Dynamic Feature Selection for Dependency Parsing

He He, Hal Daumé III and Jason Eisner

EMNLP 2013, Seattle

UNIVERSITY OF MARYLAND

JOHNS HOPKINS UNIVERSITY

Structured Prediction in NLP

Part-of-Speech Tagging

N ⟶ N ⟶ V ⟶ Det ⟶ N

↓ ↓ ↓ ↓ ↓

Fruit flies like a banana

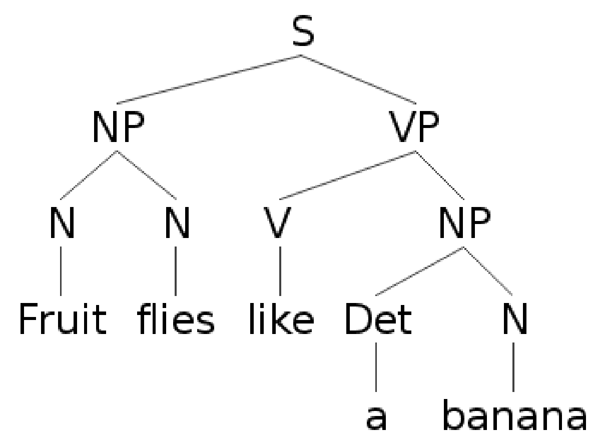
Machine Translation

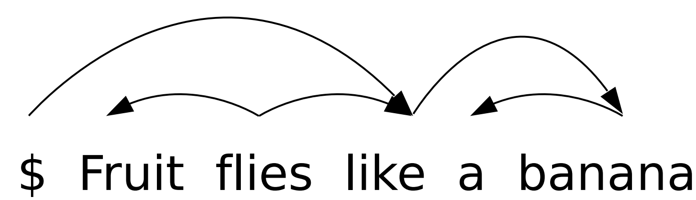
Fruit flies like a banana .

果 蝇 喜欢 香蕉 。

summarization, name entity resolution and many more ...

Parsing





Structured Prediction in NLP

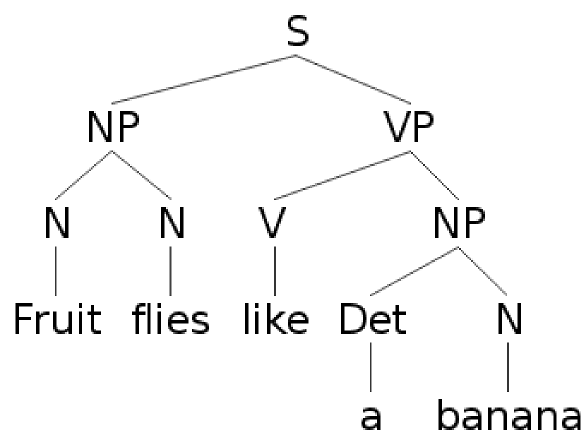
Part-of-Speech Tagging

N ⟶ N ⟶ V ⟶ Det ⟶ N

↓ ↓ ↓ ↓ ↓

Fruit flies like a banana

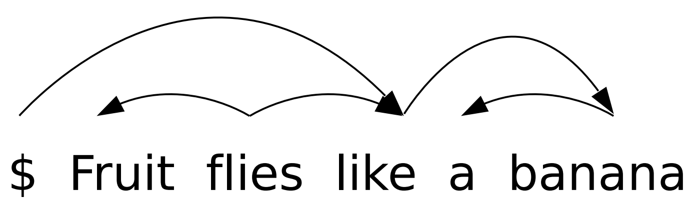
Parsing



Machine Translation

Fruit flies like a banana .

果 蝇 喜欢 香蕉 。

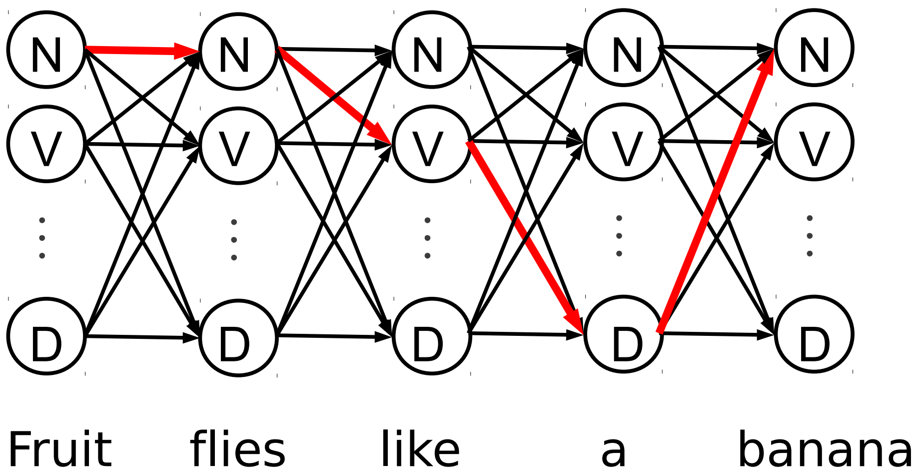


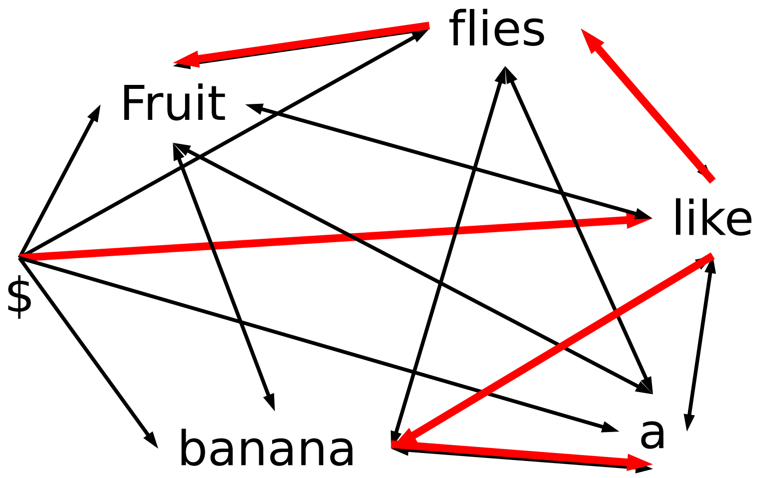
summarization, name entity resolution and many more ...

Exponentially increasing search space

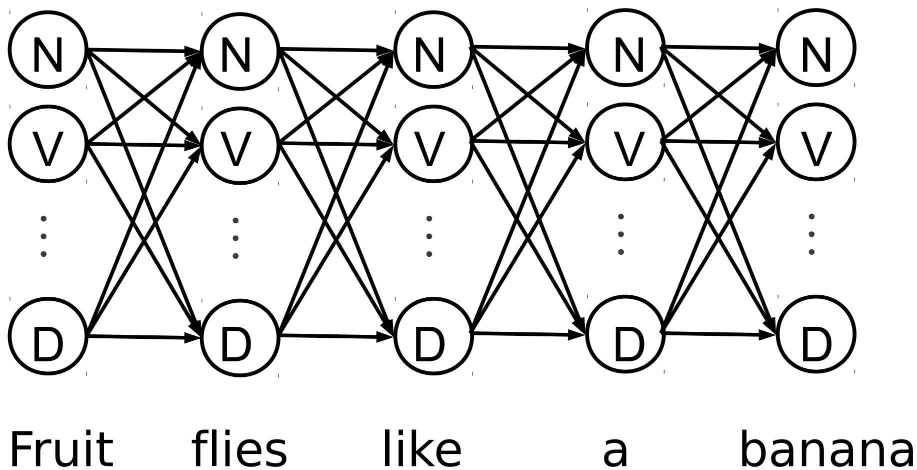
Millions of features for scoring

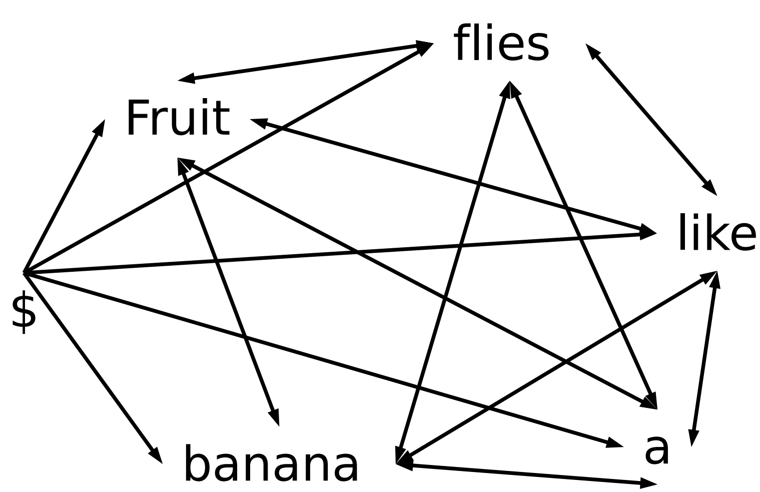
Structured Prediction in NLP





Structured Prediction in NLP

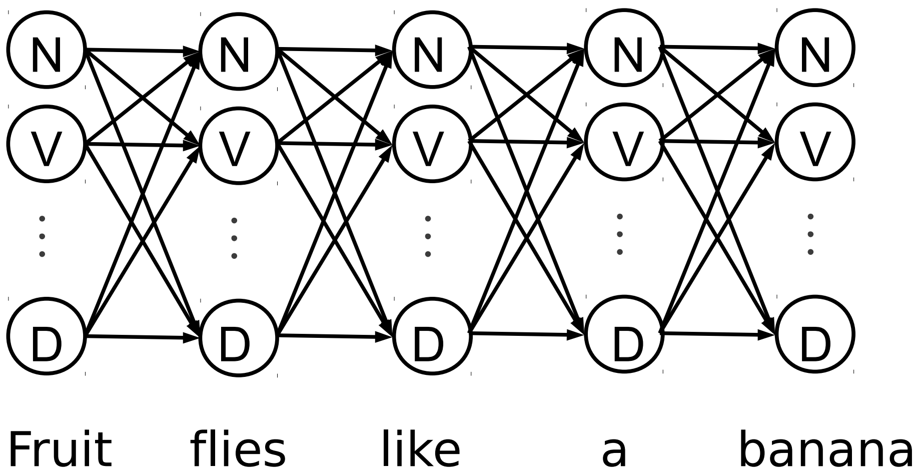


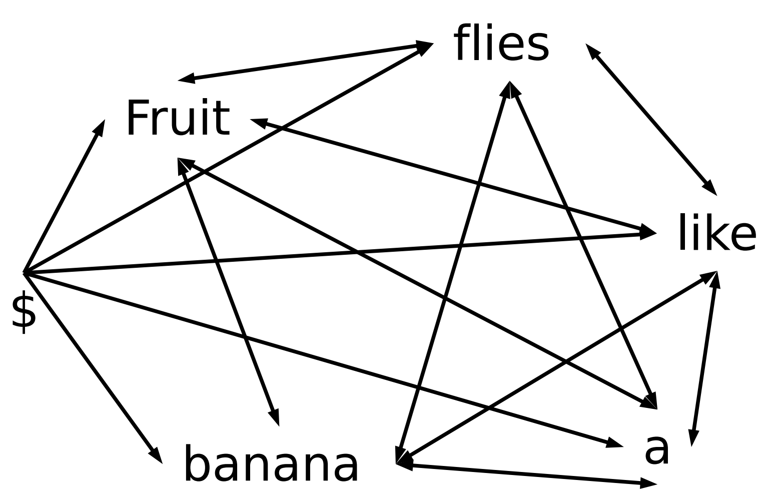


Feature templates per edge

token
left token
right token
in-between token
stem
form
bigram
tag
coarse tag
length
direction
...

Structured Prediction in NLP





Feature templates per edge

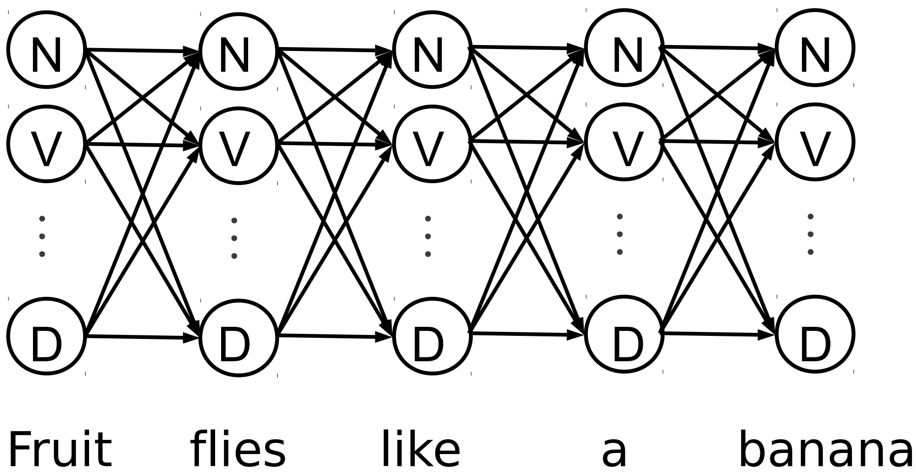
- token
- left token
- right token
- in-between token
- stem
- form
- bigram
- tag
- coarse tag
- length
- direction
- …

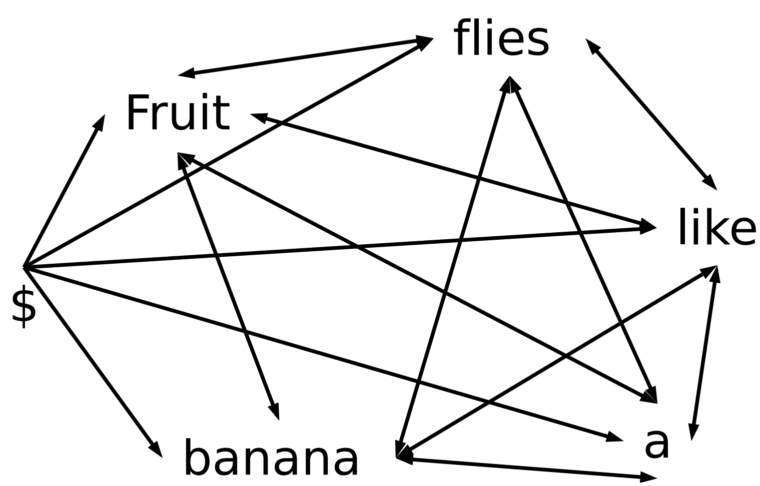
(head_token + mod_token)

X

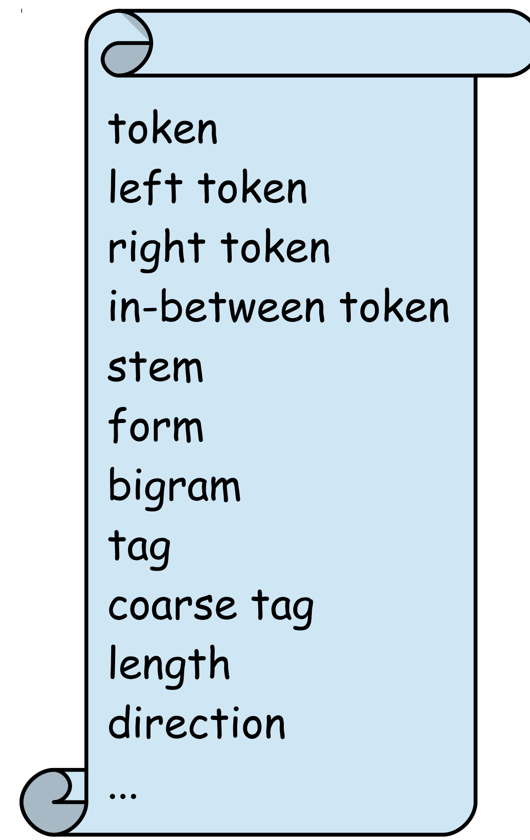
(head_tag +mod_tag)

Structured Prediction in NLP





Feature templates per edge



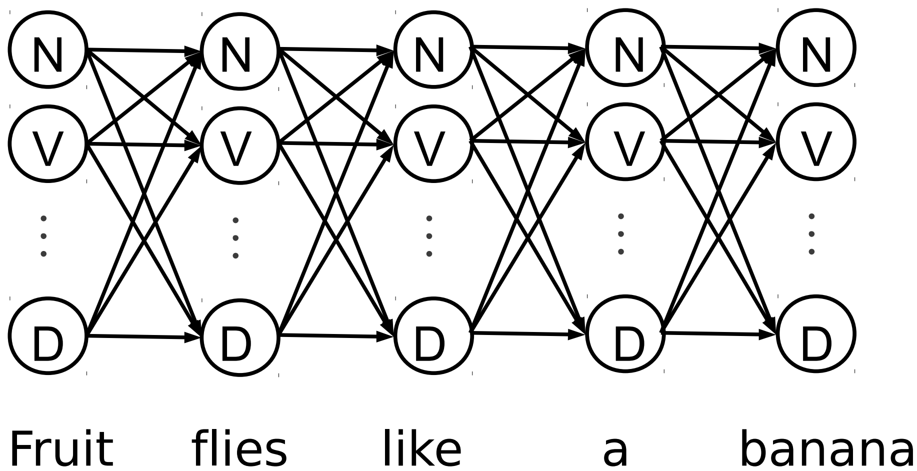
HUGE

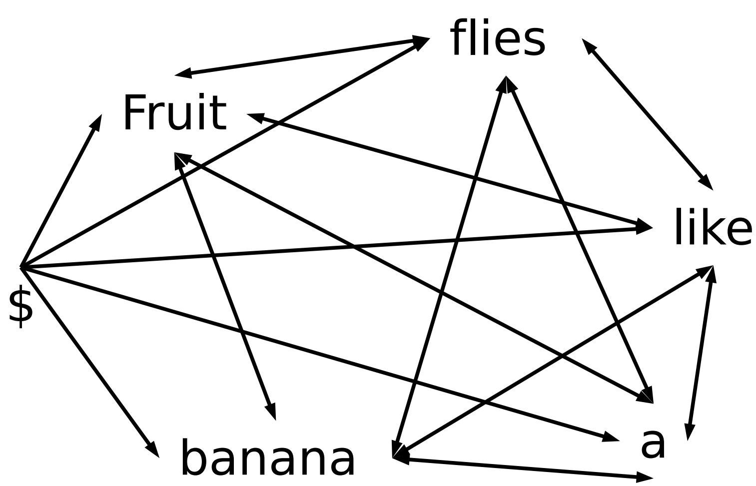
(head_token + mod_token)

X

(head_tag +mod_tag)

Structured Prediction in NLP





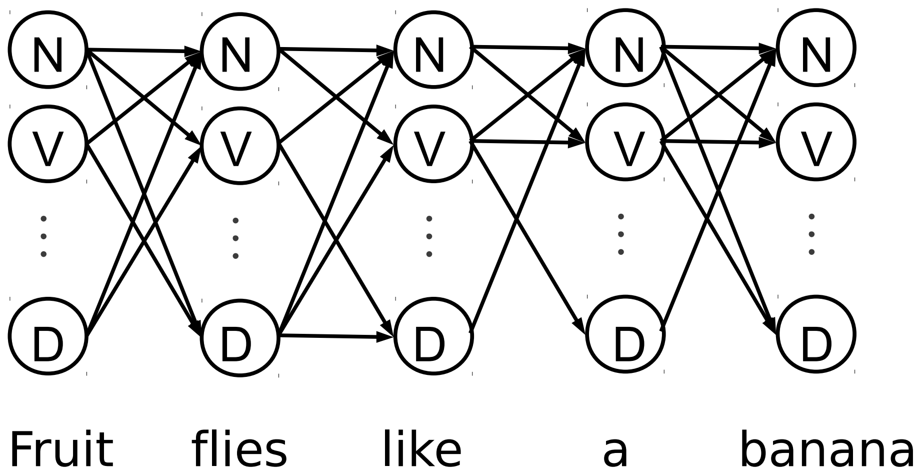
Feature templates per edge

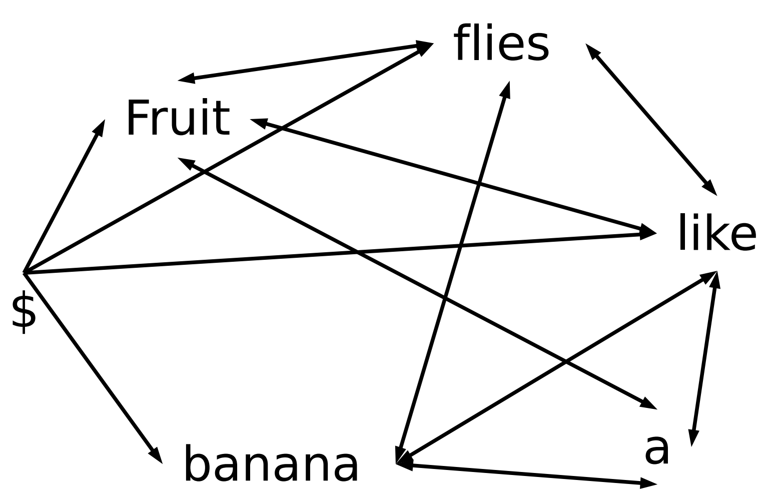
token

tag
coarse tag
length
direction
…

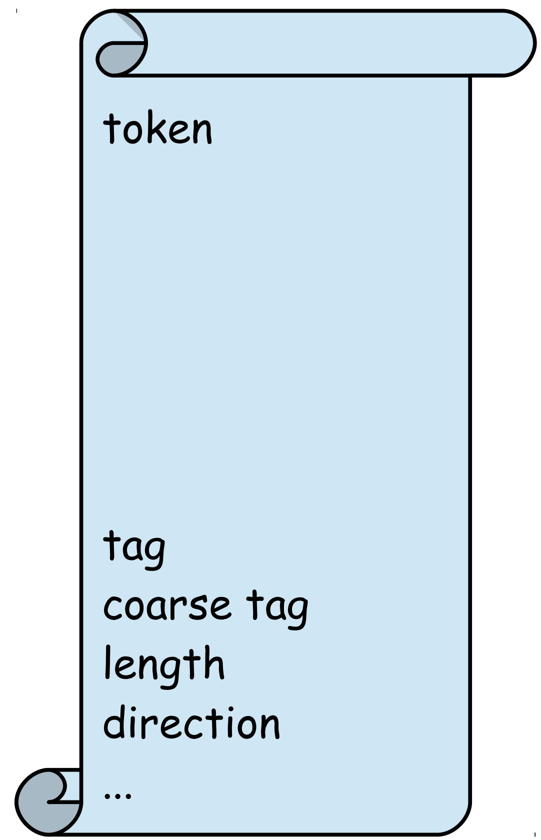
Do you need all features everywhere ?

