$ Phrase

born $\Rightarrow$ BirthPlaceOf

born $\Rightarrow$ BirthDateOf

born $\Rightarrow$ OrganizationFounded

Binary $\Rightarrow$ Binary Token

born in $\Rightarrow$ BirthPlaceOf

born in $\Rightarrow$ BirthDate
Goal: control parsing order

\[
\begin{array}{ccc}
\text{abraham} & \text{lincoln} & \text{born} \\
2 & 1 & 3 \\
\end{array}
\]

(lexicon) Set ⇒ Phrase  \( abraham \ lincoln \) ⇒ AbeLincoln
(lexicon) Set ⇒ Phrase  \( abraham \ lincon \) ⇒ LincolnMonument
(lexicon) Set ⇒ Phrase  \( lincon \) ⇒ AbeLincoln
(lexicon) Binary ⇒ Phrase  \( born \) ⇒ BirthPlaceOf
(lexicon) Binary ⇒ Phrase  \( born \) ⇒ BirthDateOf
(lexicon) Binary ⇒ Phrase  \( born \) ⇒ OrganizationFounded
(skip) Binary ⇒ Binary Token  \( born \ in \) ⇒ BirthPlaceOf
(skip) Binary ⇒ Binary Token  \( born \ in \) ⇒ BirthDate
(skip) Binary ⇒ Binary Token  \( born \ in \) ⇒ OrganizationFounded
Goal: control parsing order

abraham lincoln born in

(lexicon) Set ⇒ Phrase
(lexicon) Set ⇒ Phrase
(lexicon) Set ⇒ Phrase
(lexicon) Binary ⇒ Phrase
(lexicon) Binary ⇒ Phrase
(lexicon) Binary ⇒ Phrase
(skip) Binary ⇒ Binary Token
(skip) Binary ⇒ Binary Token
(skip) Binary ⇒ Binary Token
(join) Set ⇒ Set Binary

abraham lincoln ⇒ AbeLincoln
abraham lincon ⇒ LincolnMonument
lincon ⇒ AbeLincoln
born ⇒ BirthPlaceOf
born ⇒ BirthDateOf
born ⇒ OrganizationFounded
born in ⇒ BirthPlaceOf
born in ⇒ BirthDate
born in ⇒ OrganizationFounded
BirthPlaceOf.AbeLincoln
Goal: control parsing order

abraham lincoln born in

(lexicon) Set ⇒ Phrase
abraham lincoln ⇒ AbeLincoln
(lexicon) Set ⇒ Phrase
abraham lincon ⇒ LincolnMonument
(lexicon) Set ⇒ Phrase
lincon ⇒ AbeLincoln
(lexicon) Binary ⇒ Phrase
born ⇒ BirthPlaceOf
(lexicon) Binary ⇒ Phrase
born ⇒ BirthDateOf
(lexicon) Binary ⇒ Phrase
born ⇒ OrganizationFounded
(skip) Binary ⇒ Binary Token
born in ⇒ BirthPlaceOf
(skip) Binary ⇒ Binary Token
born in ⇒ BirthDate
(skip) Binary ⇒ Binary Token
born in ⇒ OrganizationFounded
(join) Set ⇒ Set Binary
BirthPlaceOf.AbeLincoln
(join) Set ⇒ Set Binary
BirthDateOf.AbeLincoln
Goal: control parsing order

abraham lincoln born in

2

1 3

K

(lexicon) Set ⇒ Phrase abraham lincoln ⇒ AbeLincoln
(lexicon) Set ⇒ Phrase abraham lincon ⇒ LincolnMonument
(lexicon) Set ⇒ Phrase lincon ⇒ AbeLincoln
(lexicon) Binary ⇒ Phrase born ⇒ BirthPlaceOf
(lexicon) Binary ⇒ Phrase born ⇒ BirthDateOf
(lexicon) Binary ⇒ Phrase born ⇒ OrganizationFounded
(skip) Binary ⇒ Binary Token born in ⇒ BirthPlaceOf
(skip) Binary ⇒ Binary Token born in ⇒ BirthDate
(skip) Binary ⇒ Binary Token born in ⇒ OrganizationFounded
(join) Set ⇒ Set Binary BirthPlaceOf.AbeLincoln
(join) Set ⇒ Set Binary BirthDateOf.AbeLincoln

Agenda-based parsing

Control order of derivation generation!

[Kay, 1986; Klein and Manning, 2003; Pauls and Klein, 2009]
Agenda-based parsing

Control order of derivation generation!

Parsing action
Choose derivation (based on $p_\theta(d \mid x)$)

[Kay, 1986; Klein and Manning, 2003; Pauls and Klein, 2009]
Agenda-based parsing

Control order of derivation generation!

