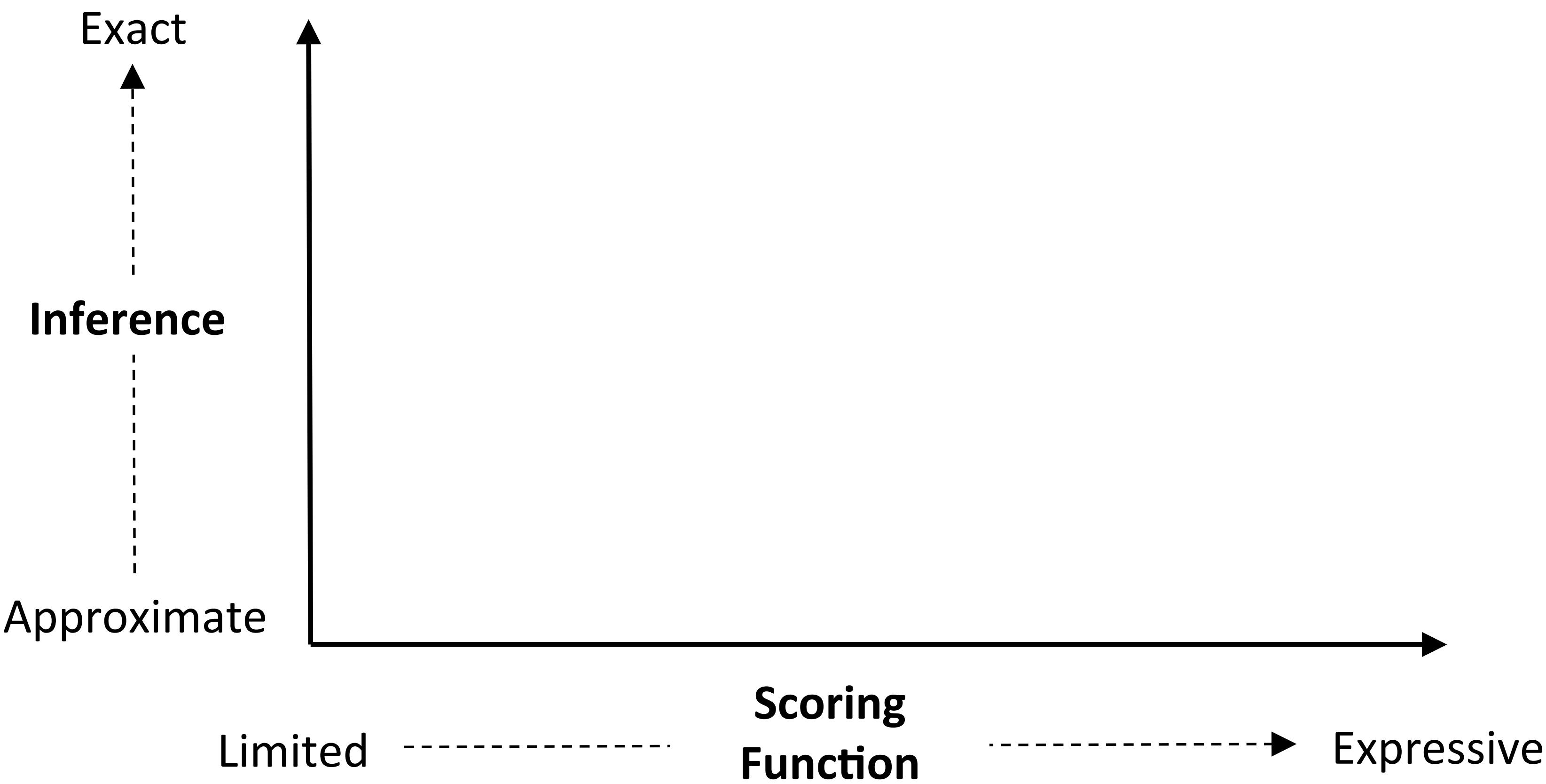
Greed is Good if Randomized: New Inference for Dependency Parsing

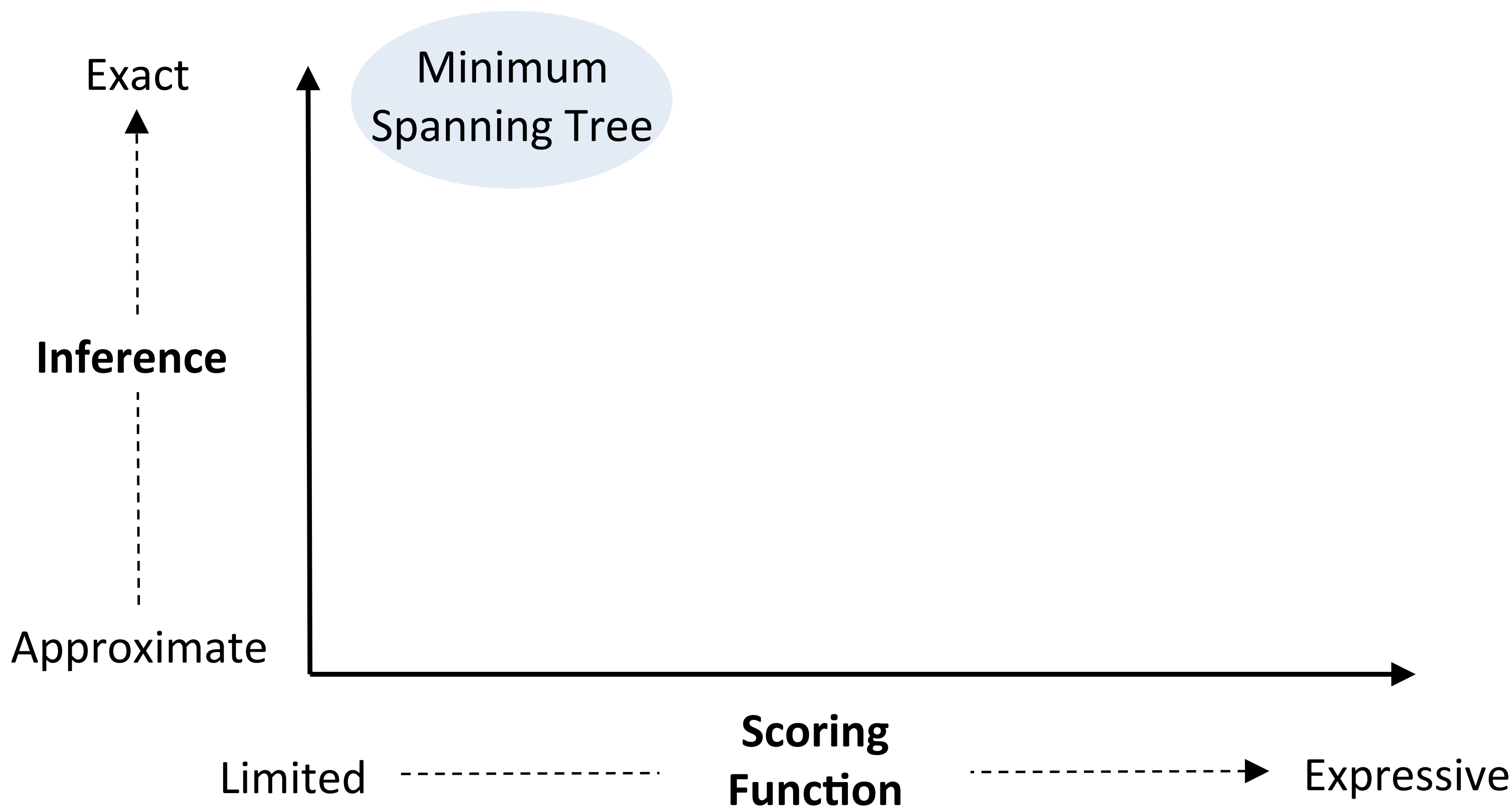
Yuan Zhang
CSAIL, MIT

Joint work with Tao Lei,
Regina Barzilay, and Tommi Jaakkola

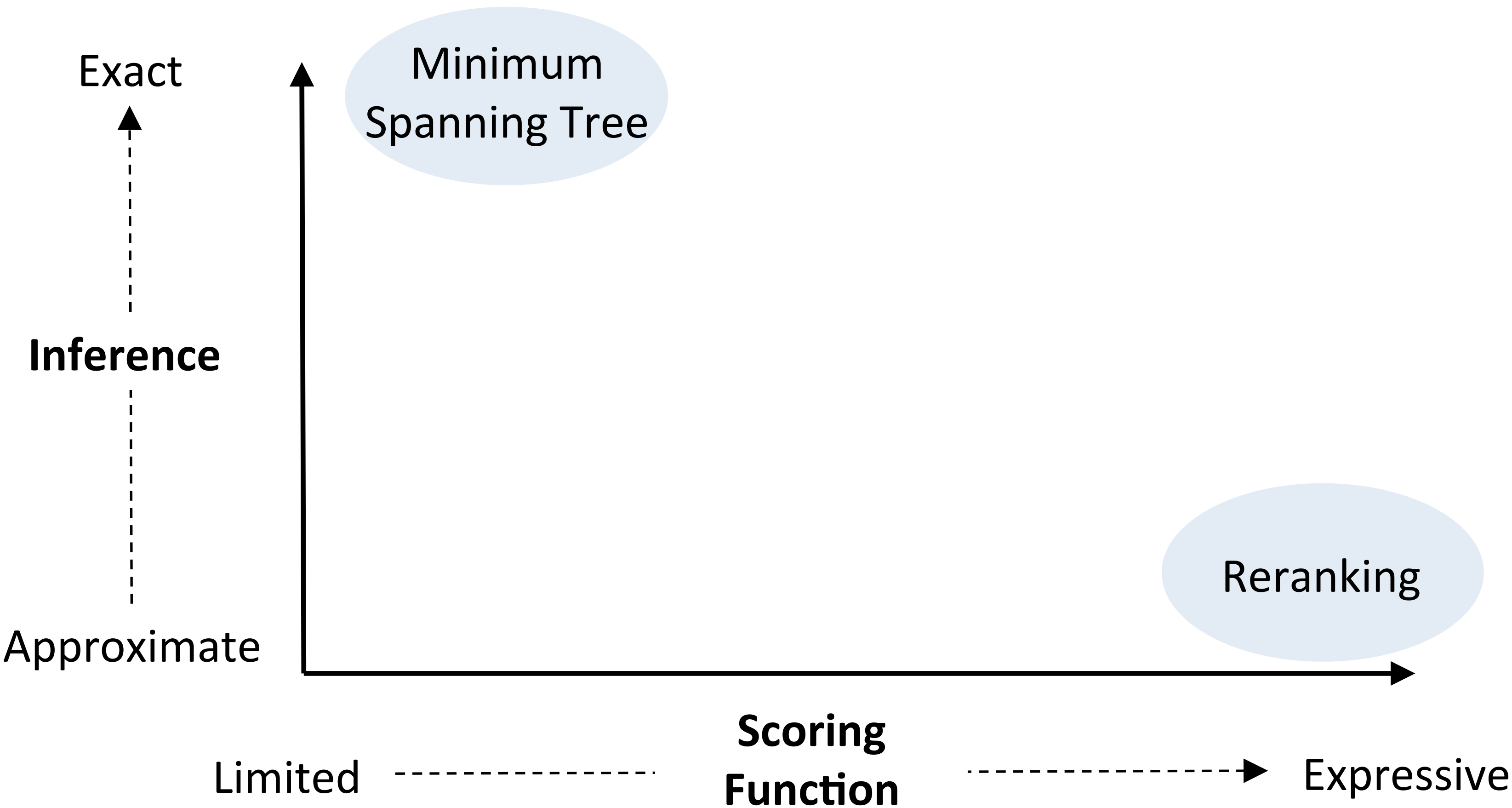
Inference vs. Scoring



Inference vs. Scoring

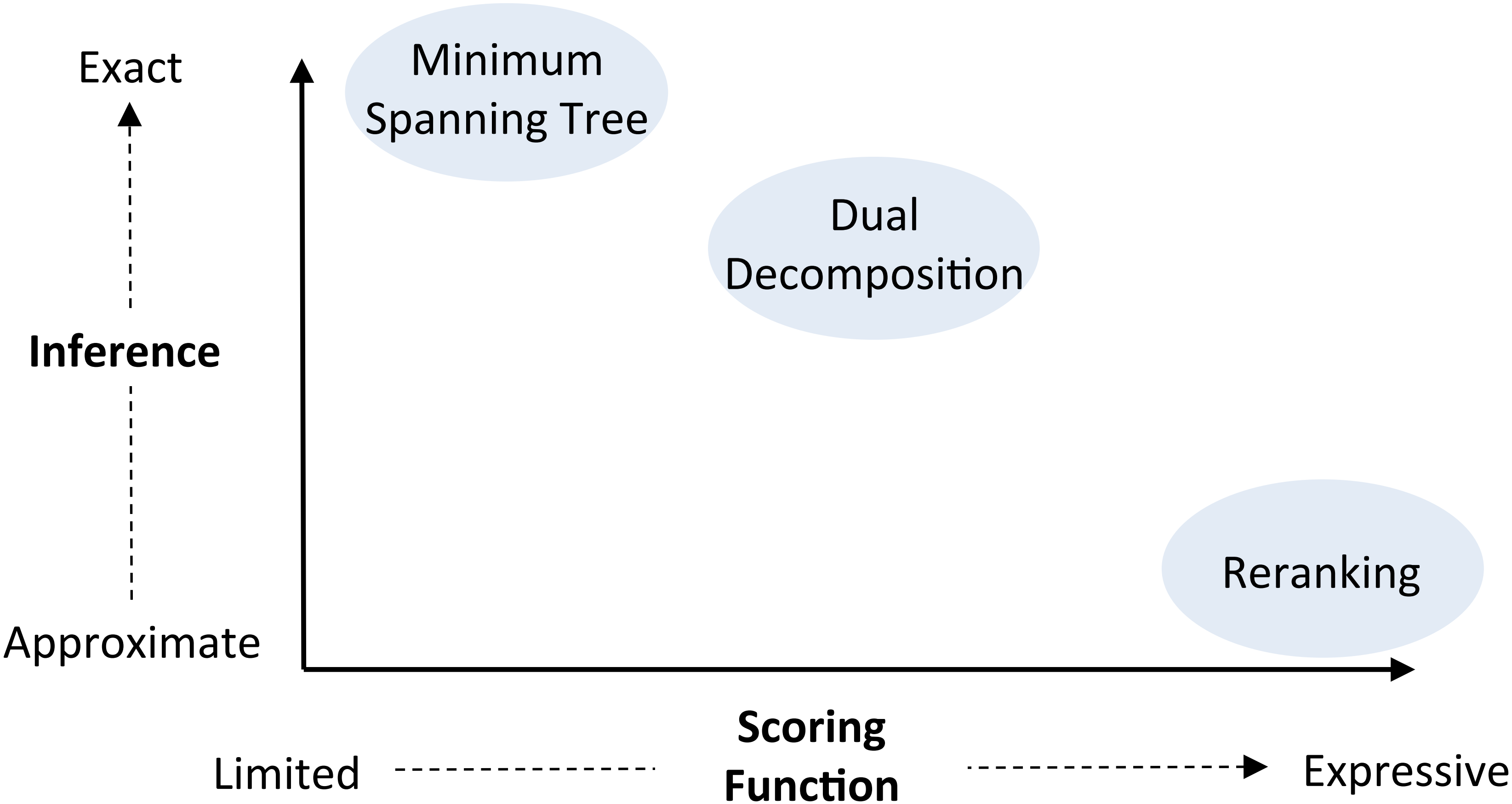


Inference vs. Scoring



- Reranking: incorporate arbitrary features

Inference vs. Scoring



- Reranking: incorporate arbitrary features
- Dual Decomposition: search in full space

