Steps to Excellence: Simple Inference with Refined Scoring of Dependency Trees

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, Amir Globerson

MIT, Hebrew University
Exact Inference vs. Expressive Scoring Function

Inference

Approximate

Exact

Limited

Scoring Function

Expressive
Exact Inference vs. Expressive Scoring Function

Inference

Approximate

Exact

Dynamic Programming

Inference

Limited

Scoring Function

Expressive
Exact Inference vs. Expressive Scoring Function

Inference

Approximate

Exact

Inference

Dynamic Programming

Limited

Scoring Function

Expressive

Reranking
Exact Inference vs. Expressive Scoring Function

- Dynamic Programming
- Dual Decomposition
- Reranking

Inference

- Exact
- Approximate

Scoring Function

- Limited
- Expressive
Exact Inference vs. Expressive Scoring Function

- Dynamic Programming
- Dual Decomposition
- Our Approach
- Reranking

- Limited
- Expressive

- Search in full parse space
- Easily incorporate arbitrary features
Our Approach

• Method: a sampling-based dependency parser
  – Decoding: climb to the optimum in small steps
  – Proposal distributions:
    ➢ Gibbs
    ➢ Metropolis-Hastings
  – Learning via SampleRank: satisfy constraints based on samples
Our Approach

• Method: a sampling-based dependency parser
  – Decoding: climb to the optimum in small steps
  – Proposal distributions:
    ➢ Gibbs
    ➢ Metropolis-Hastings
  – Learning via SampleRank: satisfy constraints based on samples

• Advantages:
  – Achieve top parsing performance
  – Readily extendable to joint prediction tasks
Sampling-Based Decoding Algorithm

• Generate a sequence of samples to climb towards the optimum in small stochastic steps

```
ROOT  I  eat  apples  y^{(0)}  -- any initial tree
```
Sampling-Based Decoding Algorithm

• Generate a sequence of samples to climb towards the optimum in small stochastic steps

\[
\begin{align*}
\text{ROOT} & \quad I \quad \text{eat} \quad \text{apples} \\
\text{ROOT} & \quad I \quad \text{eat} \quad \text{apples}
\end{align*}
\]

\[
y^{(0)} \quad \Rightarrow \quad q(y^{(1)} | x, y^{(0)}, T^{(0)}, \theta)
\]

\[
y^{(1)}
\]

\[
y^{(2)} \quad \Rightarrow \quad q(y^{(2)} | x, y^{(1)}, T^{(1)}, \theta)
\]

\[
\text{\ldots}
\]

\[
q(\cdot | x, y, T, \theta) : \text{proposal distr. which governs the climb}
\]
Sampling-Based Decoding Algorithm

- Generate a sequence of samples to climb towards the optimum in small stochastic steps

\[
y^{(0)} \quad \rightarrow 
\]

\[
y^{(1)} \quad \rightarrow 
\]

\[
y^{(2)} \quad \rightarrow 
\]

\[\vdots\]

\[
y^{(m)} \approx \arg\max_y \theta \cdot f(x, y) \quad \text{(Geman, 1984)}
\]
Proposal Distribution: Gibbs Sampling

• Change one edge each time
• Sample from a conditional distribution

\[ p(y_j \mid x, y_{-j}, T, \theta) \propto \exp(\theta \cdot f(x, y_j, y_{-j}) / T) \]
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[ p(y_j \mid x, y_{-j}, T, \theta) \propto \exp(\theta \cdot f(x, y_j, y_{-j}) / T) \]
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[ p(y_j \mid x, y_{-j}, T, \theta) \propto \exp(\theta \cdot \frac{f(x, y_j, y_{-j})}{T}) \]

- Arbitrary features in scoring function
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[
p(y_j | x, y_{-j}, T, \theta) \propto \exp(\theta \cdot \frac{f(x, y_j, y_{-j})}{T})
\]

- Arbitrary features in scoring function

ROOT I like dogs and cats
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[ p(y_j | x, y_{-j}, T, \theta) \propto \exp(\theta \cdot f(x, y_j, y_{-j}) / T) \]

- Arbitrary features in scoring function

ROOT I like dogs and cats

\[ p = 0.0 \]
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[ p(y_j \mid x, y_{-j}, T, \theta) \propto \exp(\theta f(x, y_j, y_{-j}) / T) \]

- Arbitrary features in scoring function

ROOT \rightarrow I \leftarrow like \rightarrow dogs \leftarrow and \rightarrow cats

\[ p = 0.5 \]

温度缩放 (temperature scaling)
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[
p(y_j \mid x, y_{-j}, T, \theta) \propto \exp(\theta \cdot f(x, y_j, y_{-j}) / T)
\]