Parsing action

Choose derivation (based on $p_\theta(d \mid x)$)

Execute action:
  - Add derivation to chart
Agenda-based parsing

Control order of derivation generation!

Parsing action

Choose derivation (based on $p_\theta(d \mid x)$)

Execute action:

- Add derivation to chart
- Combine and add to agenda
Agenda-based parsing

Control order of derivation generation!

 Parsing action

Choose derivation (based on $p_\theta(d \mid x)$)

Execute action:

- Add derivation to chart
- Combine and add to agenda

Stop when $K$ root derivations, rerank, and return best

[Kay, 1986; Klein and Manning, 2003; Pauls and Klein, 2009]
Learning a scoring function

Scoring function: A distribution over full derivations $p_\theta(d \mid x)$:
Learning a scoring function

Scoring function: A distribution over full derivations $p_\theta(d \mid x)$:

But search over partial derivations:

LincolnTown

USSLincoln

BirthplaceOf.Lincoln

AbeLincoln

BirthplaceOf

lincoln

lincoln

lincoln

born in

born in
Learning a scoring function

Scoring function: A distribution over full derivations $p_\theta(d \mid x)$:

But search over partial derivations:

Idea: learn explicitly to make good search choices
Markov Decision Process

state $s_1$ derivations on chart and agenda
Markov Decision Process

State $s$ derivations on chart and agenda

Action $a$ choosing an agenda derivation $p_\theta(a \mid s)$
Markov Decision Process

State $s$ derivations on chart and agenda
Action $a$ choosing an agenda derivation $p_\theta(a \mid s)$
History $h = (s_1, a_1, \ldots, a_T, s_{T+1})$ $p_\theta(h) = \prod_{t=1}^{T} p(a_t \mid s_t)$

[Sutton et al., 1999]
Markov Decision Process

State $s$ derivations on chart and agenda
Action $a$ choosing an agenda derivation $p_\theta(a \mid s)$

History $h = (s_1, a_1, \ldots, a_T, s_{T+1})$ $p_\theta(h) = \prod_{t=1}^{T} p(a_t \mid s_t)$

Objective: $\mathcal{O}(\theta) = \mathbb{E}_{p_\theta} [\text{Acc}(h)] = \sum_h p_\theta(h)\text{Acc}(h)$

Explicitly model all parsing actions in all states.

[Sutton et al., 1999]
Learning is hard!

\[ \nabla O(\theta) = \mathbb{E}_{p_\theta(h)} \left[ \text{Acc}(h) \sum_{t=1}^{T} \nabla \log p_\theta(a_t | s_t) \right] \]
Learning is hard!

\[
\nabla O(\theta) = \mathbb{E}_{p_\theta(h)} \left[ \text{Acc}(h) \sum_{t=1}^{T} \nabla \log p_\theta(a_t | s_t) \right]
\]

**Challenge:**

- Sampling from \( p_\theta \) leads to slow learning.
Learning is hard!

\[
\nabla O(\theta) = E_{p_\theta(h)} \left[ \text{Acc}(h) \sum_{t=1}^{T} \nabla \log p_\theta(a_t | s_t) \right]
\]

**Challenge:**

- Sampling from \( p_\theta \) leads to slow learning.

**Imitation Learning:** follow a proposal distribution.
Proposal distribution - local reweighting

Assume a oracle $d^*$:

$$q_\theta(a \mid s) \propto p_\theta(a \mid s) \cdot \exp[\beta \mathbb{I}[a \text{ in } d^*]]$$
Proposal distribution - local reweighting

Assume a oracle $d^*$:

$q_\theta(a \mid s) \propto p_\theta(a \mid s) \cdot \exp[\beta \mathbb{1}[a \text{ in } d^*]]$

Prefer sampling from oracle sub-derivations
Proposal distribution - compression

\[ a_4 = d^* \]

\[ a_1 \quad a_3 \]

\[ h : \]

\[ s_1 \quad s_2 \quad s_3 \quad s_4 \quad s_5 \]

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \]

compressed history \[ c(h) = (s_1, a_1, s_3, a_3, s_4, a_4) \]
Proposal distribution - compression

\[ a_4 = d^* \]

\[ \mathbf{h} : s_1 \rightarrow a_1 \rightarrow s_2 \rightarrow a_2 \rightarrow s_3 \rightarrow a_3 \rightarrow s_4 \rightarrow a_4 \rightarrow s_5 \]

compressed history \( c(\mathbf{h}) = (s_1, a_1, s_3, a_3, s_4, a_4) \)

\[ q_\theta(\mathbf{h}) = \sum_{\mathbf{h}' : c(\mathbf{h}') = \mathbf{h}} p_\theta(\mathbf{h}) \]
Proposal distribution - compression

\[ a_4 = d^* \]

\[ \begin{aligned}
    a_1 &\quad & a_3 \\
    s_1 &\quad & s_3 \\
\end{aligned} \]

compressed history \( c(h) = (s_1, a_1, s_3, a_3, s_4, a_4) \)

\[ q_\theta(h) = \sum_{h' : c(h') = h} p_\theta(h) \]