Parsing Complexity

- High-order parsing is NP-hard (McDonald et al., 2006)
- Hypothesis: parsing is easy on average
- Many NP-hard problems are easy on average
  - MAX-SAT (Resende et al., 1997)
  - Set cover (Hochbaum, 1982)

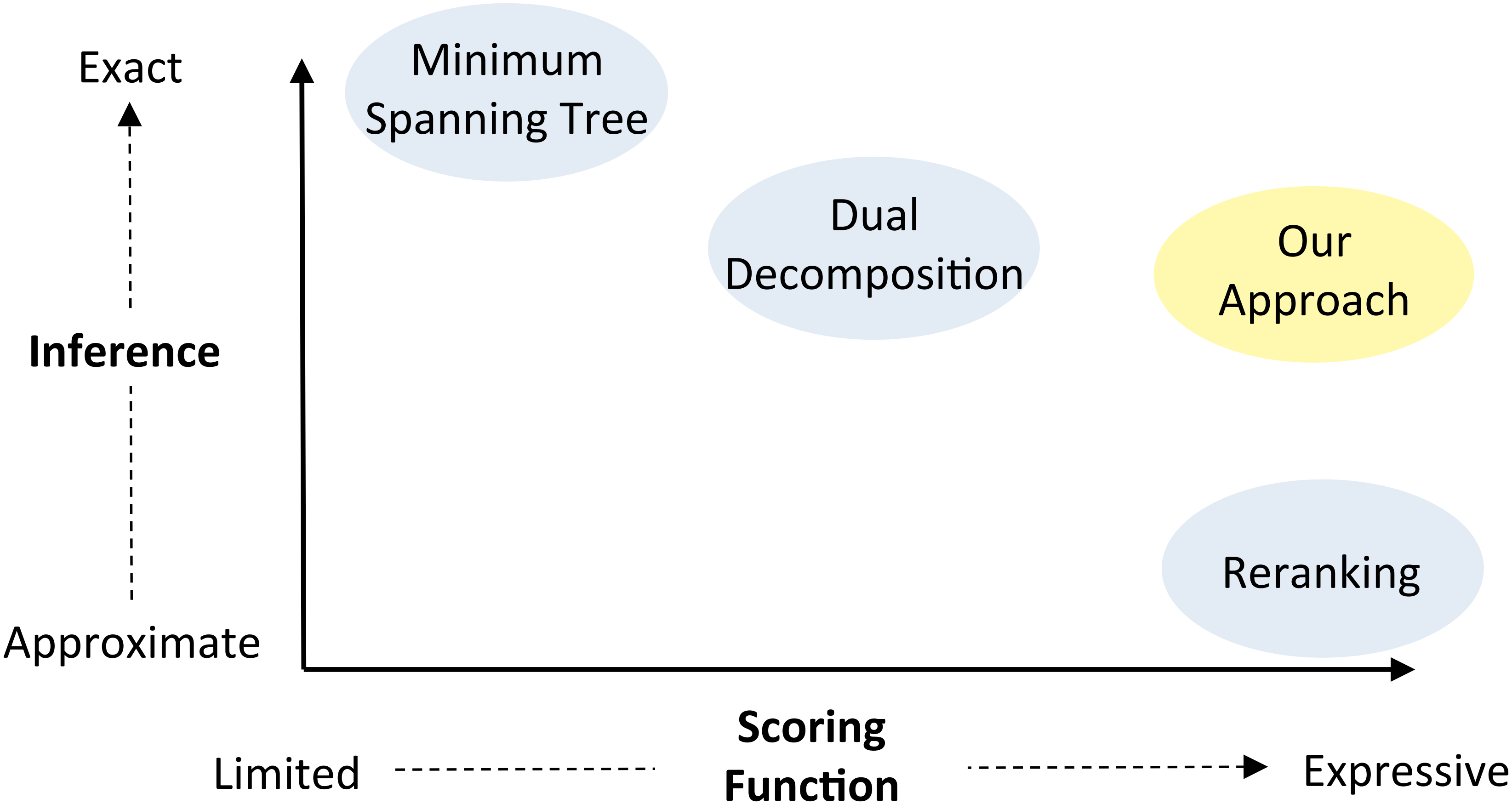
Parsing Complexity

- High-order parsing is NP-hard (McDonald et al., 2006)
- Hypothesis: parsing is easy on average
- Many NP-hard problems are easy on average
  - MAX-SAT (Resende et al., 1997)
  - Set cover (Hochbaum, 1982)

We show

- Analysis on average parsing complexity
- A simple inference algorithm based on the analysis

Our Approach



- Reranking: incorporate arbitrary features
- Dual Decomposition: search in full space

Core Idea

- Climb to the optimal tree in a few small greedy steps

Randomized Hill-climbing

For k = 1 to K

1) Randomly sample a dependency tree
2) Greedily improve the tree one edge at a time
3) Repeat (2) until converge

Select the tree with the highest score

Core Idea

- Climb to the optimal tree in a few small greedy steps

Randomized Hill-climbing

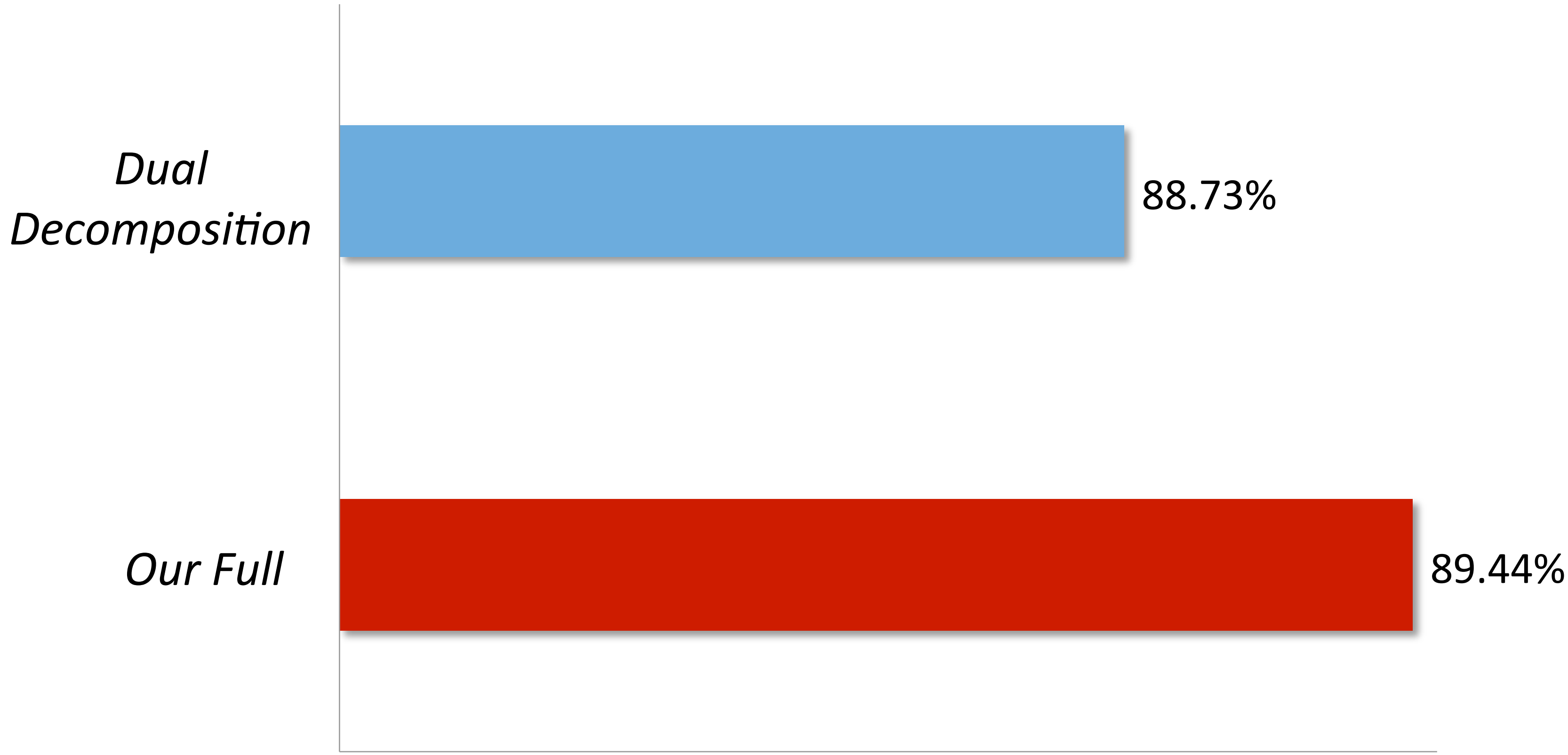
For $k = 1$ to K

1) Randomly sample a dependency tree
2) Greedily improve the tree one edge at a time
3) Repeat (2) until converge

Select the tree with the highest score

That's it!

It Works!



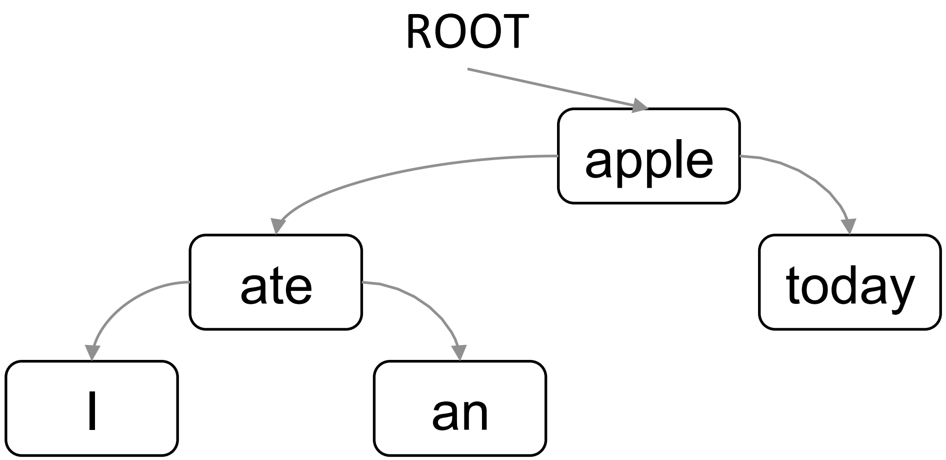
Parsing Performance on CoNLL Dataset

Example

" I ate an apple today"

Example

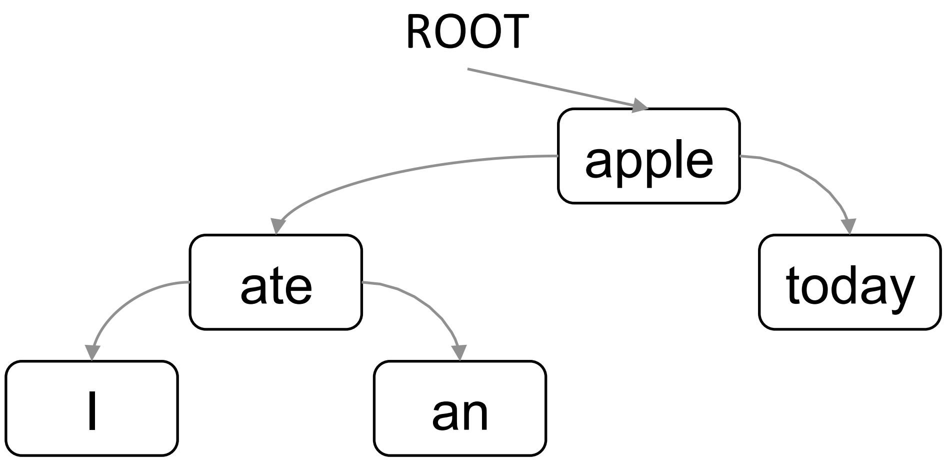
Initial tree



" I ate an apple today"

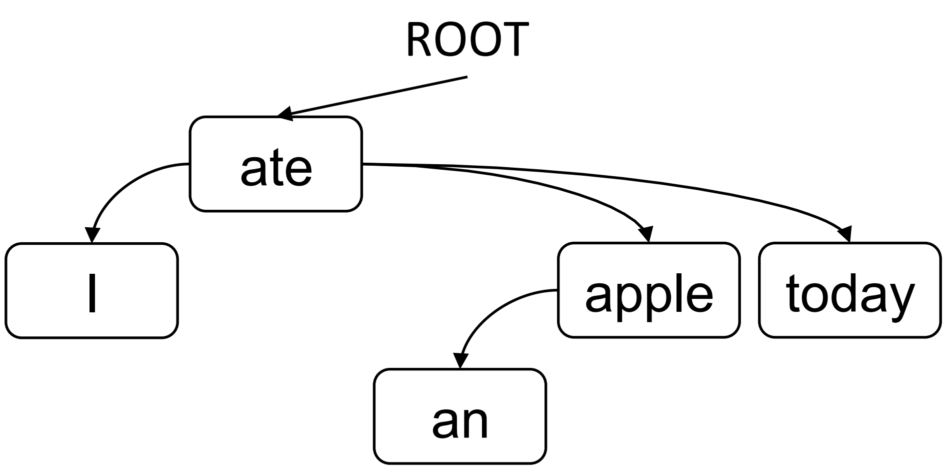
Example

Initial tree



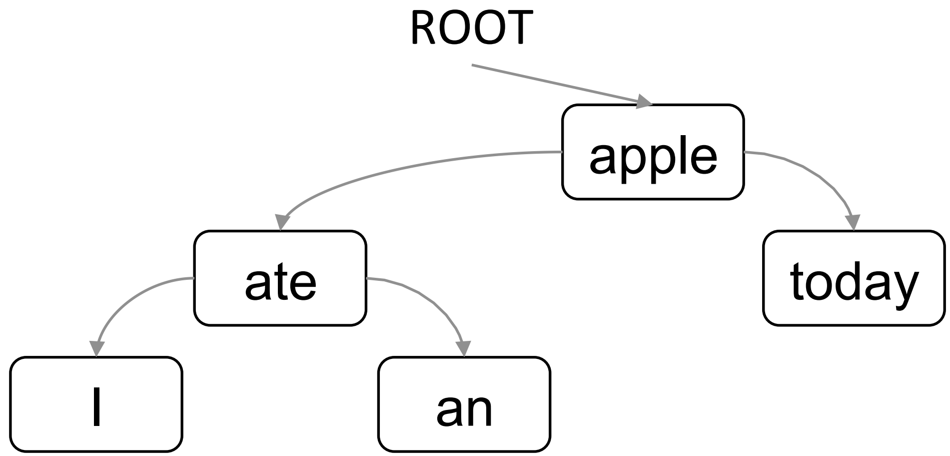
“ I ate an apple today”

