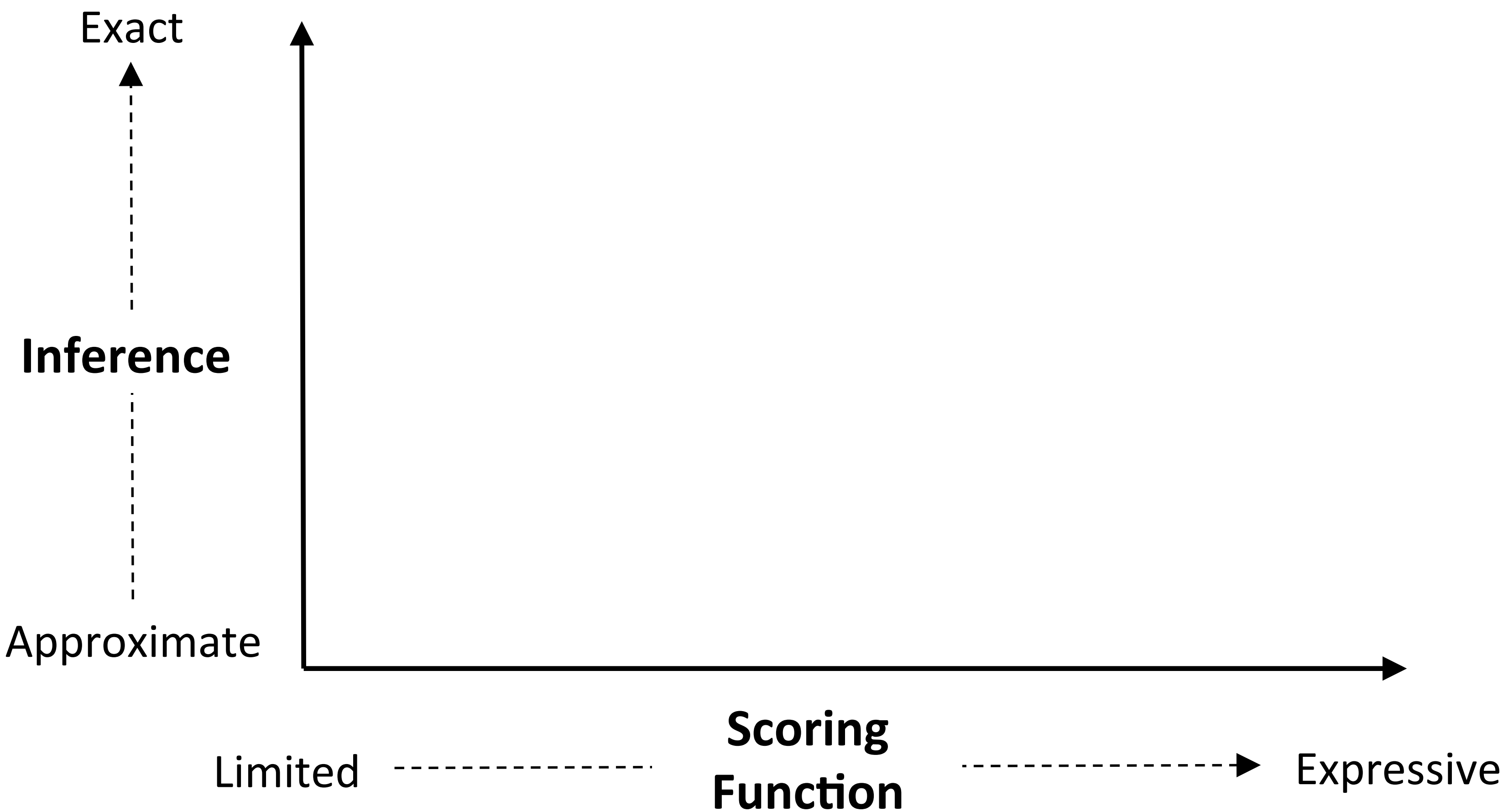
Steps to Excellence: Simple Inference with Refined Scoring of Dependency Trees

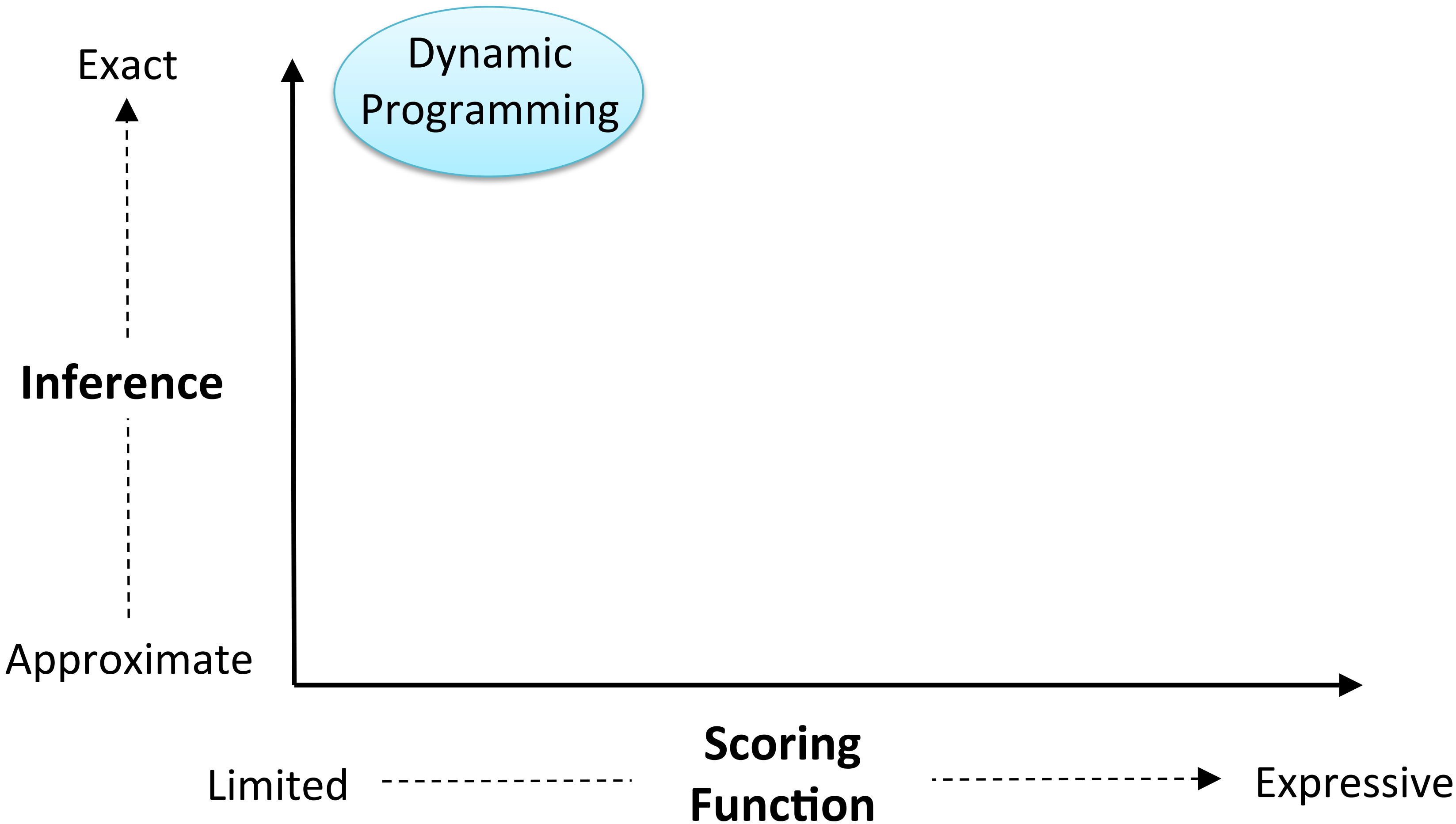
Yuan Zhang, Tao Lei, Regina Barzilay,
Tommi Jaakkola, Amir Globerson

MIT, Hebrew University

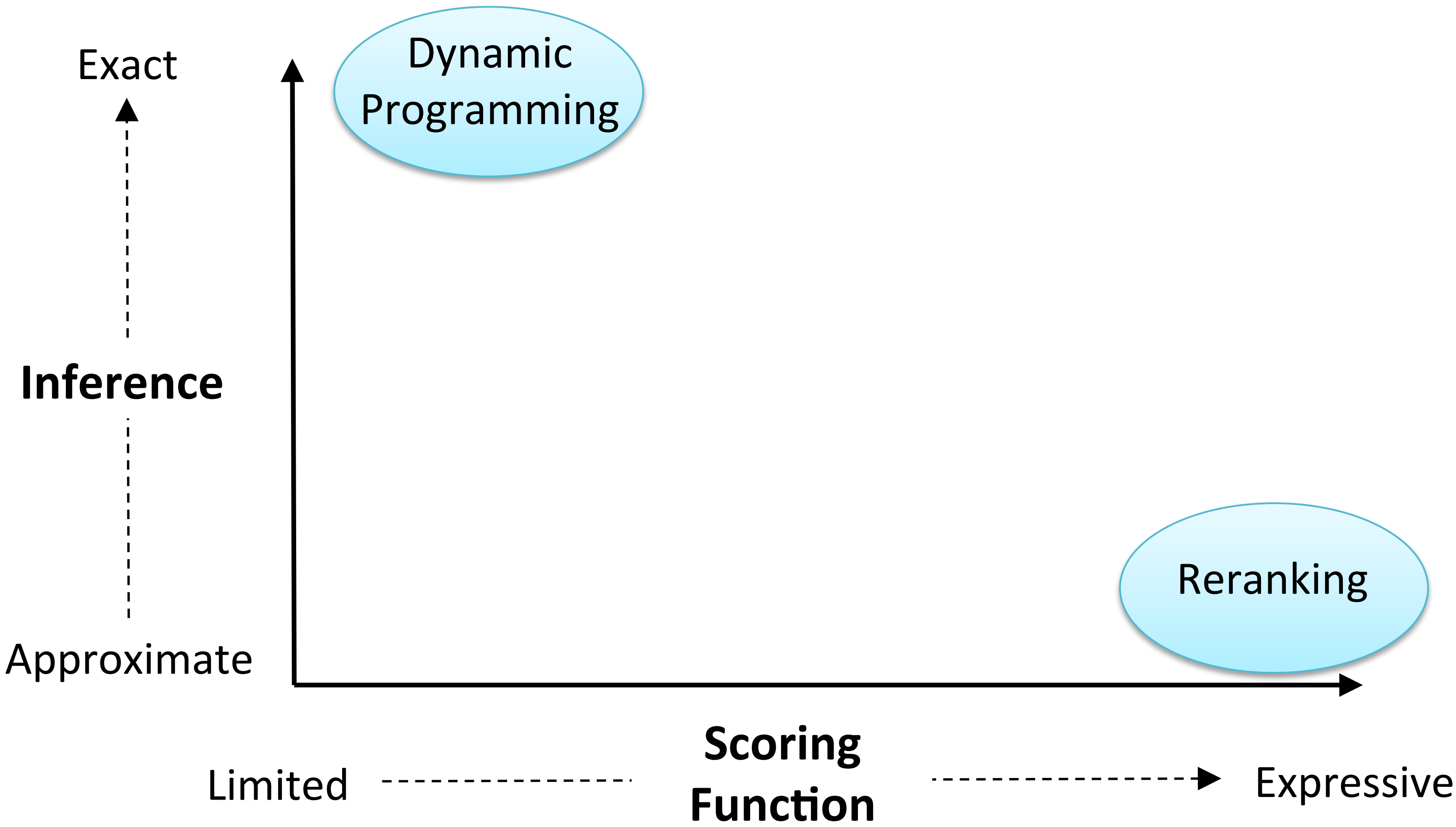
Exact Inference vs. Expressive Scoring Function



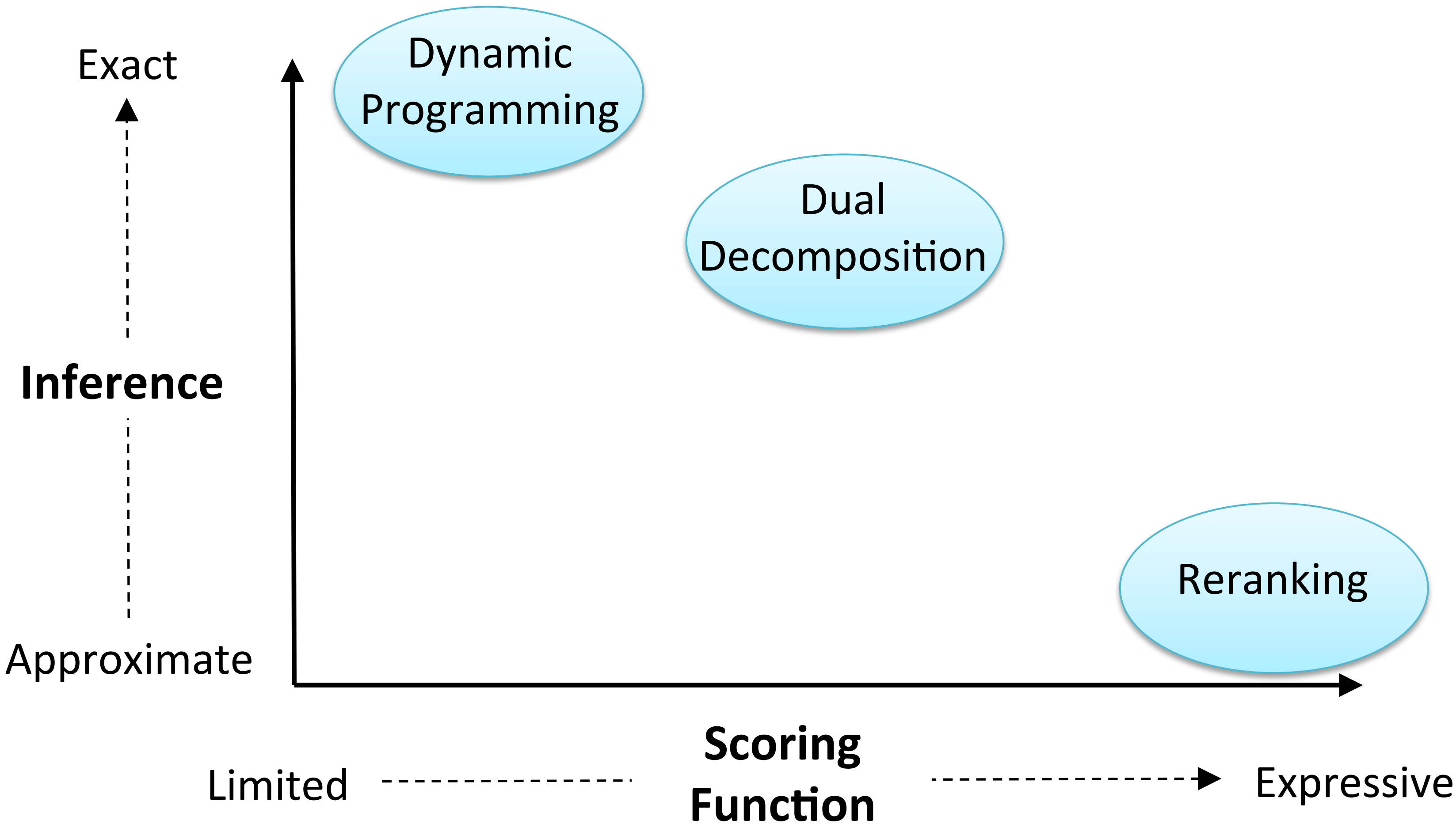
Exact Inference vs. Expressive Scoring Function