Structured Prediction in NLP



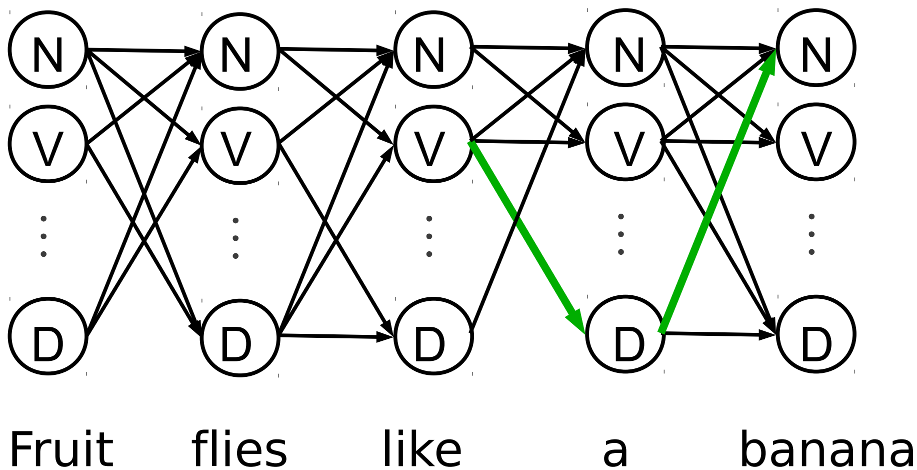


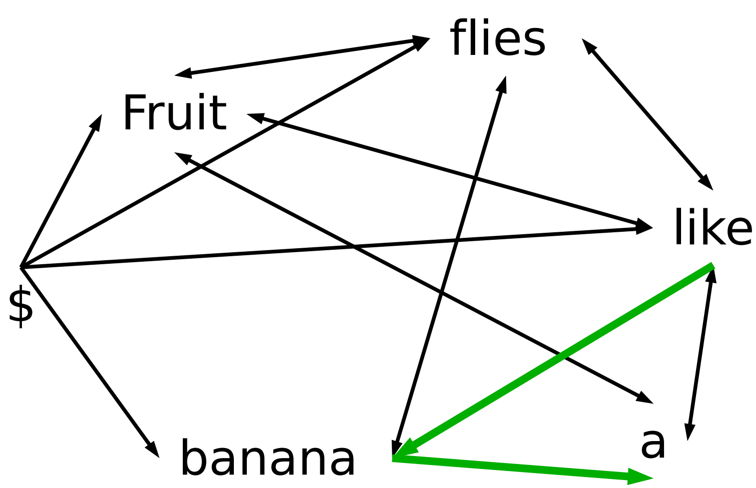
Feature templates per edge



Do you need all features everywhere ?

Structured Prediction in NLP

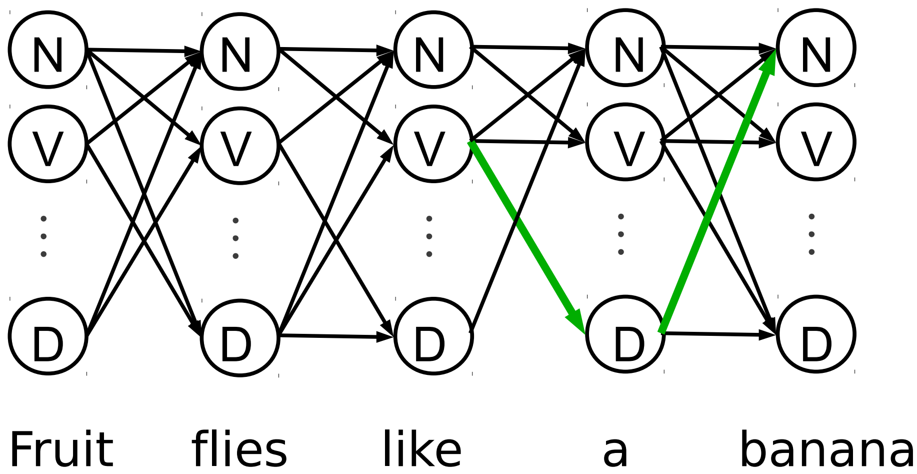




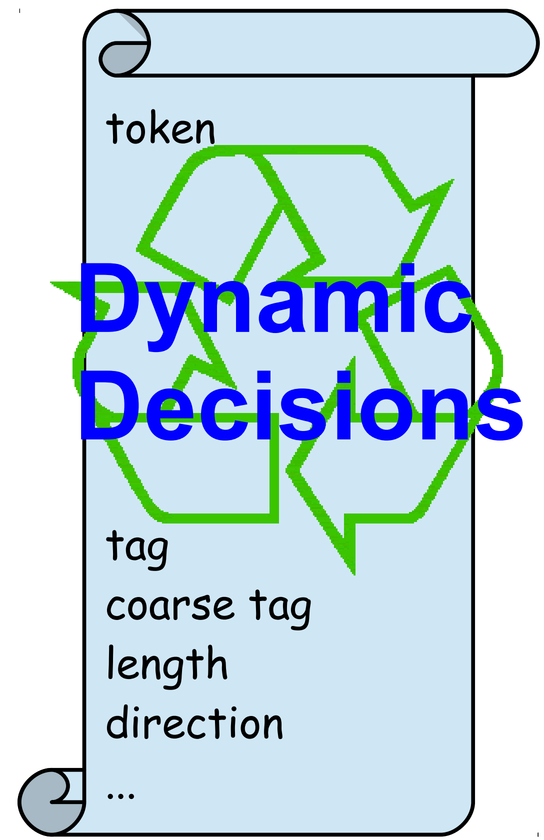
Feature templates per edge

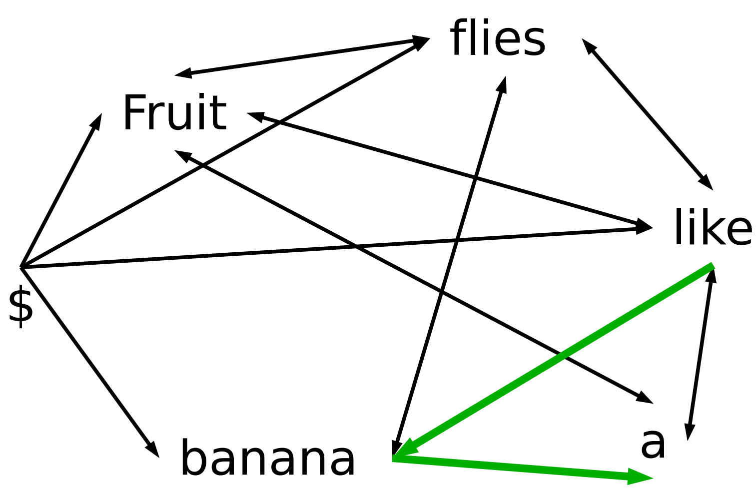
Do you need all features everywhere ?

Structured Prediction in NLP

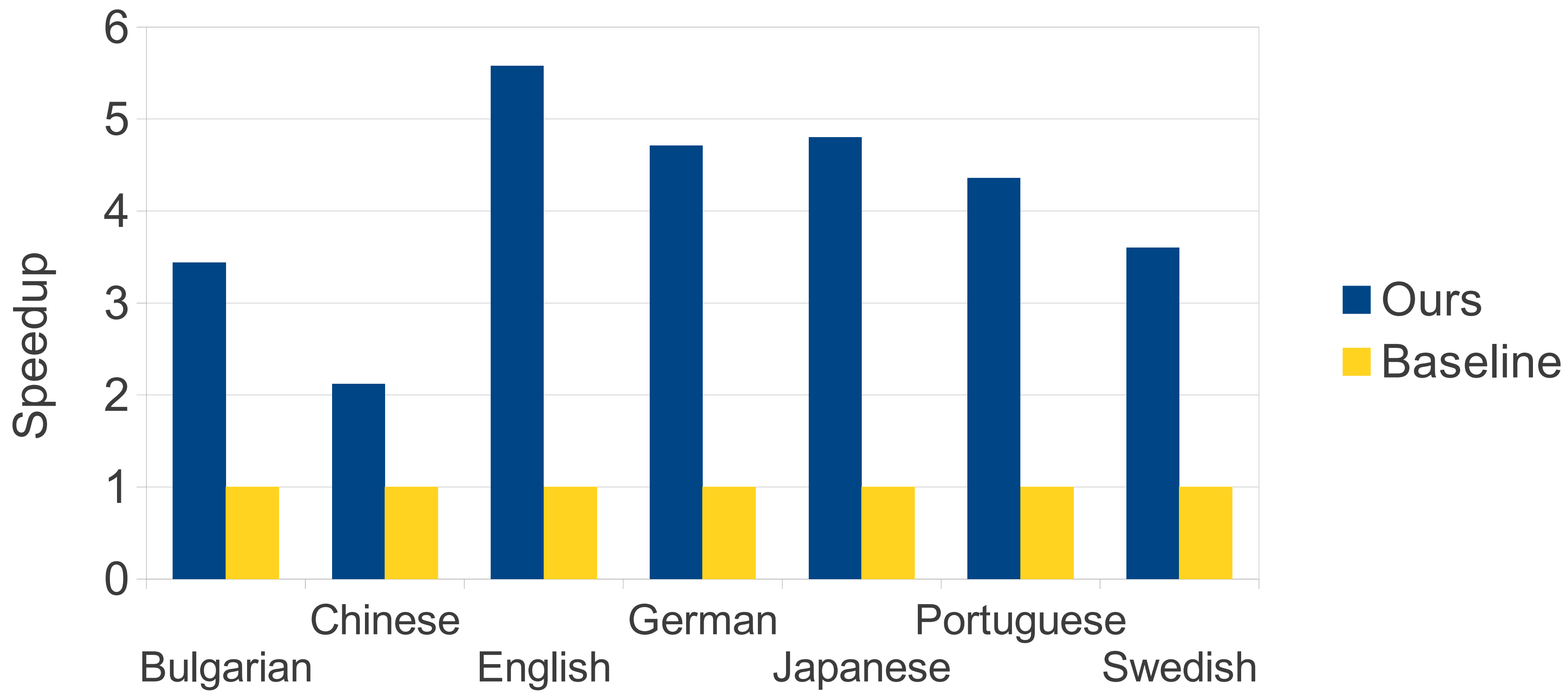


Feature templates per edge



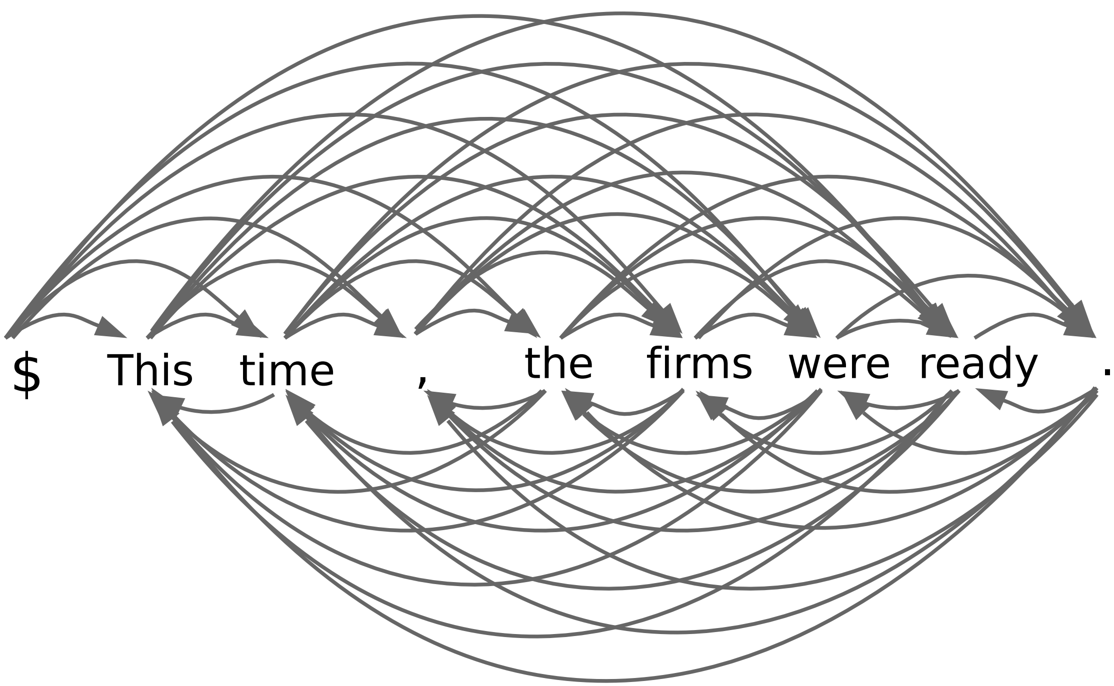


Case Study: Dependency Parsing



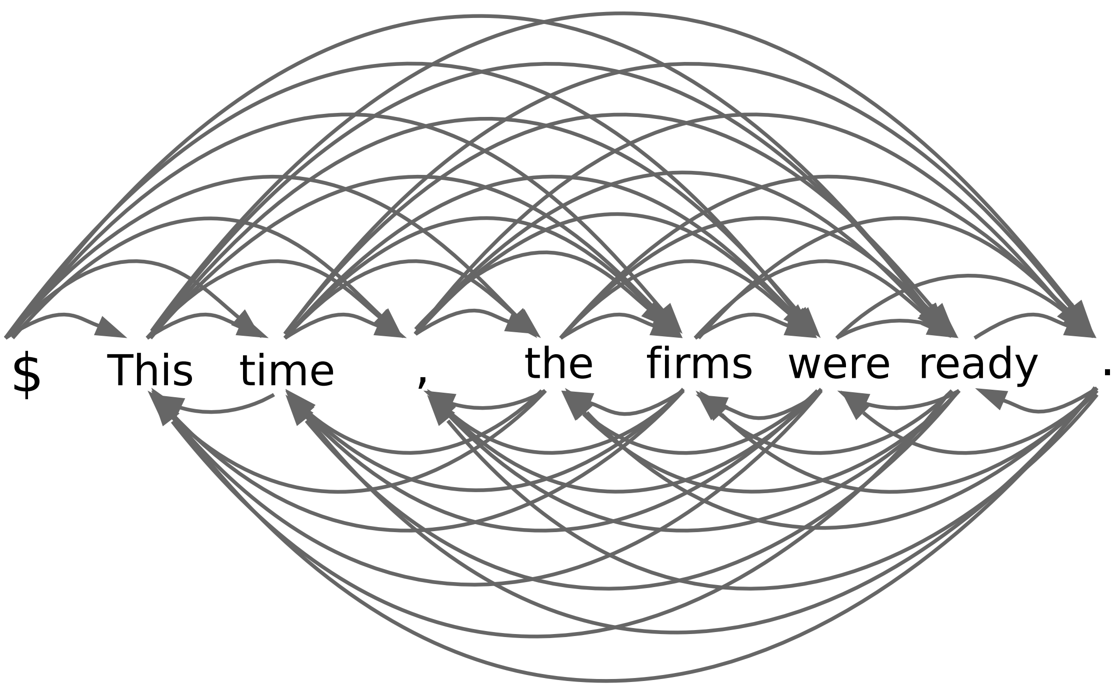
2x to 6x speedup with little loss in accuracy

Graph-based Dependency Parsing



Scoring: $\phi(E) \cdot w$

Graph-based Dependency Parsing

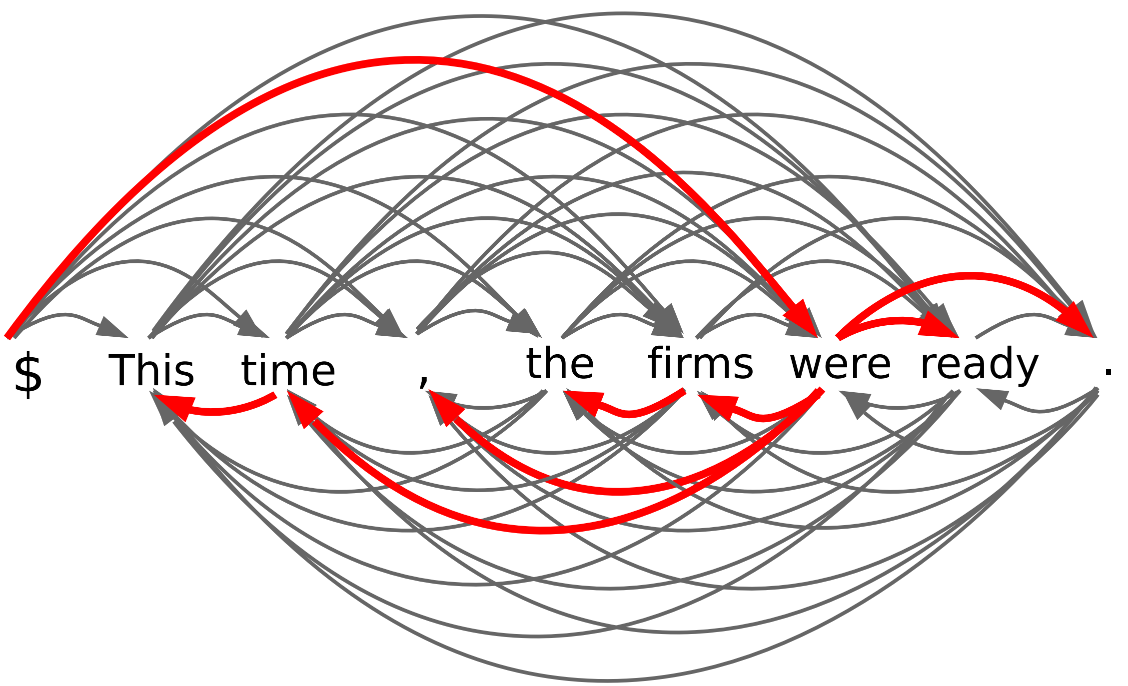


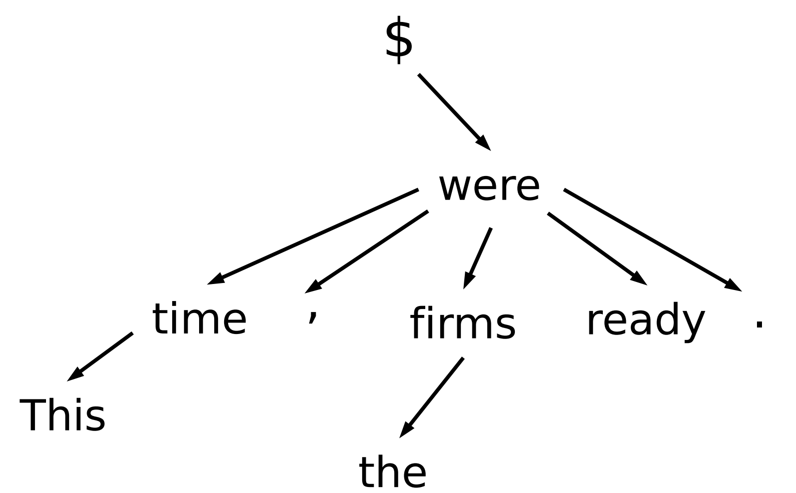
length: 1
direction: right
modifier_token: were
head_token: firms
head_tag: noun
⋮

And hundreds more!