Target tree



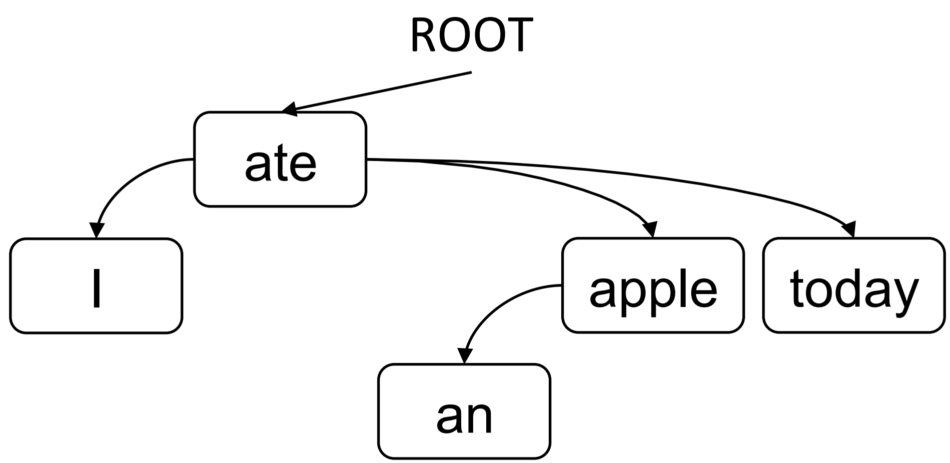
Example

Initial tree

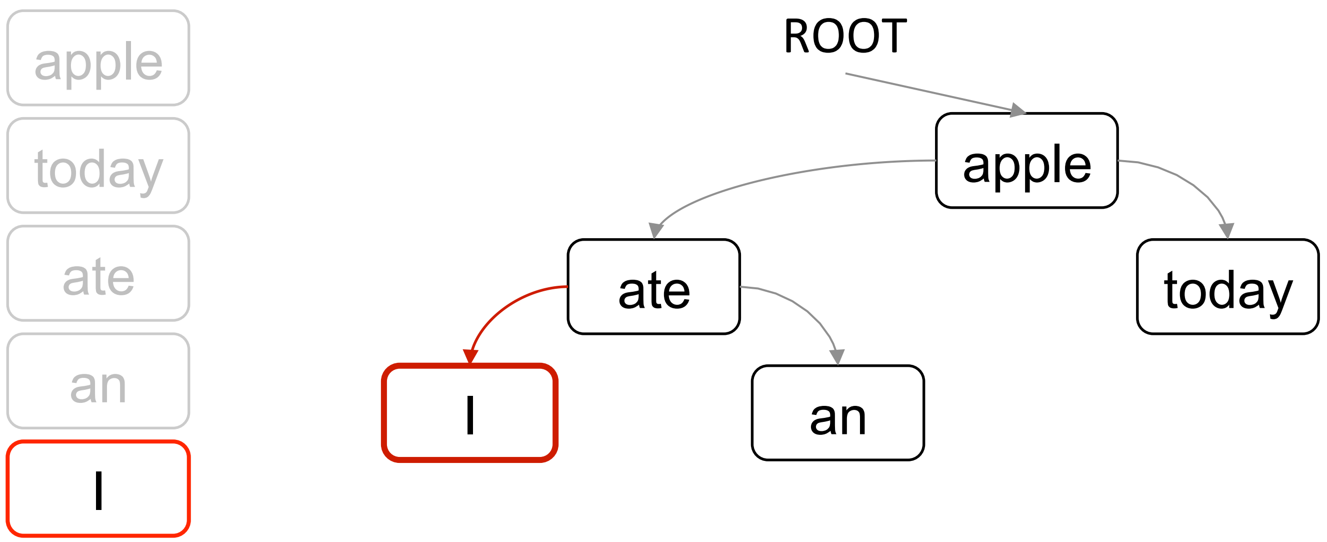


“ I ate an apple today”

Target tree

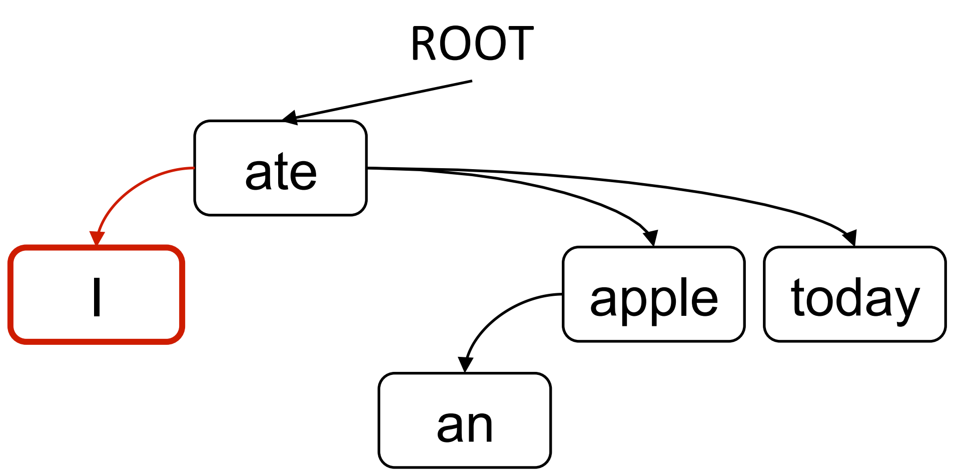


Example

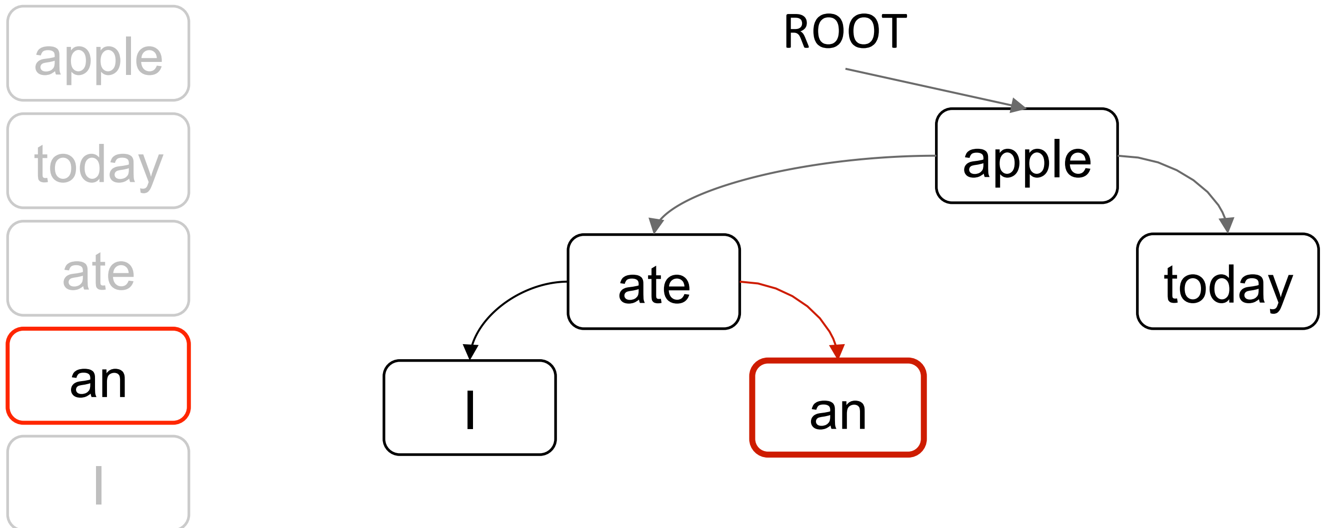


“I ate an apple today”

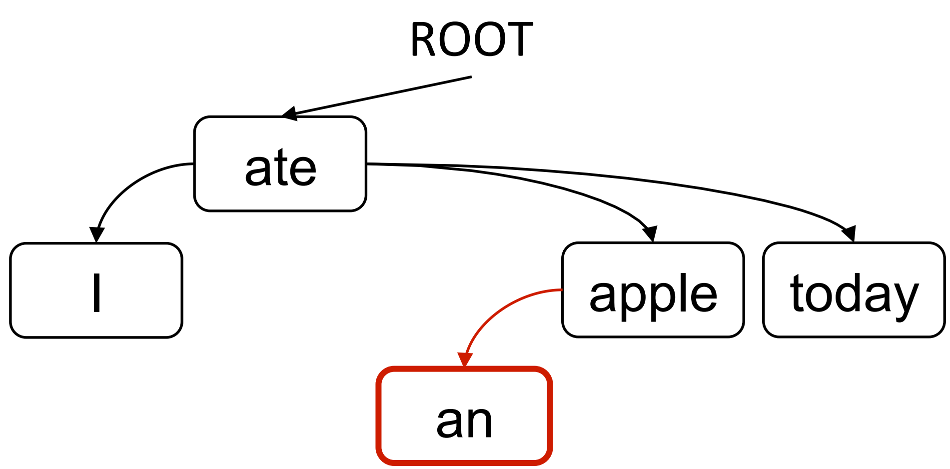
Target tree



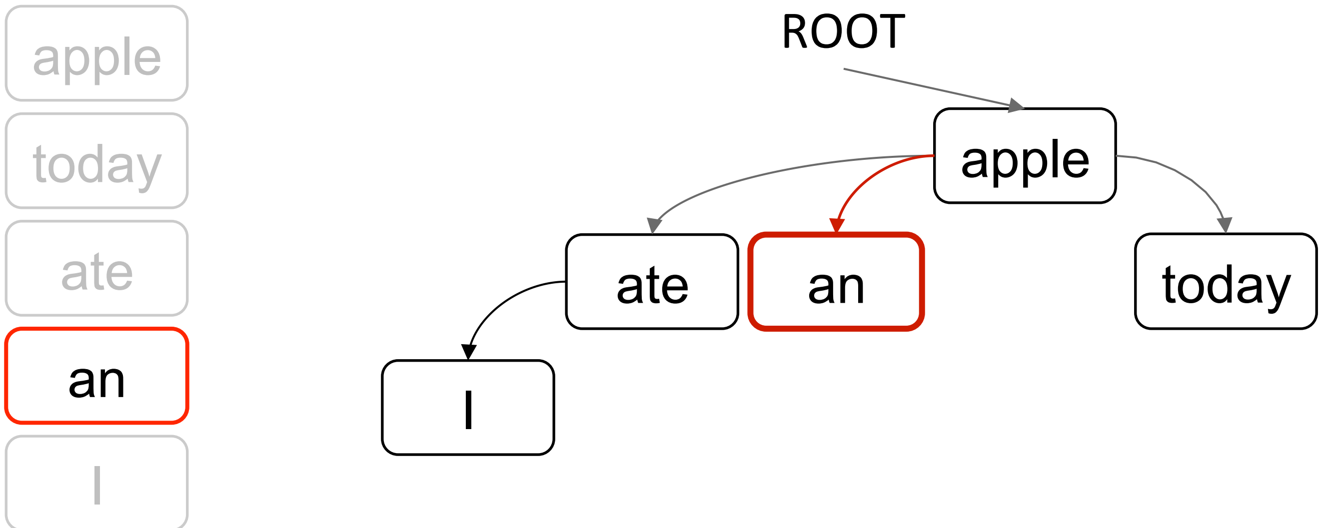
Example



“ I ate an apple today”

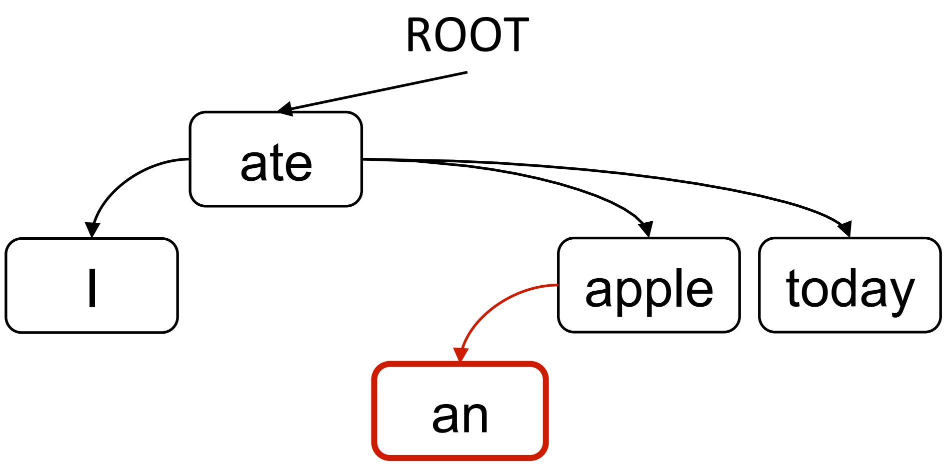


Example

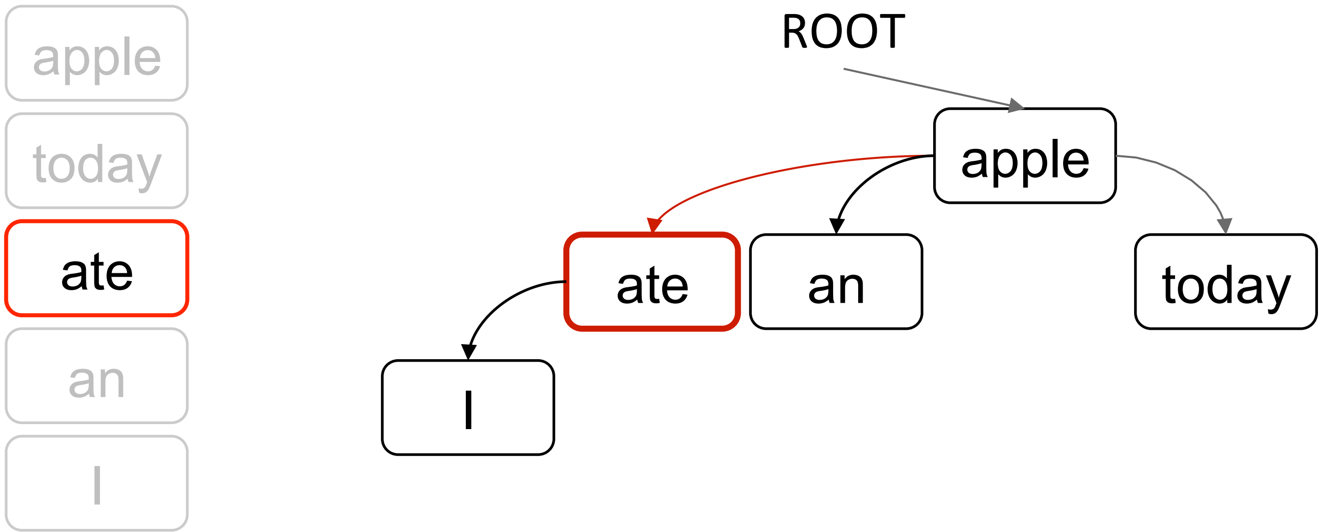


"I ate an apple today"