- Arbitrary features in scoring function

ROOT I like dogs and cats

- Temperature scaling

\[
p = 0.3
\]

\[
p = 0.5
\]

\[
p = 0.0
\]
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[ p(y_j \mid x, y_{-j}, T, \theta) \propto \exp(\theta f(x, y_j, y_{-j}) / T) \]

- Arbitrary features in scoring function

\[
p = 0.3
\]
\[
p = 0.5
\]
\[
p = 0.5
\]
\[
p = 0.0
\]
\[
p = 0.0
\]
\[
p = 0.2
\]
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[
p(y_j | x, y_{-j}, T, \theta) \propto \exp(\theta \cdot \frac{f(x, y_j, y_{-j})}{T})
\]

- Arbitrary features in scoring function
Proposal Distribution: Gibbs Sampling

- Change one edge each time
- Sample from a conditional distribution

\[ p(y_j \mid x, y_{-j}, T, \theta) \propto \exp(\theta \cdot f(x, y_j, y_{-j}) / T) \]

➢ Arbitrary features in scoring function
Proposal Distribution: Extended MH Sampling

- Change K edges each time
Proposal Distribution: Extended MH Sampling

• Change K edges each time

• Random Walk-based sampler (Wilson, 1996):
  – Draw samples from the first-order distribution

• Acceptance probability with full scoring
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$
2: For each node not in the tree $x_i \notin T$
3: Random walk from $x_i$ until reach a node in $T$
4: Add path into the tree $T \leftarrow T \cup path$
5: End for

original tree

ROOT  I  like  dogs  and  cats
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree \( T \leftarrow \{ROOT\} \)

2: For each node not in the tree \( x_i \notin T \)

3: Random walk from \( x_i \) until reach a node in \( T \)

4: Add path into the tree \( T \leftarrow T \cup path \)

5: End for

original tree

```plaintext
ROOT
```

```
I like dogs and cats
```
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$
2: For each node not in the tree $x_i \notin T$
3: Random walk from $x_i$ until reach a node in $T$
4: Add path into the tree $T \leftarrow T \cup path$
5: End for

walk path:

ROOT  I  like  dogs  and  cats
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{\text{ROOT}\}$
2: For each node not in the tree $x_i \notin T$
3: Random walk from $x_i$ until reach a node in $T$
4: Add path into the tree $T \leftarrow T \cup path$
5: End for

walk path:

ROOT    I like dogs and cats
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree \( T \leftarrow \{\text{ROOT}\} \)

2: For each node not in the tree \( x_i \notin T \)

3: Random walk from \( x_i \) until reach a node in \( T \)

4: Add path into the tree \( T \leftarrow T \cup path \)

5: End for

walk path: I

ROOT \[ \text{I like dogs and cats} \]
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$

2: For each node not in the tree $x_i \notin T$

3: Random walk from $x_i$ until reach a node in $T$

4: Add path into the tree $T \leftarrow T \cup path$

5: End for

walk path: $I \rightarrow like$

ROOT  I  like  dogs  and  cats
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$
2: For each node not in the tree $x_i \notin T$
3: Random walk from $x_i$ until reach a node in $T$
4: Add path into the tree $T \leftarrow T \cup path$
5: End for

walk path: $I \rightarrow like \rightarrow ROOT$
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree \( T \leftarrow \{ROOT\} \)
2: For each node not in the tree \( x_i \notin T \)
3: Random walk from \( x_i \) until reach a node in \( T \)
4: Add path into the tree \( T \leftarrow T \cup path \)
5: End for

walk path: \( I \rightarrow like \rightarrow ROOT \)
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$

2: **For each** node not in the tree $x_i \not\in T$

3: Random walk from $x_i$ until reach a node in $T$

4: Add path into the tree $T \leftarrow T \cup path$

5: **End for**

*walk path: dogs*
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$
2: For each node not in the tree $x_i \notin T$
3: Random walk from $x_i$ until reach a node in $T$
4: Add path into the tree $T \leftarrow T \cup path$
5: End for

walk path: dogs $\rightarrow$ and
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$

2: For each node not in the tree $x_i \notin T$

3: Random walk from $x_i$ until reach a node in $T$

4: Add path into the tree $T \leftarrow T \cup path$

5: End for

walk path: dogs $\rightarrow$ and $\rightarrow$ like
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$

2: **For each** node not in the tree $x_i \notin T$

3: Random walk from $x_i$ until reach a node in $T$

4: Add path into the tree $T \leftarrow T \cup path$

5: **End for**

*walk path:* dogs $\rightarrow$ and $\rightarrow$ like
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree \( T \leftarrow \{ROOT\} \)