Scoring: $\phi(E) \cdot w$

Graph-based Dependency Parsing

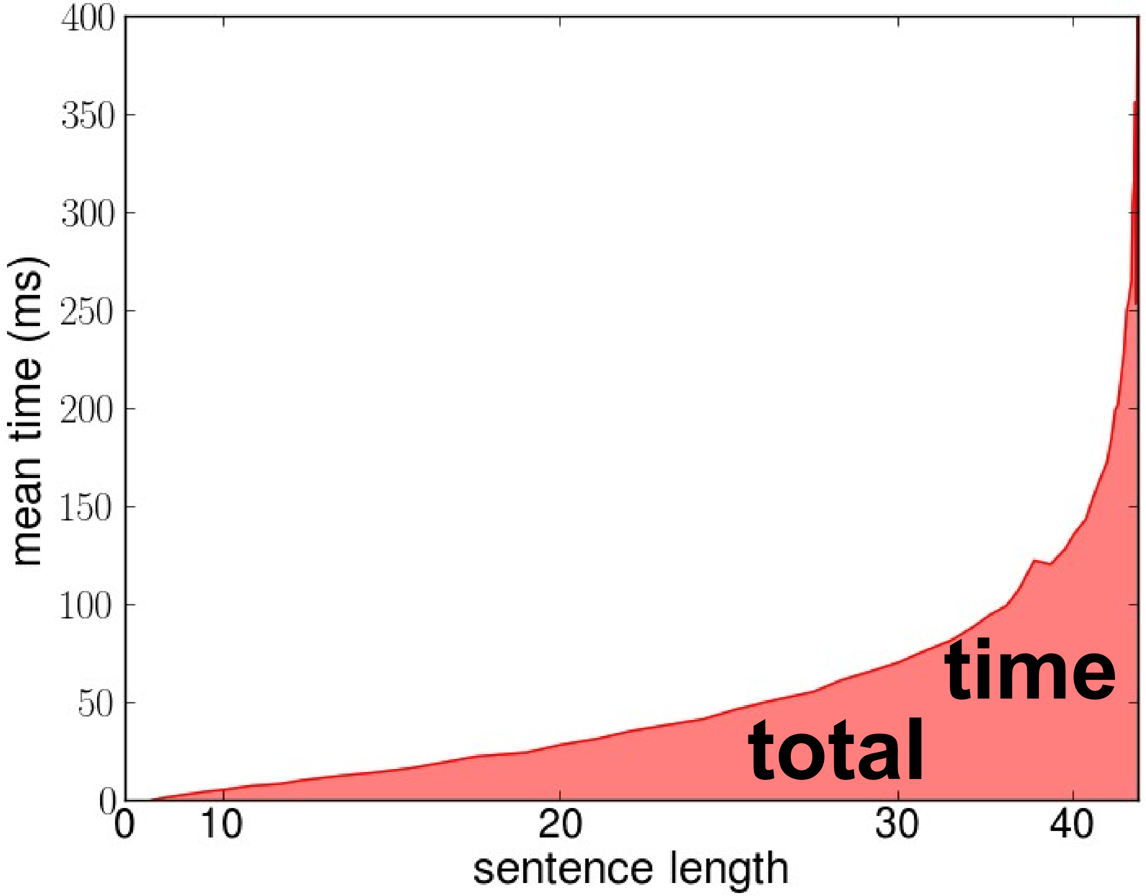




Decoding: find the highest-scoring tree

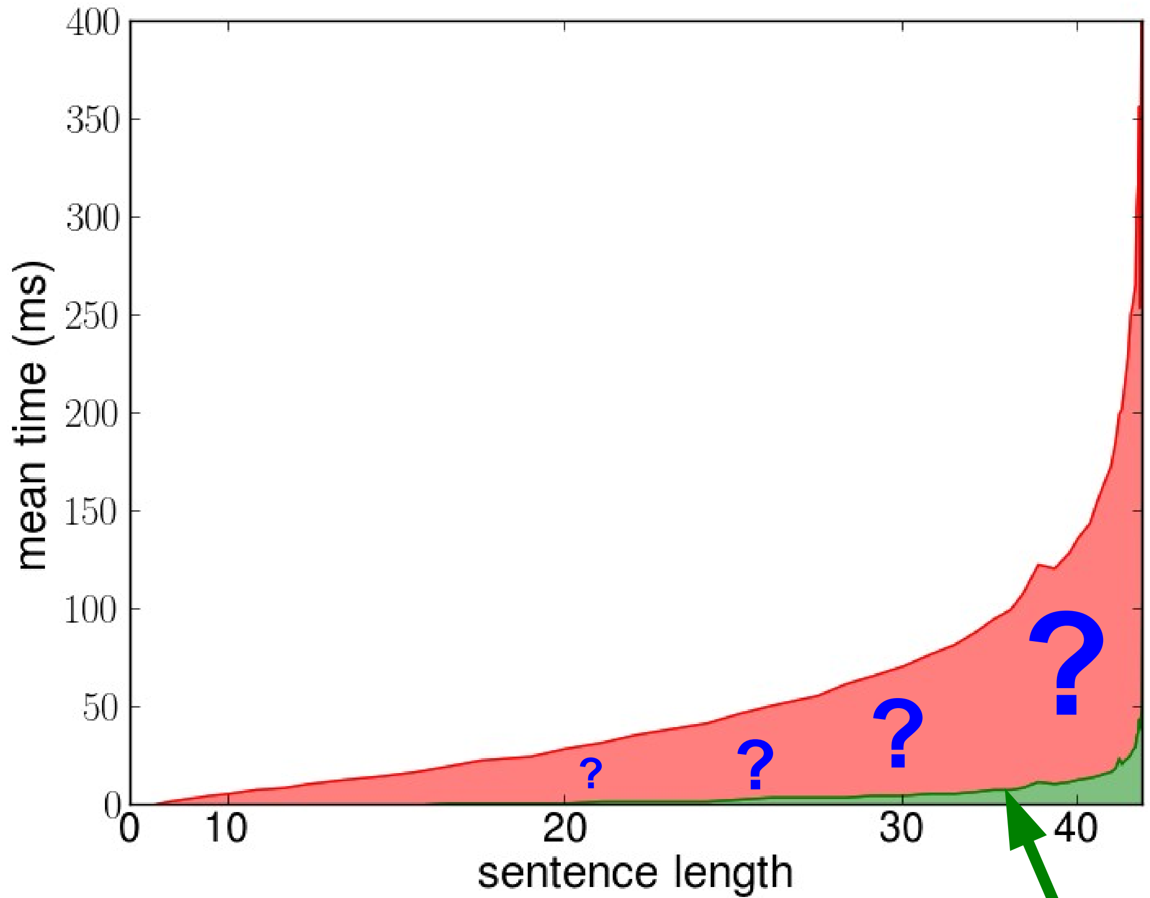
MST Dependency Parsing

(1st-order projective)



MST Dependency Parsing

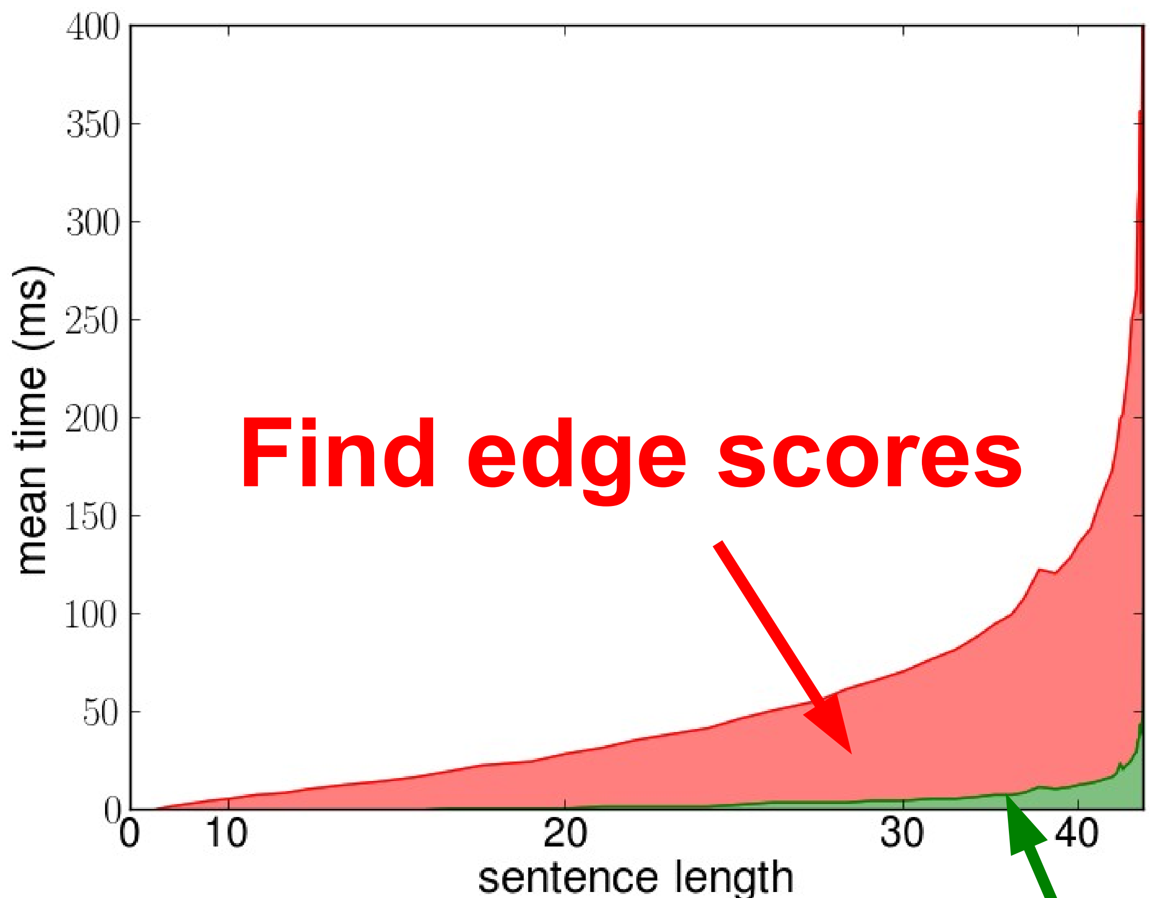
(1st-order projective)



Find highest-scoring tree $O(n^3)$

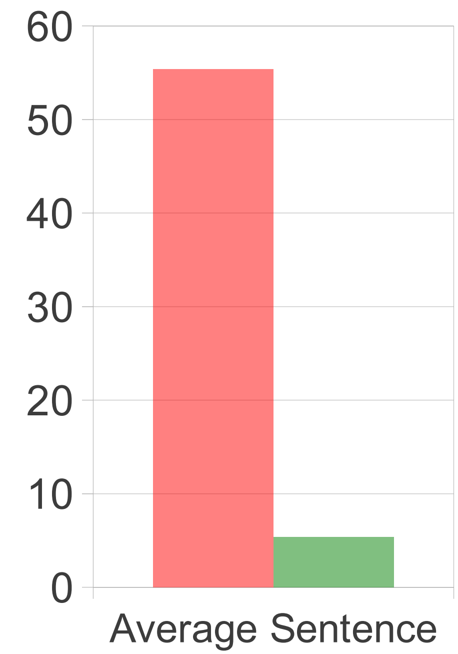
MST Dependency Parsing

(1st-order projective)



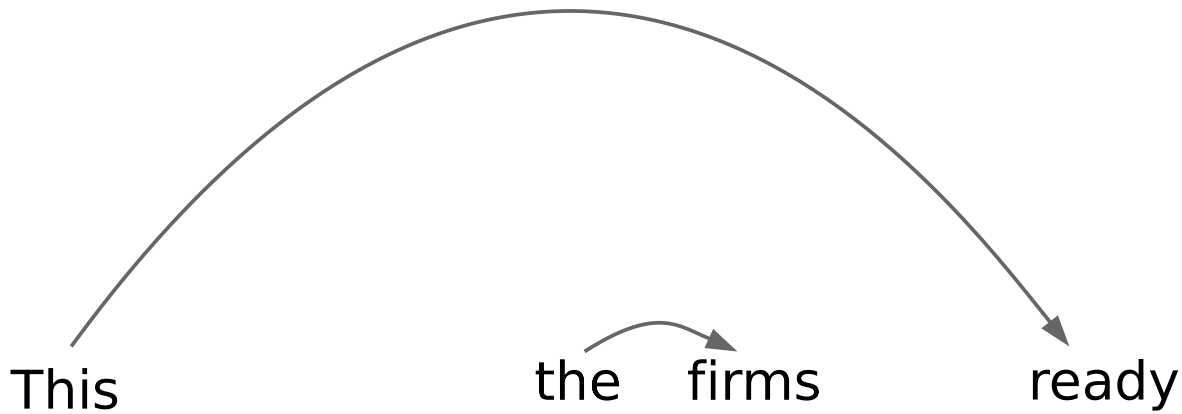
~268 feature templates

~76M features



Find highest-scoring tree $O(n^3)$

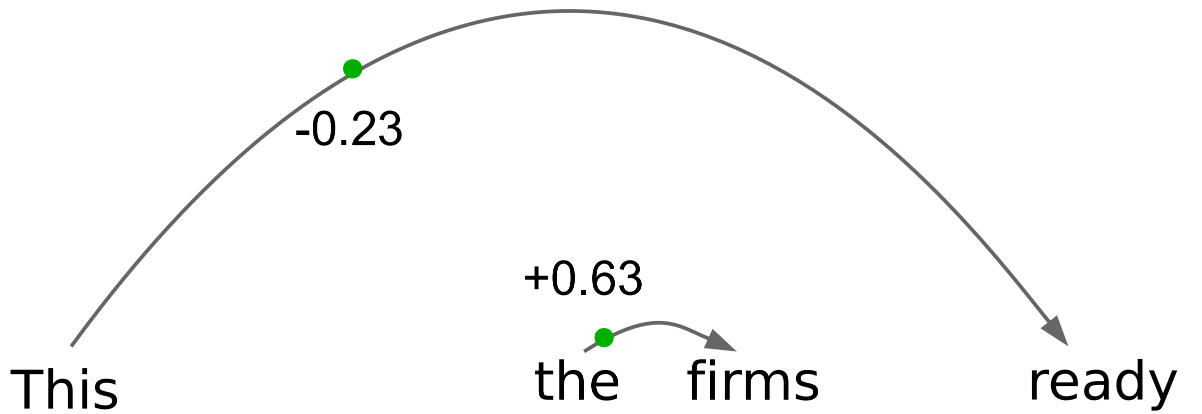
Add features only when necessary!



score(This → ready) =

score(the → firms) =

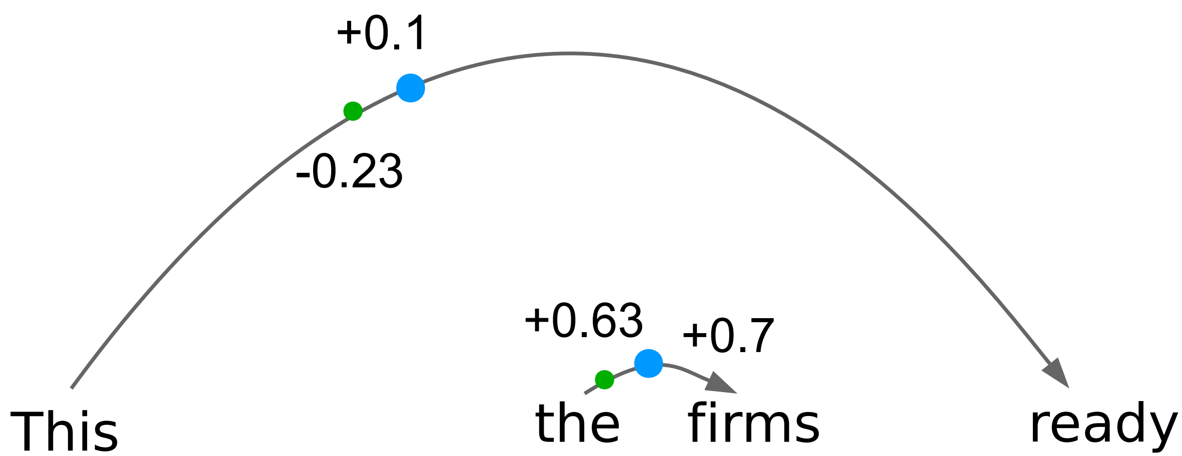
Add features only when necessary!



score(This → ready) = -0.23

score(the → firms) = 0.63

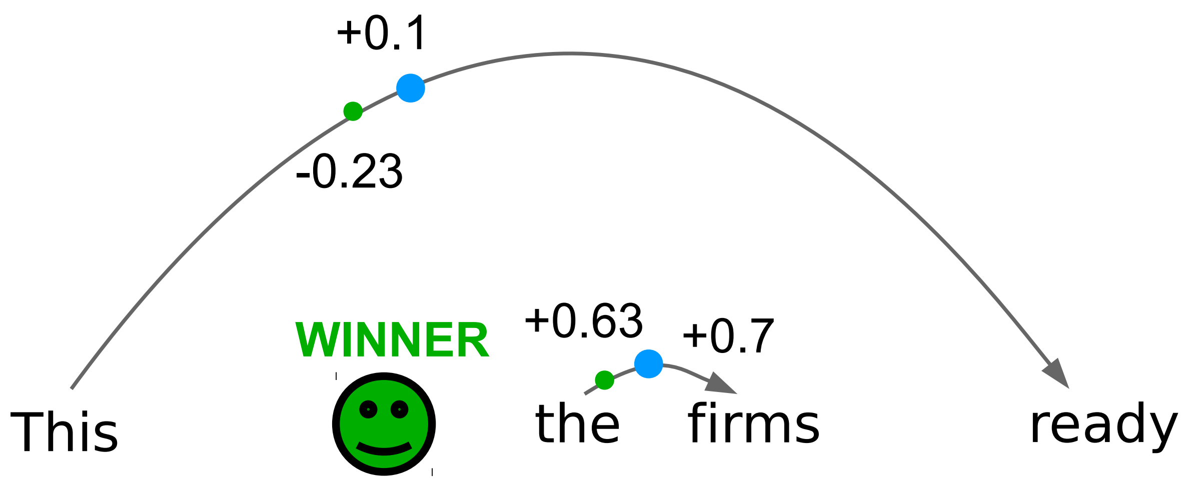
Add features only when necessary!