Target tree

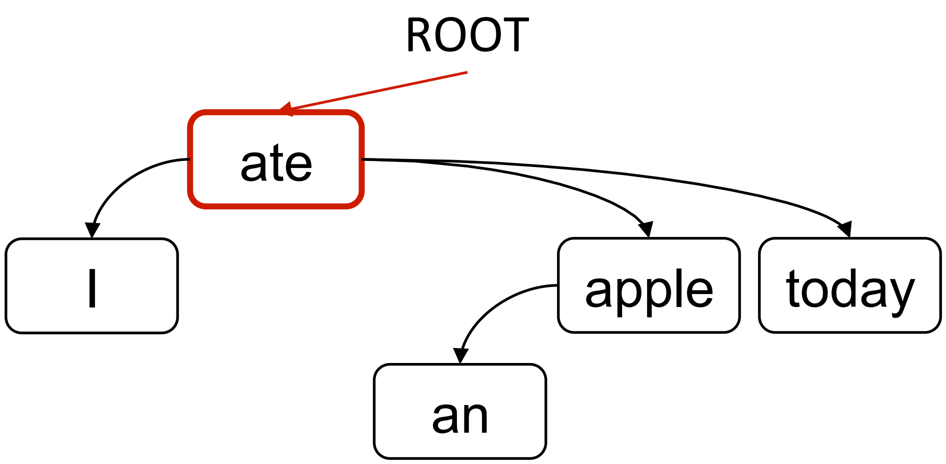


Example

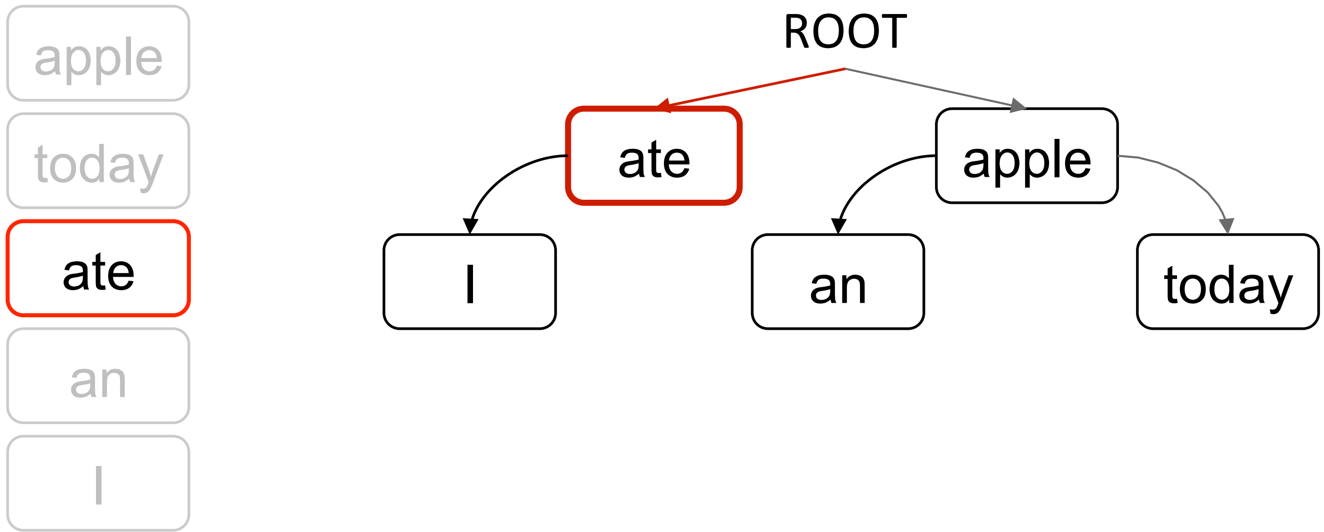


"I ate an apple today"

Target tree

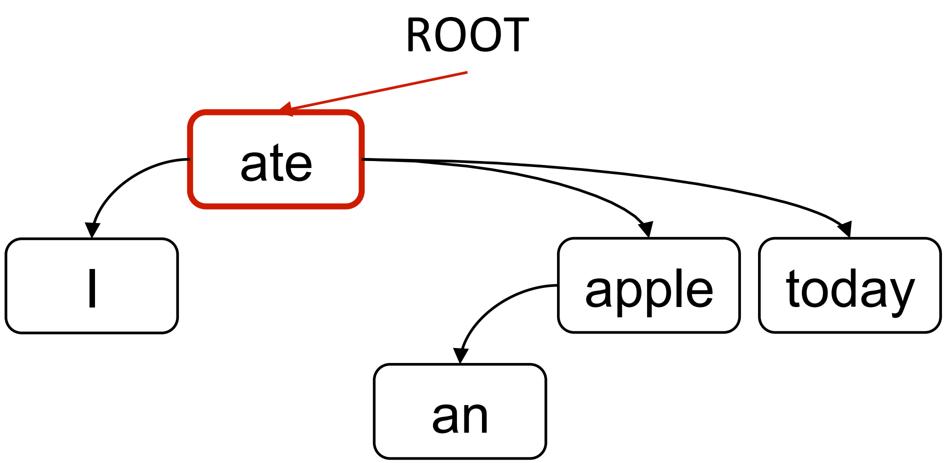


Example

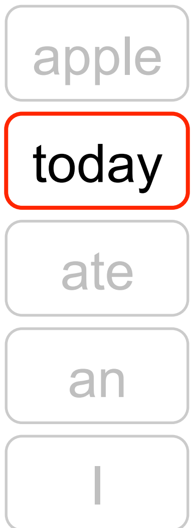


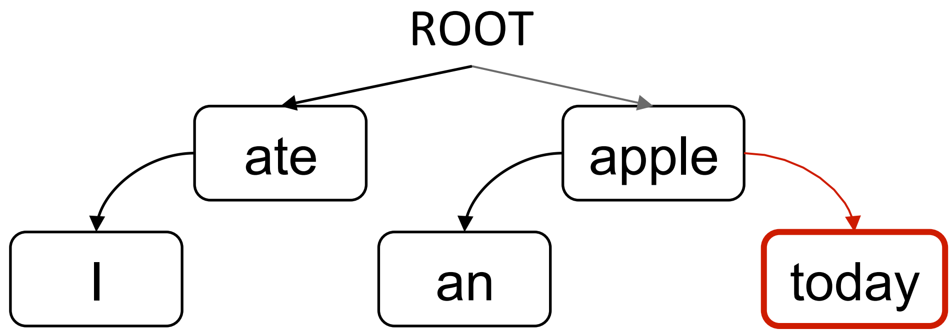
“ I ate an apple today”

Target tree



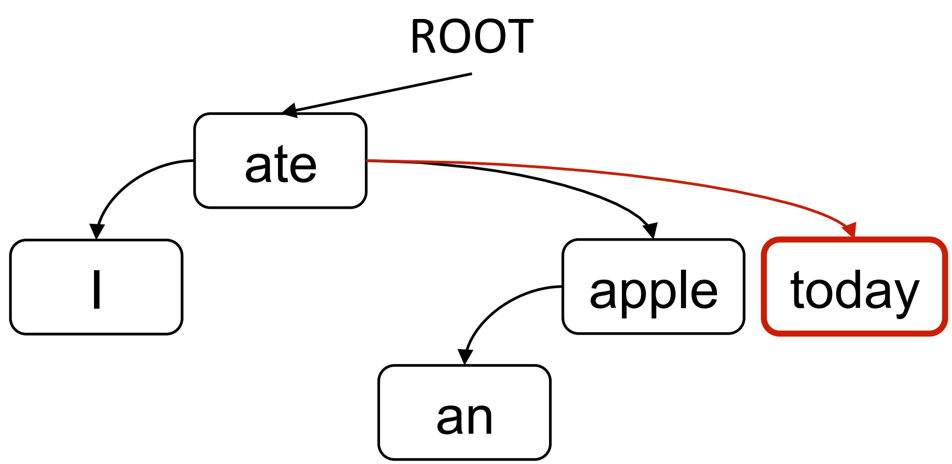
Example



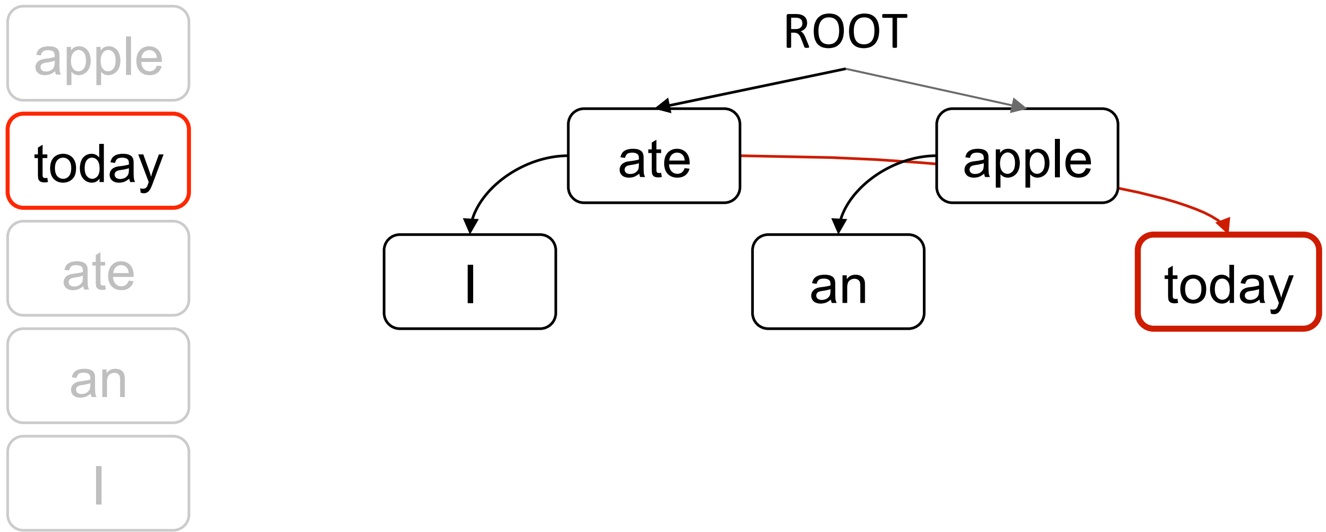


"I ate an apple today"

Target tree

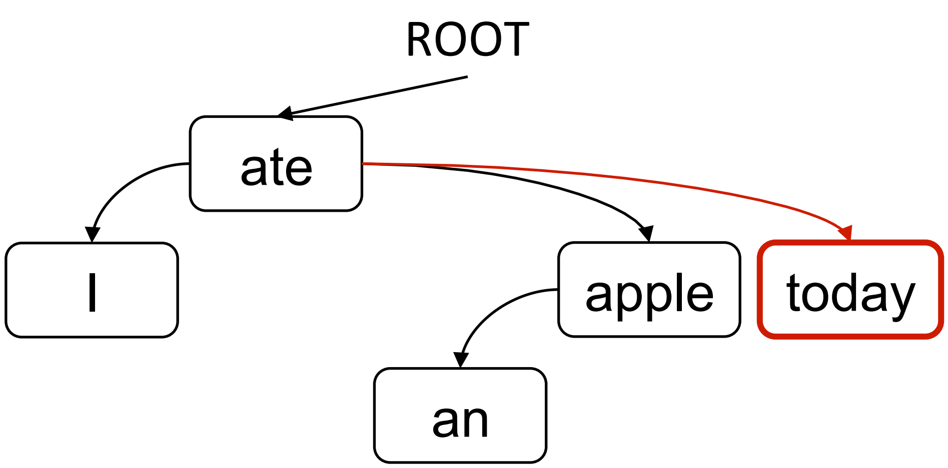


Example

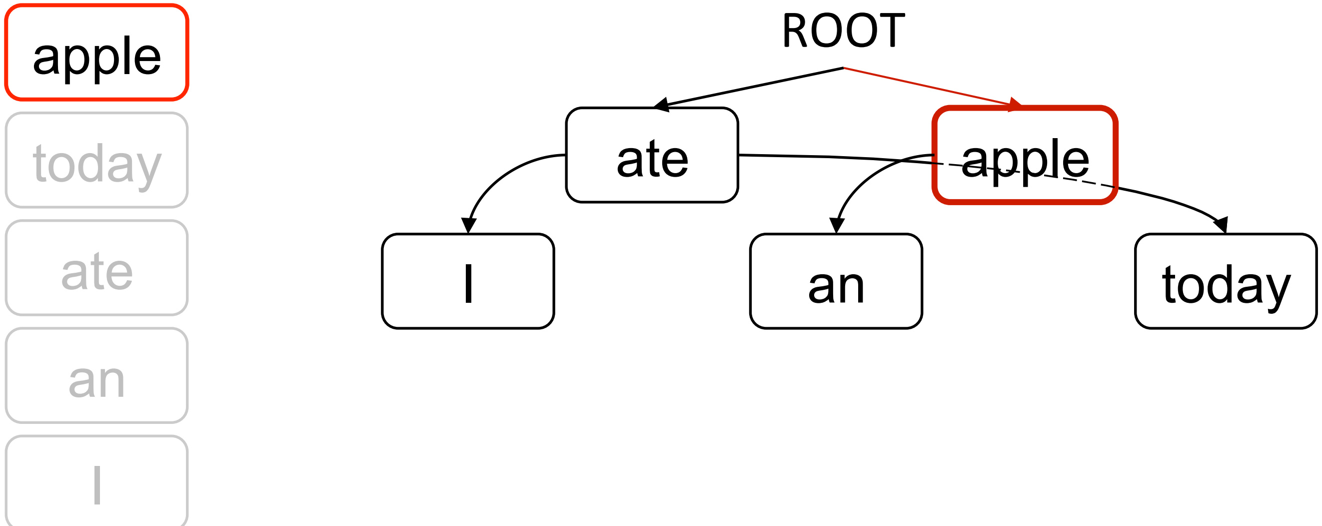


" I ate an apple today"

Target tree

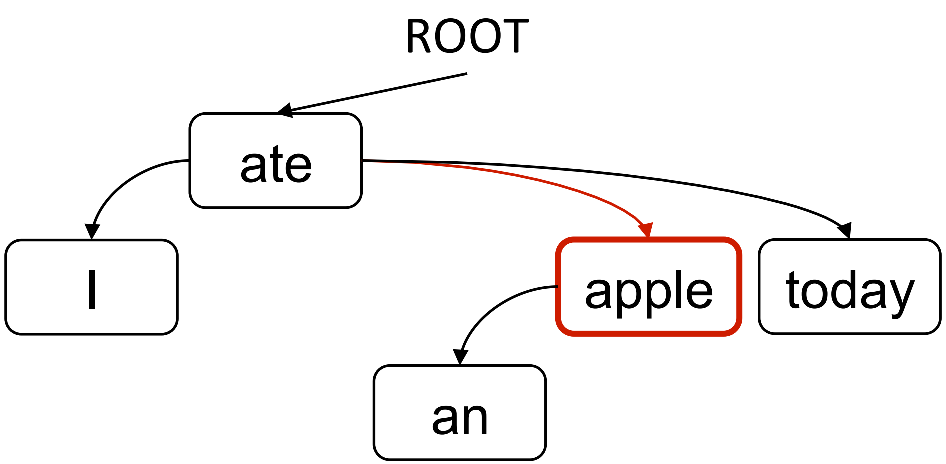


Example

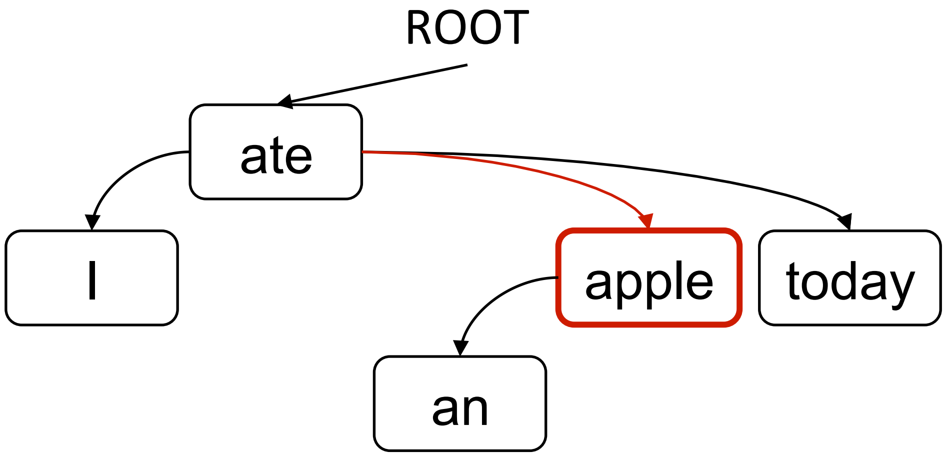


"I ate an apple today"