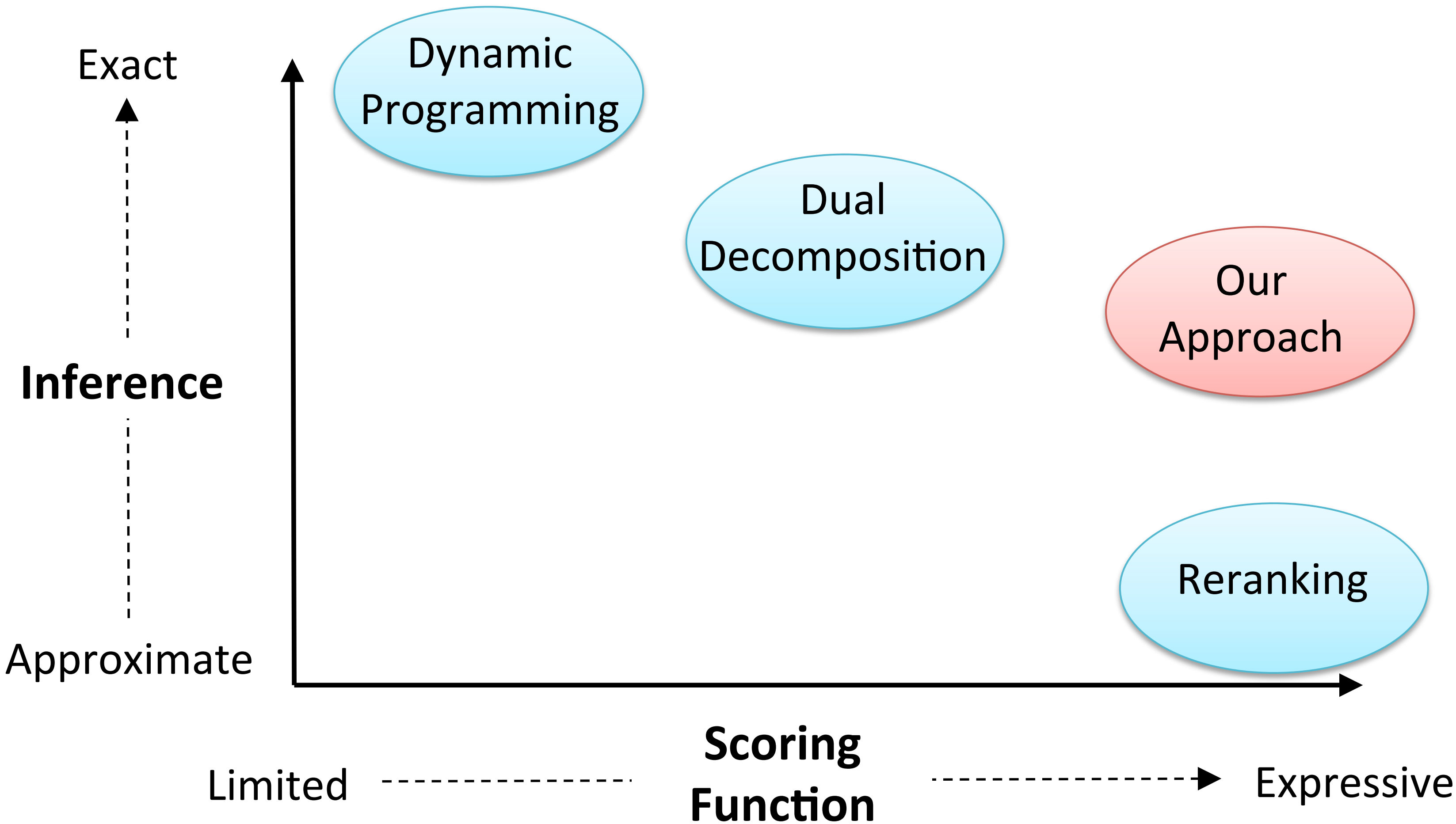
Exact Inference vs. Expressive Scoring Function



Exact Inference vs. Expressive Scoring Function



Exact Inference vs. Expressive Scoring Function



- Search in full parse space
- Easily incorporate arbitrary features

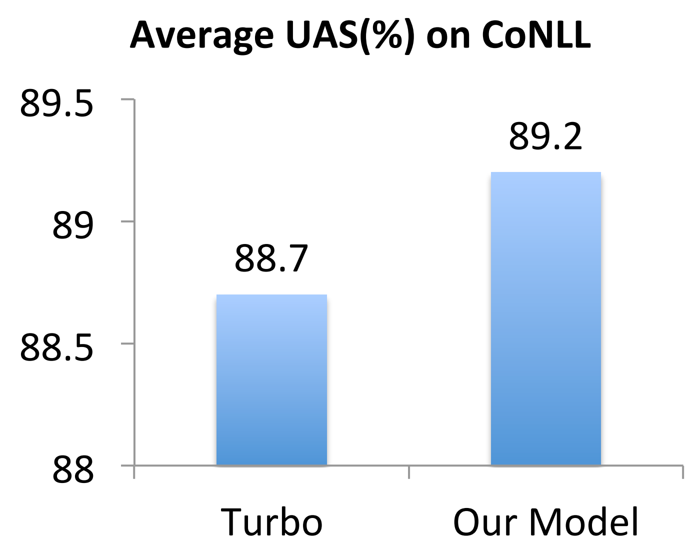
Our Approach

- 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

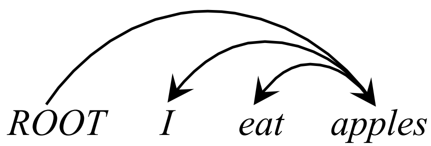
- Method: a sampling-based dependency parser
  - Decoding: climb to the optimum in small steps
  - Proposal distributions:
    - Gibbs
    - Metropolis-Hastings
  - Learning via SampleRank: satisfy constraints based on samples

- Advantages:
  - Achieve top parsing performance
  - Readily extendable to joint prediction tasks



Sampling-Based Decoding Algorithm

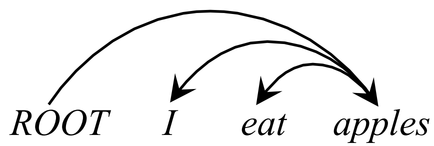
- Generate a sequence of samples to climb towards the optimum in small stochastic steps



$y^{(0)}$ -- any initial tree

Sampling-Based Decoding Algorithm

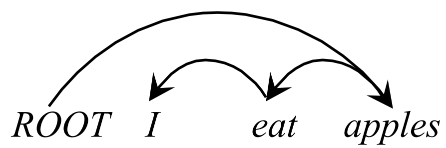
- Generate a sequence of samples to climb towards the optimum in small stochastic steps



$y^{(0)}$ -- any initial tree

$\boxed{q(y^{(1)} \mid x, y^{(0)}, T^{(0)}, \theta)}$

$q(\cdot \mid x, y, T, \theta)$: proposal distr. which governs the climb



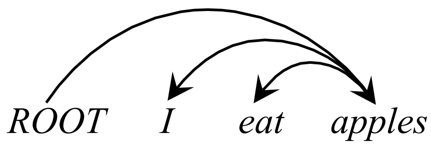
$y^{(1)}$

$q(y^{(2)} \mid x, y^{(1)}, T^{(1)}, \theta)$

$\vdots$

Sampling-Based Decoding Algorithm

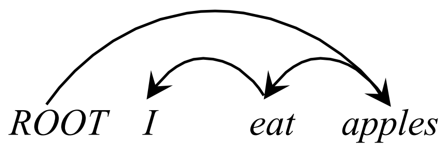
- Generate a sequence of samples to climb towards the optimum in small stochastic steps



$y^{(0)}$ -- any initial tree

$q(y^{(1)} \mid x, y^{(0)}, T^{(0)}, \theta)$

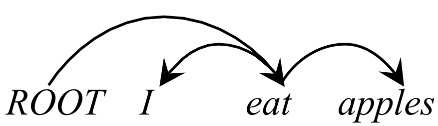
$q(\cdot \mid x, y, T, \theta)$: proposal distr. which governs the climb



$y^{(1)}$

$q(y^{(2)} \mid x, y^{(1)}, T^{(1)}, \theta)$

⋮



$y^{(m)} \approx \underset{y}{\operatorname{argmax}}\, \theta \cdot f(x, y)$ (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