Ignore non-oracle actions
Proposal distribution

*city lincoln born*         Hodgenville, KY
Proposal distribution

*city lincoln born* Hodgenville, KY
Proposal distribution

city lincoln born Hodgenville, KY

\[ \mathcal{H} \subseteq \mathcal{H}_{KY} \]

\[ d^* \]

Type.CityTown \sqcap \text{BirthPlaceOf}.ABeLincoln

Type.CityTown

BirthPlaceOf.ABeLincoln

\[ \text{city} \]

ABeLincoln

BirthPlaceOf

\[ \text{lincoln} \]

\[ \text{born} \]
Proposal distribution

city lincoln born

Hodgenville, KY

Prefer sampling subtrees of $d^*$
Learning

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

Output: \( \theta \)
Learning

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

Output: \( \theta \)

\( \theta \leftarrow 0 \)
Learning

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

Output: $\theta$

$\theta \leftarrow 0$

for iteration $\tau$ and example $i$

$h_0 \leftarrow \text{parse } x_i \text{ sampling from model distribution}$
Input: $\{x_i, y_i\}_{i=1}^n$

Output: $\theta$

$\theta \leftarrow 0$

for iteration $\tau$ and example $i$

$h_0 \leftarrow \text{parse } x_i \text{ sampling from model distribution}$

$d^* \leftarrow \text{choose oracle from } h_0$
Learning

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

Output: \( \theta \)

\( \theta \leftarrow 0 \)

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

\( h_0 \leftarrow \) parse \( x_i \) sampling from model distribution

\( d^* \leftarrow \) choose oracle from \( h_0 \)

\( h \leftarrow \) parse \( x_i \) sampling from proposal distribution
Learning

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

Output: \( \theta \)

\( \theta \leftarrow 0 \)

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

\( h_0 \leftarrow \) parse \( x_i \) sampling from model distribution

\( d^* \leftarrow \) choose oracle from \( h_0 \)

\( h \leftarrow \) parse \( x_i \) sampling from proposal distribution

Update \( \theta \) using proposed history \( h \)
Learning

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

Output: \( \theta \)

\[ \theta \leftarrow 0 \]

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

\[ h_0 \leftarrow \text{parse } x_i \text{ sampling from model distribution} \]

\[ d^* \leftarrow \text{choose oracle from } h_0 \]

\[ h \leftarrow \text{parse } x_i \text{ sampling from proposal distribution} \]

Update \( \theta \) using proposed history \( h \)

**Test time: standard agenda-based parsing**
Summary

• Agenda-based parsing

• Learning to search

• Oracle (imitation learning) guides learning
Evaluation: WebQuestions dataset

What character did Natalie Portman play in Star Wars? ⇒ Padmé Amidala

What kind of money to take to Bahamas? ⇒ Bahamian dollar

What currency do you use in Costa Rica? ⇒ Costa Rican colón

What did Obama study in school? ⇒ political science

What do Michelle Obama do for a living? ⇒ writer, lawyer

What killed Sammy Davis Jr? ⇒ throat cancer
## Comparison with beam parsing (test)

<table>
<thead>
<tr>
<th>Test</th>
<th>Train</th>
<th>actions</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+learn +parse AgendaIL</td>
<td>49.7</td>
<td>61.1</td>
<td>1,346</td>
</tr>
<tr>
<td>-learn -parse FixedOrder</td>
<td>49.6</td>
<td>60.6</td>
<td>18,127</td>
</tr>
</tbody>
</table>

**Comparable accuracy**

**13 times fewer actions**

**6 times faster**


## Learning analysis (dev)

<table>
<thead>
<tr>
<th>Dev</th>
<th>actions</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+learn +parse AgendaIL</td>
<td>48.0</td>
<td>1,421</td>
</tr>
<tr>
<td>-learn -parse FixedOrder</td>
<td>49.1</td>
<td>18,259</td>
</tr>
<tr>
<td>-learn +parse Agenda</td>
<td>45.9</td>
<td>6,211</td>
</tr>
</tbody>
</table>

AgendaIL is faster and better
Beam size analysis (dev)
WebQuestions SOTA

[Yao, 2015] [Yih et al., 2015] [BH, 2015] [this work] [Reddy et al., 2016]

44.3 48.4 49.4 49.7 50.3 52.5

[Yao, 2015] [Yih et al., 2015] [BH, 2015] [this work] [Reddy et al., 2016] [Yih et al., 2015]