score(This → ready) = -0.13

score(the → firms) = 1.33

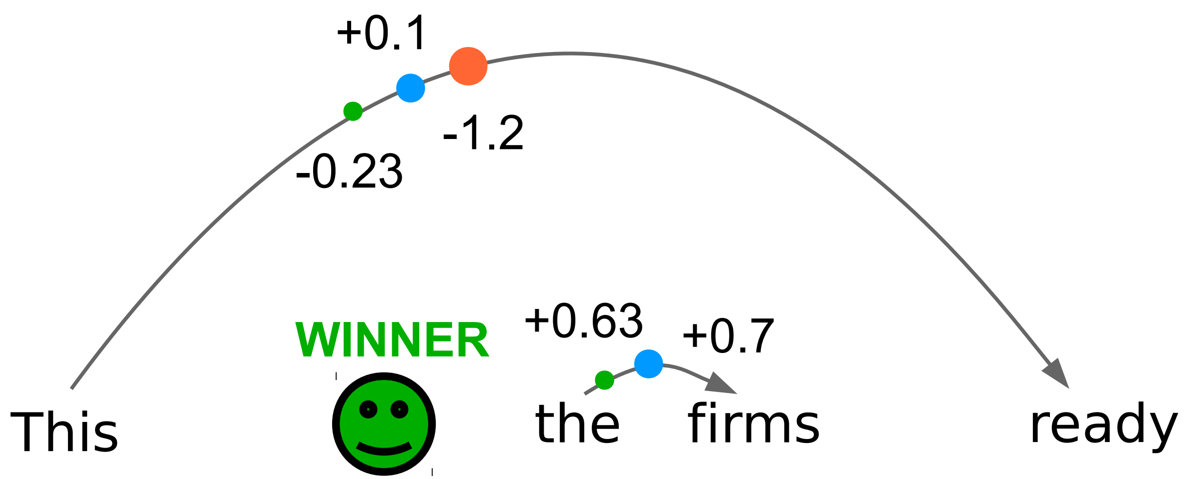
Add features only when necessary!



score(This → ready) = -0.13

score(the → firms) = 1.33

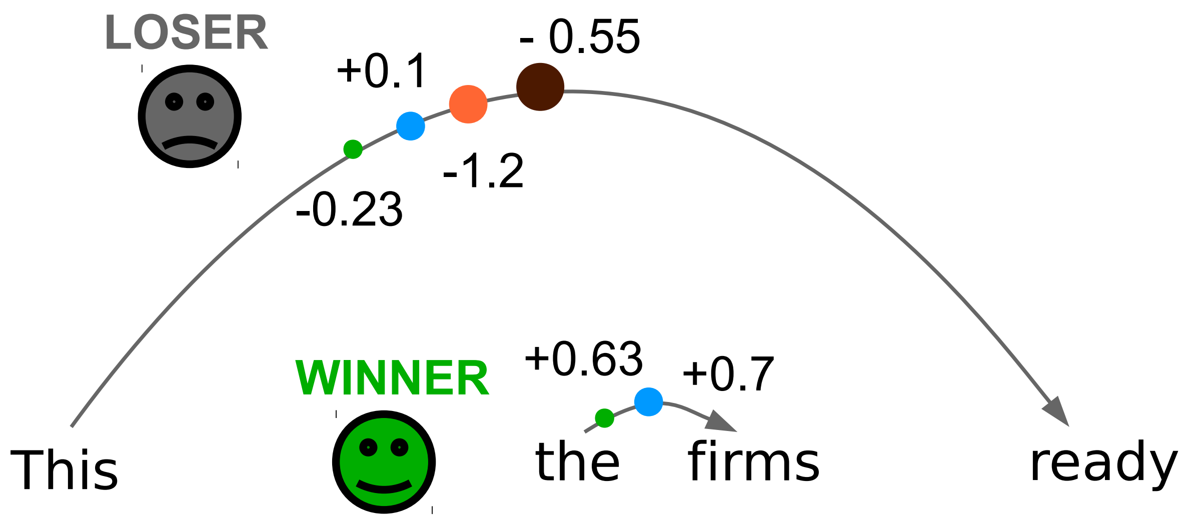
Add features only when necessary!



score(This → ready) = -1.33

score(the → firms) = 1.33

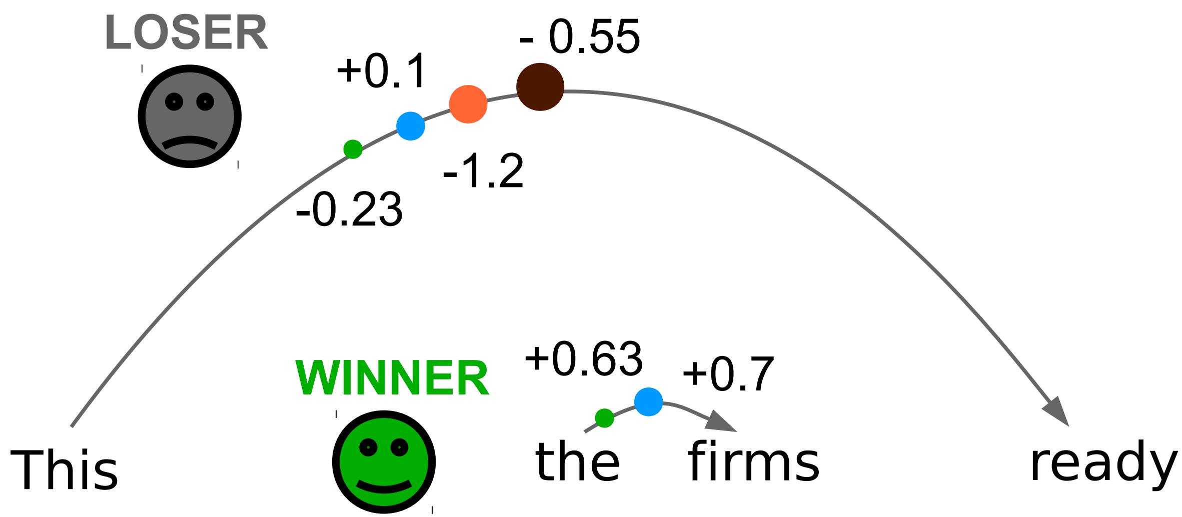
Add features only when necessary!



score(This → ready) = -1.88

score(the → firms) = 1.33

Add features only when necessary!



score(This → ready) = -1.88

score(the → firms) = 1.33

This is a **structured** problem!
Should not look at scores independently.

Dynamic Dependency Parsing

1. Find the highest-scoring tree after adding some features *fast non-projective decoding*

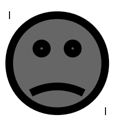
Dynamic Dependency Parsing

1. Find the highest-scoring tree after adding some features *fast non-projective decoding*
2. Only edges in the current best tree can win

Dynamic Dependency Parsing

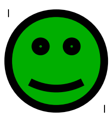
1. Find the highest-scoring tree after adding some features *fast non-projective decoding*
2. Only edges in the current best tree can win

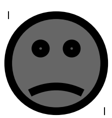
 are chosen by a classifier ≤ n *decisions*

 are killed because they fight with the winners

Dynamic Dependency Parsing

1. Find the highest-scoring tree after adding some features *fast non-projective decoding*
2. Only edges in the current best tree can win

 are chosen by a classifier ≤ n *decisions*

 are killed because they fight with the winners

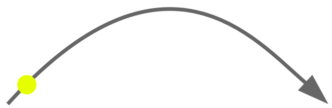
3. Add features to undetermined edges *by group*

Dynamic Dependency Parsing

1. Find the highest-scoring tree after adding some features *fast non-projective decoding*
2. Only edges in the current best tree can win
   - are chosen by a classifier ≤ n *decisions*
   - are killed because they fight with the winners
3. Add features to undetermined edges *by group*

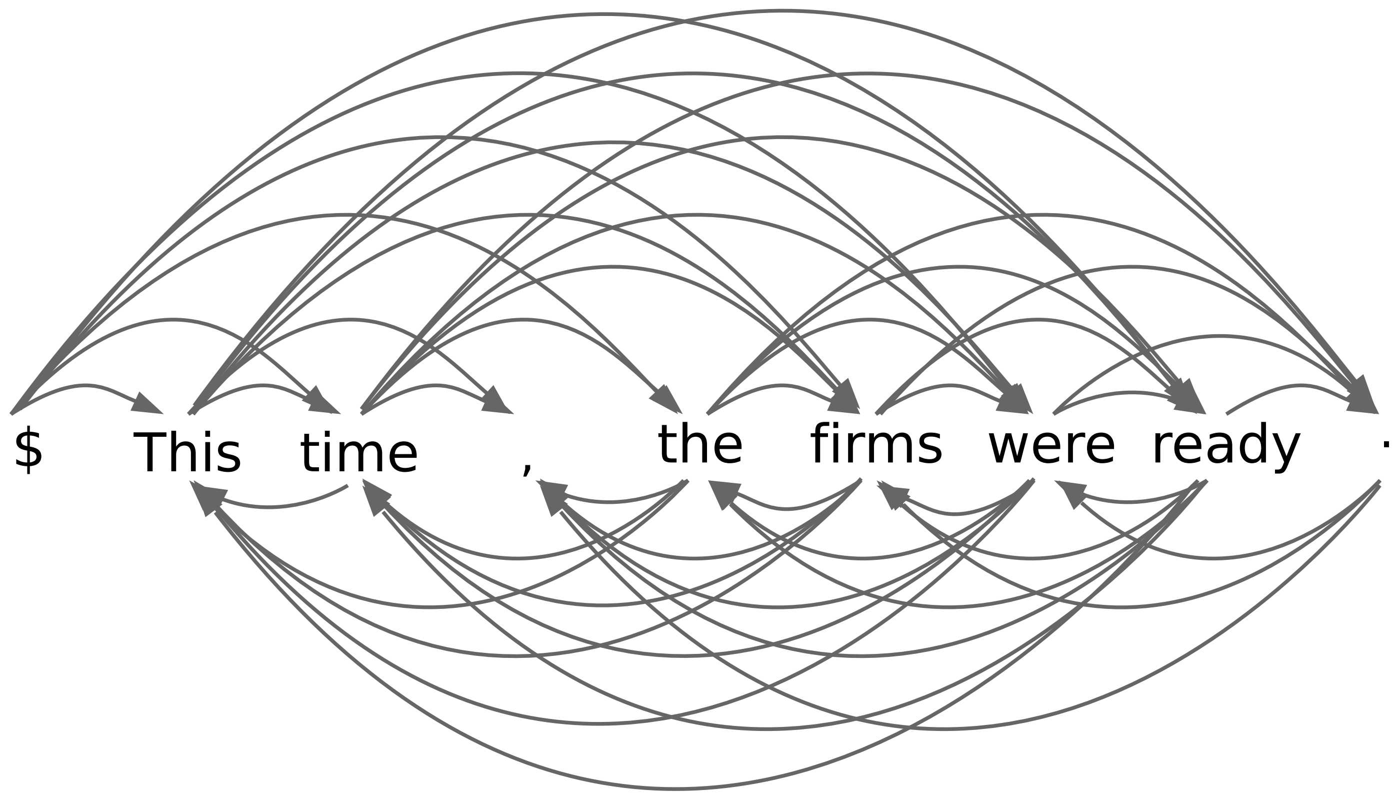
Max # of iterations = # of feature groups

+ first feature group



51 gray edges with unknown fate...

5 features per gray edge

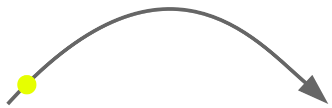


Undetermined edge

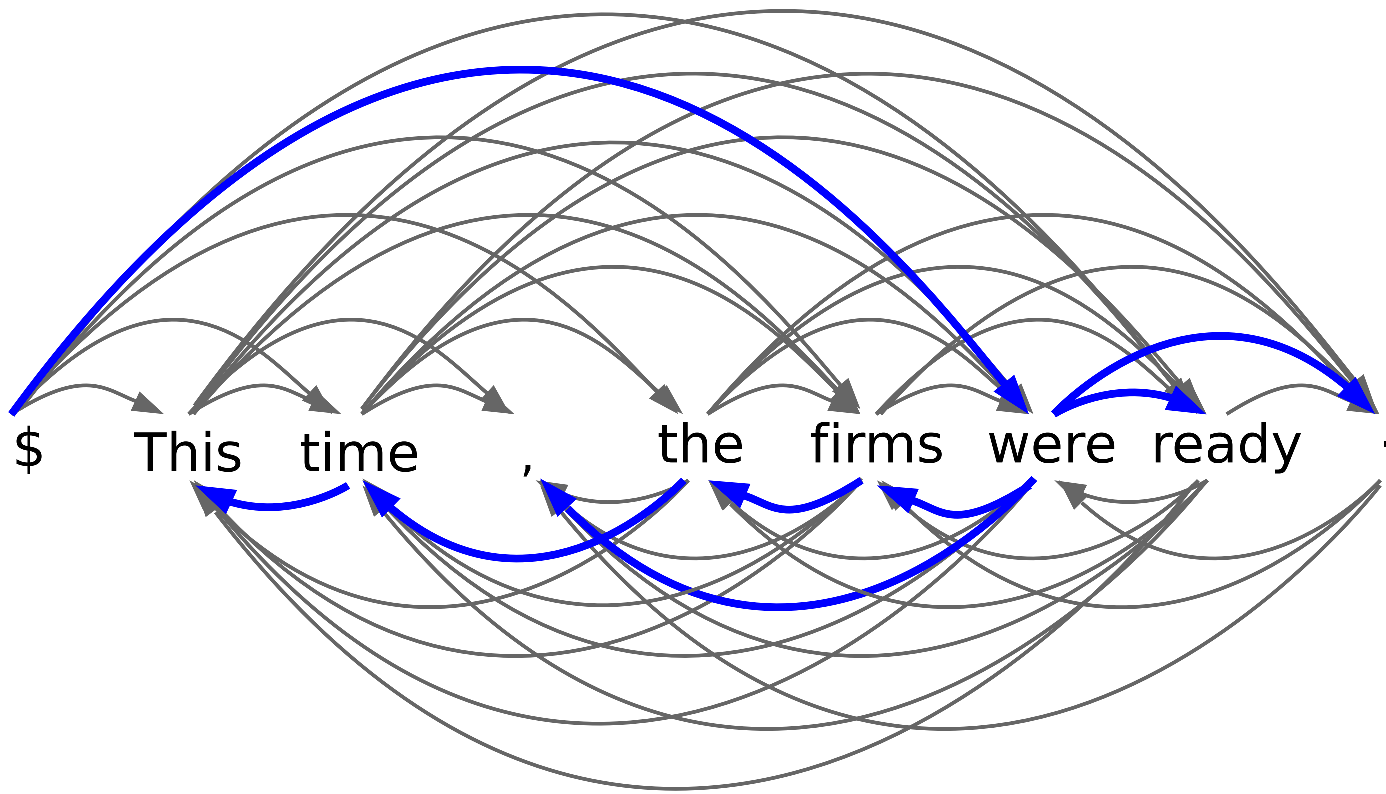
Current 1-best tree

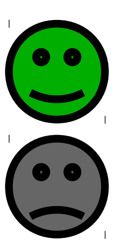
Winner edge
(permanently in 1-best tree)

Loser edge



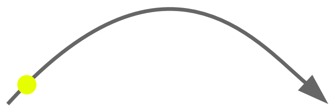
51 gray edges with unknown fate...
5 features per gray edge





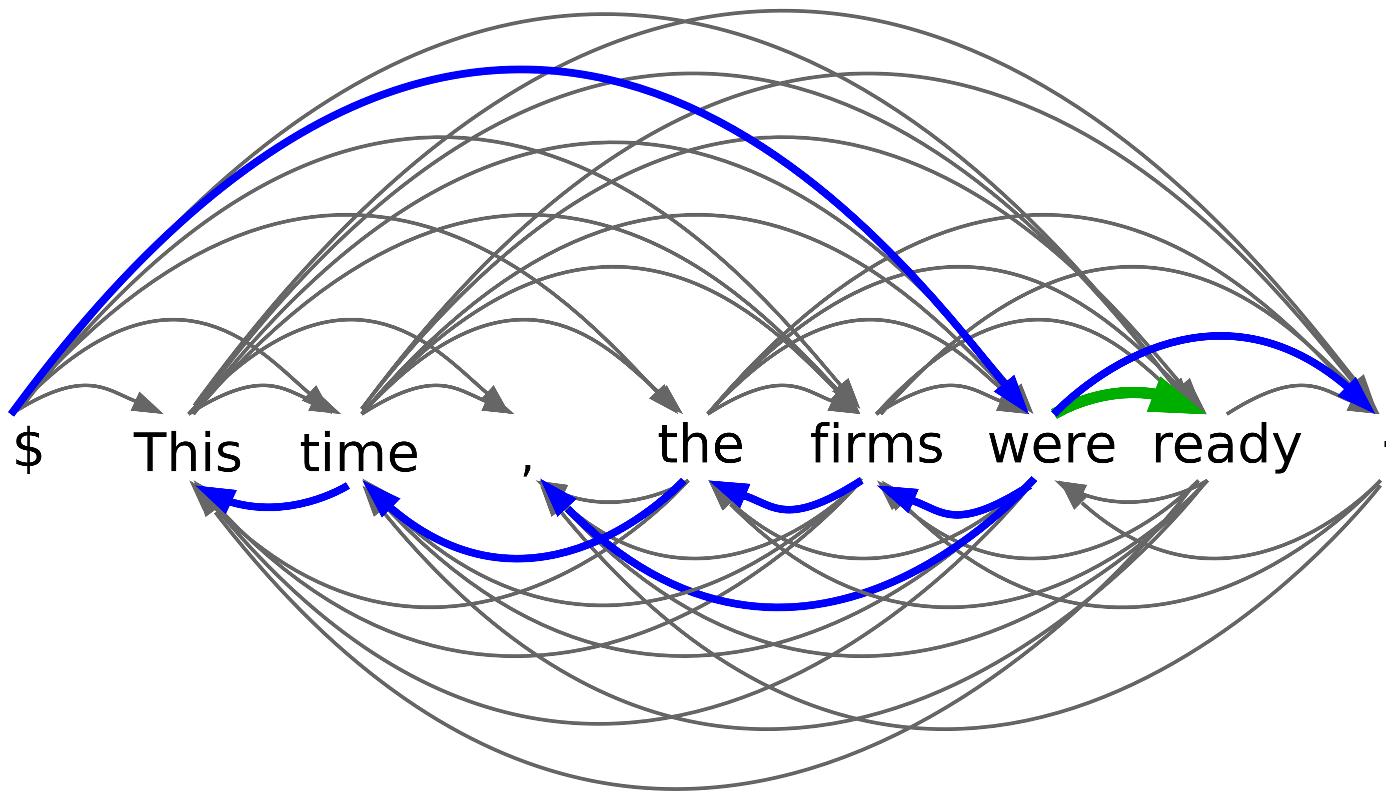
Undetermined edge
Current 1-best tree
Winner edge
(permanently in 1-best tree)
Loser edge

Non-projective decoding to find new 1-best tree



50 gray edges with unknown fate...