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

temperature scaling

Proposal Distribution: Gibbs Sampling

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

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

temperature scaling

➢ Arbitrary features in scoring function

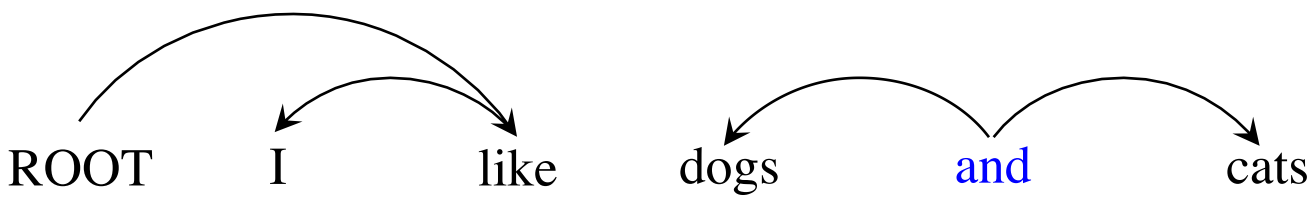
Proposal Distribution: Gibbs Sampling

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

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

temperature scaling

➢ Arbitrary features in scoring function



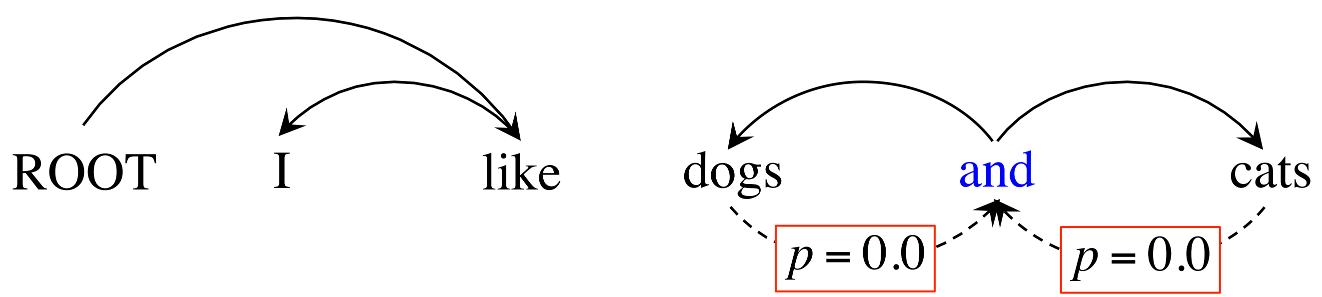
Proposal Distribution: Gibbs Sampling

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

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

temperature scaling

➢ Arbitrary features in scoring function



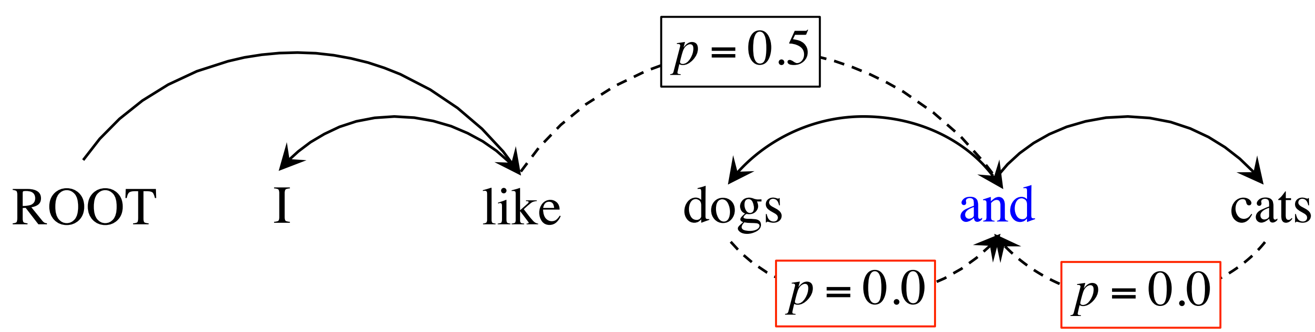
Proposal Distribution: Gibbs Sampling

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

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

temperature scaling

➢ Arbitrary features in scoring function



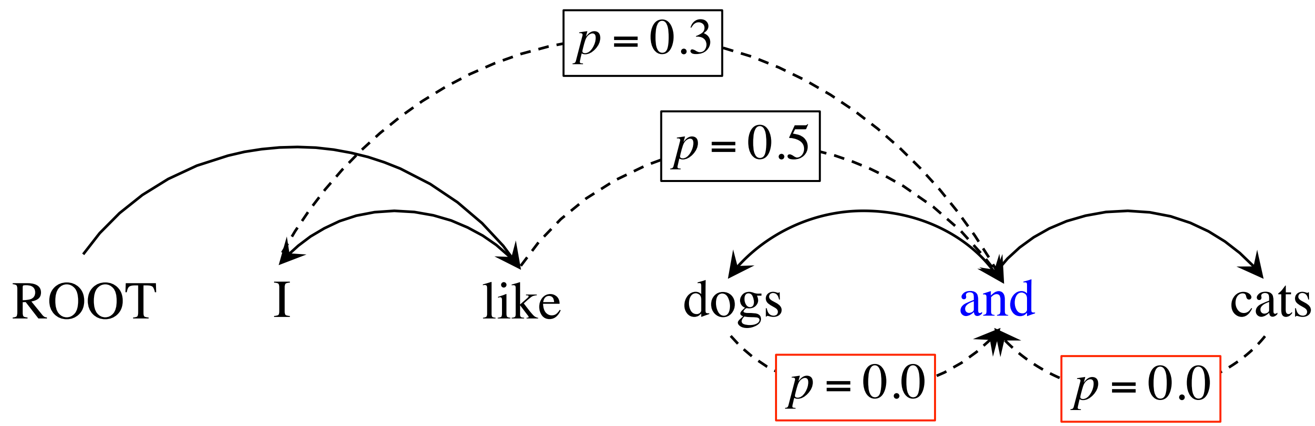
Proposal Distribution: Gibbs Sampling

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

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

temperature scaling

➢ Arbitrary features in scoring function



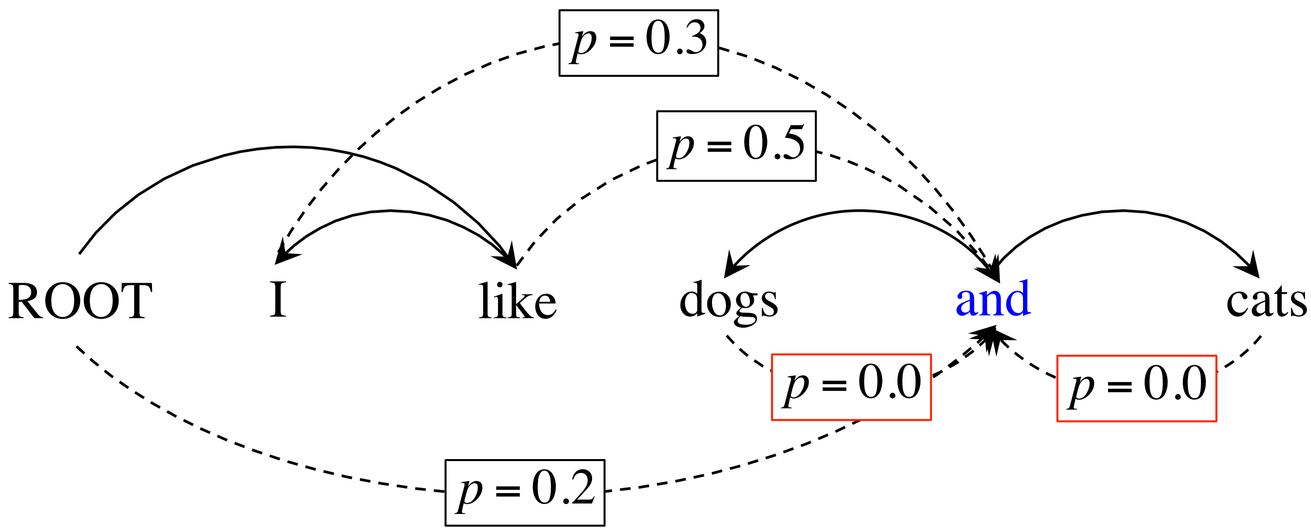
Proposal Distribution: Gibbs Sampling

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

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

temperature scaling

➢ Arbitrary features in scoring function



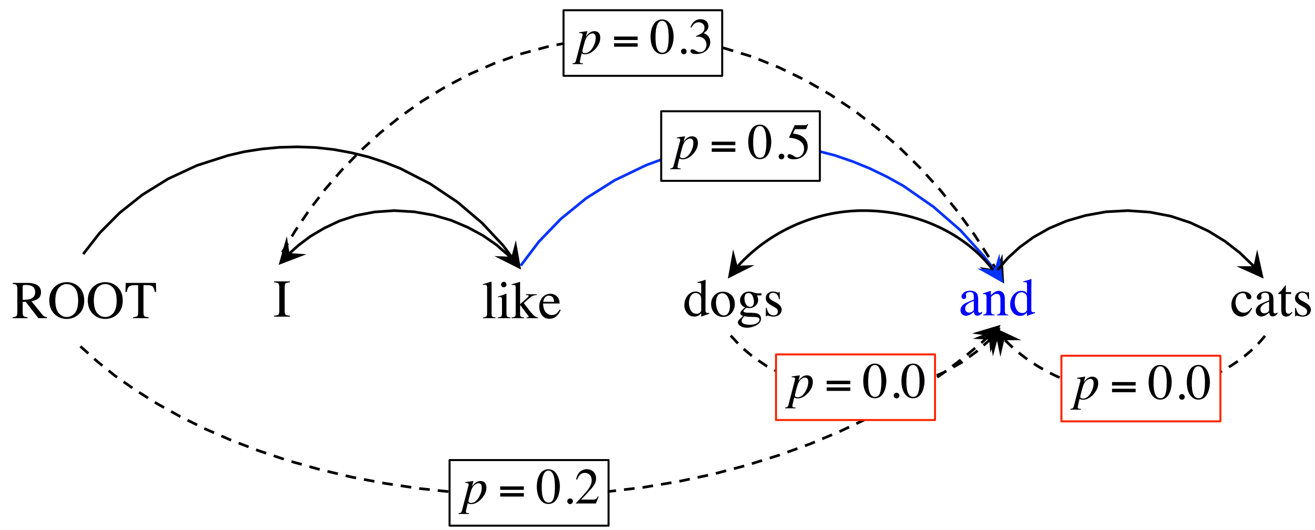
Proposal Distribution: Gibbs Sampling

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

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

temperature scaling

➢ Arbitrary features in scoring function

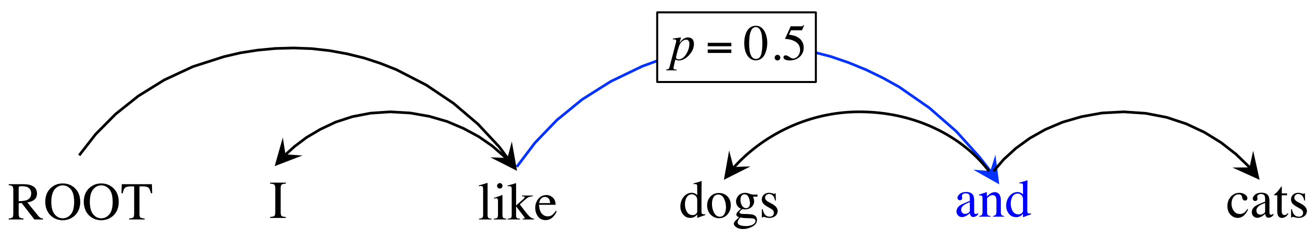


Proposal Distribution: Gibbs Sampling

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

$$p(y_j \mid x, y_{-j}, T, \theta) \propto \exp(\theta \cdot \boxed{f(x, y_j, y_{-j})} \underbrace{\boxed{/T}}_{\text{temperature scaling}})$$

➢ 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)

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

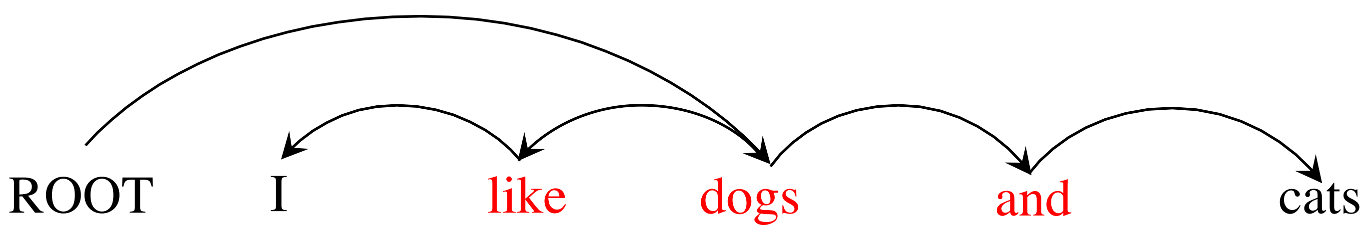
For each node not in the tree $x_i \notin T$

  Random walk from x_i until reach a node in T

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

End for

original tree



Random Walk-Based Sampler (Wilson 1996)

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

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

    Random walk from x_i until reach a node in T

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

End for

original tree

Random Walk-Based Sampler (Wilson 1996)

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

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

    Random walk from x_i until reach a node in T

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

End for

---

walk path:

ROOT   I   like   dogs   and   cats

Random Walk-Based Sampler (Wilson 1996)

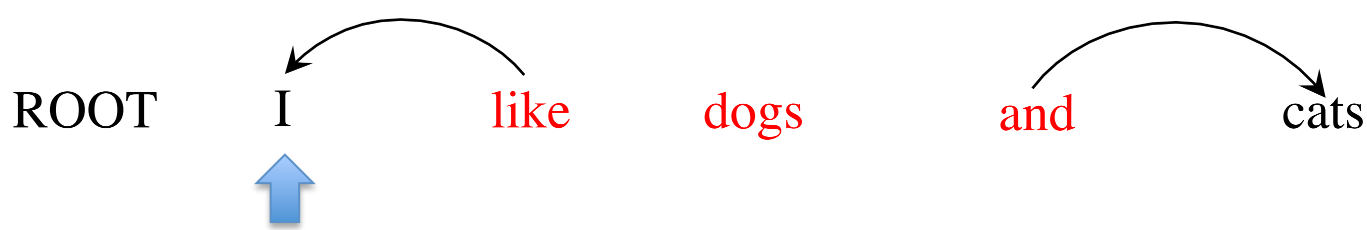
Initial tree $T \leftarrow \{ROOT\}$
For each node not in the tree $x_i \notin T$
    Random walk from x_i until reach a node in T
    Add path into the tree $T \leftarrow T \cup path$
End for

walk path:

Random Walk-Based Sampler (Wilson 1996)

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

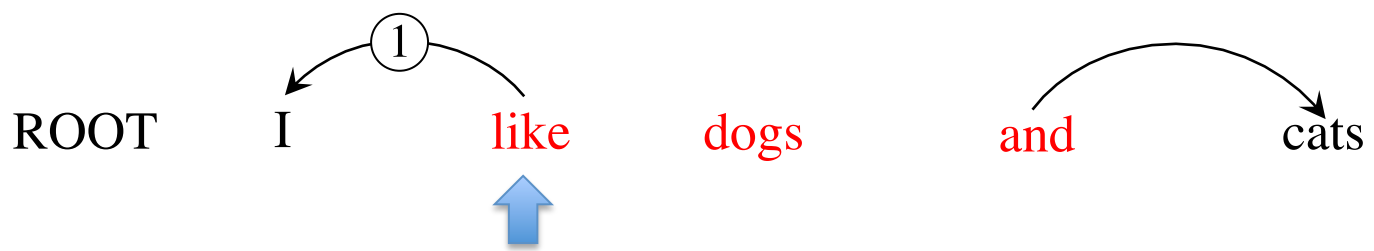
walk path: I



Random Walk-Based Sampler (Wilson 1996)

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

walk path: $I \rightarrow like$



Random Walk-Based Sampler (Wilson 1996)

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

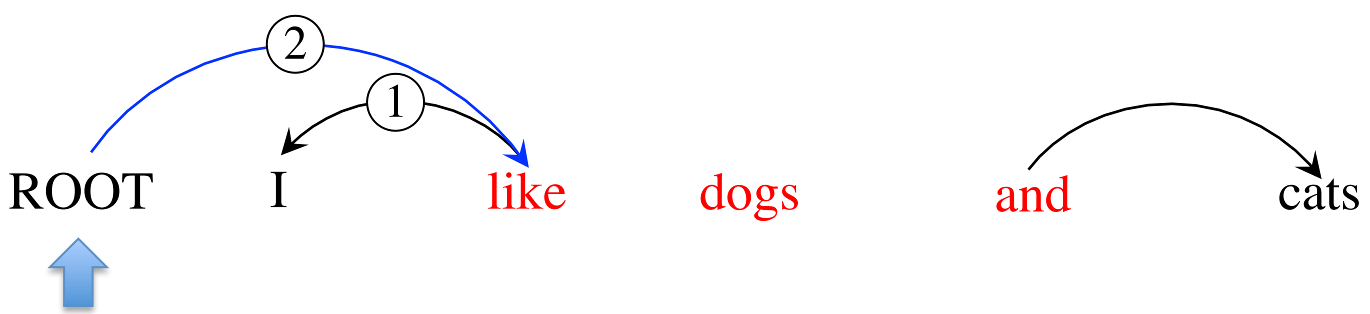
2: **For each** node not in the tree $x_i \notin T$