2: For each node not in the tree \( x_i \notin T \)

3: Random walk from \( x_i \) until reach a node in \( T \)

4: Add path into the tree \( T \leftarrow T \cup \text{path} \)

5: End for

walk path: cats \( \rightarrow \) and

ROOT  I  like  dogs  and  cats
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$
2: For each node not in the tree $x_i \notin T$
3: Random walk from $x_i$ until reach a node in $T$
4: Add path into the tree $T \leftarrow T \cup path$
5: End for

walk path: cats → and
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$

2: For each node not in the tree $x_i \notin T$

3: Random walk from $x_i$ until reach a node in $T$

4: Add path into the tree $T \leftarrow T \cup \text{path}$

5: End for

walk path: cats → and
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$

2: For each node not in the tree $x_i \notin T$

3: Random walk from $x_i$ until reach a node in $T$

4: Add path into the tree $T \leftarrow T \cup path$

5: End for

I like dogs and cats

new tree
Random Walk-Based Sampler (Wilson 1996)

1: Initial tree $T \leftarrow \{ROOT\}$

2: **For each** node not in the tree $x_i \notin T$

3: Random walk from $x_i$ until reach a node in $T$

4: Add path into the tree $T \leftarrow T \cup path$

5: **End for**

```
new tree
```

- Extended MH performs better than Gibbs given constrained time
- Both reach the same result given enough time
Sampling-Based Learning Algorithm

- Generate a sequence of samples

\[ y^{(0)} \xrightarrow{q(\cdot|y^{(0)})} y^{(1)} \xrightarrow{q(\cdot|y^{(1)})} y^{(2)} \xrightarrow{q(\cdot|y^{(2)})} y^{(3)} \rightarrow \ldots \]

- Satisfy two types of constraints based on random samples (SampleRank: Wick et al. 2011)
Sampling-Based Learning Algorithm

- Generate a sequence of samples

\[ y^{(0)} \xrightarrow{q(\cdot|y^{(0)})} y^{(1)} \xrightarrow{q(\cdot|y^{(1)})} y^{(2)} \xrightarrow{q(\cdot|y^{(2)})} y^{(3)} \rightarrow \ldots \]

- Satisfy two types of constraints based on random samples (SampleRank: Wick et al. 2011)

- More efficient than a standard structure learning algorithm because full decoding is not required
1) Constraints between samples and the gold tree

\[ s(x, \hat{y}) - s(x, y^{(t)}) \geq \text{Err}(y^{(t)}) \]

Score of the gold tree  Score of the sample  # errors in the sample
Constraints in Learning

1) Constraints between samples and the gold tree

\[ s(x, \hat{y}) - s(x, y^{(t)}) \geq \text{Err}(y^{(t)}) \]

- Score of the gold tree
- Score of the sample
- # errors in the sample

2) Constraints between neighboring samples

Markov chain: \[ y^{(0)} \rightarrow y^{(1)} \rightarrow y^{(2)} \rightarrow y^{(3)} \rightarrow y^{(4)} \ldots \]

If \( y^{(3)} \) is more accurate than \( y^{(2)} \),

\[ s(x, y^{(3)}) - s(x, y^{(2)}) \geq \text{Err}(y^{(2)}) - \text{Err}(y^{(3)}) \]
Constraints in Learning

1) Constraints between samples and the gold tree

\[ s(x, \hat{y}) - s(x, y^{(t)}) \geq Err(y^{(t)}) \]

- Score of the gold tree
- Score of the sample
- # errors in the sample

2) Constraints between neighboring samples

Markov chain: \[ y^{(0)} \rightarrow y^{(1)} \rightarrow y^{(2)} \rightarrow y^{(3)} \rightarrow y^{(4)} \ldots \]

- if \( y^{(3)} \) is more accurate than \( y^{(2)} \)

\[ s(x, y^{(3)}) - s(x, y^{(2)}) \geq Err(y^{(2)}) - Err(y^{(3)}) \]

- None of the samples are necessarily the argmax
First- to Third-Order Features

• Similar features used in previous work

arc

consecutive sibling

grandparent

head bigram

tri-siblings

grand-sibling

outer-sibling-grandchild

inner-sibling-grandchild
Global Features

• Conjuncts consistency
  – POS tag consistency

  ![Diagram showing correct and incorrect POS tag consistency]
Global Features

- Conjuncts consistency
  - POS tag consistency
  - Span length consistency

```
NOUN and NOUN  ✔️

NOUN and VERB ✗

NOUN and NOUN  ✔️

NOUN and NOUN ✗
```

Global Features

• Conjuncts consistency
  – POS tag consistency
    - NOUN and NOUN (Consistent)
    - NOUN and VERB (Inconsistent)
  – Span length consistency
    - NOUN and NOUN (Consistent)
    - NOUN and NOUN (Inconsistent)