5 features per gray edge



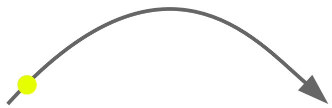
Undetermined edge

Current 1-best tree

Winner edge
(permanently in 1-best tree)

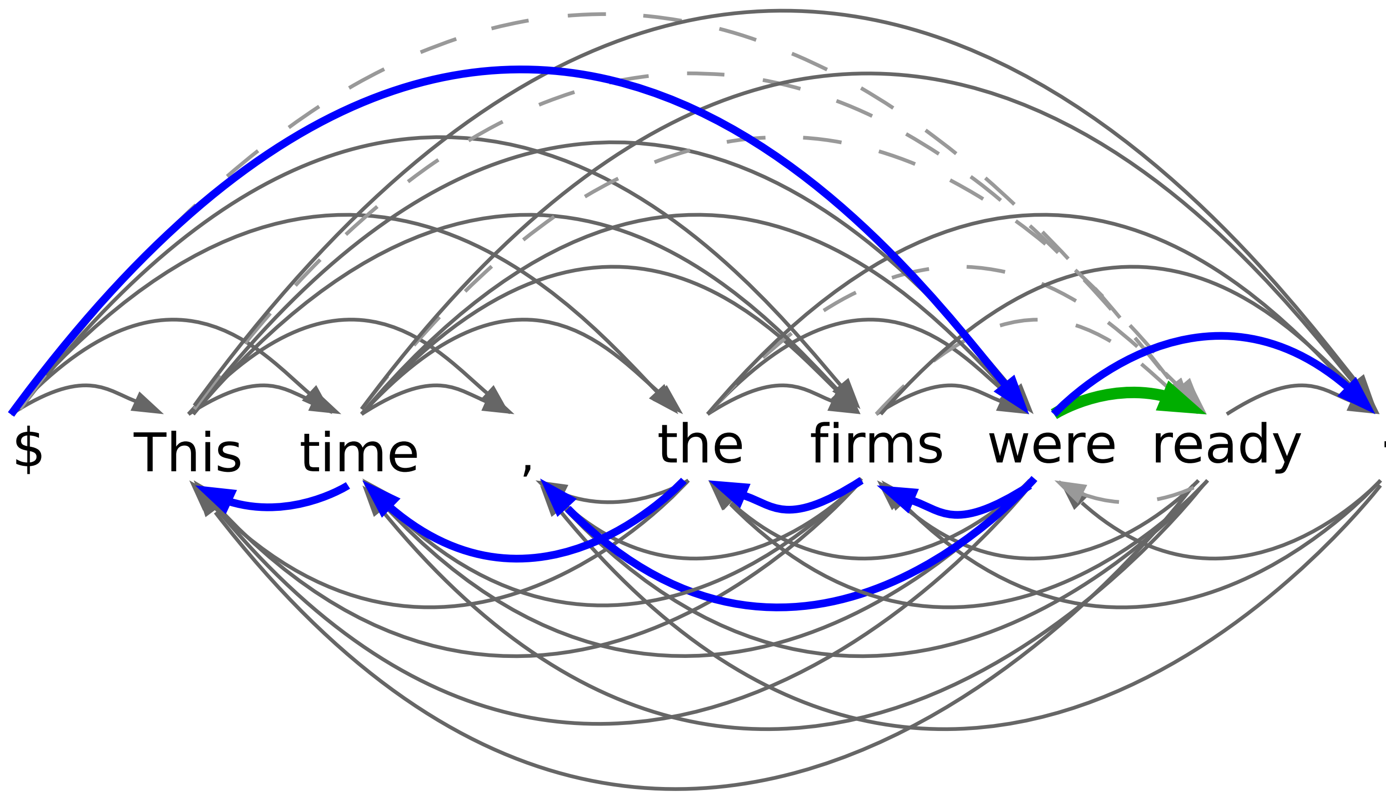
Loser edge

Classifier picks winners among the blue edges



44 gray edges with unknown fate...

5 features per gray edge



Undetermined edge

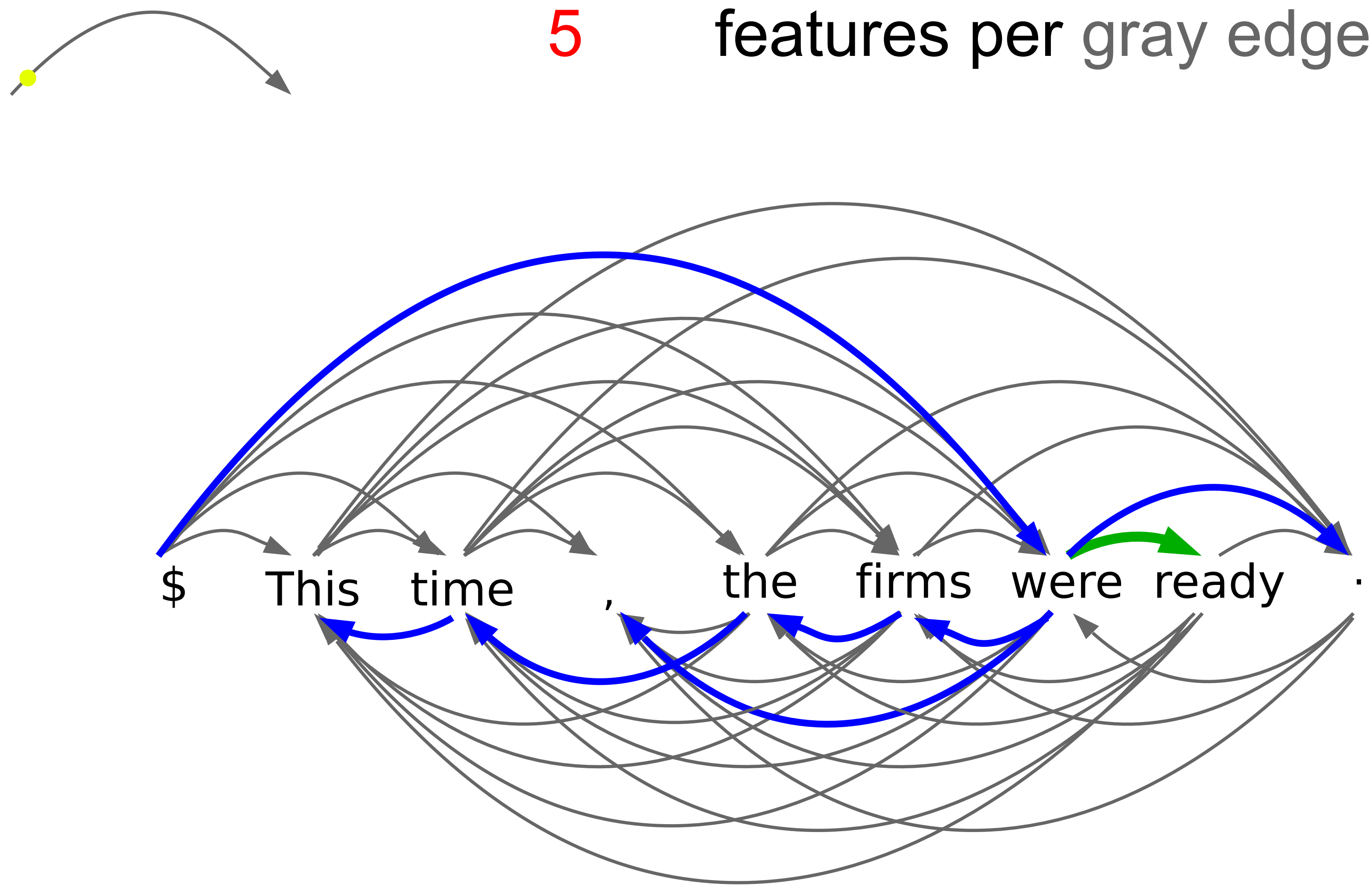
Current 1-best tree

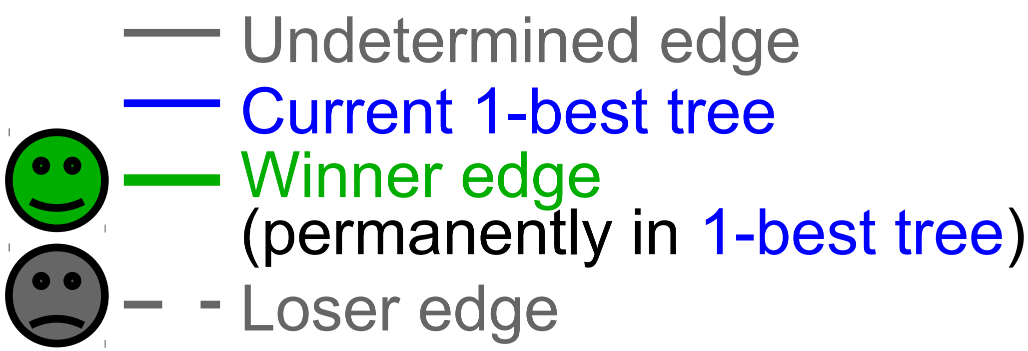
Winner edge
(permanently in 1-best tree)

Loser edge

Remove losers in conflict with the winners

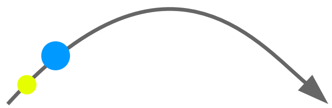
44 gray edges with unknown fate...
5 features per gray edge





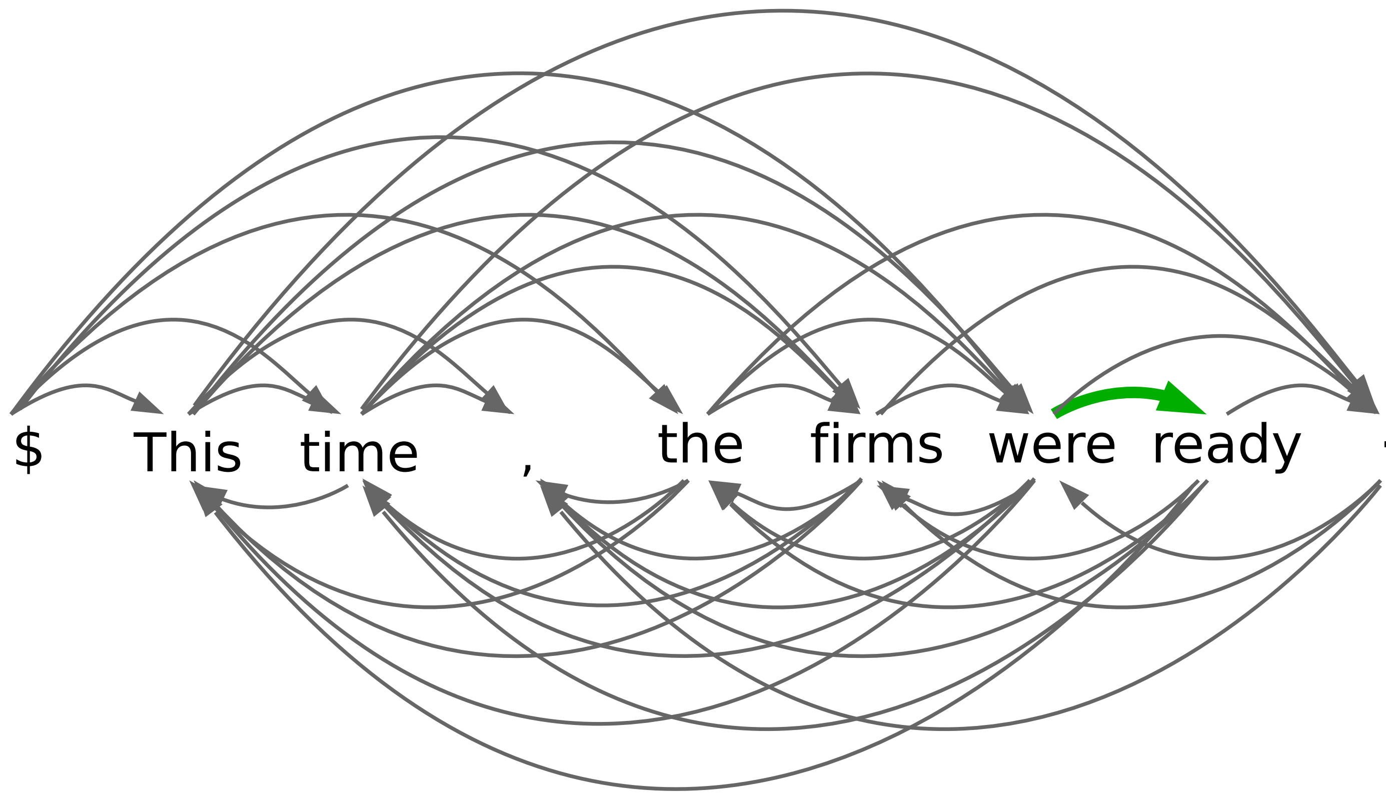
Remove losers in conflict with the winners

+ next feature group



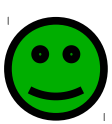
44 gray edges with unknown fate...

27 features per gray edge

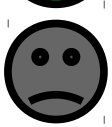


Undetermined edge

Current 1-best tree

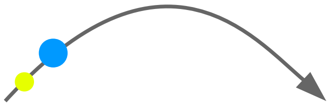


Winner edge
(permanently in 1-best tree)