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: $I \rightarrow like \rightarrow ROOT$



Random Walk-Based Sampler (Wilson 1996)

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

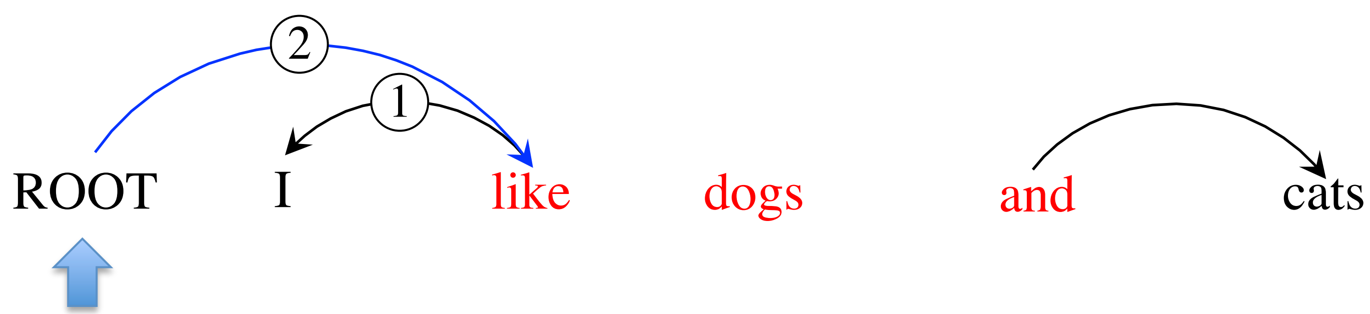
For each node not in the tree $x_i \notin T$

  Random walk from x_i until reach a node in T

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

End for

walk path: $I \rightarrow like \rightarrow ROOT$



Random Walk-Based Sampler (Wilson 1996)

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

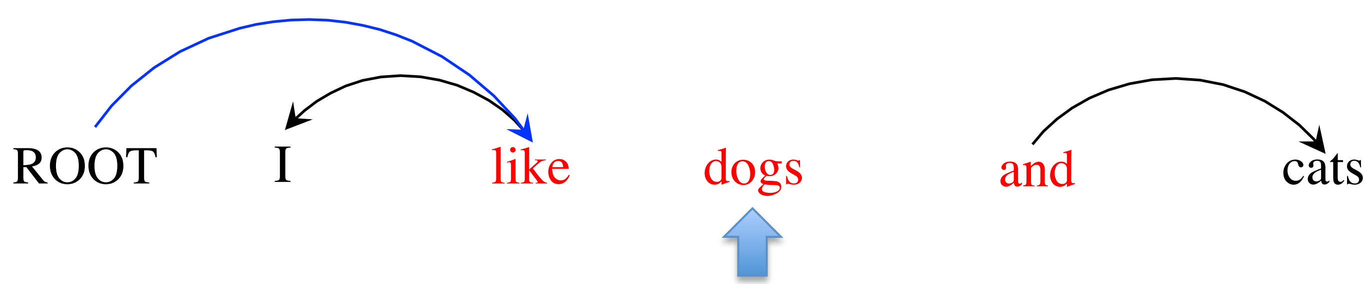
For each node not in the tree $x_i \notin T$

    Random walk from x_i until reach a node in T

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

End for

walk path: dogs



Random Walk-Based Sampler (Wilson 1996)

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

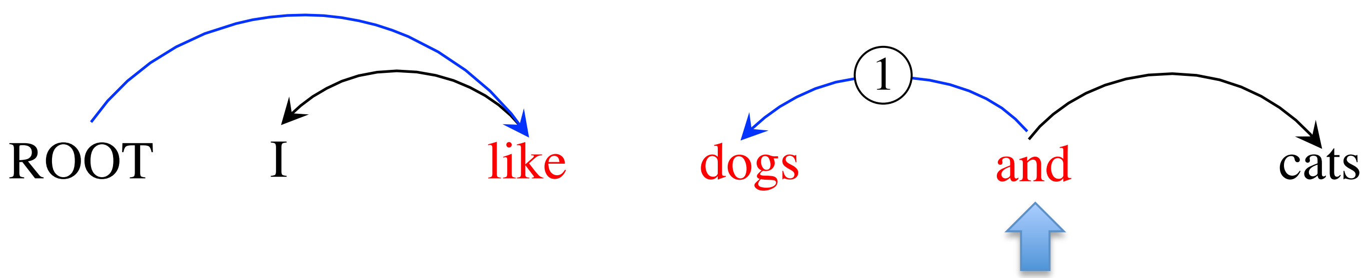
2: **For each** node not in the tree $x_i \notin T$

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 $T \leftarrow \{ROOT\}$

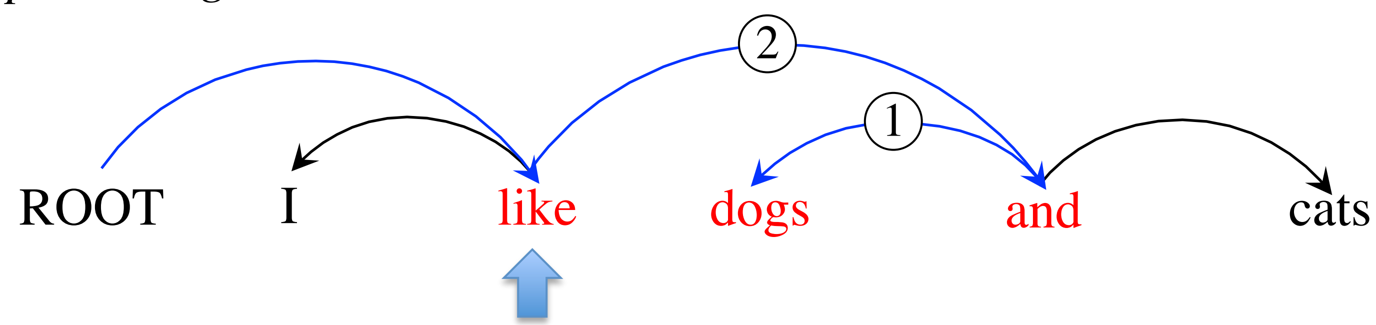
2: **For each** node not in the tree $x_i \notin T$

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 $T \leftarrow \{ROOT\}$

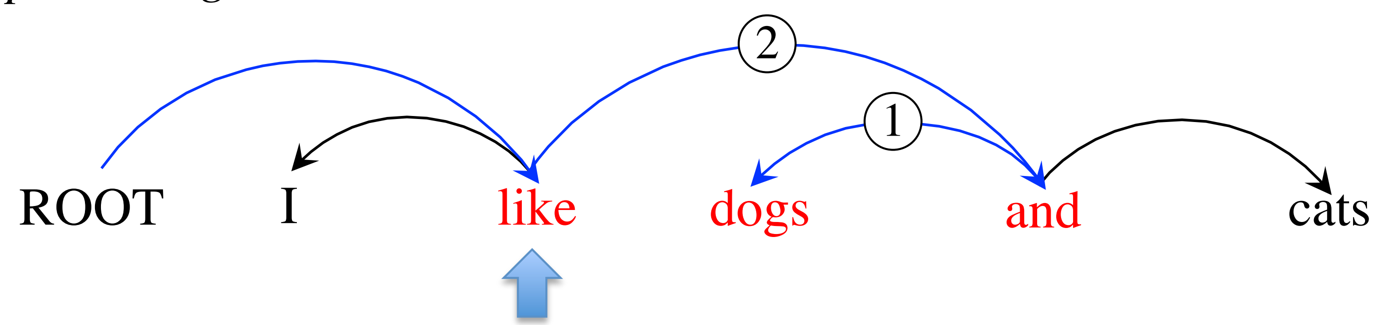
2: **For each** node not in the tree $x_i \notin T$

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 $T \leftarrow \{ROOT\}$

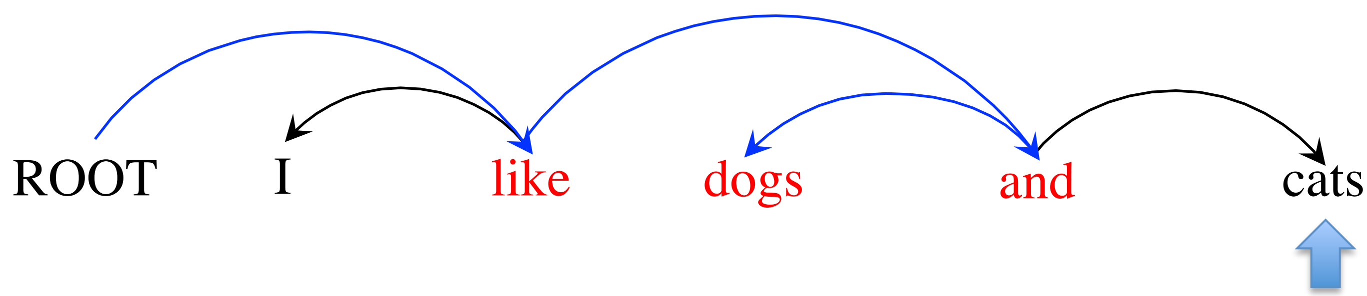
2: **For each** node not in the tree $x_i \notin T$

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

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

5: **End for**

walk path: $cats \rightarrow and$



Random Walk-Based Sampler (Wilson 1996)

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

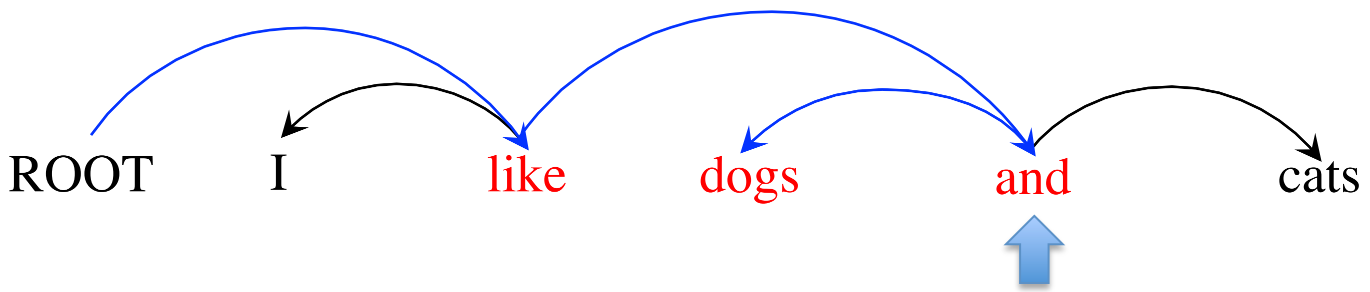
2: **For each** node not in the tree $x_i \notin T$

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: $cats \rightarrow and$



Random Walk-Based Sampler (Wilson 1996)

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

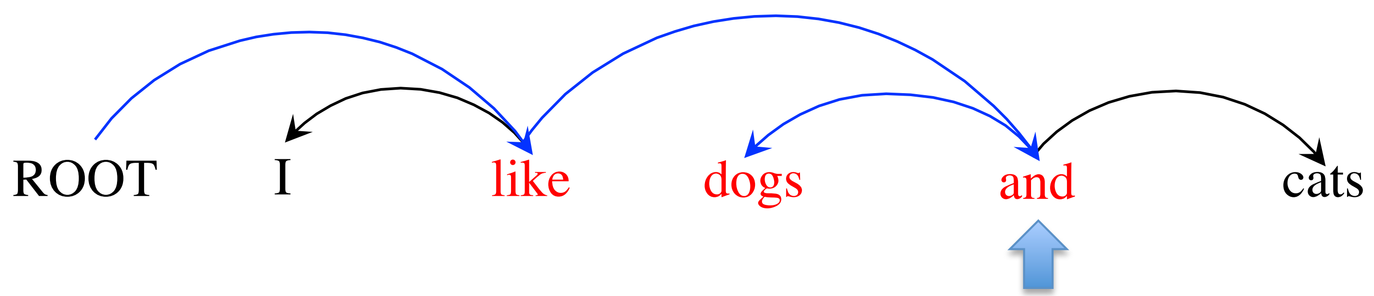
2: **For each** node not in the tree $x_i \notin T$

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: $cats \rightarrow and$



Random Walk-Based Sampler (Wilson 1996)

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

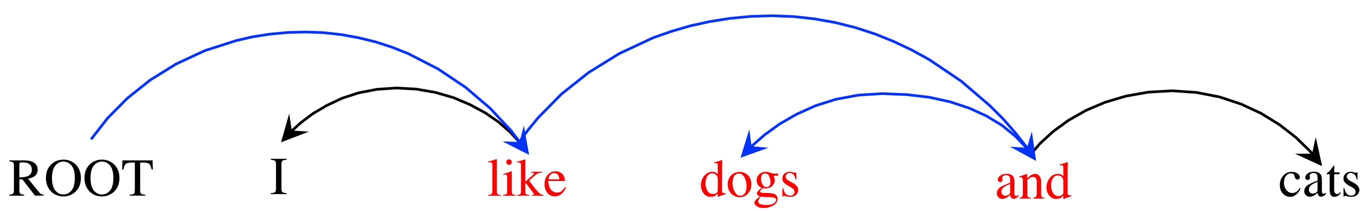
2: **For each** node not in the tree $x_i \notin T$

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**

new tree



Random Walk-Based Sampler (Wilson 1996)