Target tree

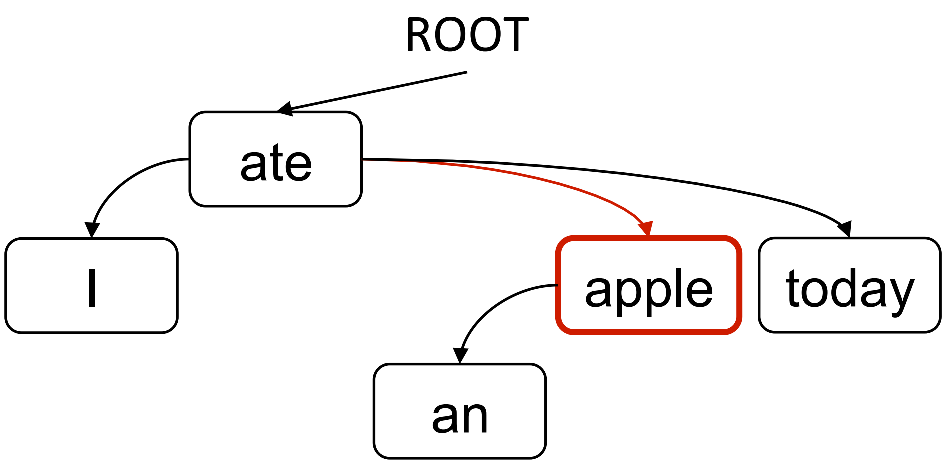


Example

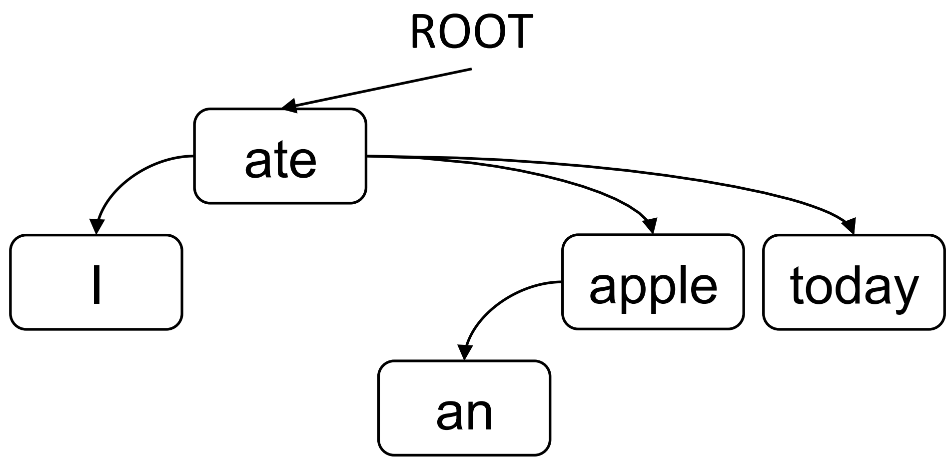


" I ate an apple today"

Target tree

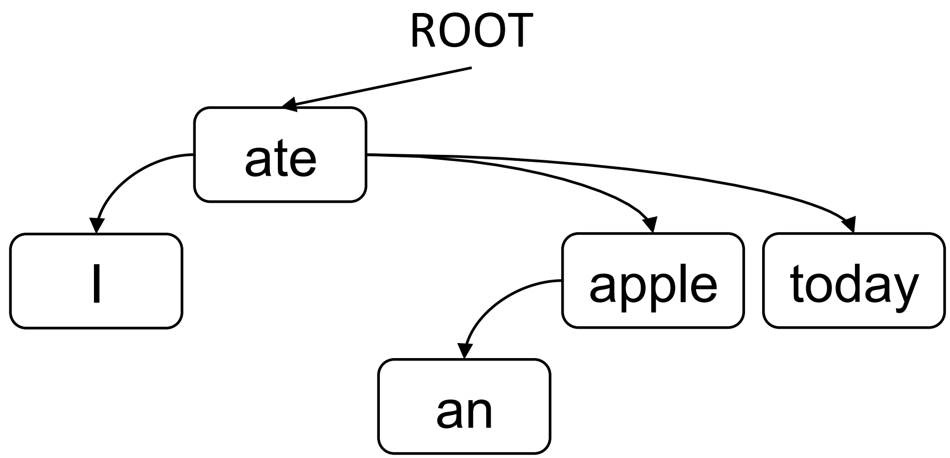


Example

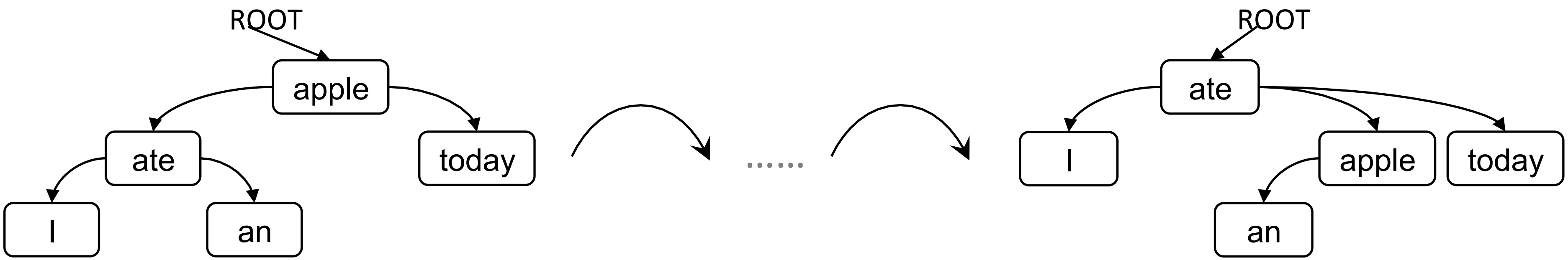


" I ate an apple today"

Target tree



Why Greedy Has a Chance to Work

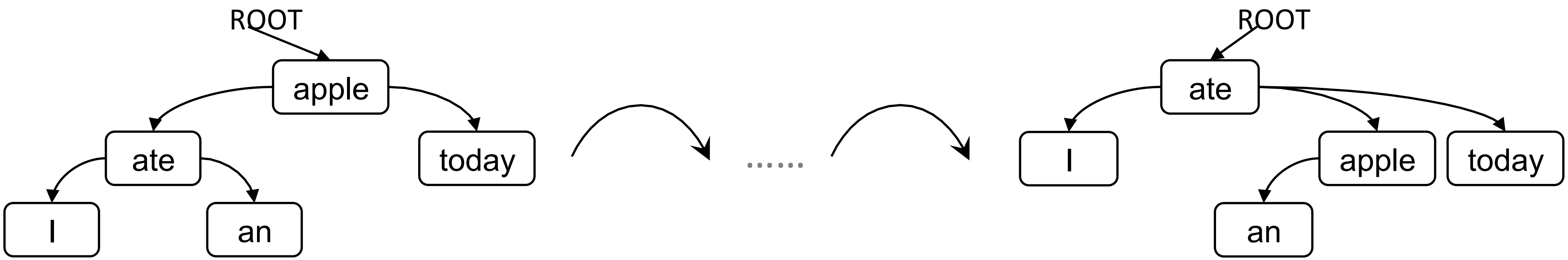


$$y^{(0)} \dashrightarrow y^{(T)}$$

Reachability: transforming any tree to any other tree

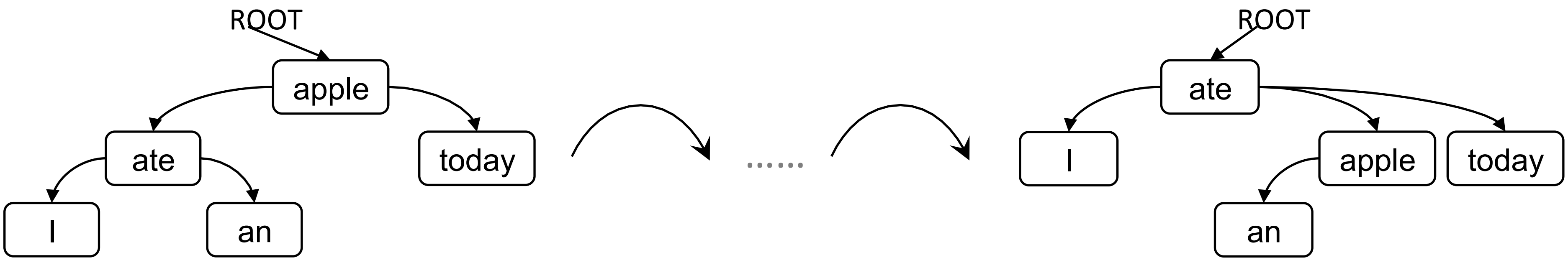
- maintaining the structure a valid tree at any point
- using as few as *d* steps (*d* : head differences/hamming distance)

Greedy Hill-climbing



$y^{(0)}$ ------ increase $S(x, y^{(t)})$ ------> $y^{(T)}$

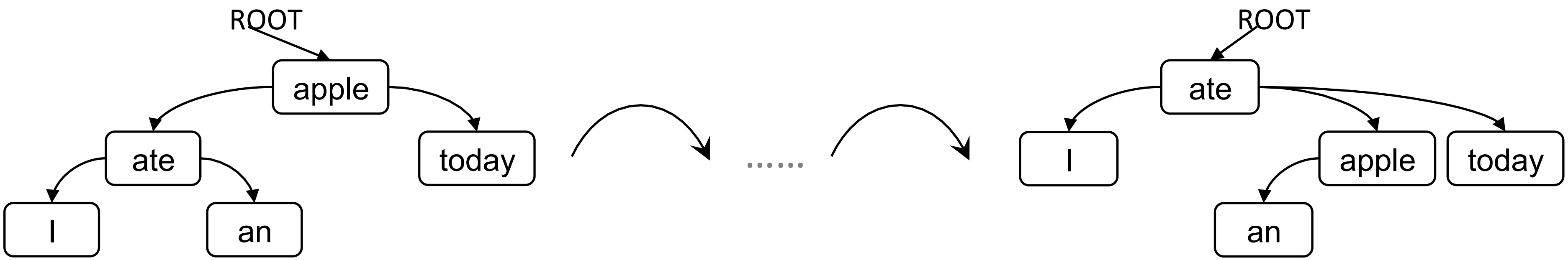
Greedy Hill-climbing



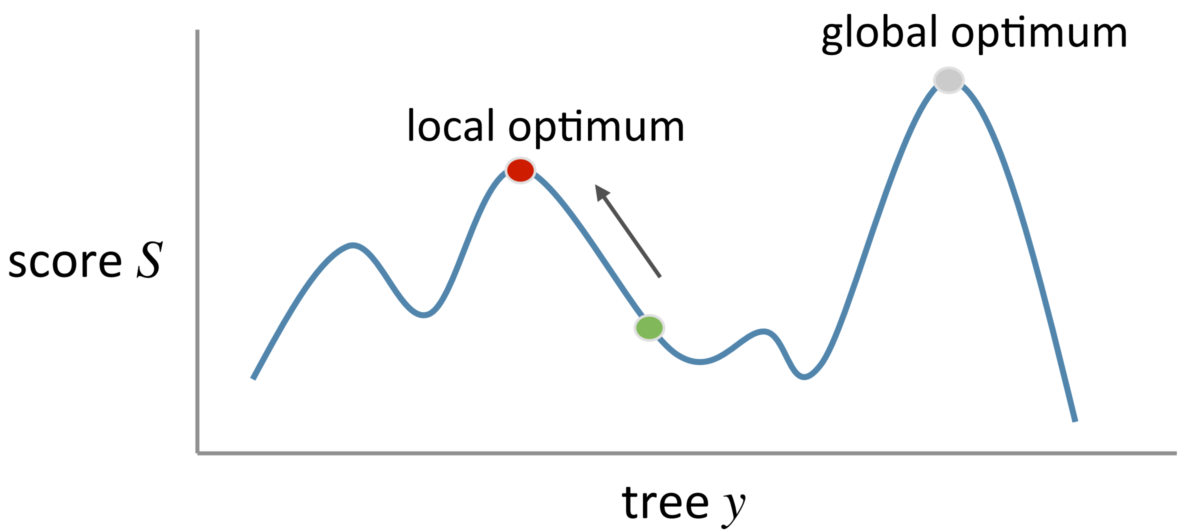
$y^{(0)}$ - - - - - increase $S(x, y^{(t)})$ - - - - - → $y^{(T)}$