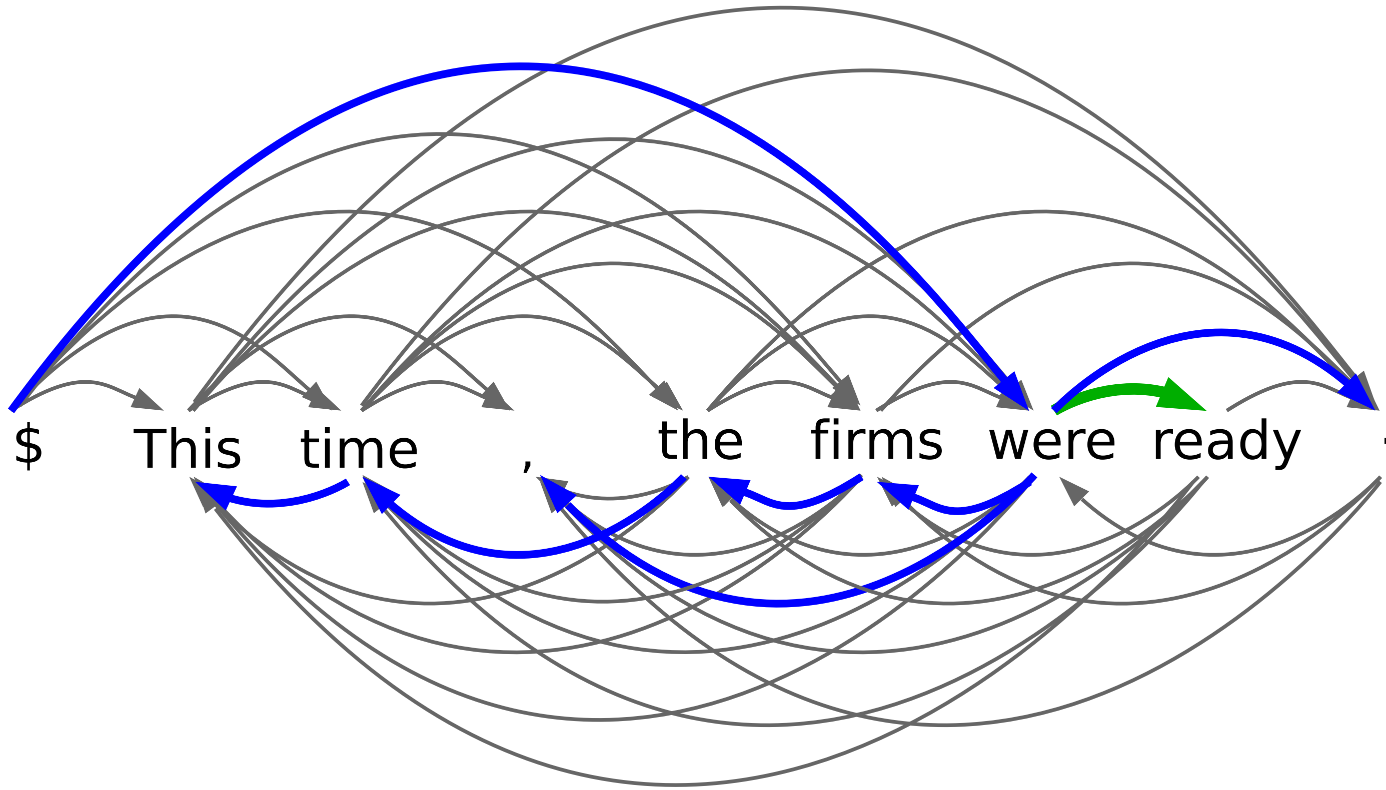


Loser edge

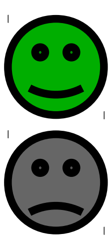
+ next feature group



44 gray edges with unknown fate...
27 features per gray edge



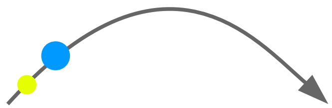
Undetermined edge
Current 1-best tree

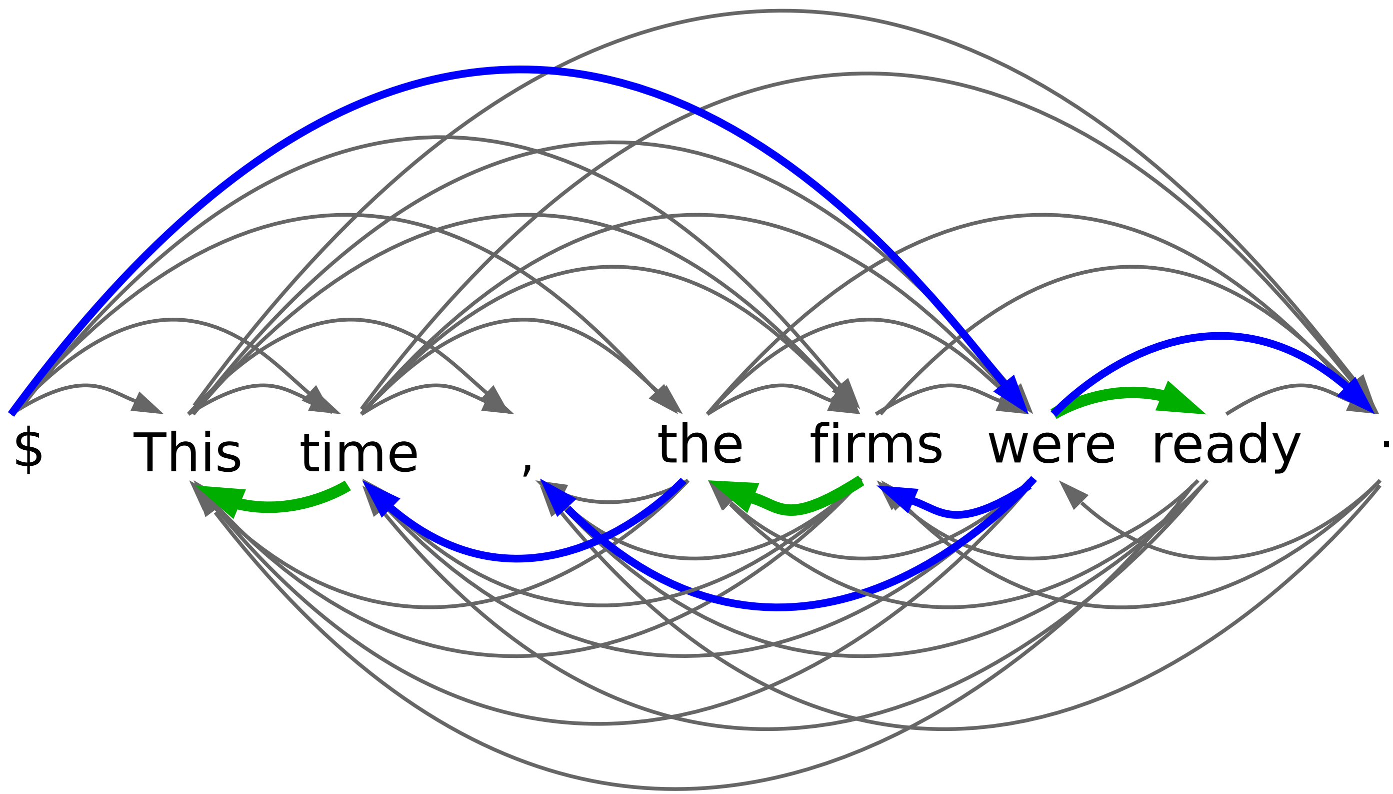


Winner edge
(permanently in 1-best tree)
Loser edge

Non-projective decoding to find new 1-best tree

42 gray edges with unknown fate...
27 features per gray edge

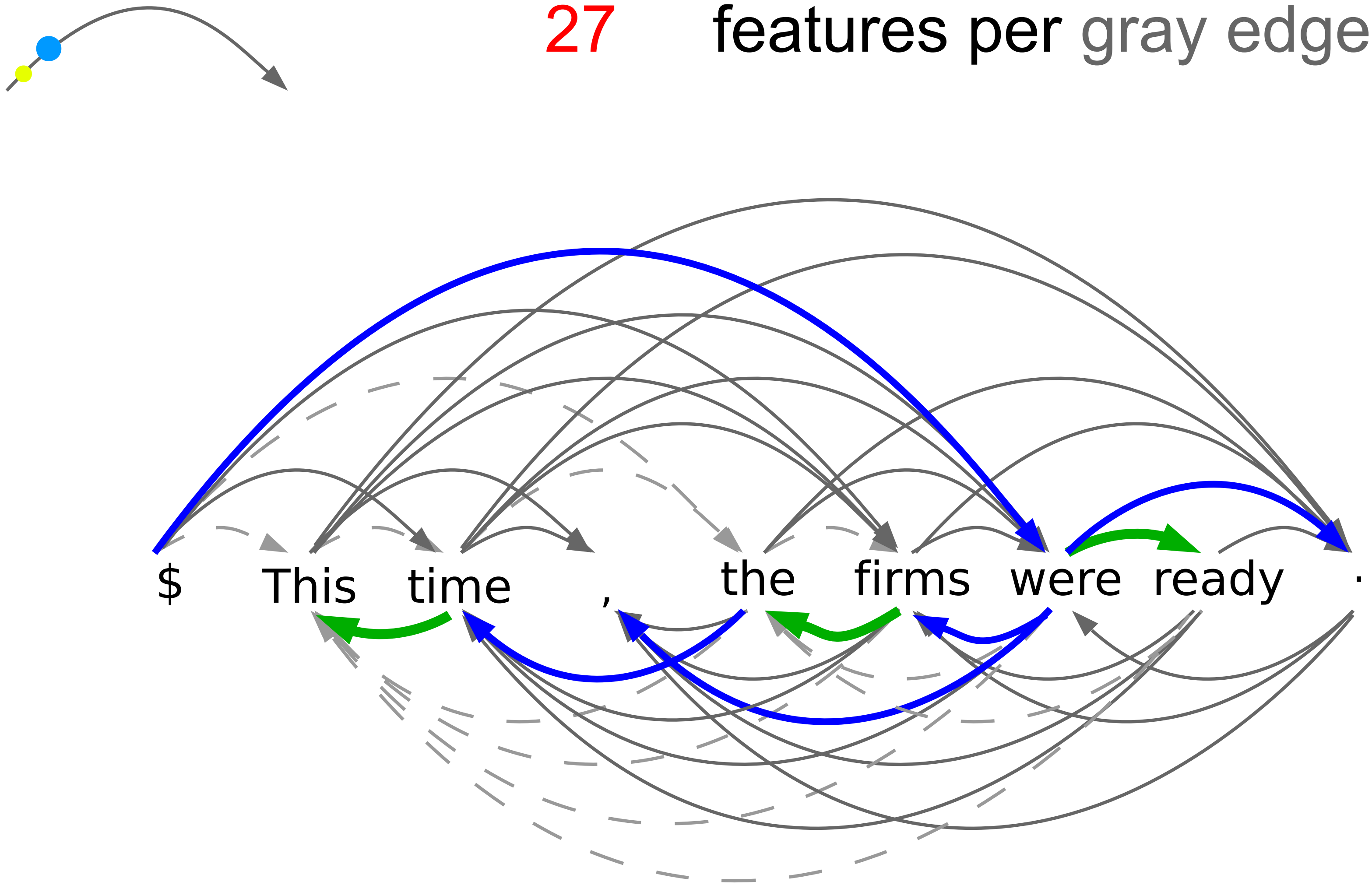


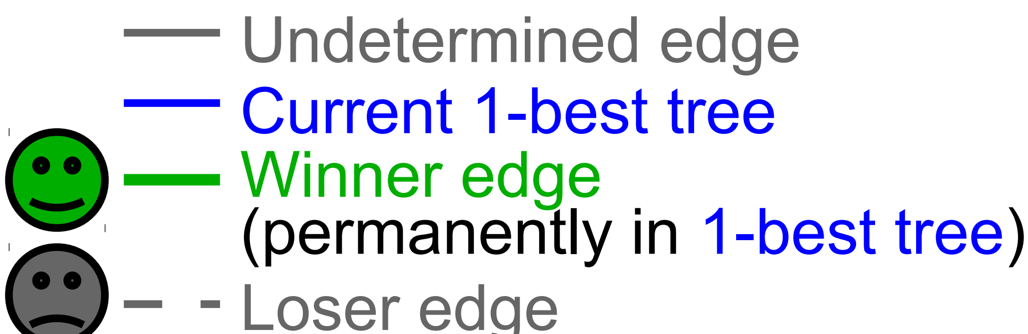


Undetermined edge
Current 1-best tree
Winner edge
(permanently in 1-best tree)
Loser edge

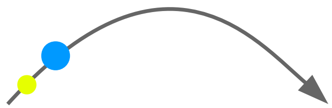
Classifier picks winners among the blue edges

31 gray edges with unknown fate...
27 features per gray edge

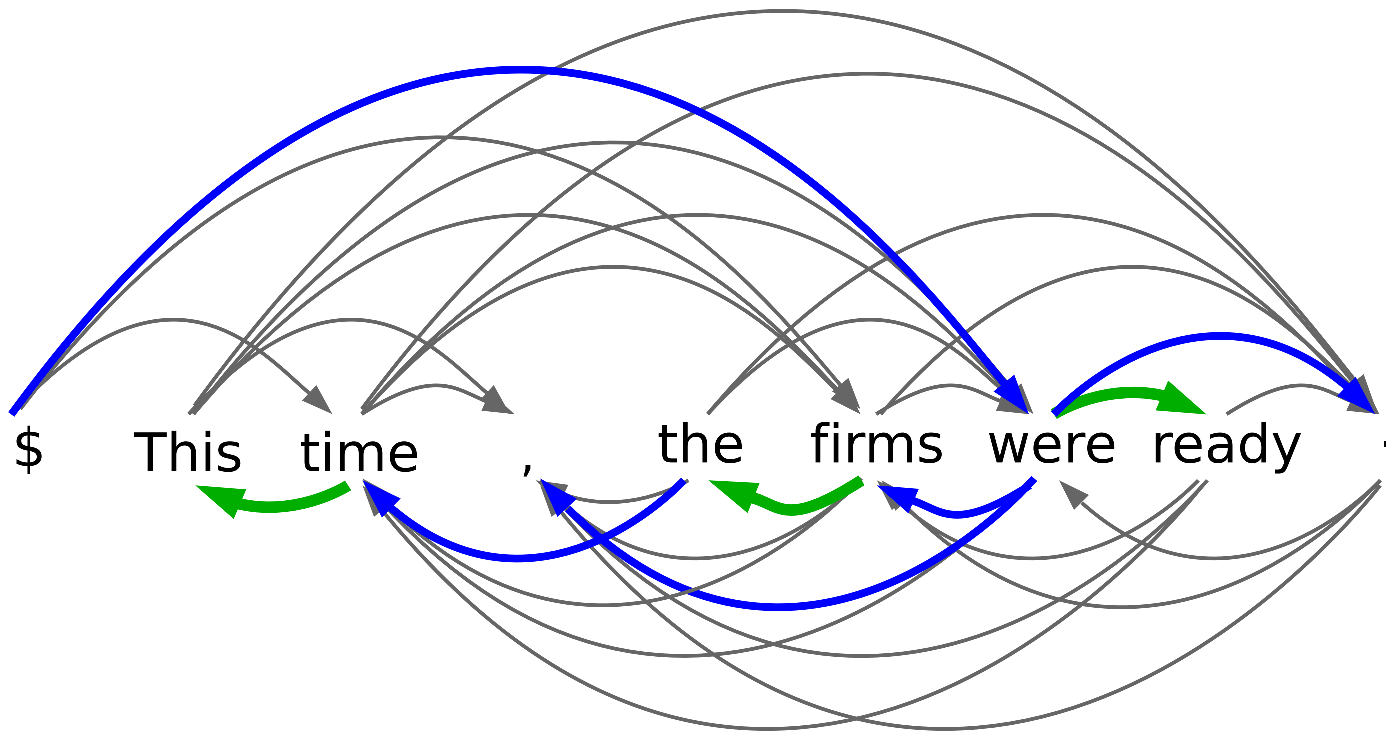




Remove losers in conflict with the winners



31 gray edges with unknown fate...
27 features per gray edge

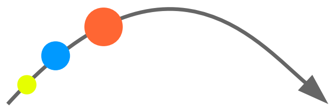


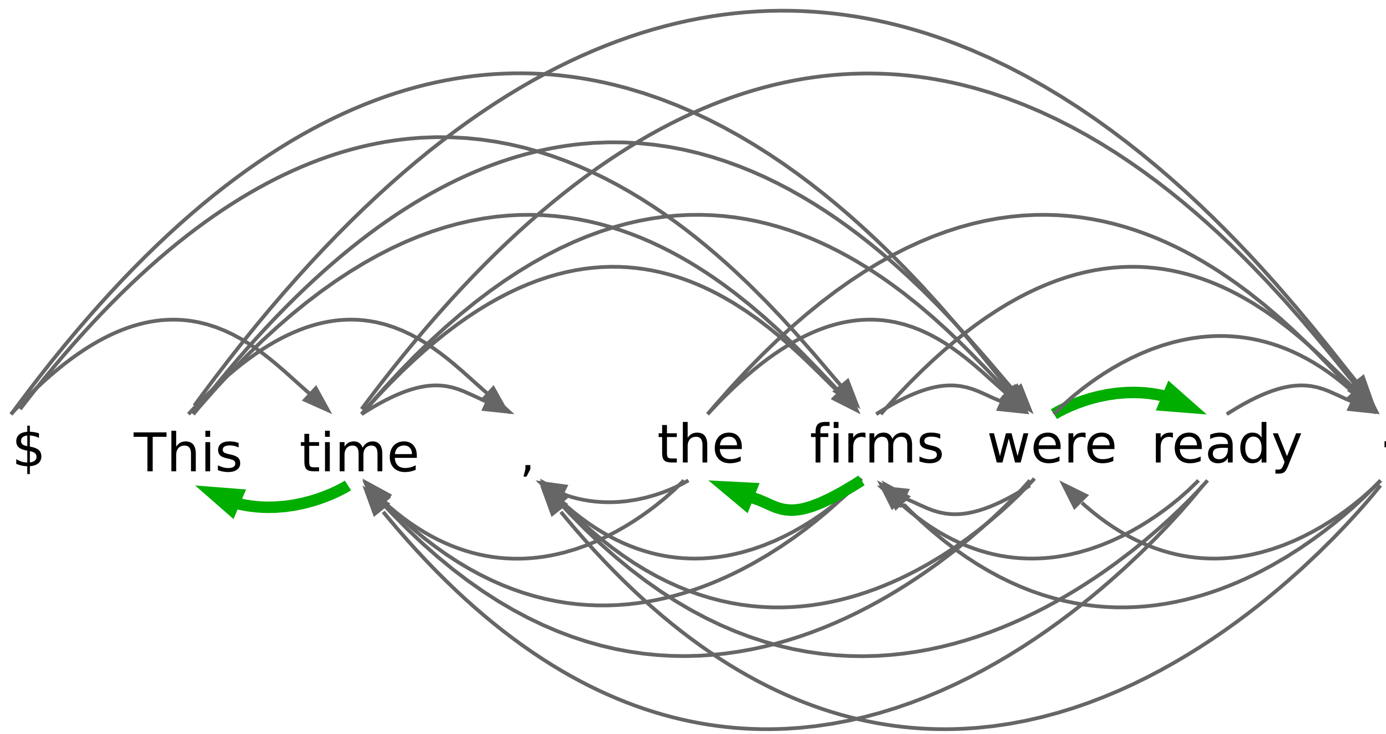
Undetermined edge
Current 1-best tree
Winner edge
(permanently in 1-best tree)
Loser edge

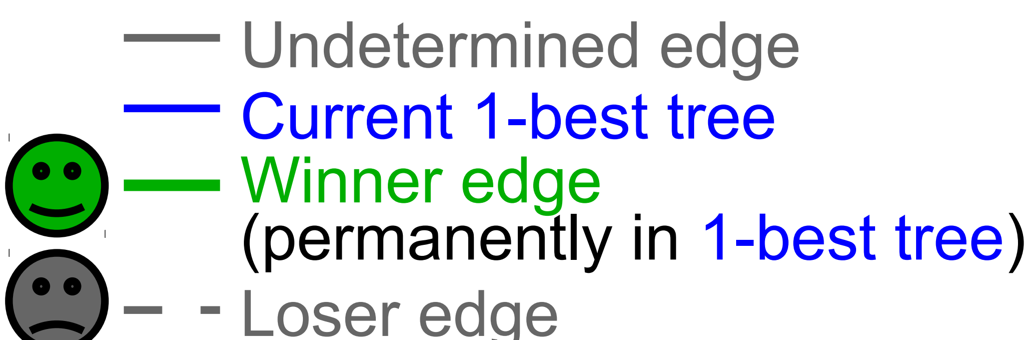
Remove losers in conflict with the winners

+ next feature group 31 gray edges with unknown fate...

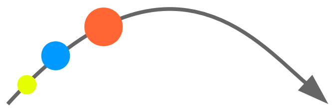
74 features per gray edge

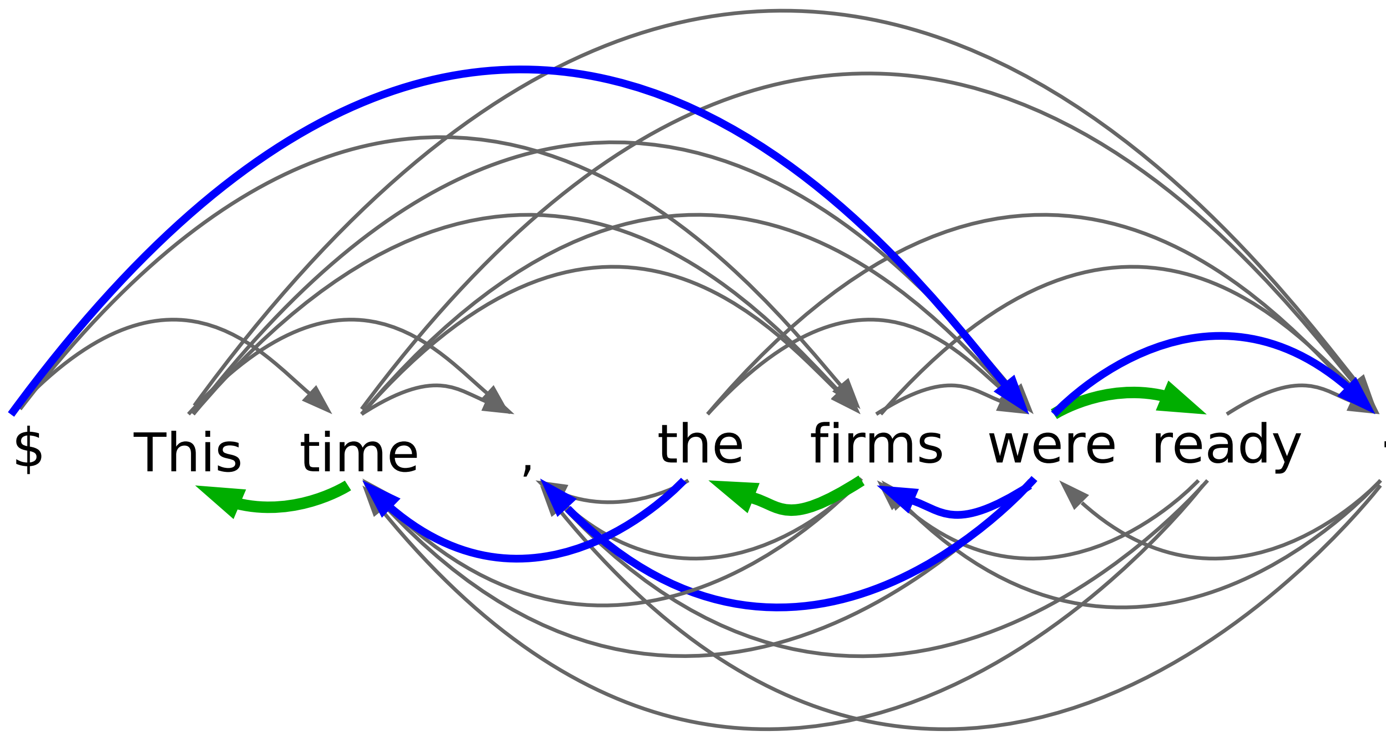






31 gray edges with unknown fate...
74 features per gray edge

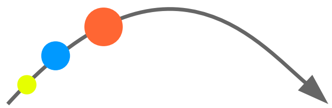


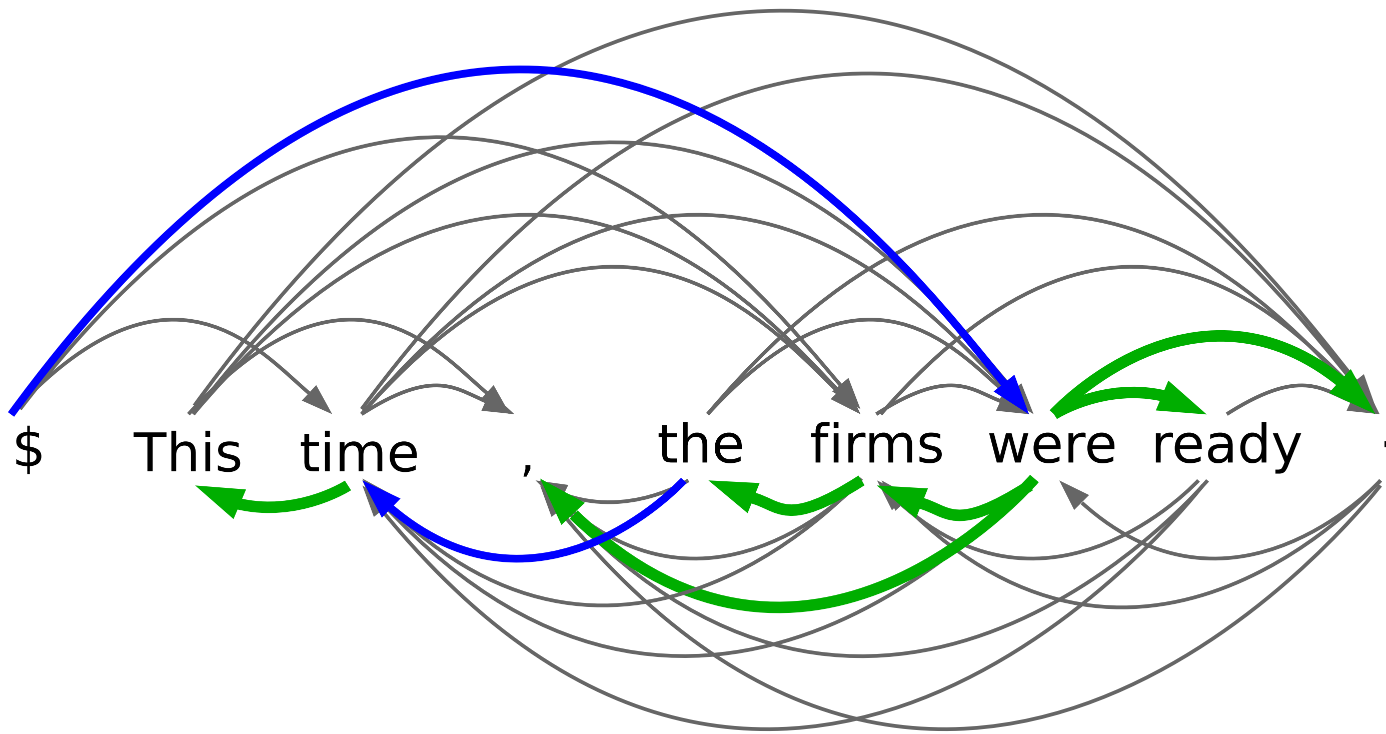


Undetermined edge
Current 1-best tree
Winner edge
(permanently in 1-best tree)
Loser edge