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

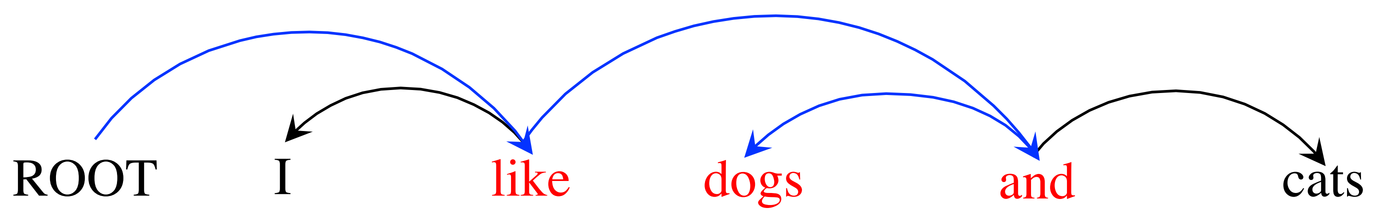
2: **For each** node not in the tree $x_i \notin T$

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

$$y^{(0)} \xrightarrow{q(\cdot|y^{(0)})} y^{(1)} \xrightarrow{q(\cdot|y^{(1)})} y^{(2)} \xrightarrow{q(\cdot|y^{(2)})} y^{(3)} \rightarrow \cdots$$

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

Sampling-Based Learning Algorithm

- Generate a sequence of samples

$$y^{(0)} \xrightarrow{q(\cdot|y^{(0)})} y^{(1)} \xrightarrow{q(\cdot|y^{(1)})} y^{(2)} \xrightarrow{q(\cdot|y^{(2)})} y^{(3)} \rightarrow \cdots$$

➢ 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

Constraints in Learning

1) Constraints between samples and the gold tree

$$\underbrace{s(x,\hat{y})}_{\text{Score of the gold tree}} - \underbrace{s(x,y^{(t)})}_{\text{Score of the sample}} \geq \underbrace{Err(y^{(t)})}_{\text{\# errors in the sample}}$$

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

$$\text{Markov chain:} \quad y^{(0)} \rightarrow y^{(1)} \rightarrow \boxed{y^{(2)} \rightarrow y^{(3)}} \rightarrow y^{(4)} \cdots$$

if $y^{(3)}$ *is more accurate than* $y^{(2)}$

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

Constraints in Learning

1) Constraints between samples and the gold tree

$$\boxed{s(x,\hat{y})} - \boxed{s(x,y^{(t)})} \geq \boxed{Err(y^{(t)})}$$

Score of the gold tree

Score of the sample

errors in the sample

2) Constraints between neighboring samples

Markov chain: $y^{(0)} \rightarrow y^{(1)} \rightarrow \boxed{y^{(2)} \rightarrow y^{(3)}} \rightarrow y^{(4)} \cdots$

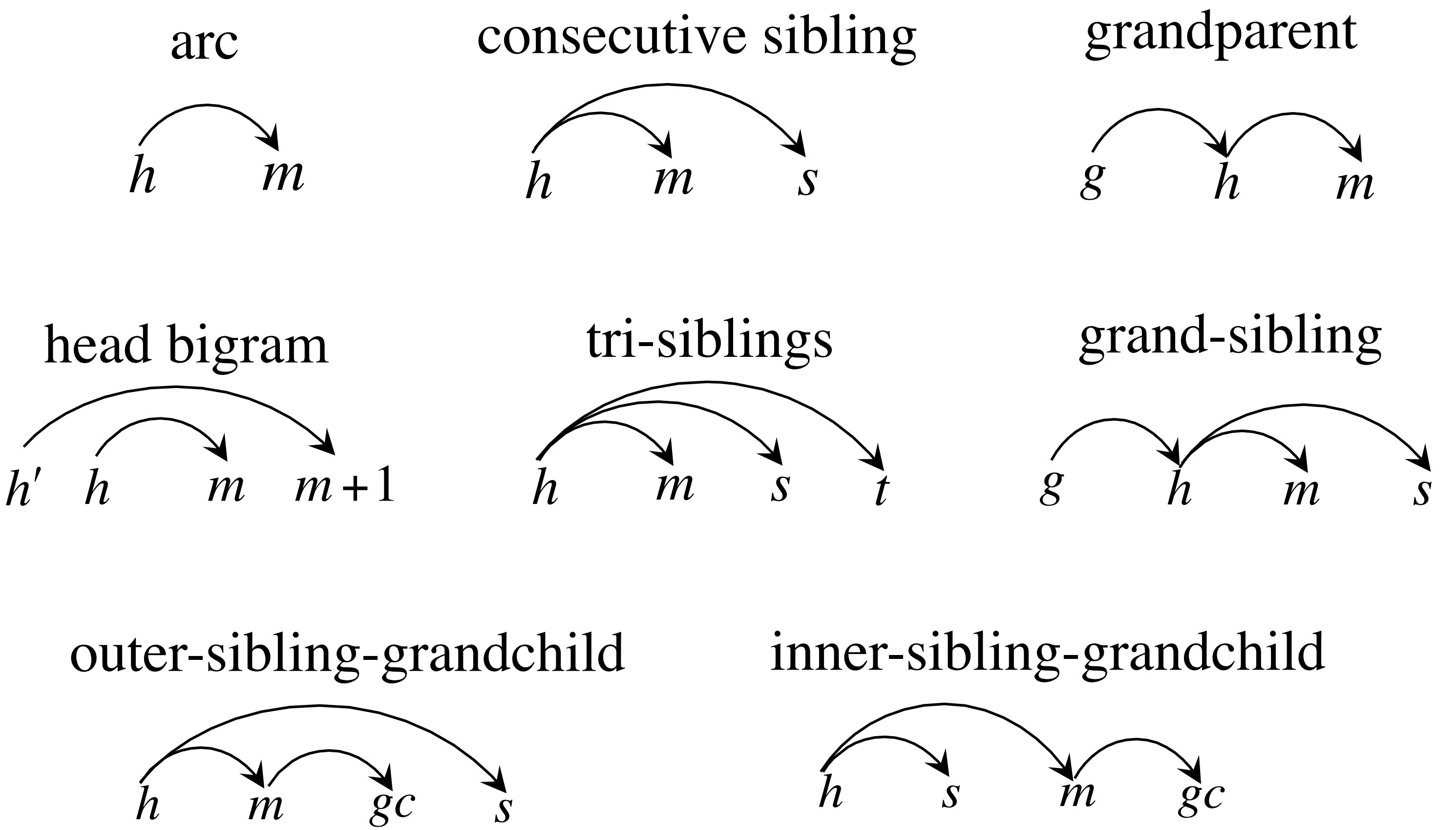
if $y^{(3)}$ *is more accurate than* $y^{(2)}$

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

➢ None of the samples are necessarily the argmax

First- to Third-Order Features

- Similar features used in previous work



Global Features

- Conjuncts consistency
  - POS tag consistency

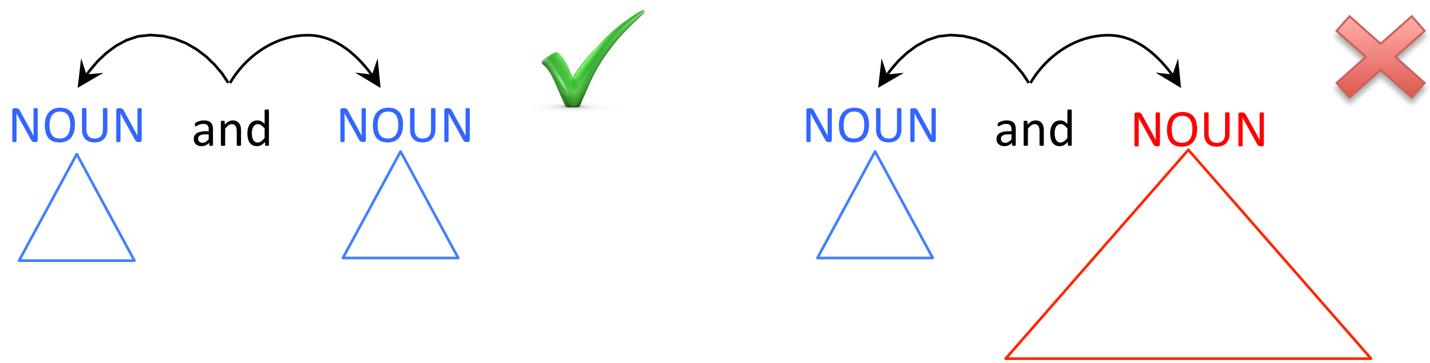
NOUN and NOUN

NOUN and VERB

Global Features

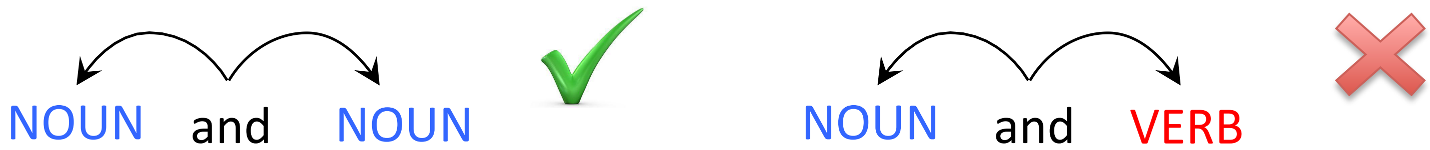
- Conjuncts consistency
  - POS tag consistency

  - Span length consistency

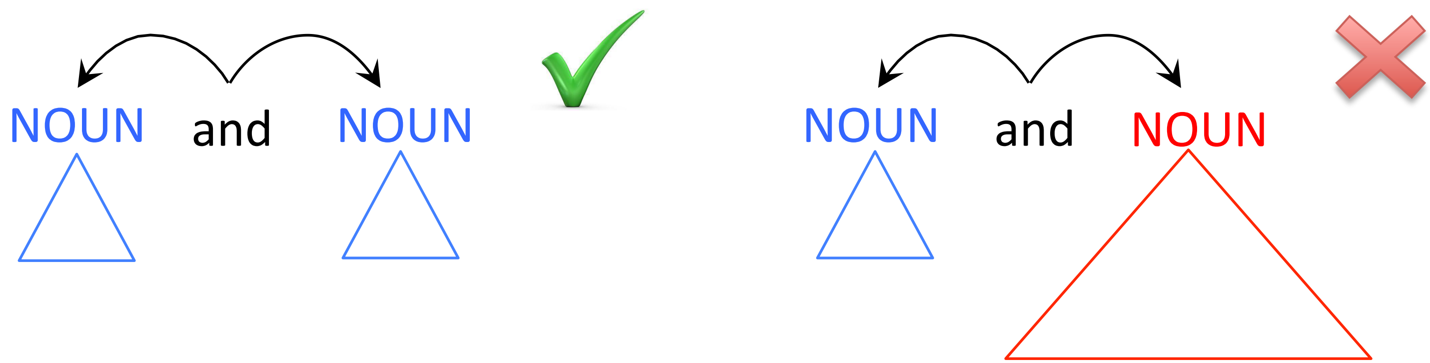


Global Features

- Conjuncts consistency
  - POS tag consistency



  - Span length consistency



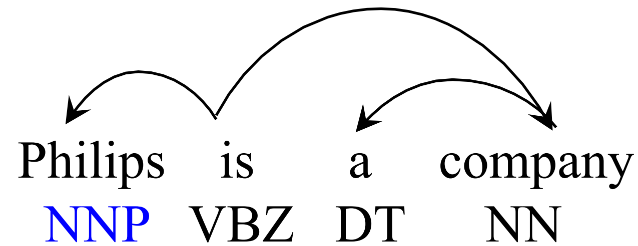
- Right branching, PP attachment, neighbors, valency, non-projective arcs

Joint Parsing and POS Correction

- Task:

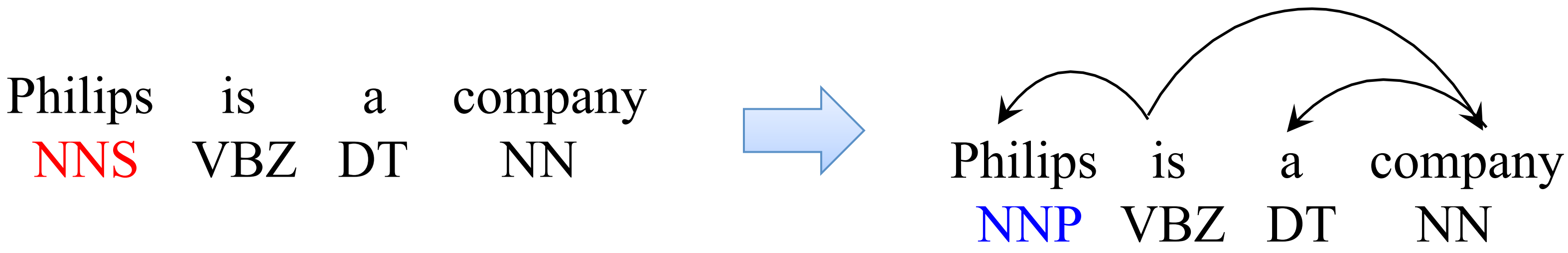
Philips is a company
NNS VBZ DT NN

→



Joint Parsing and POS Correction

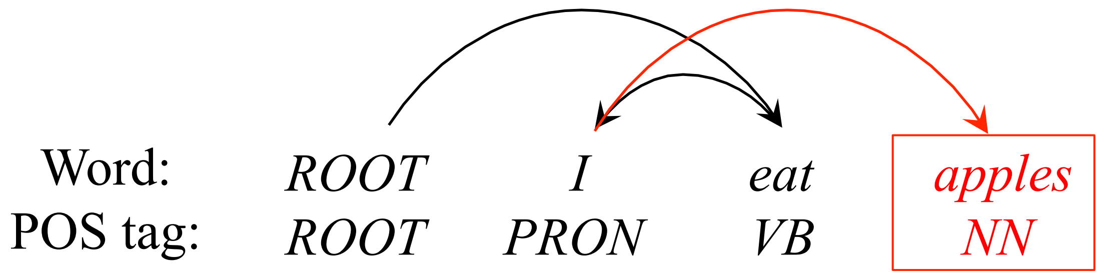
- Task:



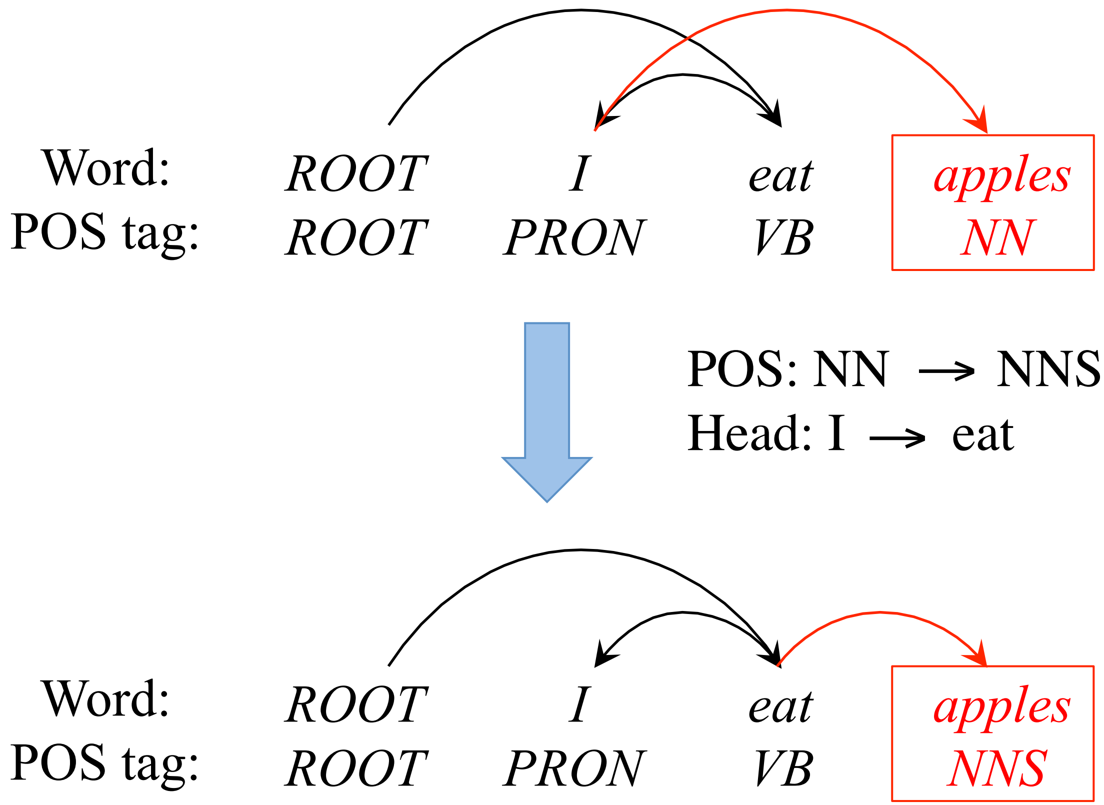
- Our approach: simple extension of our parsing model
  - Sample new heads y_j and POS tags t_j simultaneously

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

Example



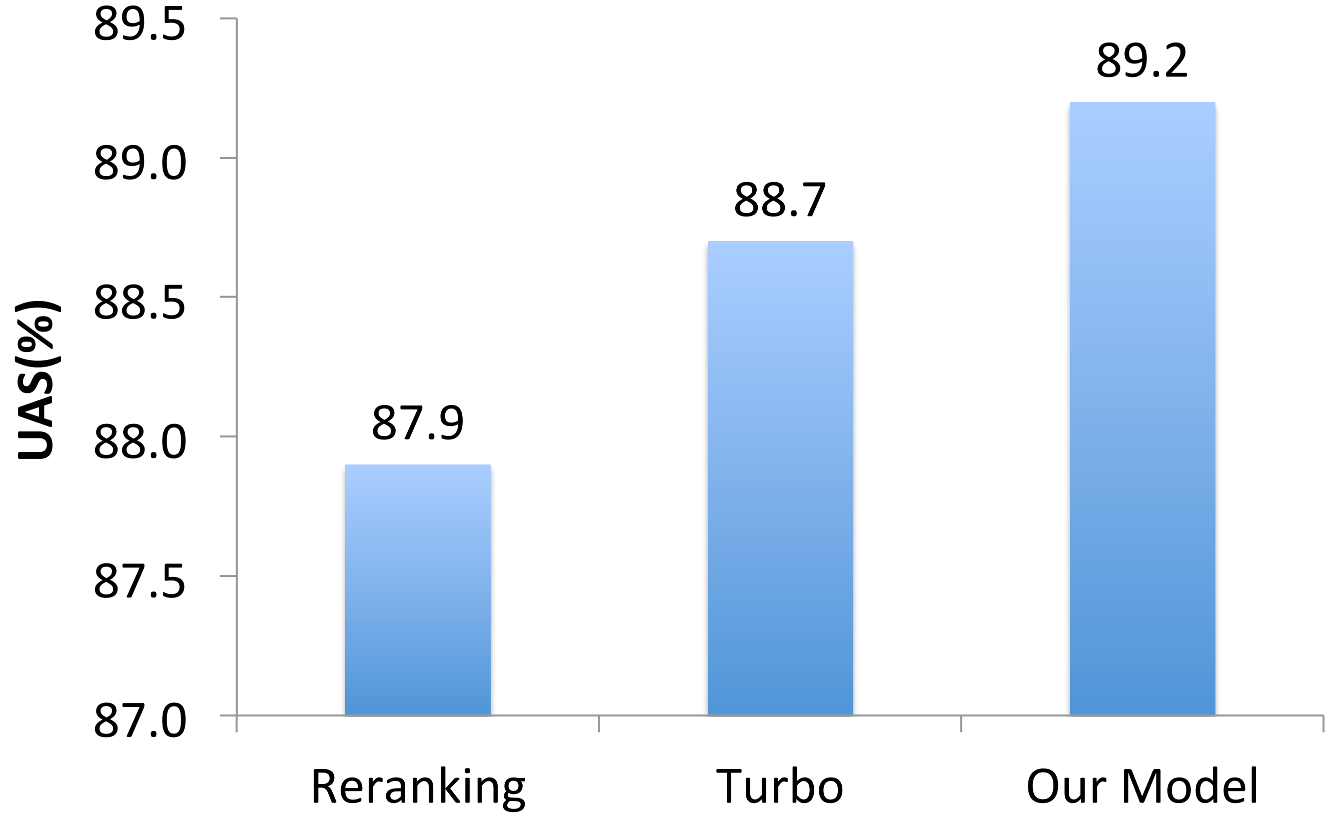
Example


Word: ROOT I eat apples
POS tag: ROOT PRON VB NN
POS: NN → NNS
Head: I → eat
Word: ROOT I eat apples
POS tag: ROOT PRON VB NNS