Arbitrary features in the scoring function

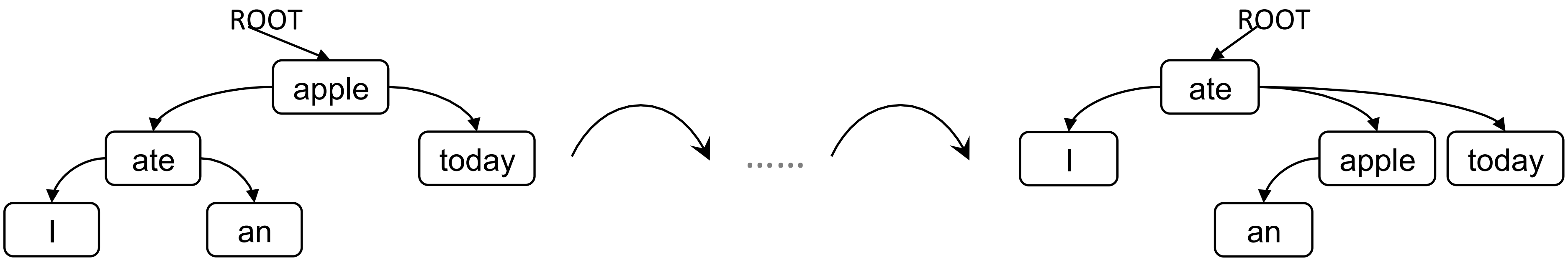
Challenge: Local Optimum

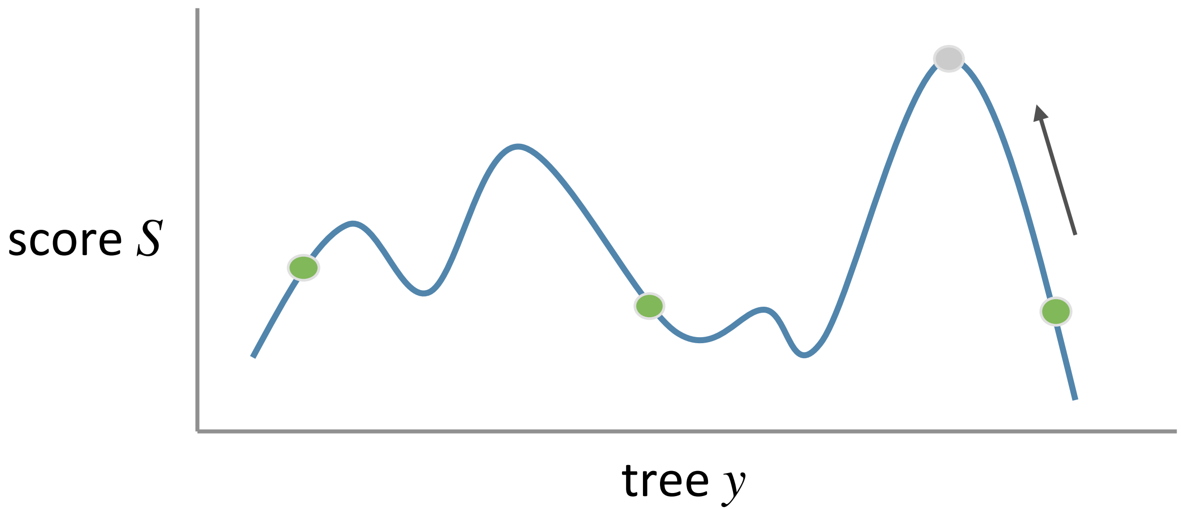


$y^{(0)}$ - - - - - - increase $S(x, y^{(t)})$ - - - - - -→ $y^{(T)}$



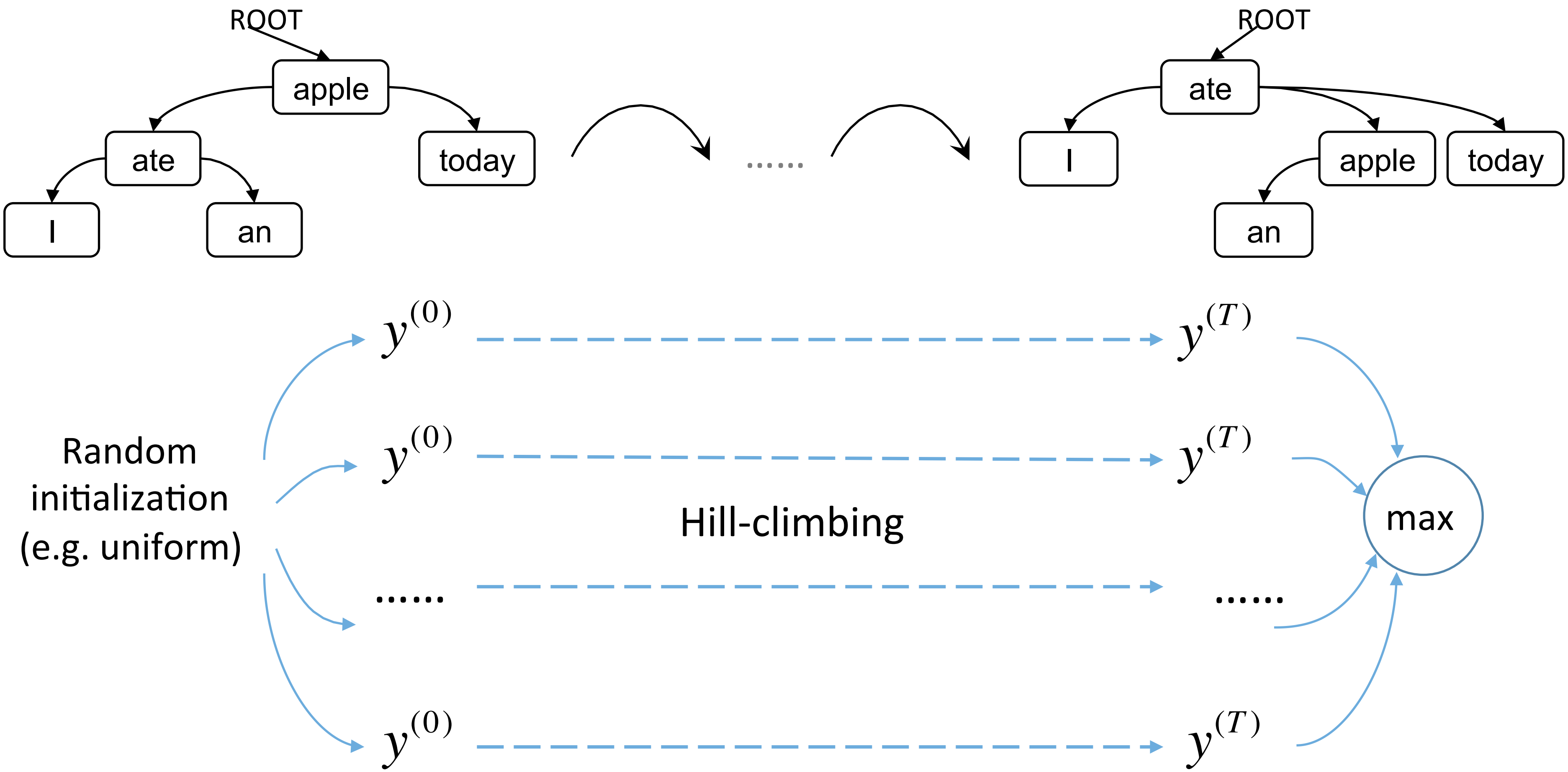
Hill-climbing with Restarts





Overcome local optima via restarts

Hill-climbing with Restarts



Overcome local optima via restarts

Learning Algorithm

- Follow common max-margin framework

$$\forall y \in T(x) \qquad S(x,\hat{y}) \geq S(x,y) + |\hat{y} - y| - \xi$$

  - $\hat{y}$ is the gold tree

Learning Algorithm

- Follow common max-margin framework

$$\forall y \in T(x) \qquad S(x,\hat{y}) \geq S(x,y) + |\hat{y} - y| - \xi$$

  - $\hat{y}$ is the gold tree

- Adopt passive-aggressive online learning framework (Crammer et al. 2006)

- Decode with our randomized greedy algorithm

Analysis

Analysis

	Theoretical	Empirical
First-order	✓	✓

Analysis

	Theoretical	Empirical
First-order	✓	✓
High-order	?	✓

Analysis

	Theoretical	Empirical
First-order	✓	✓
High-order	?	✓

Search Space Complexity: First-order

10 words

Search Space Complexity: First-order

≈ 2 billion trees

10 words

Search Space Complexity: First-order

≈ 2 billion trees

10 words

< 512 local optima

Search Space Complexity: First-order

Theorem: For any first-order scoring function:

- there are at most 2^{n-1} locally optimal trees
- this upper bound is **tight**

Search Space Complexity: First-order

Theorem: For any first-order scoring function:

- there are at most 2^{n-1} locally optimal trees
- this upper bound is **tight**

➡ 2^{n-1} is still a lot, but it is the worst case

Search Space Complexity: First-order

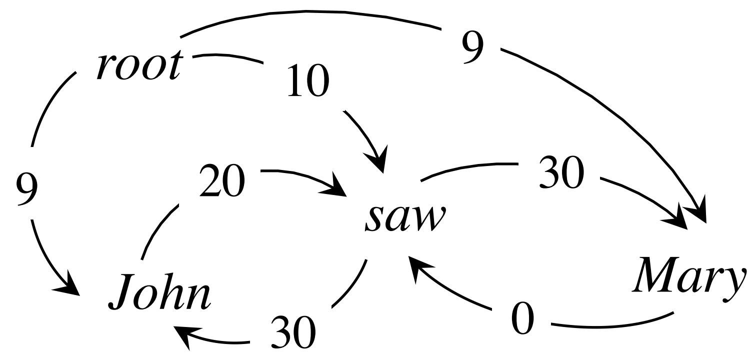
Theorem: For any first-order scoring function:

- there are at most 2^{n-1} locally optimal trees
- this upper bound is **tight**

➡ 2^{n-1} is still a lot, but it is the worst case

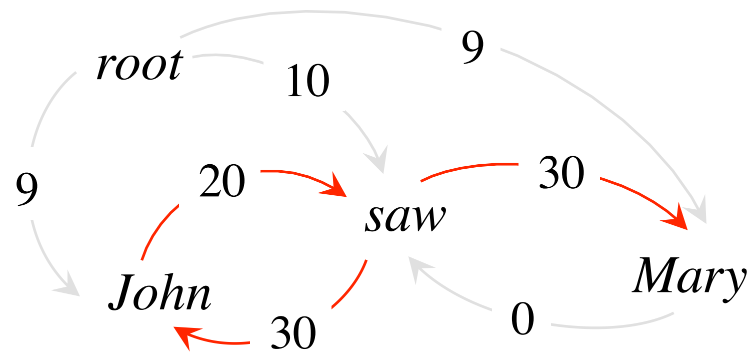
What about the average case?

Algorithm for Counting Local Optima



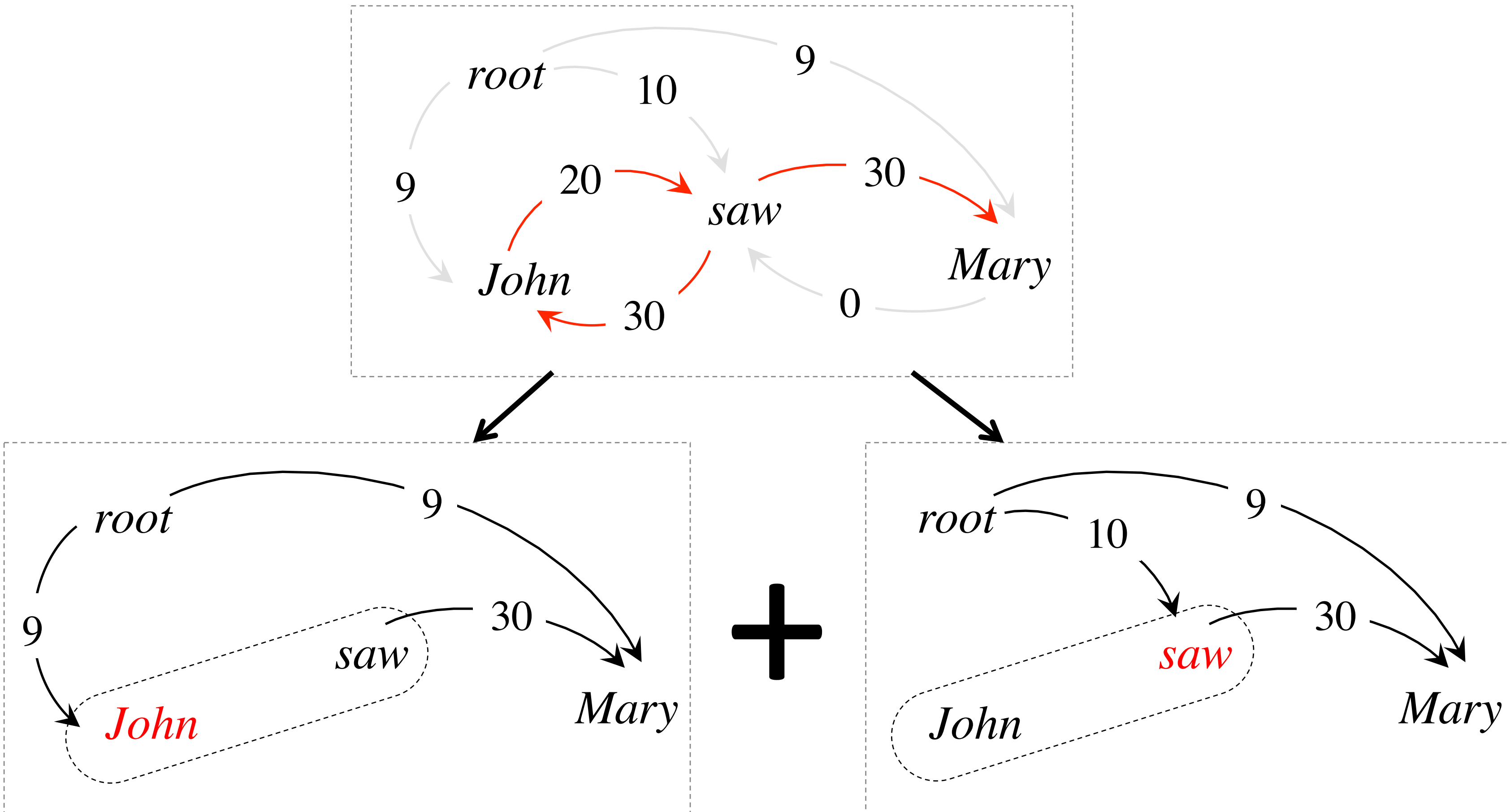
Algorithm for Counting Local Optima

The method is based on Chu-Liu-Edmonds algorithm



- Select the best heads independently

Algorithm for Counting Local Optima

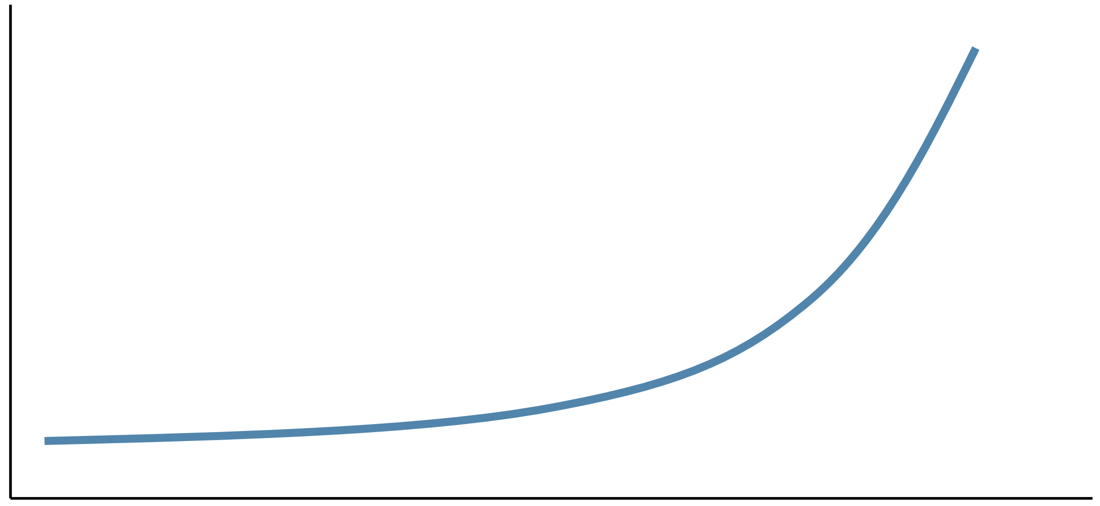


- Contract the cycle and recursively count the local optima
  - ➢ Any local optimum exactly reassigns one edge in the cycle

Empirical Results: First-order

How many local optima in real data?

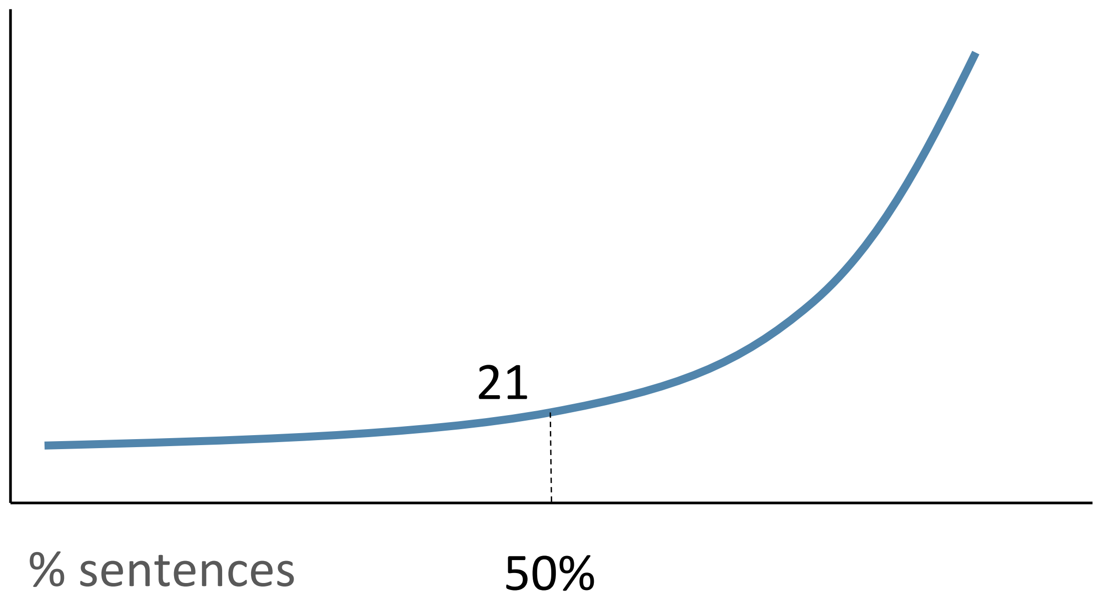
 # Optima on English Dataset



Empirical Results: First-order

How many local optima in real data?