Non-projective decoding to find new 1-best tree

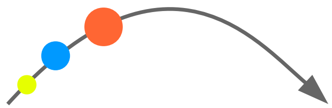
28 gray edges with unknown fate...
74 features per gray edge



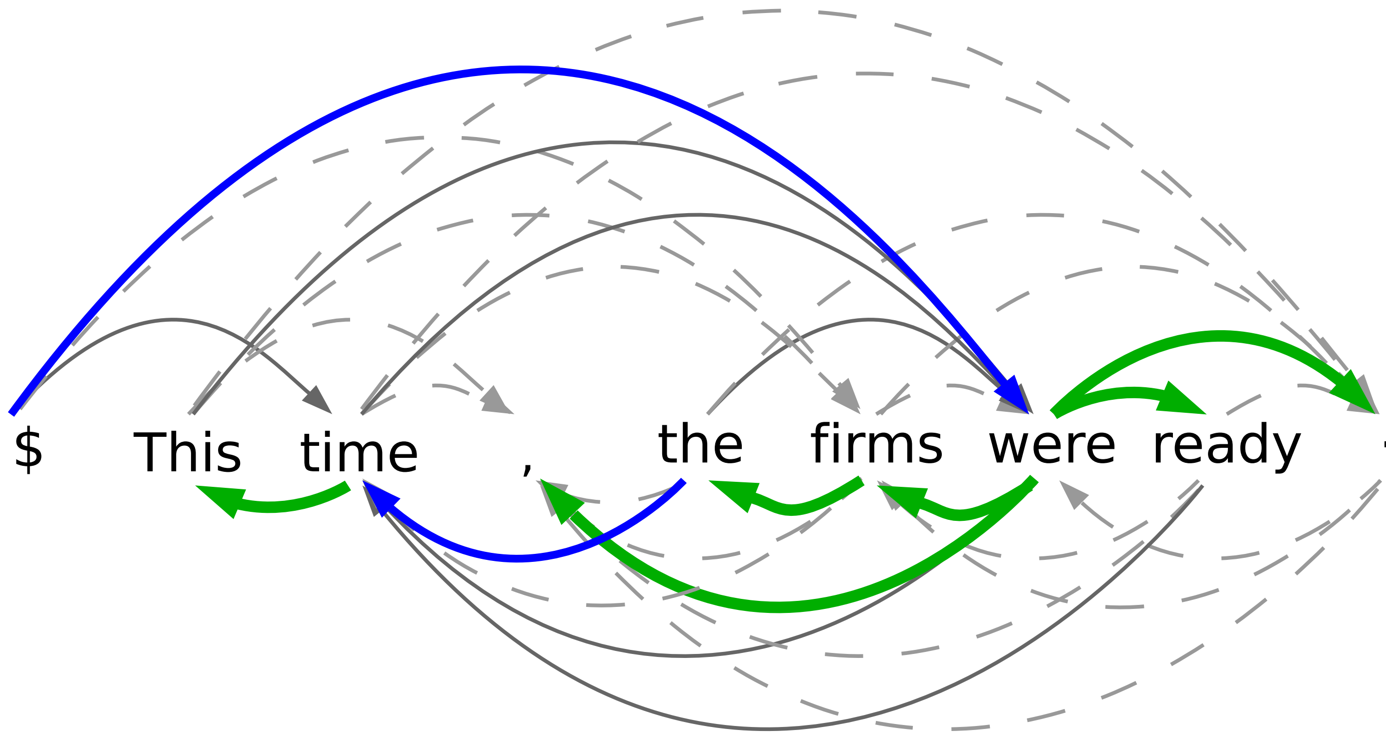


Undetermined edge
Current 1-best tree
Winner edge (permanently in 1-best tree)
Loser edge

Classifier picks winners among the blue edges

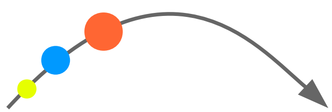


8 gray edges with unknown fate...
74 features per gray edge



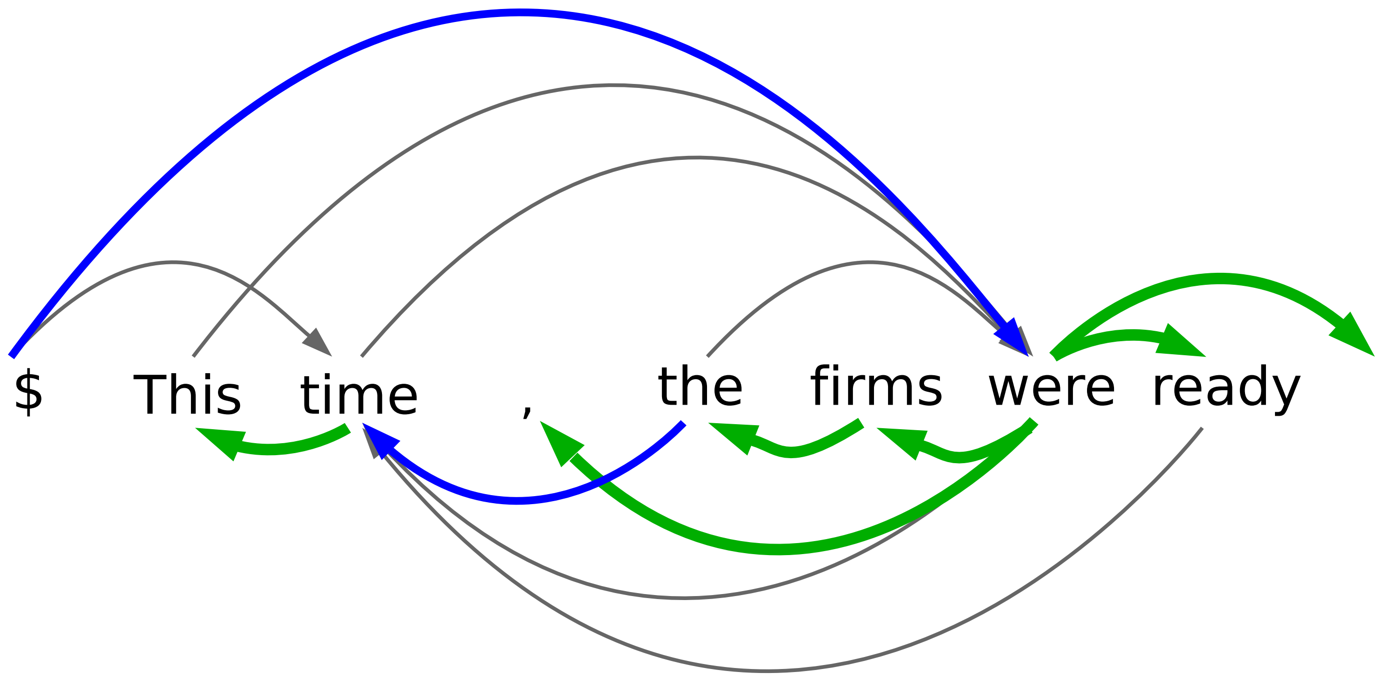
Undetermined edge
Current 1-best tree
Winner edge
(permanently in 1-best tree)
Loser edge

Remove losers in conflict with the winners



8 gray edges with unknown fate...

74 features per gray edge



Undetermined edge

Current 1-best tree

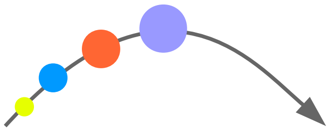
Winner edge
(permanently in 1-best tree)

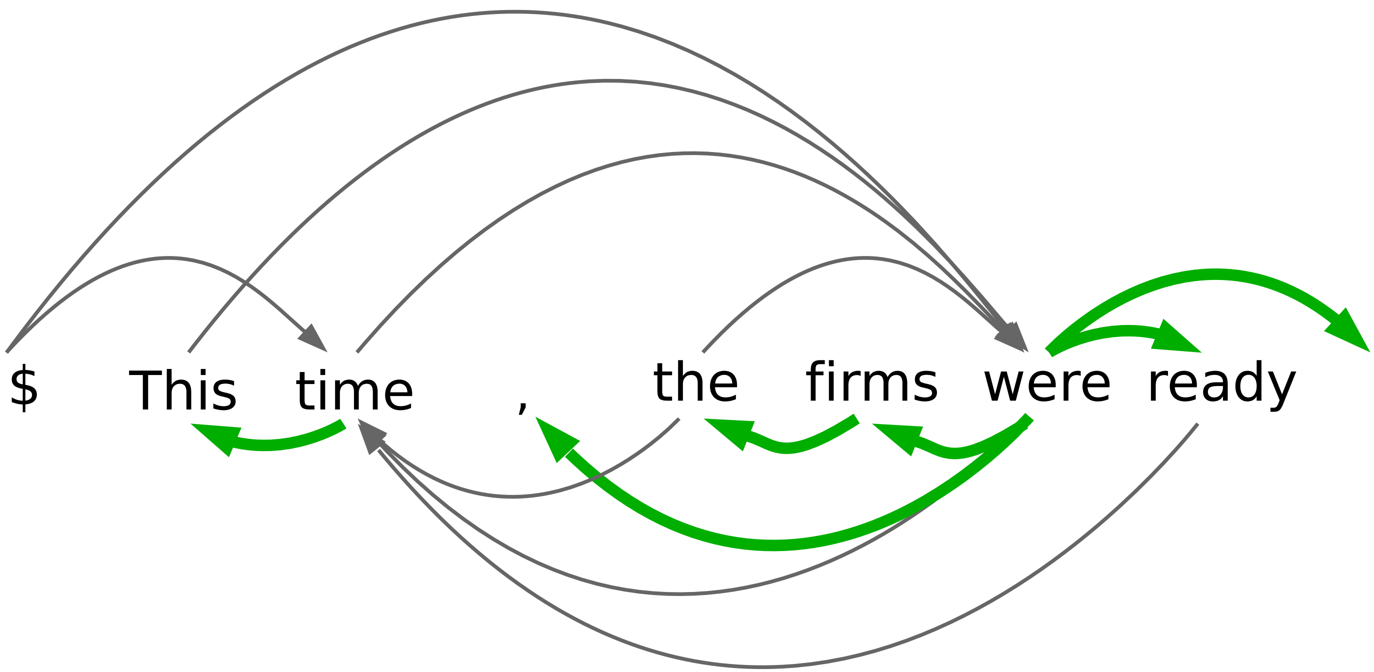
Loser edge

Remove losers in conflict with the winners

+ next feature group 8 gray edges with unknown fate...

107 features per gray edge



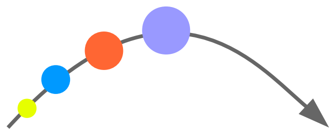


Undetermined edge

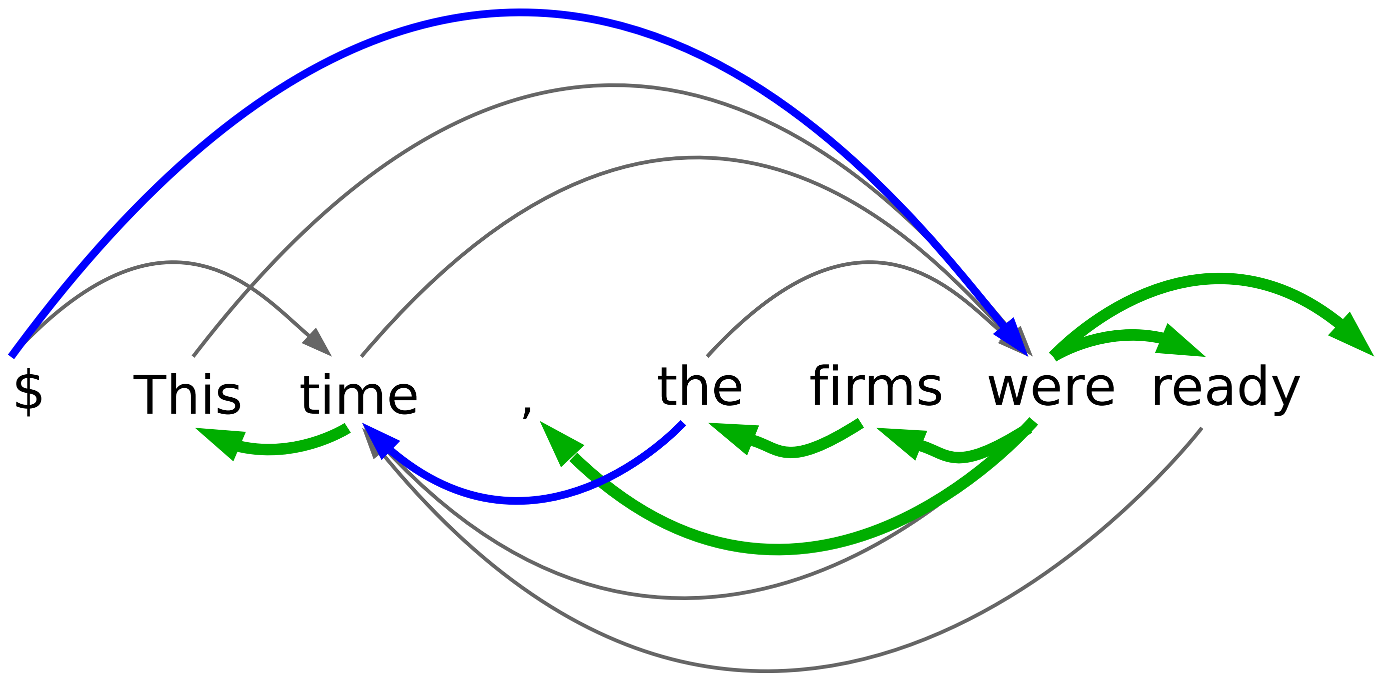
Current 1-best tree

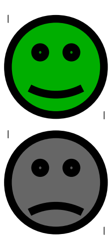
Winner edge
(permanently in 1-best tree)

Loser edge



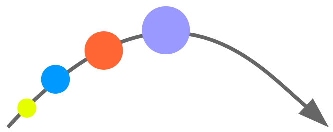
8 gray edges with unknown fate...
107 features per gray edge



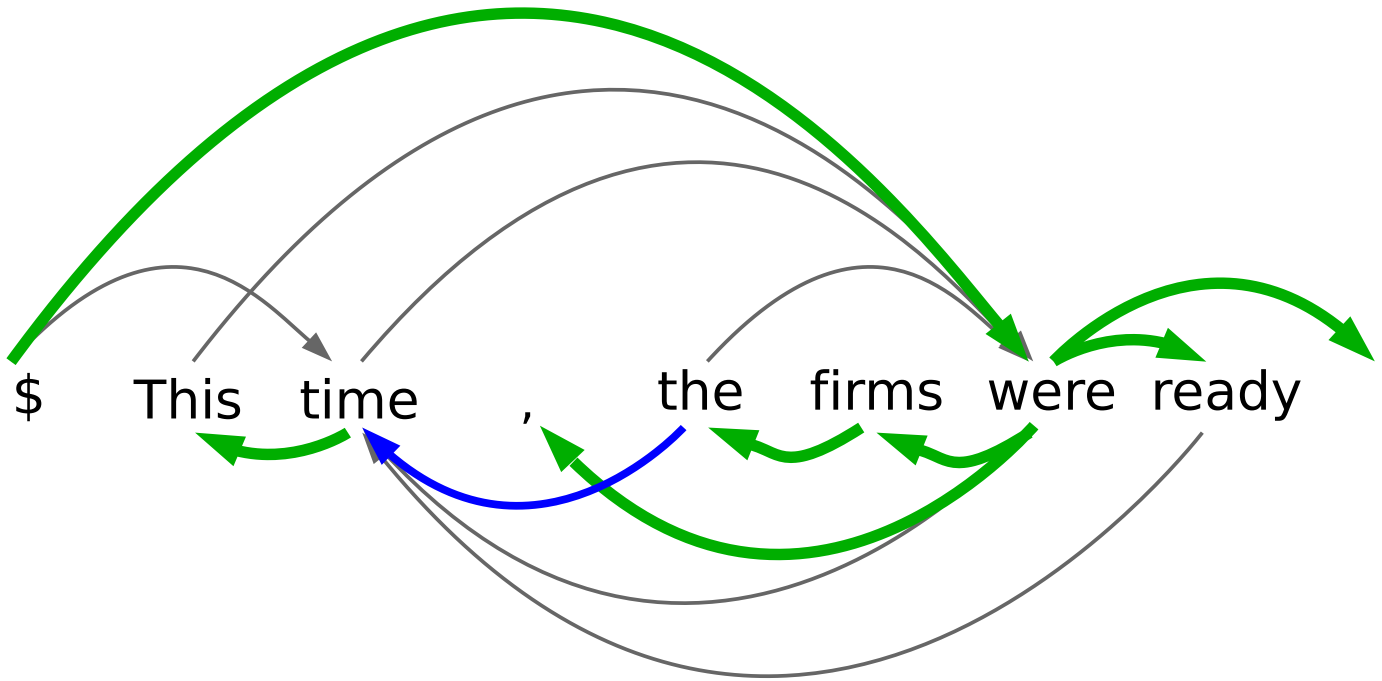


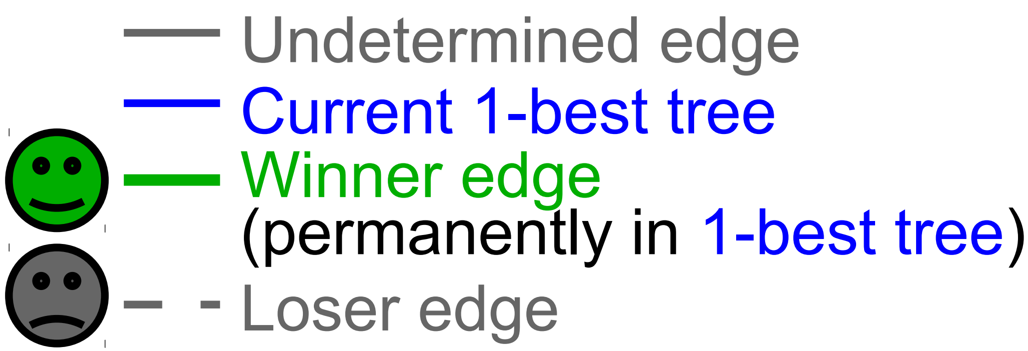
Undetermined edge
Current 1-best tree
Winner edge
(permanently in 1-best tree)
Loser edge

Non-projective decoding to find new 1-best tree

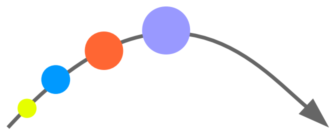


7 gray edges with unknown fate...
107 features per gray edge



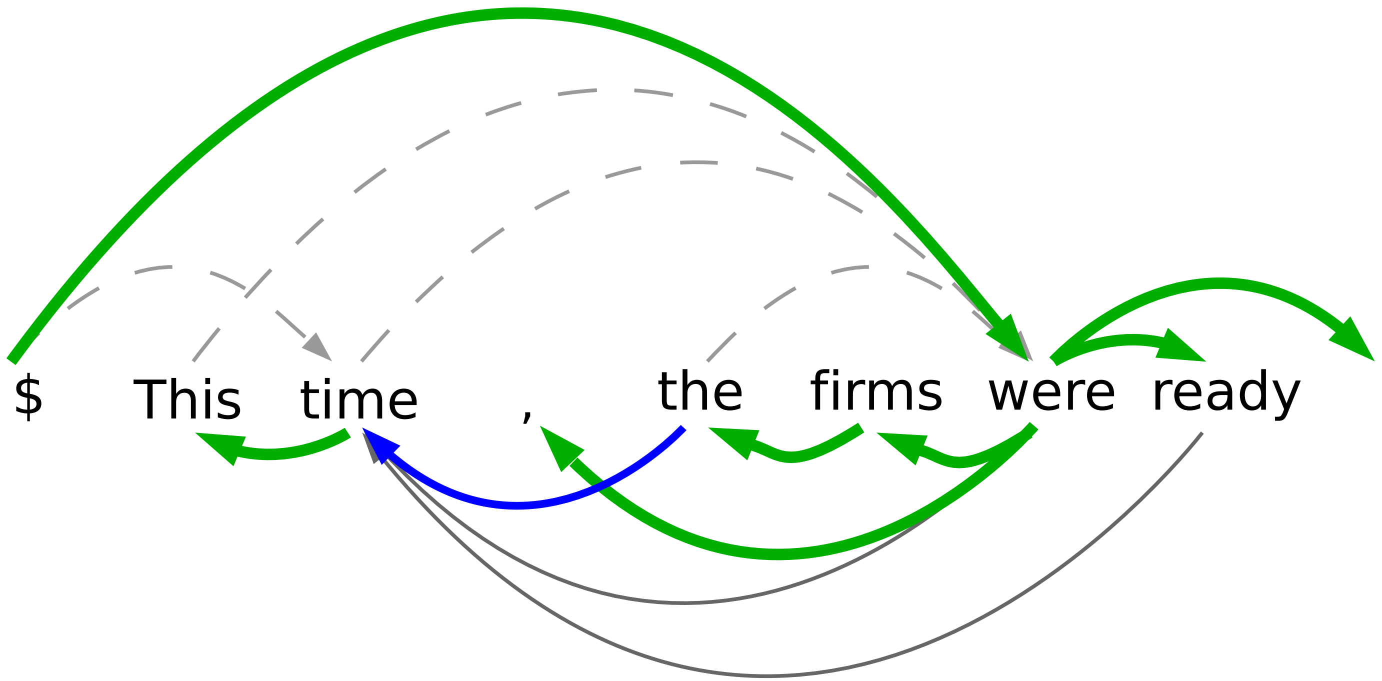


Classifier picks winners among the blue edges



3 gray edges with unknown fate...