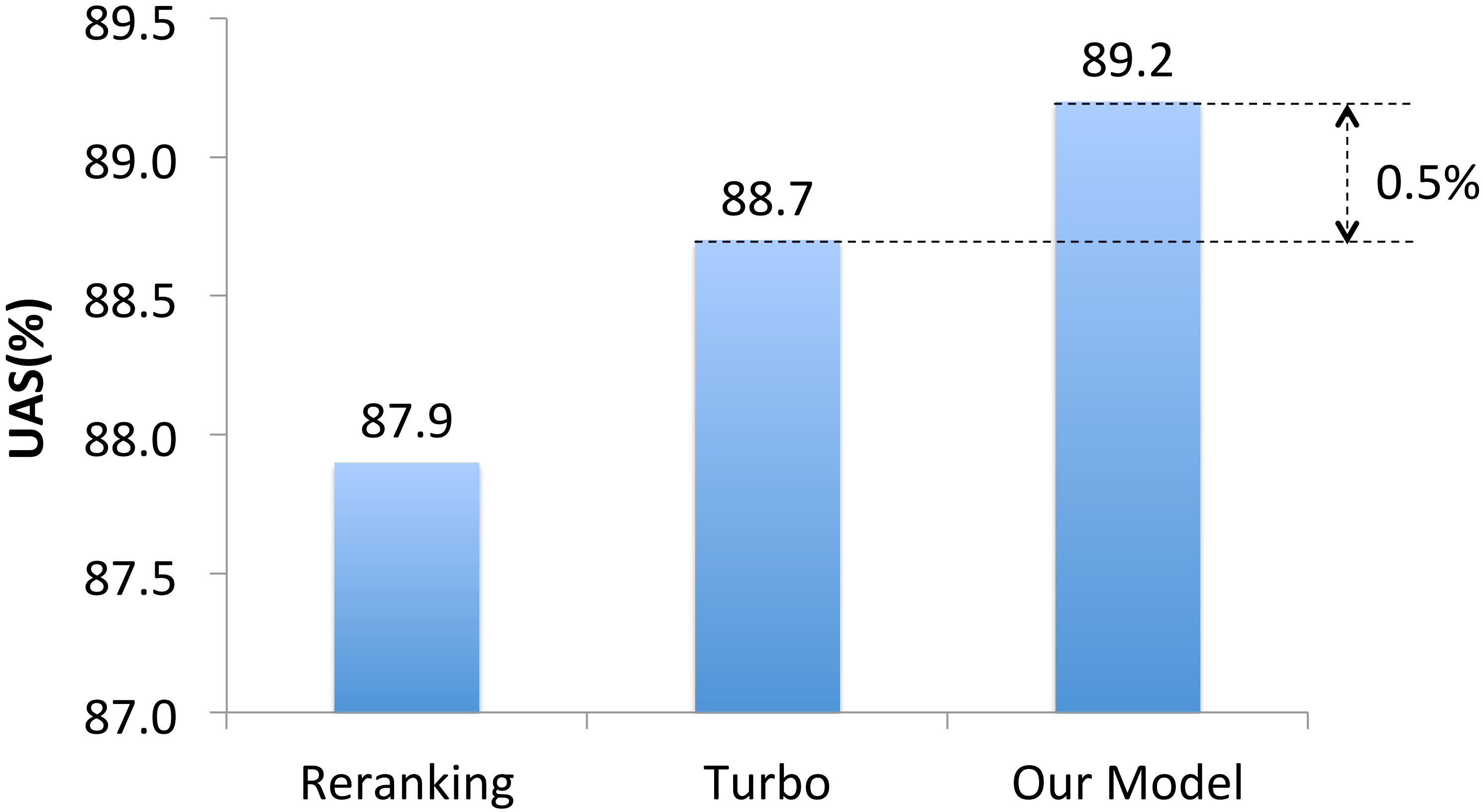
Experimental Setup for Parsing

- 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

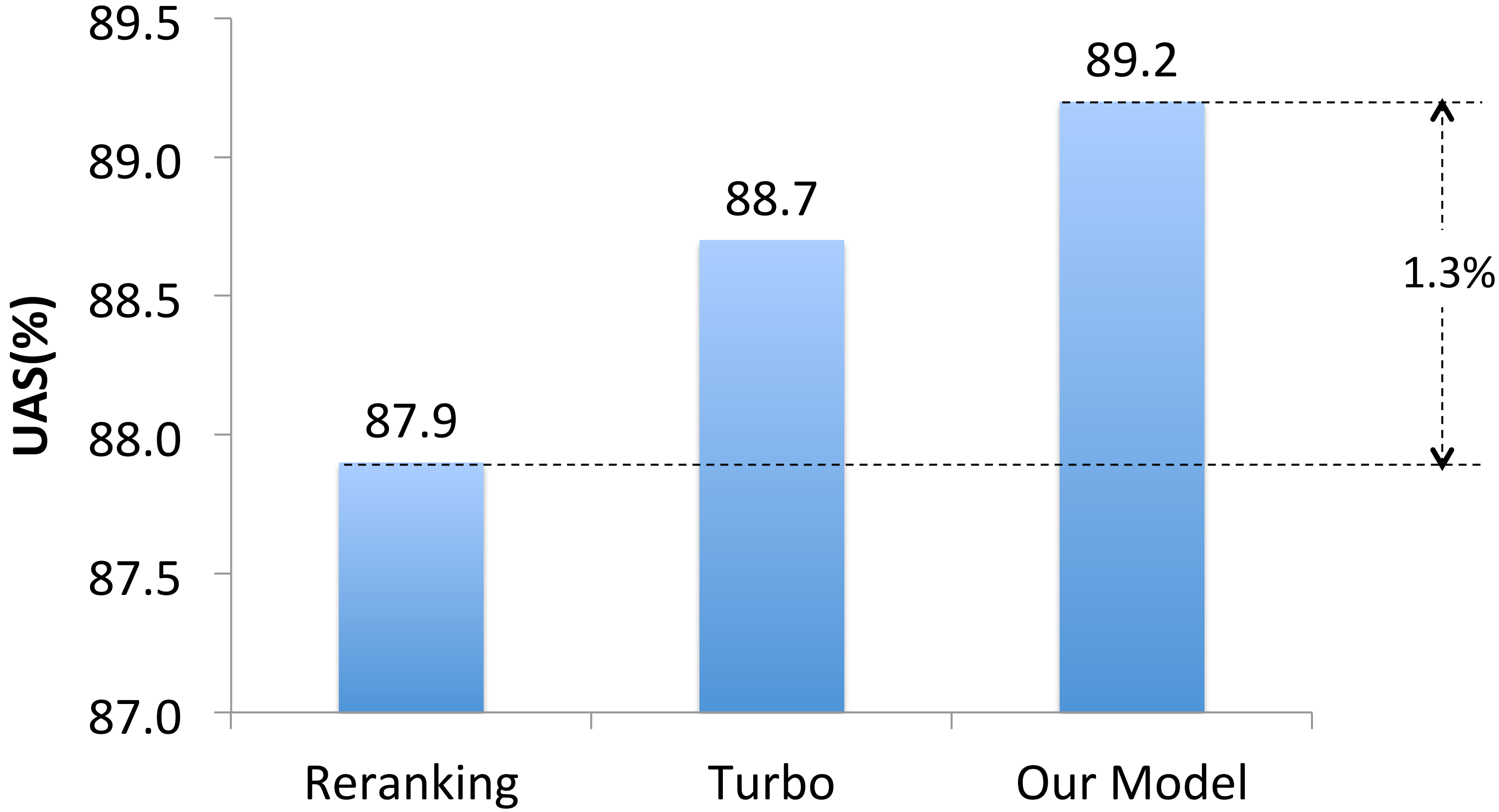
Results on CoNLL Dataset



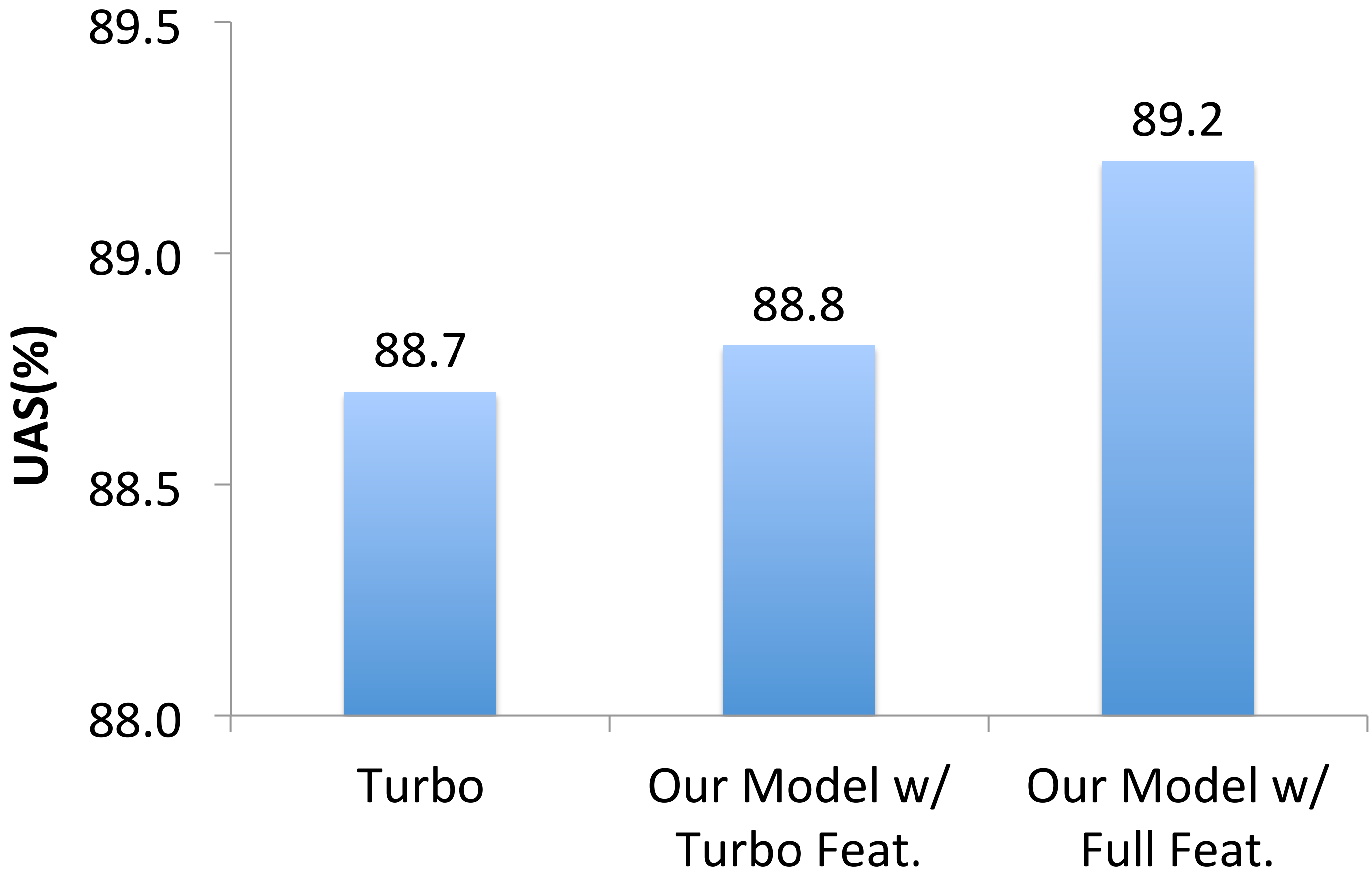
Results on CoNLL Dataset



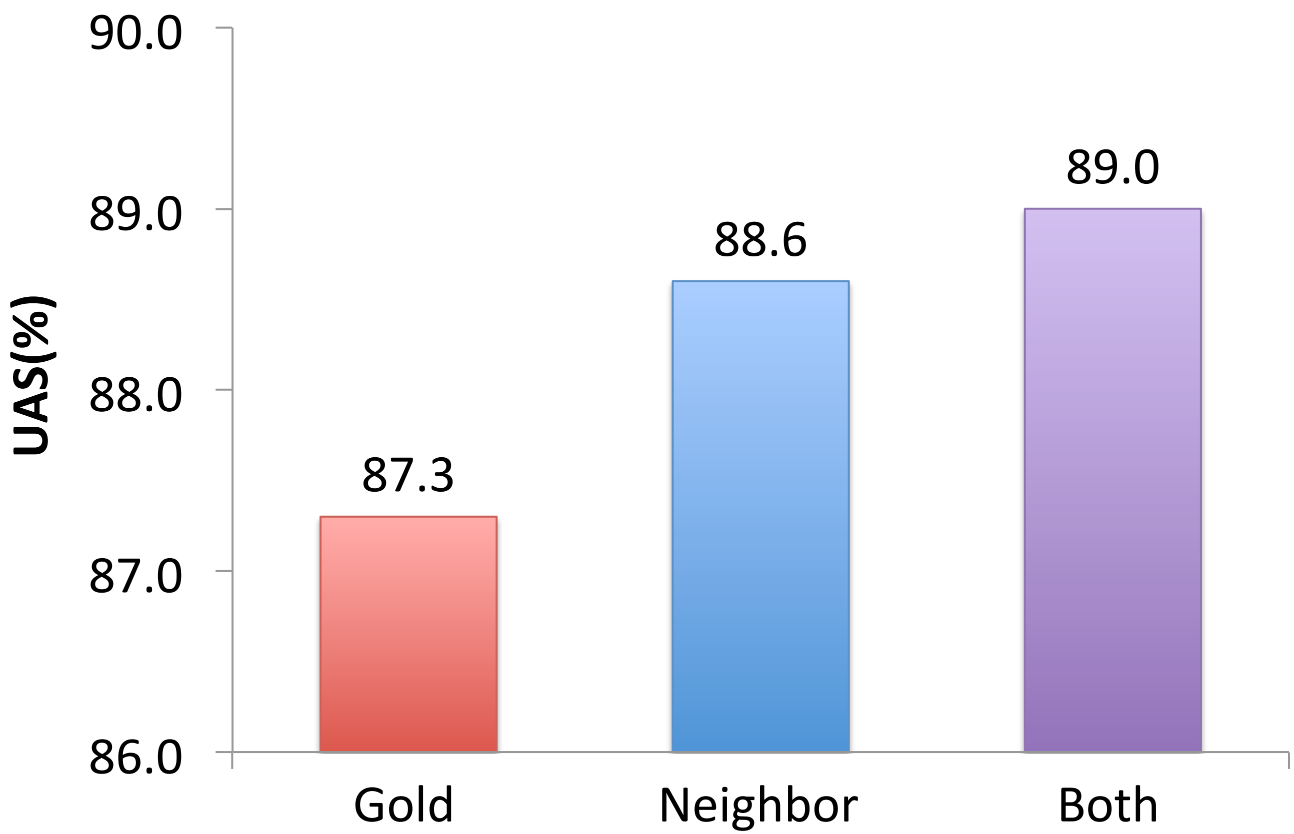
Results on CoNLL Dataset



Comparison with Turbo: Impact of Feature Sets

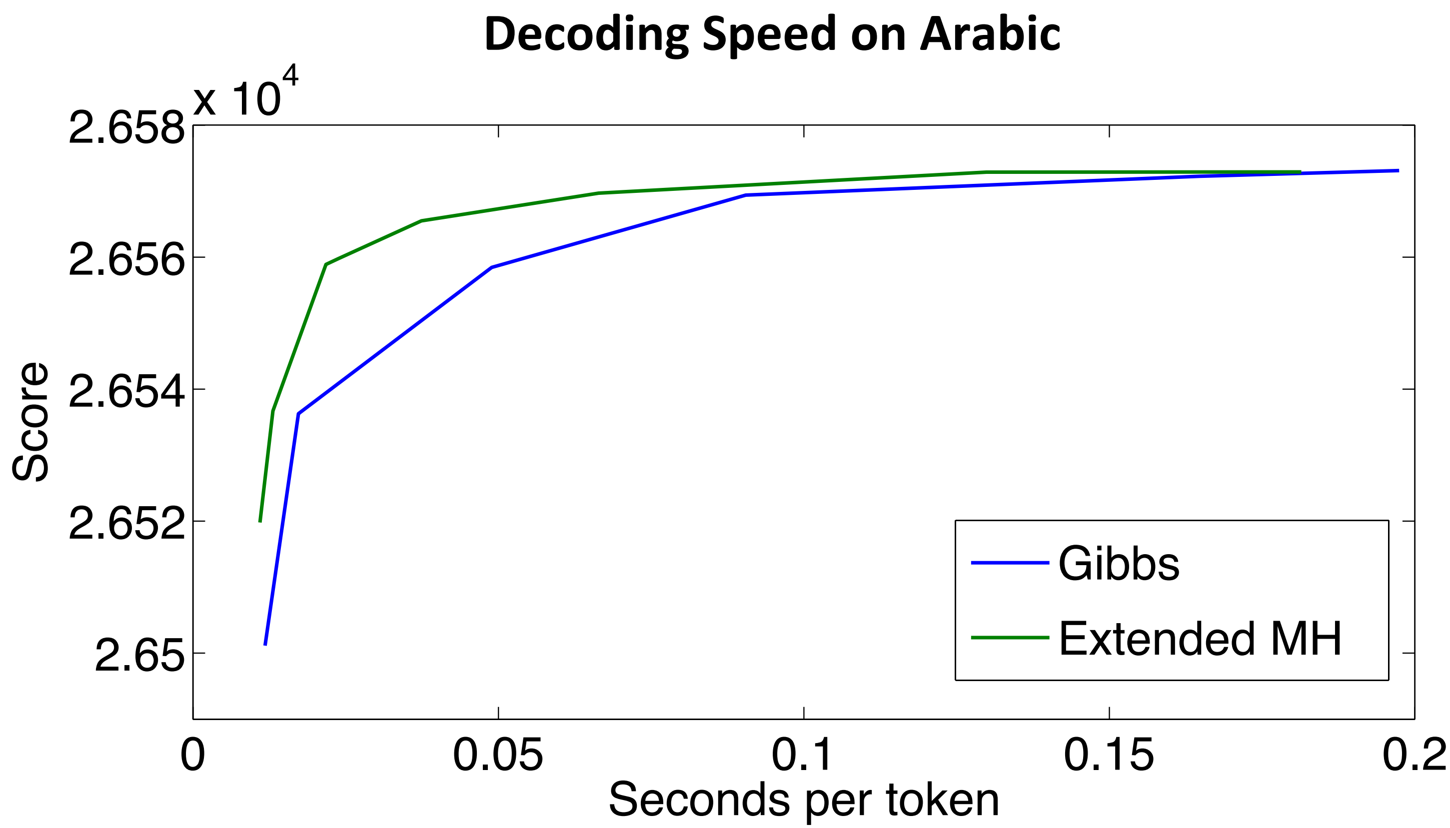


The Effect of Constraints in Learning



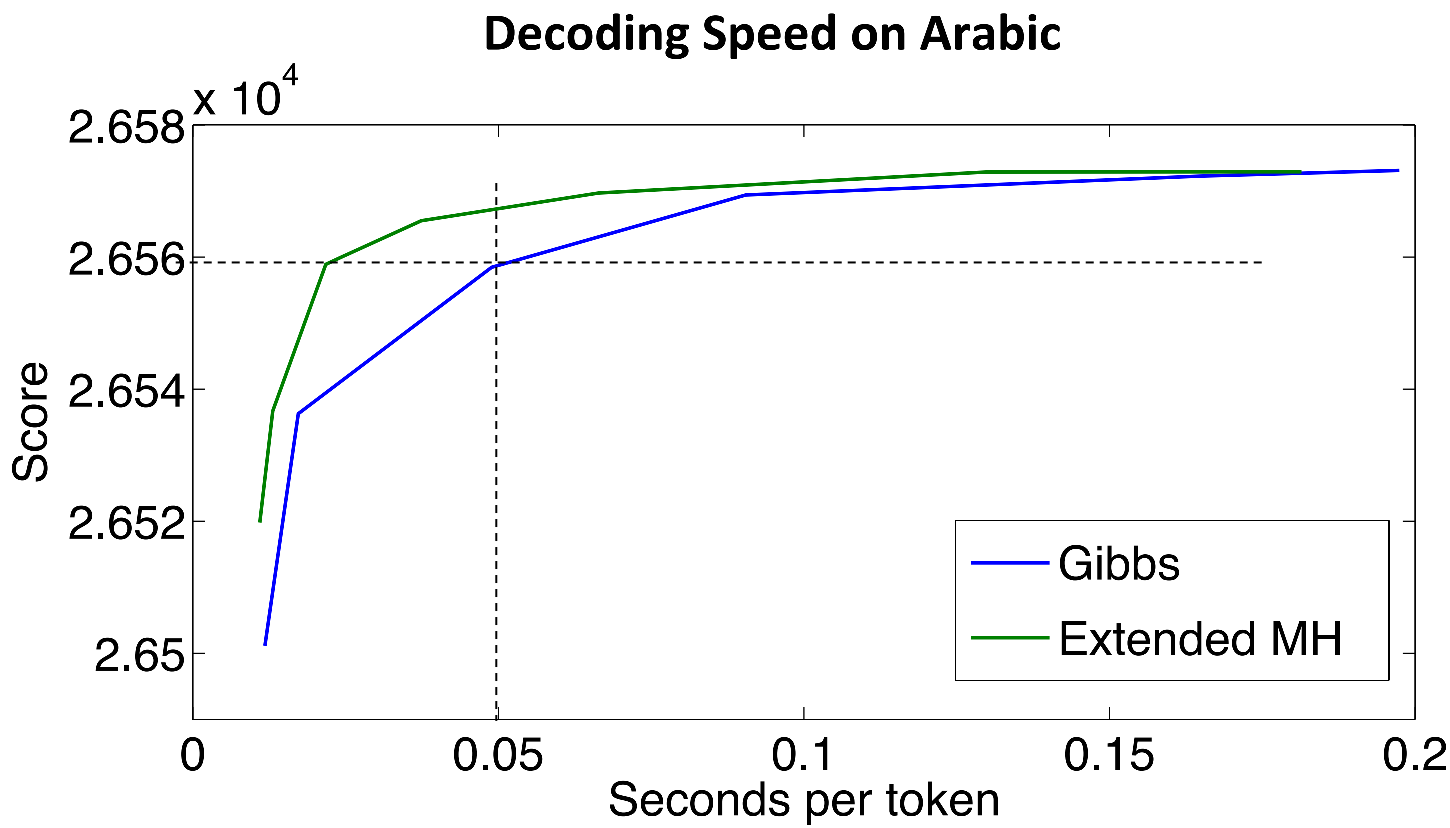
- Gold: constraints between samples and *gold* trees
- Neighbor: constraints between *neighboring* samples

Impact of Different Proposal Distributions



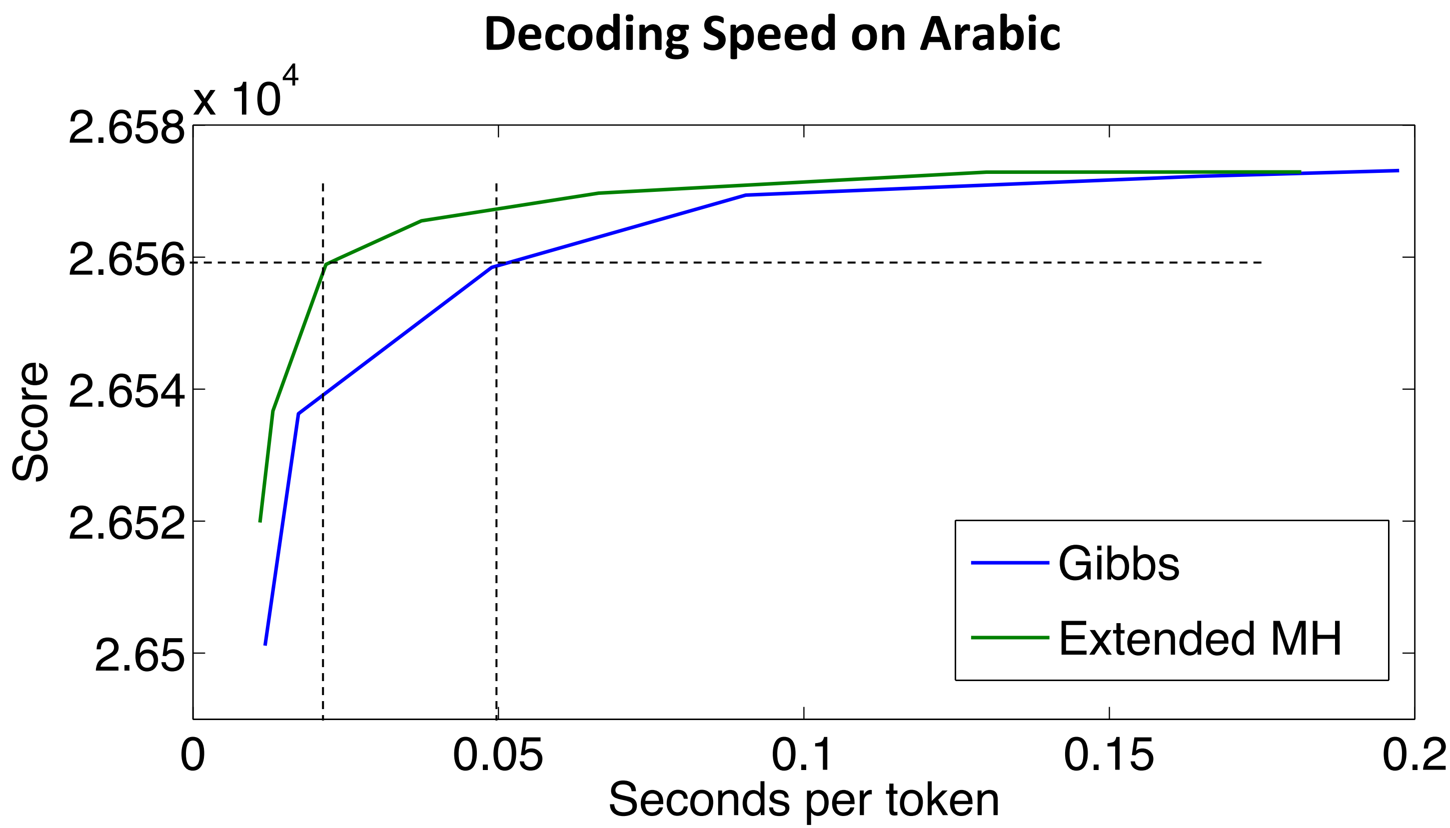
- We decode in different speed by controlling converge iterations