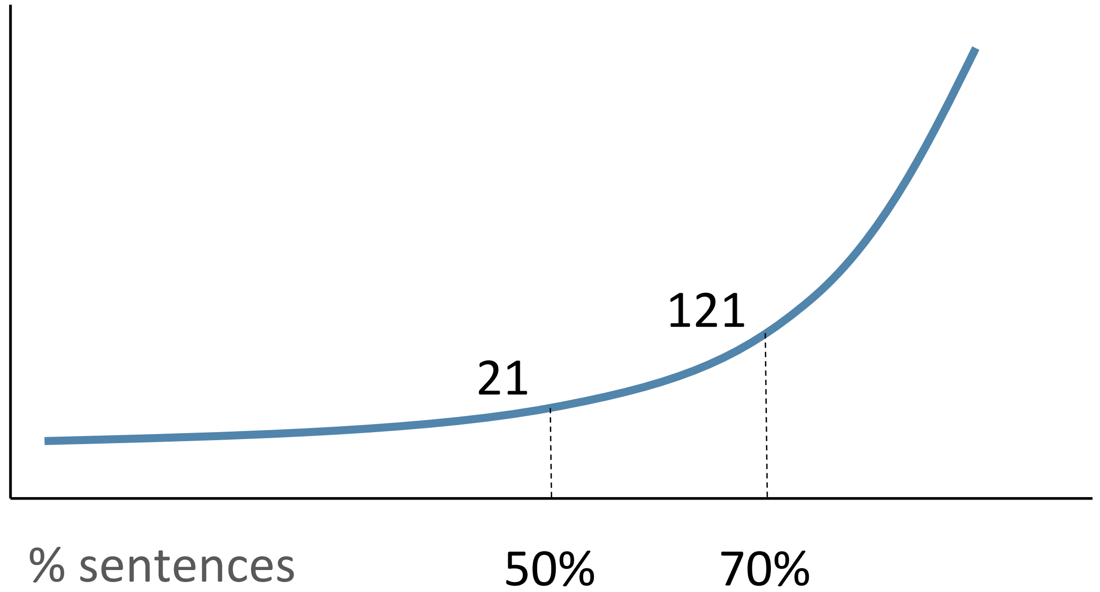
 # Optima on English Dataset



Empirical Results: First-order

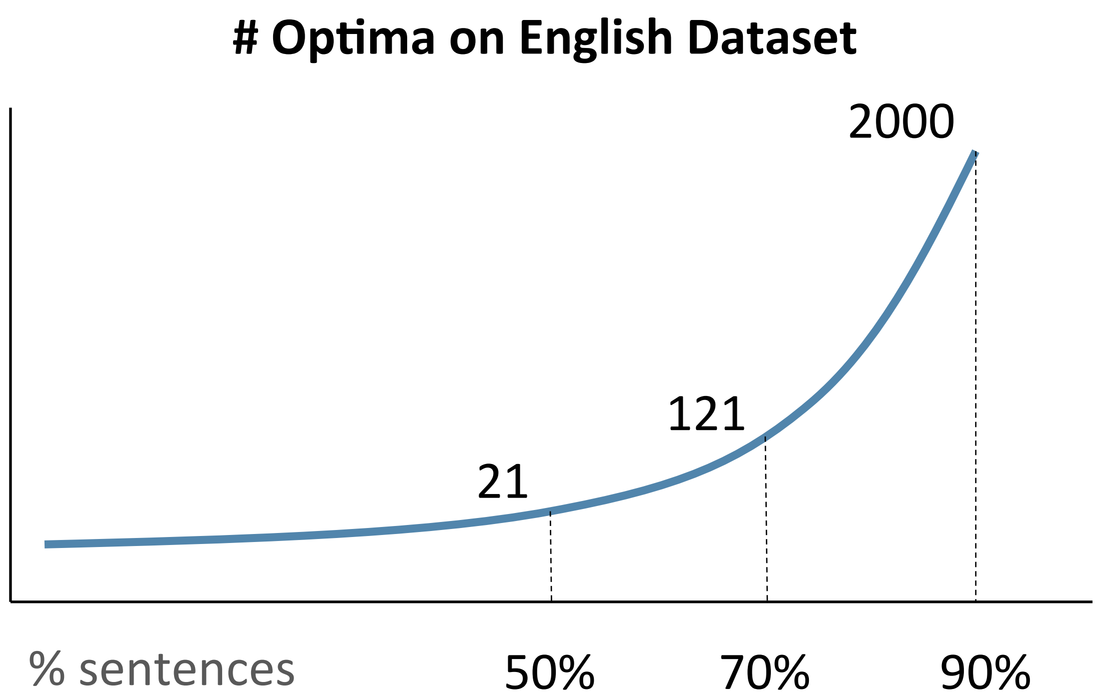
How many local optima in real data?

 # Optima on English Dataset



Empirical Results: First-order

How many local optima in real data?



Empirical Results: First-order

Does the hill-climbing find the argmax?

Finding Global Optimum on English

Empirical Results: First-order

Does the hill-climbing find the argmax?

Finding Global Optimum on English

Len. ≤ 15
100%

Len. > 15
99.3%

Easy search space leads to successful decoding

Empirical Results: High-order

Does the hill-climbing find the argmax?

Comparison on English Given DD Cert.

	% Certificate
Dual decomposition (Koo et al., 2010)	94.5%

	$S_{DD} = S_{HC}$
Given a certificate by DD	99.8%

Empirical Results: High-order

Does the hill-climbing find the argmax?

Overall Comparison on English

Empirical Results: High-order

Does the hill-climbing find the argmax?

Overall Comparison on English

$S_{DD} = S_{HC}$
98.7%

Empirical Results: High-order

Does the hill-climbing find the argmax?

Overall Comparison on English

$S_{DD} = S_{HC}$
98.7%

$S_{DD} < S_{HC}$
1.0%

$S_{DD} > S_{HC}$
0.3%

Experimental Setup

Datasets

- 14 languages in CoNLL 2006 & 2008 shared tasks

Features

- Up to 3rd-order (three arcs) features used in MST/Turbo parsers
- Global features used in re-ranking

Implementation

- Adaptive restarting strategy with $K = 300$

Baselines and Evaluation Measure

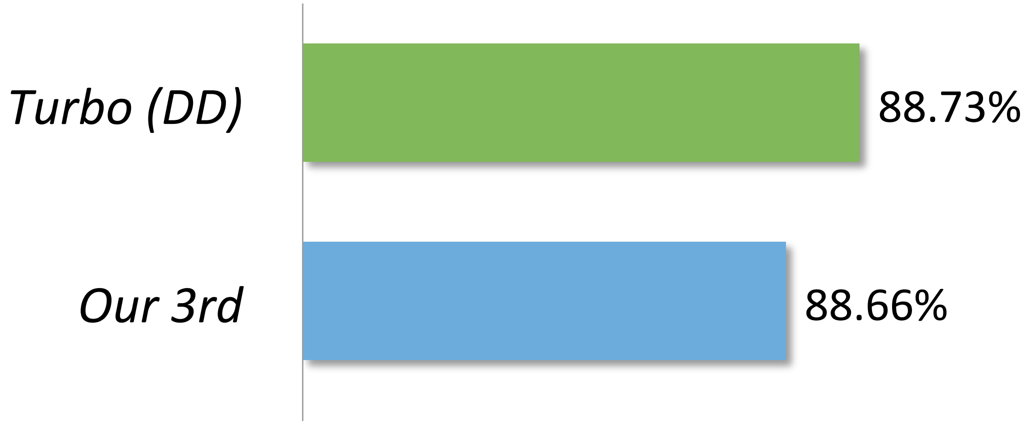
Baselines:

- Turbo Parser: Dual Decomposition with 3rd-order features (Martins et al., 2013)
- Sampling-based Parser: MCMC sampling with global features (Zhang et al., 2014)

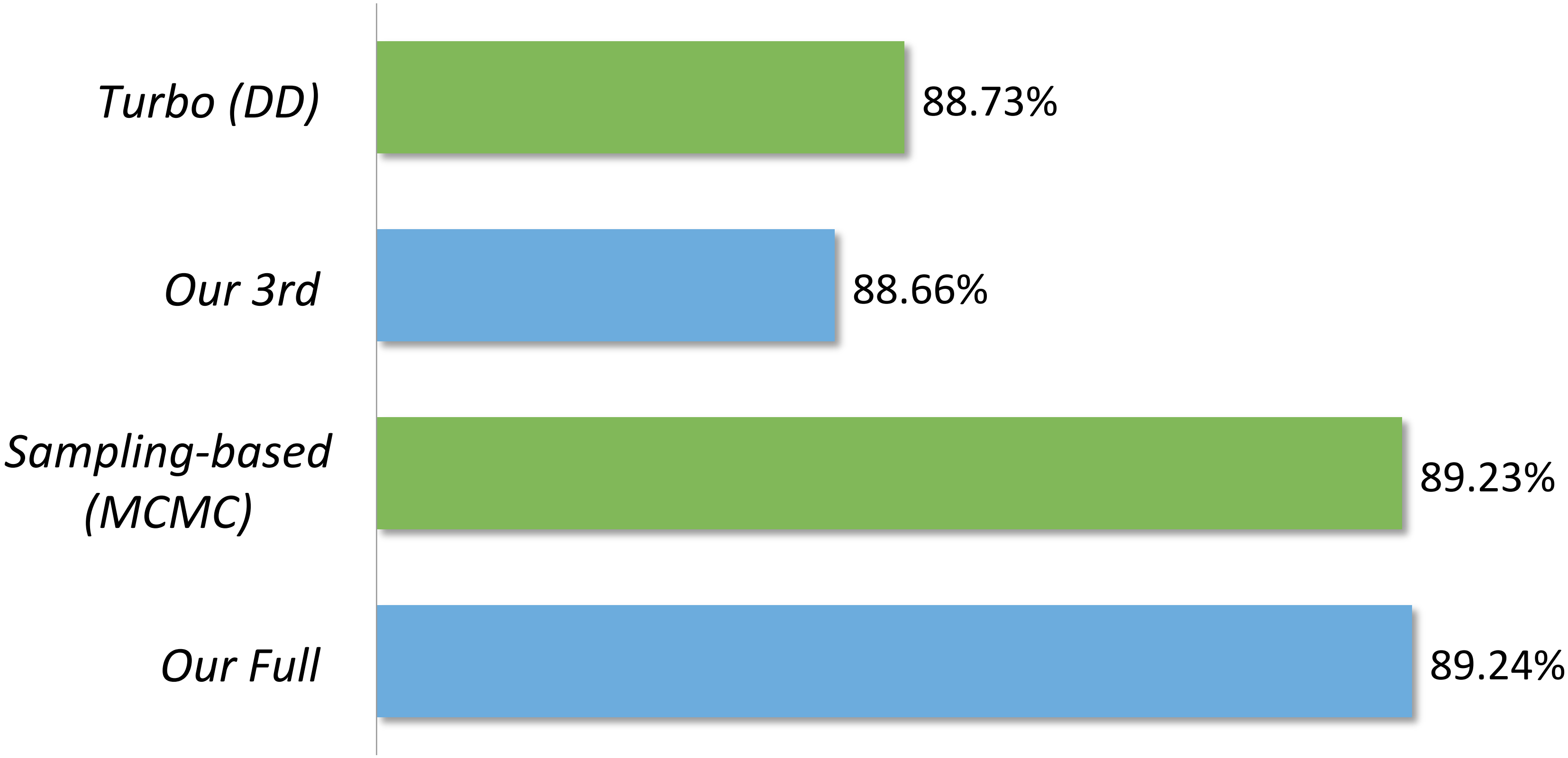
Evaluation Measure:

- Unlabeled Attachment Score (UAS), without punctuations

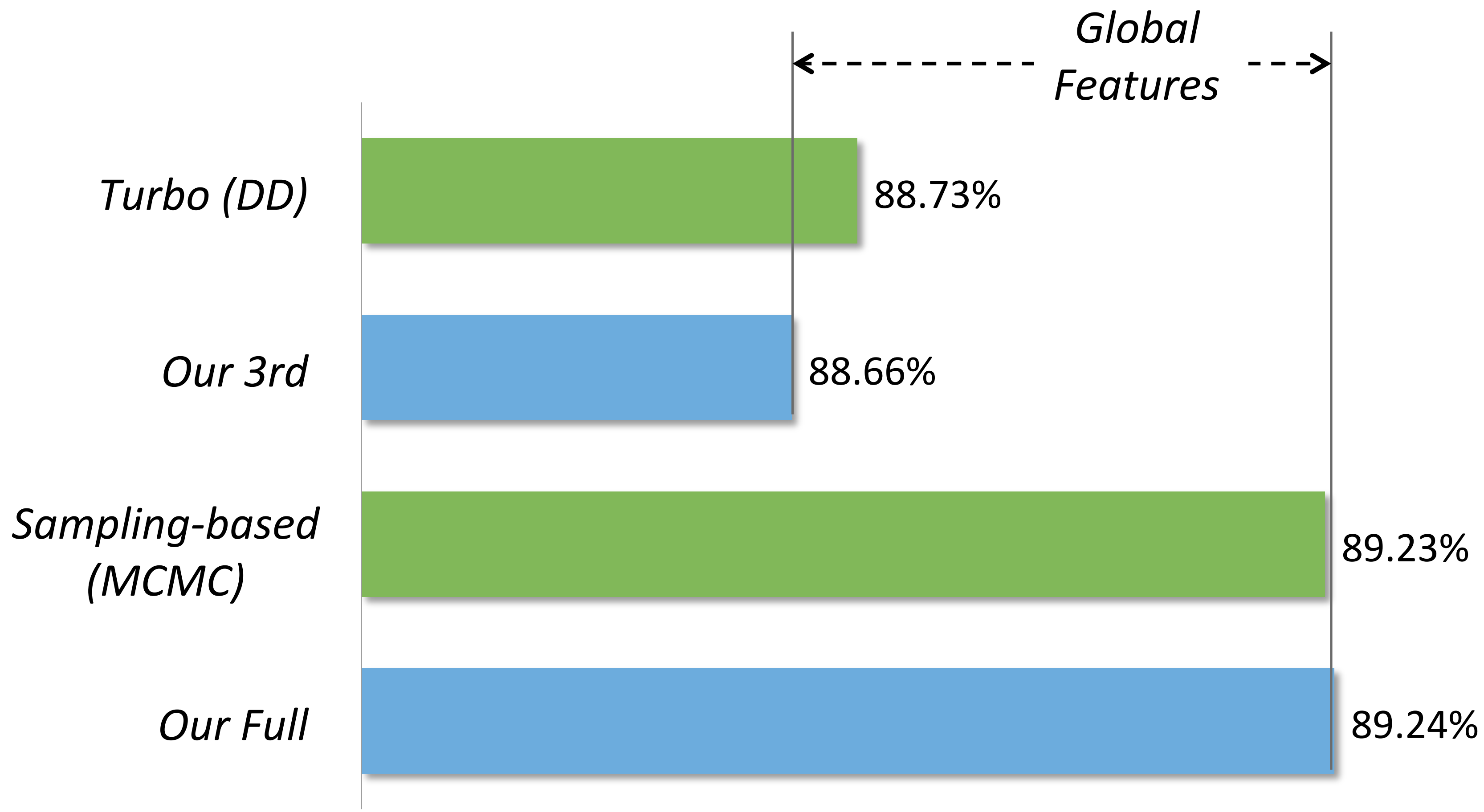
Comparing with Baselines



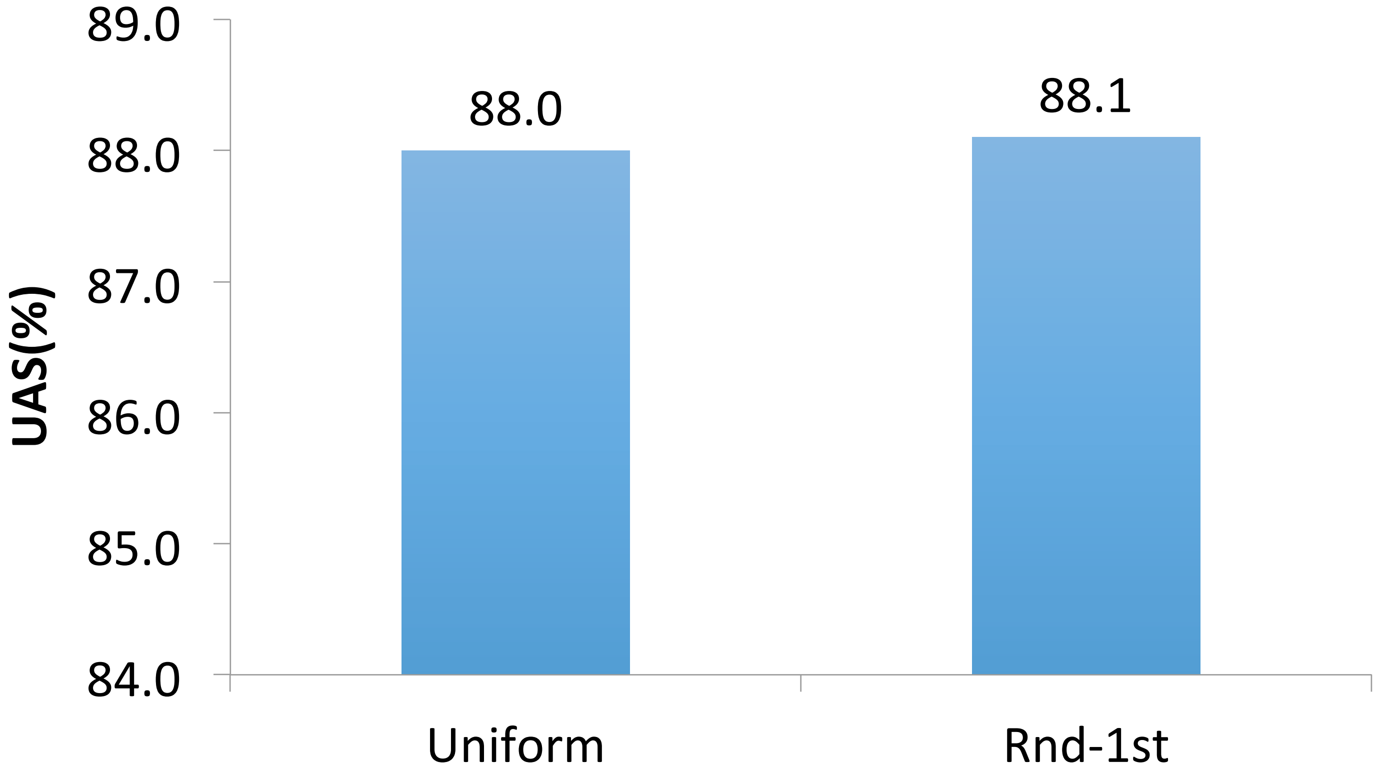
Comparing with Baselines



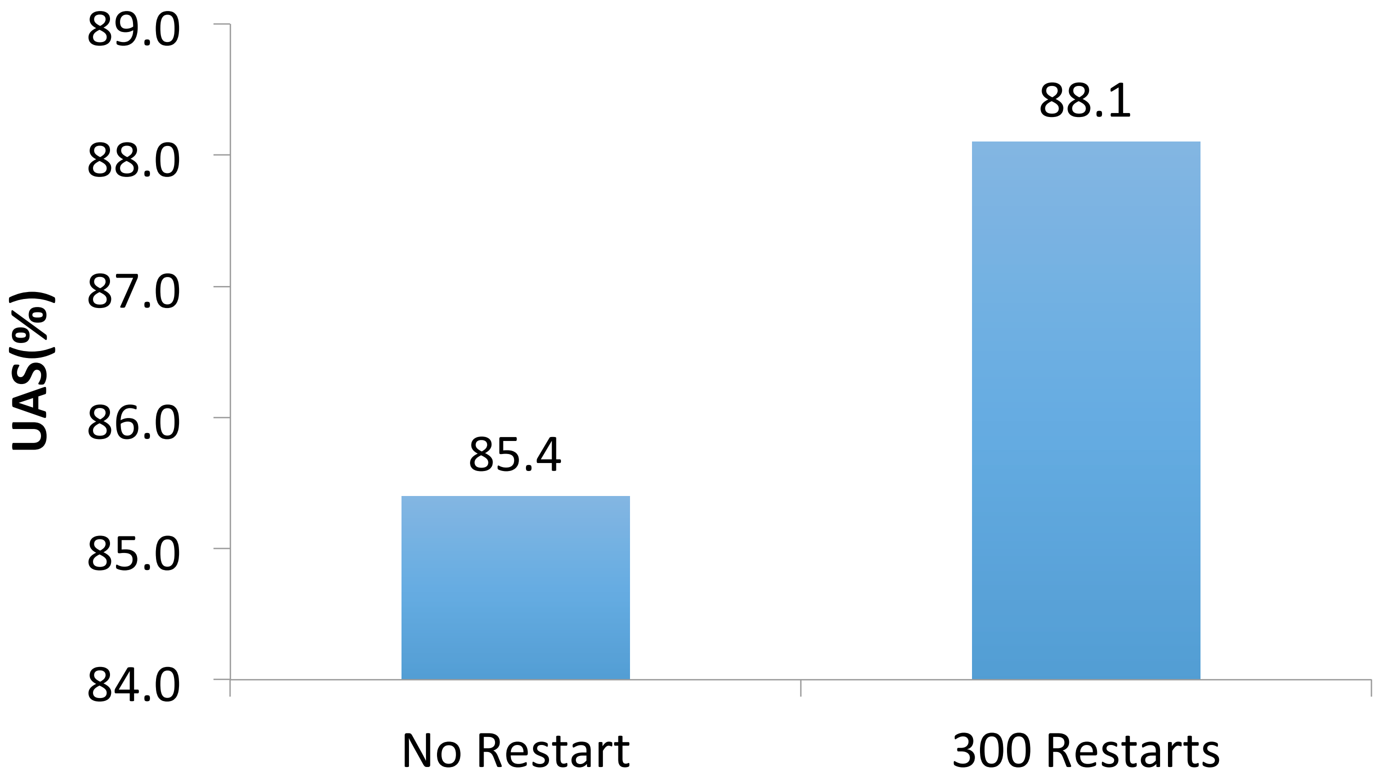
Comparing with Baselines



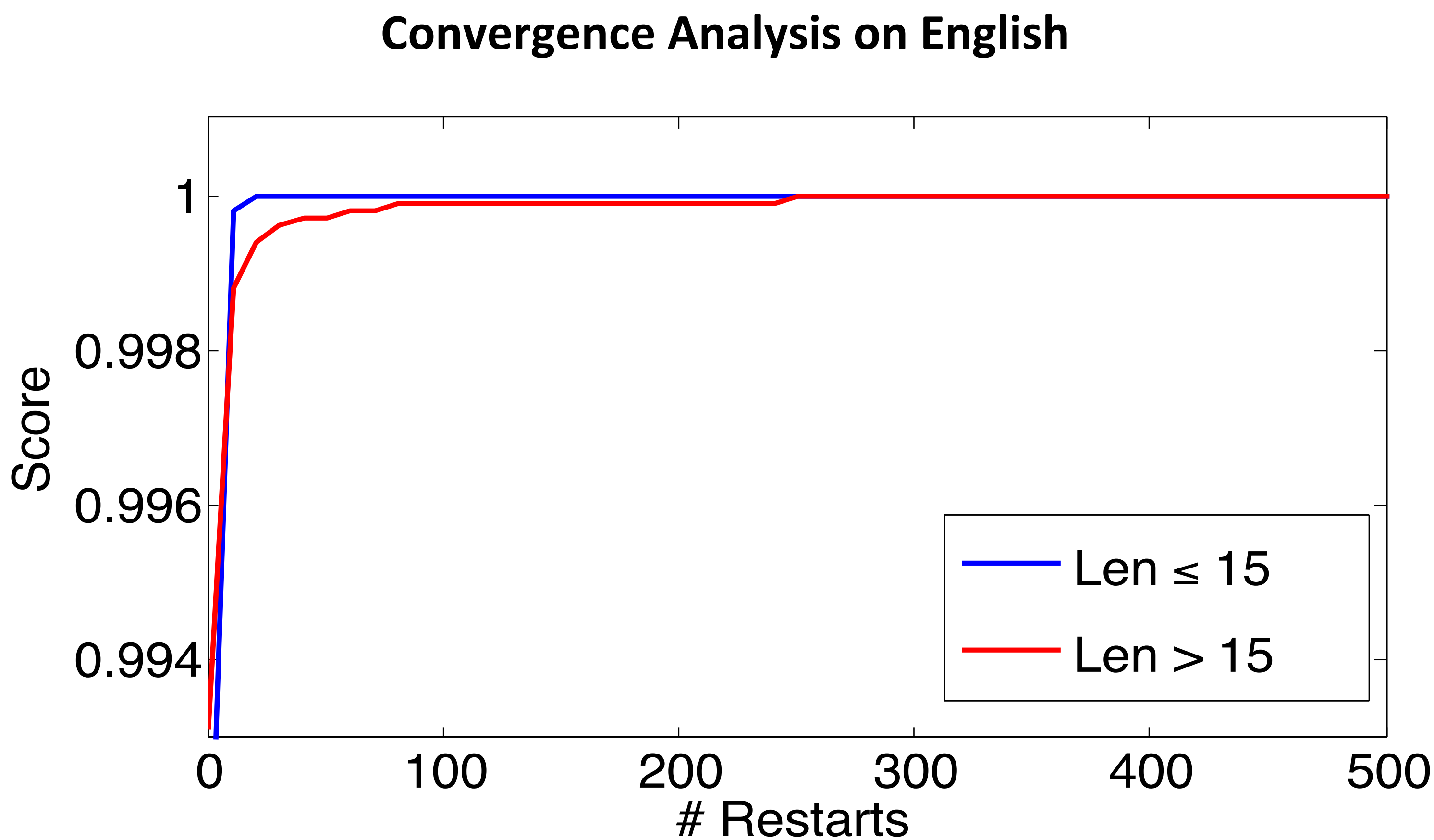
Impact of Initialization



Impact of Restarts

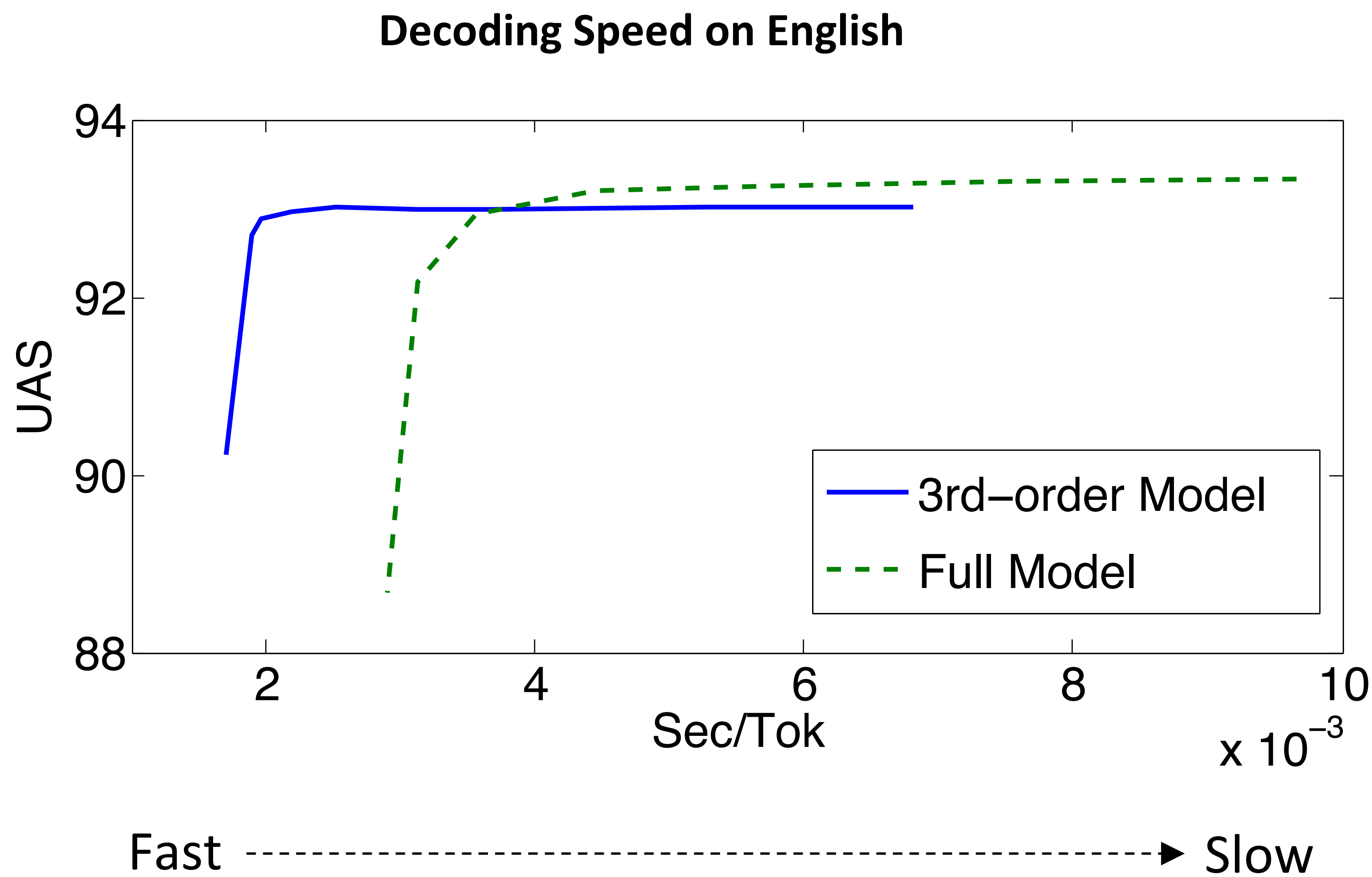


Convergence Property



- Score normalized by the highest score in 3000 restarts

Trade-off between Speed and Performance



Conclusion

- *Analysis:* we investigate average case complexity of parsing
- *Algorithm:* we introduce a simple randomized greedy inference algorithm

Source code available at:

https://github.com/taolei87/RBGParser

Thank You!