• Right branching, PP attachment, neighbors, valency, non-projective arcs
Joint Parsing and POS Correction

• Task:

Philips is a company
NNS VBZ DT NN

Philips is a company
NNP VBZ DT NN
Joint Parsing and POS Correction

- **Task:**

  Philips is a company

  \[
  p(y_j, t_j \mid x, y_{-j}, t_{-j}^T, \theta) \propto \exp(\theta \cdot f(x, y_j, y_{-j}, t_j, t_{-j}) / T)
  \]

- **Our approach: simple extension of our parsing model**
  - Sample new heads \( y_j \) and POS tags \( t_j \) simultaneously
Example

Word: eat
POS tag: VB

ROOT
I
eat

ROOT
PRON

apples
NN
Example

Word: ROOT I eat
POS tag: ROOT PRON VB

Word: ROOT I eat
POS tag: ROOT PRON VB

apples NN
POS: NN ➔ NNS
Head: I ➔ eat

apples NNS
Experimental Setup for Parsing

• Dataset
  – CoNLL datasets with 14 languages

• Evaluation Metric
  – UAS: Unlabeled Attachment Score

• Pruning
  – Prune away unlikely candidate heads based on a first-order model trained by the same method
Results on CoNLL Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>UAS(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reranking</td>
<td>87.9</td>
</tr>
<tr>
<td>Turbo</td>
<td>88.7</td>
</tr>
<tr>
<td>Our Model</td>
<td>89.2</td>
</tr>
</tbody>
</table>
Results on CoNLL Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>UAS(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reranking</td>
<td>87.9</td>
</tr>
<tr>
<td>Turbo</td>
<td>88.7</td>
</tr>
<tr>
<td>Our Model</td>
<td>89.2</td>
</tr>
</tbody>
</table>

0.5% improvement
Results on CoNLL Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>UAS(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reranking</td>
<td>87.9</td>
</tr>
<tr>
<td>Turbo</td>
<td>88.7</td>
</tr>
<tr>
<td>Our Model</td>
<td>89.2</td>
</tr>
</tbody>
</table>

The difference between Turbo and Our Model is 1.3%.
Comparison with Turbo: Impact of Feature Sets

<table>
<thead>
<tr>
<th></th>
<th>UAS(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbo</td>
<td>88.7</td>
</tr>
<tr>
<td>Our Model w/ Turbo Feat.</td>
<td>88.8</td>
</tr>
<tr>
<td>Our Model w/ Full Feat.</td>
<td>89.2</td>
</tr>
</tbody>
</table>
The Effect of Constraints in Learning

- **Gold**: constraints between samples and *gold* trees
- **Neighbor**: constraints between *neighboring* samples

![Graph showing UAS(%) for Gold, Neighbor, and Both]

- Gold: 87.3%
- Neighbor: 88.6%
- Both: 89.0%
• We decode in different speed by controlling converge iterations
• Both methods achieve the same result given enough time
• Extended MH sampler performs better given constrained time
Impact of Different Proposal Distributions

Decoding Speed on Arabic

- We decode in different speed by controlling converge iterations.
- Both methods achieve the same result given enough time.
- Extended MH sampler performs better given constrained time.
• We decode in different speed by controlling converge iterations
• Both methods achieve the same result given enough time
• Extended MH sampler performs better given constrained time
Experimental Setup for Joint Prediction Task

• Arabic dataset in SPMRL 2013
  – Train: gold and predicted POS tags, gold trees
  – Test: predicted POS tags

• Evaluation Metric
  – UAS: Unlabeled Attachment Score
  – POS tagging accuracy

• POS tags candidate list
  – Generate the POS candidate list for each word based on the confusion matrix of the training set
Results on Joint Parsing and POS Correction

POS Accuracy on SPMRL Arabic dataset

- Predicted: 96.8%
- Correction: 97.5%

Improvement: 0.7%
Results on Joint Parsing and POS Correction

UAS(%) on SPMRL Arabic dataset

<table>
<thead>
<tr>
<th></th>
<th>IMS-Single</th>
<th>w/o Correction</th>
<th>w/ Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAS(%)</td>
<td>87.0</td>
<td>87.0</td>
<td>88.4</td>
</tr>
</tbody>
</table>

1.4% improvement
Conclusion

• A simple sampling-based parser that handles arbitrary features:
  – Outperform the state-of-the-art methods on the CoNLL dataset

• A simple and effective extension for joint parsing and corrective POS tagging
  – Outperform the best single system on the Arabic dataset in SPMRL 2013

Source code available at:
Thank You!