- Both methods achieve the same result given enough time
- Extended MH sampler performs better given constrained time

Impact of Different Proposal Distributions



- 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



- 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

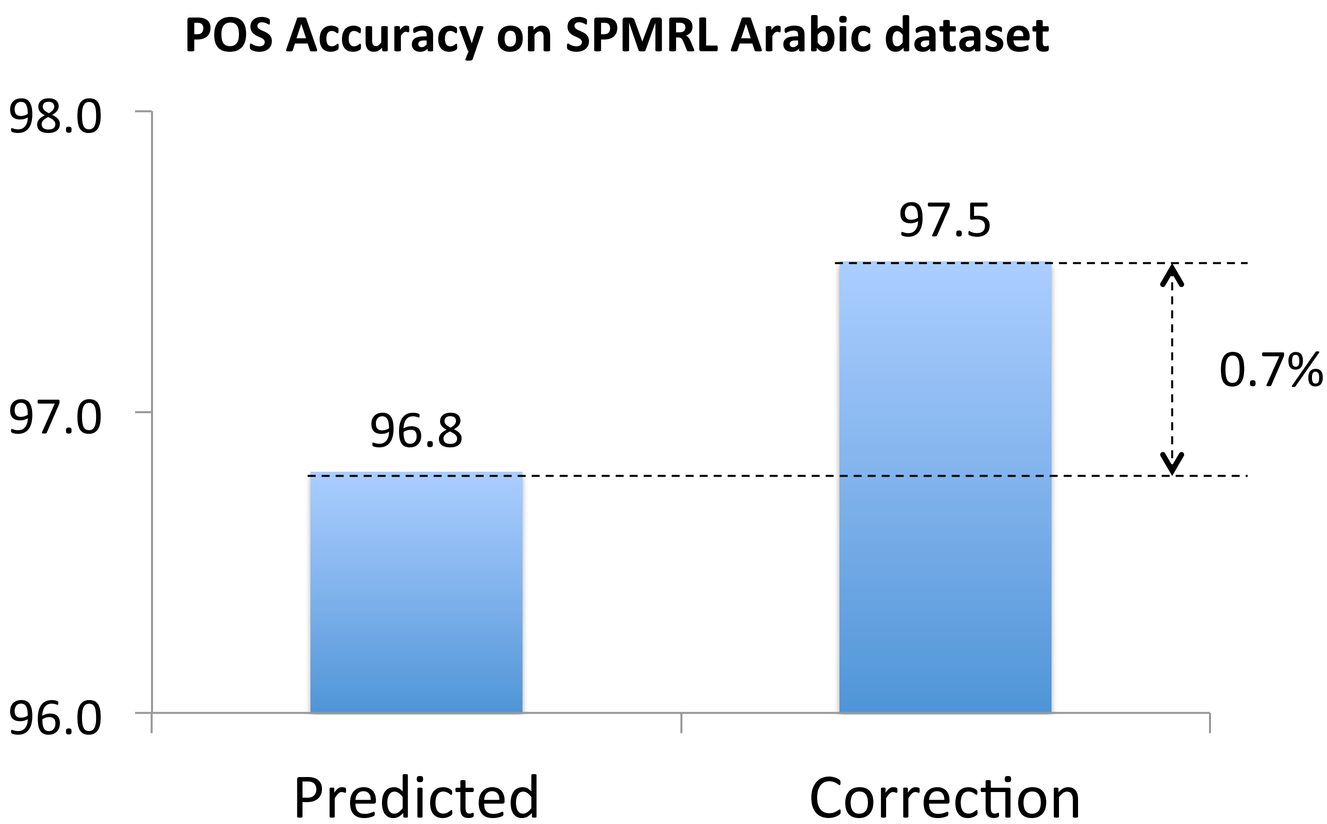
Experimental Setup for Joint Prediction Task

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

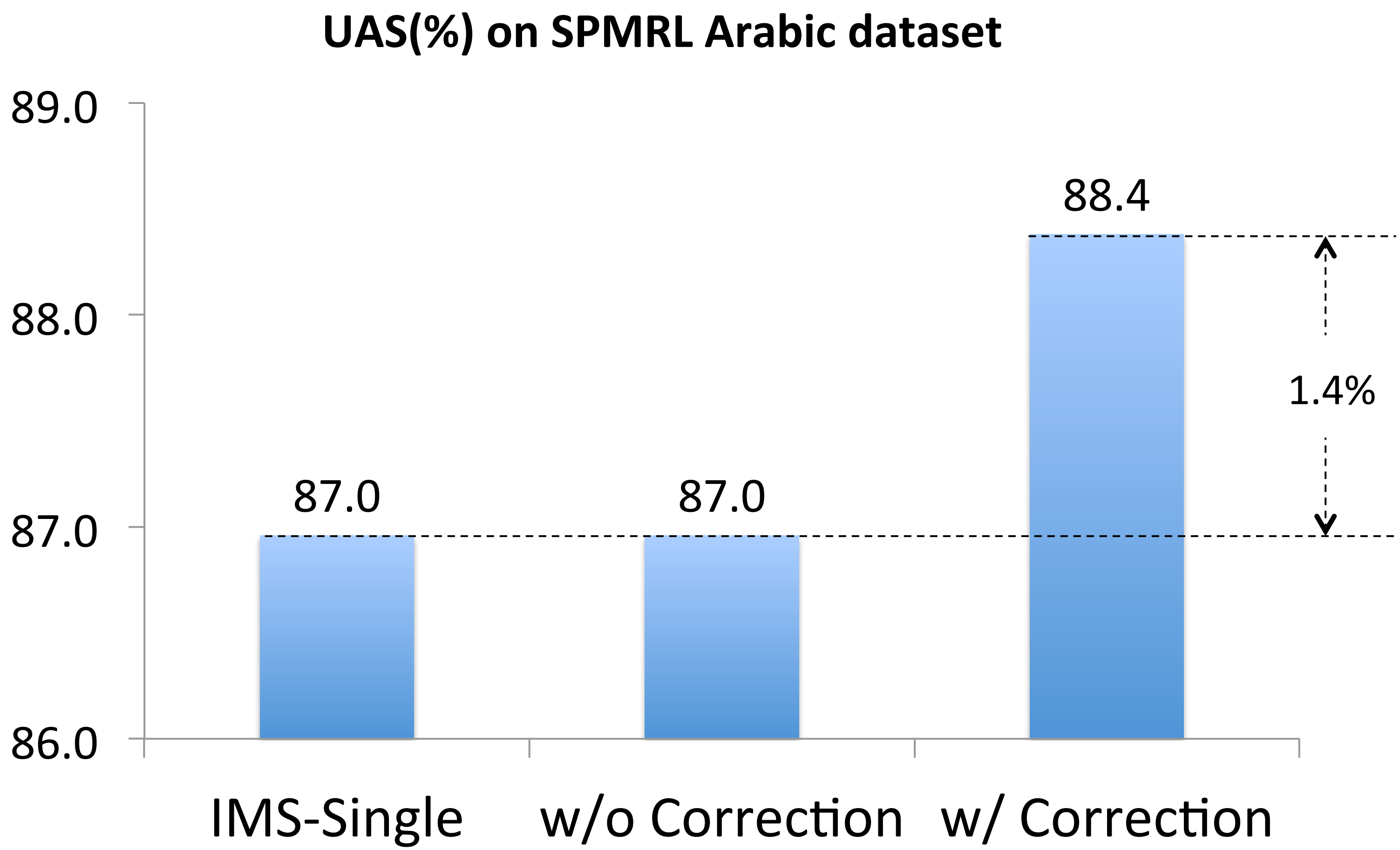
- Evaluation Metric
  - UAS: Unlabeled Attachment Score
  - POS tagging accuracy

- 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



Results on Joint Parsing and POS Correction



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:

http://groups.csail.mit.edu/rbg/code/global/acl2014

Thank You!