107 features per gray edge



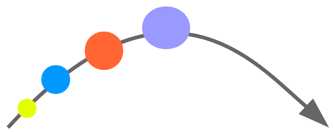
Undetermined edge

Current 1-best tree

Winner edge (permanently in 1-best tree)

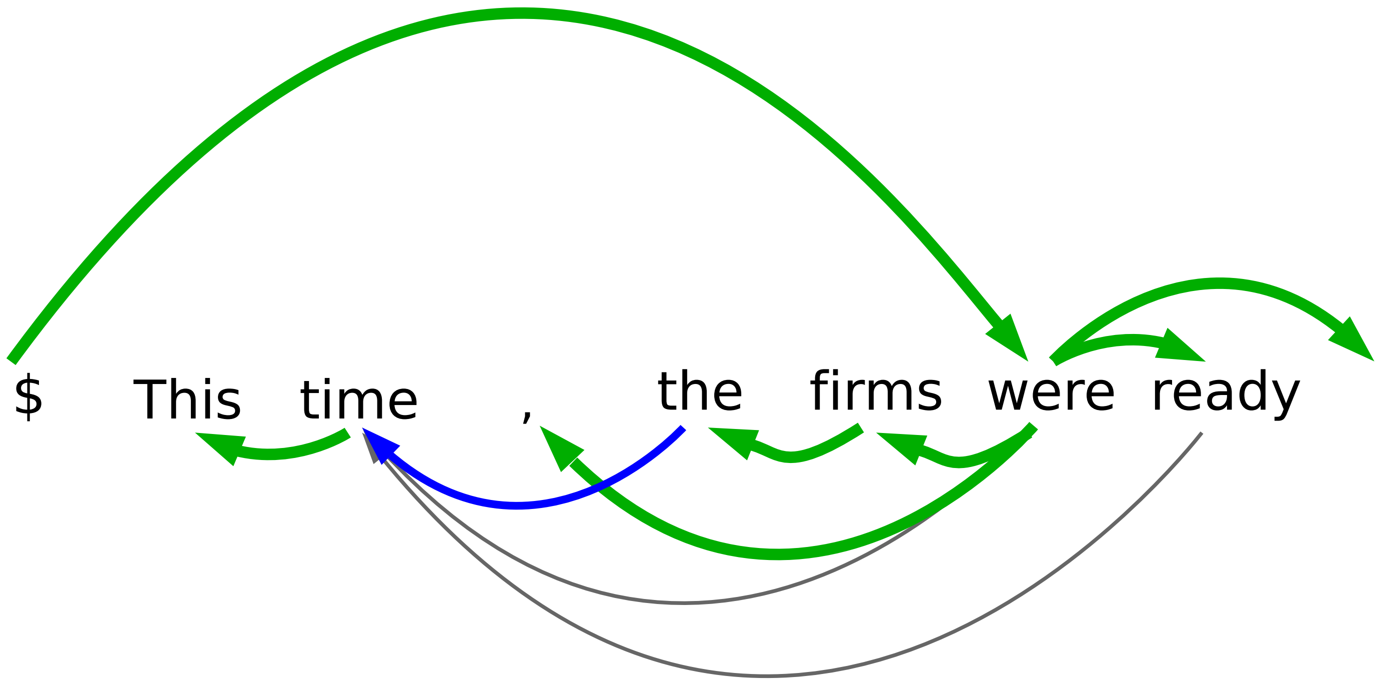
Loser edge

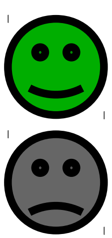
Remove losers in conflict with the winners



3 gray edges with unknown fate...

107 features per gray edge





Undetermined edge

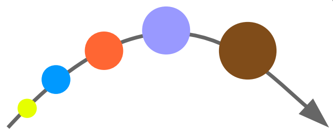
Current 1-best tree

Winner edge
(permanently in 1-best tree)

Loser edge

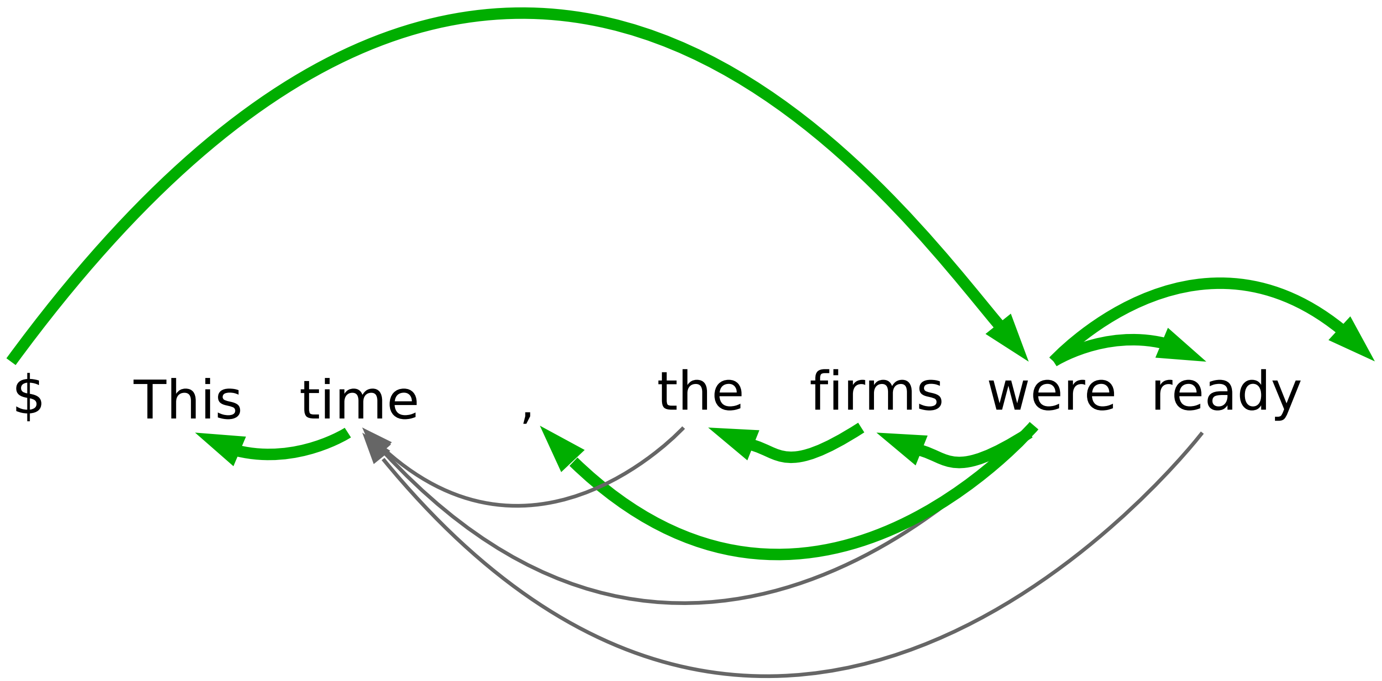
Remove losers in conflict with the winners

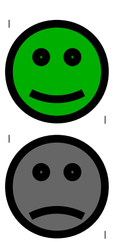
+ last feature group



3 gray edges with unknown fate...

268 features per gray edge



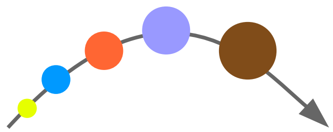


Undetermined edge

Current 1-best tree

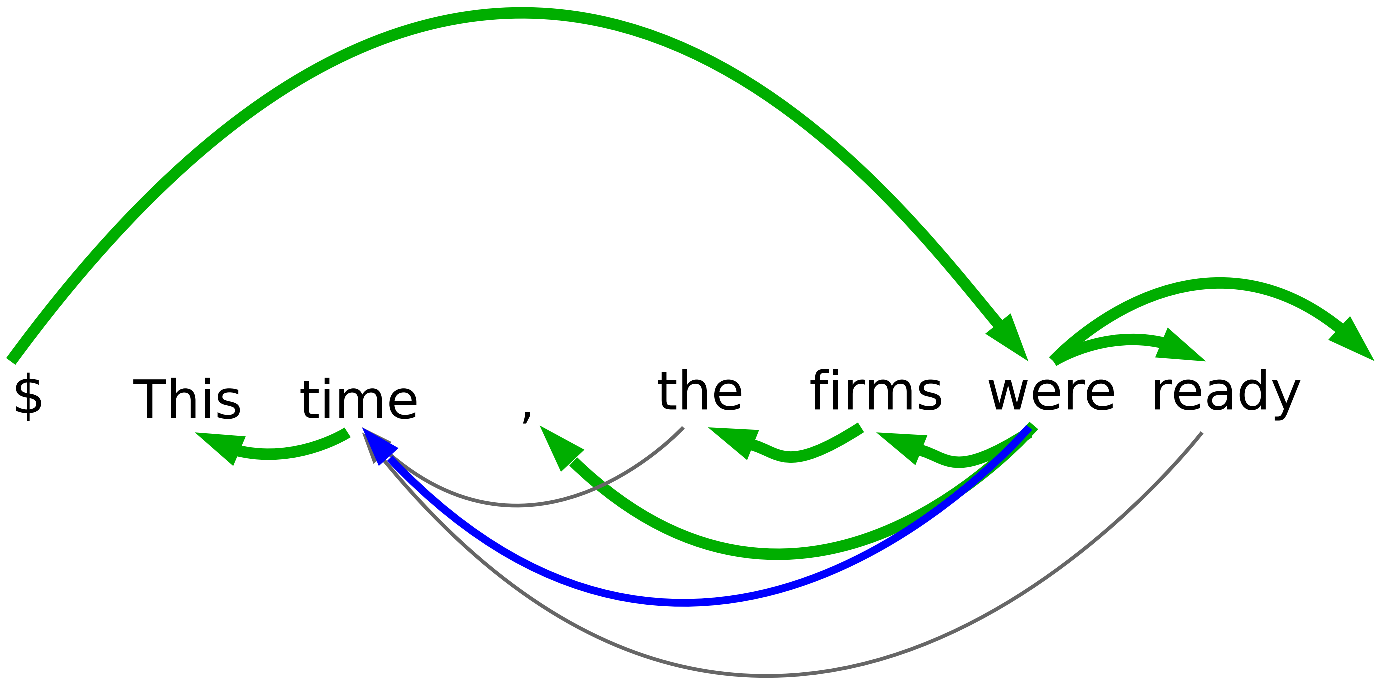
Winner edge
(permanently in 1-best tree)

Loser edge



0 gray edge with unknown fate...

268 features per gray edge



Undetermined edge

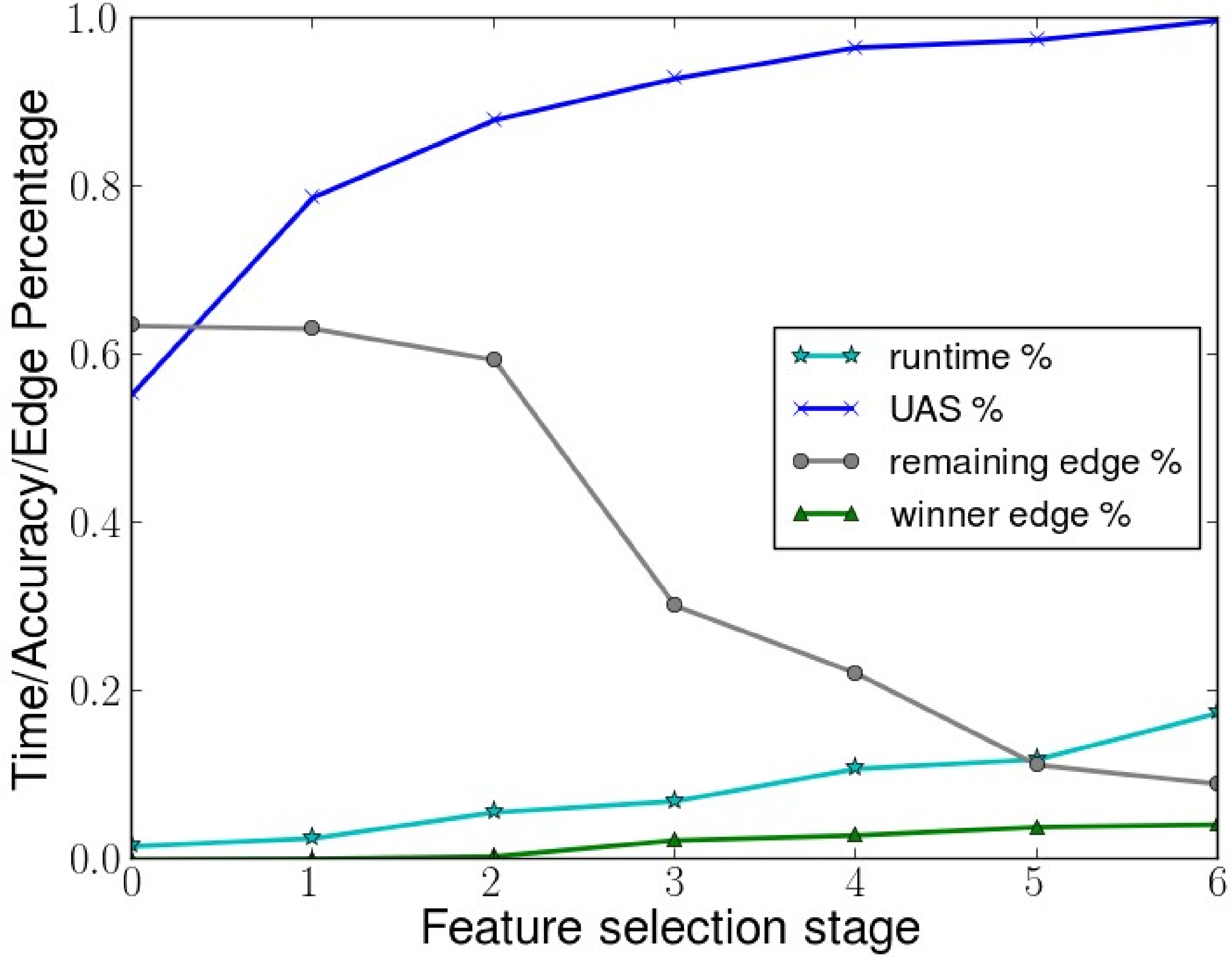
Current 1-best tree

Winner edge (permanently in 1-best tree)

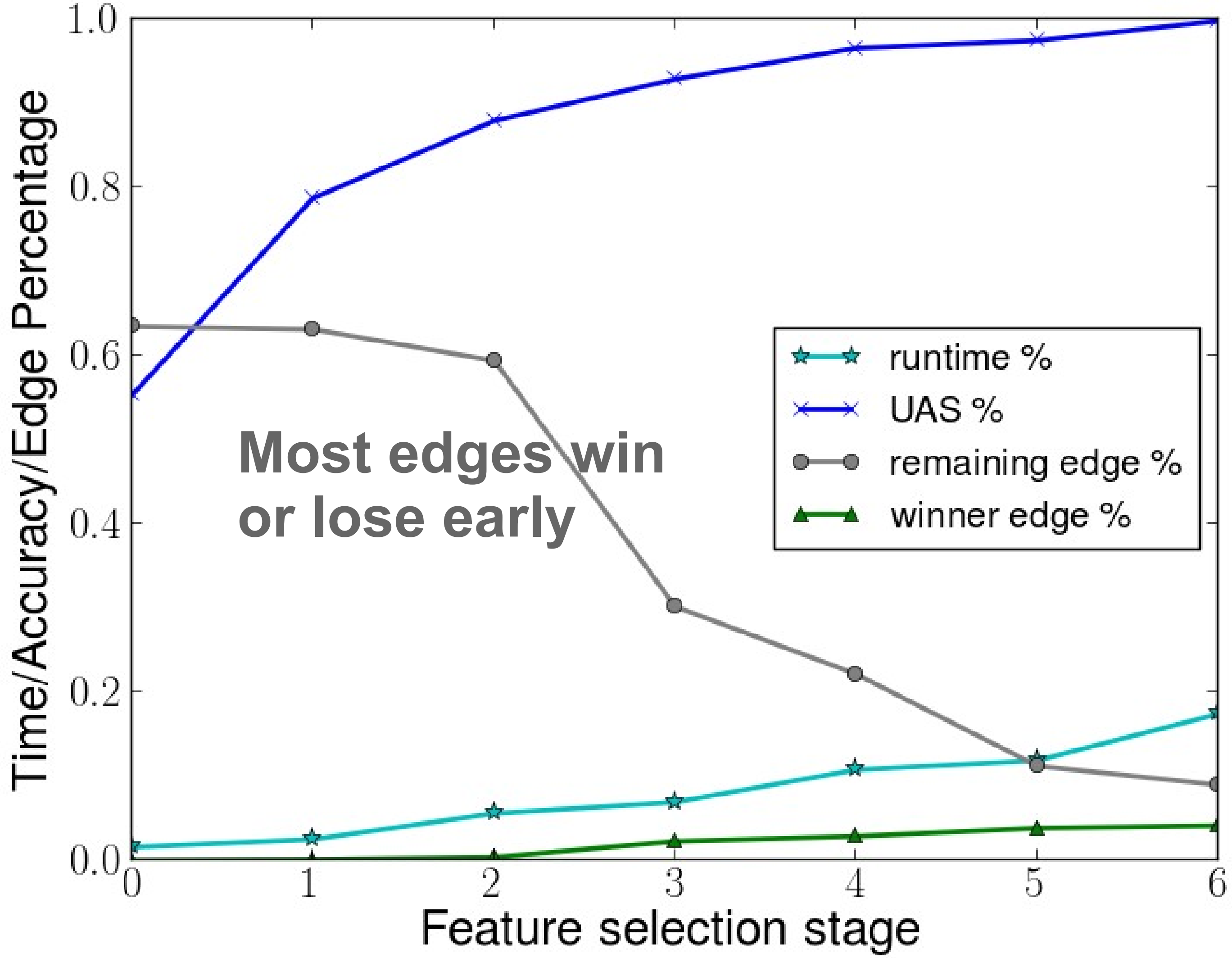
Loser edge

Projective decoding to find final 1-best tree

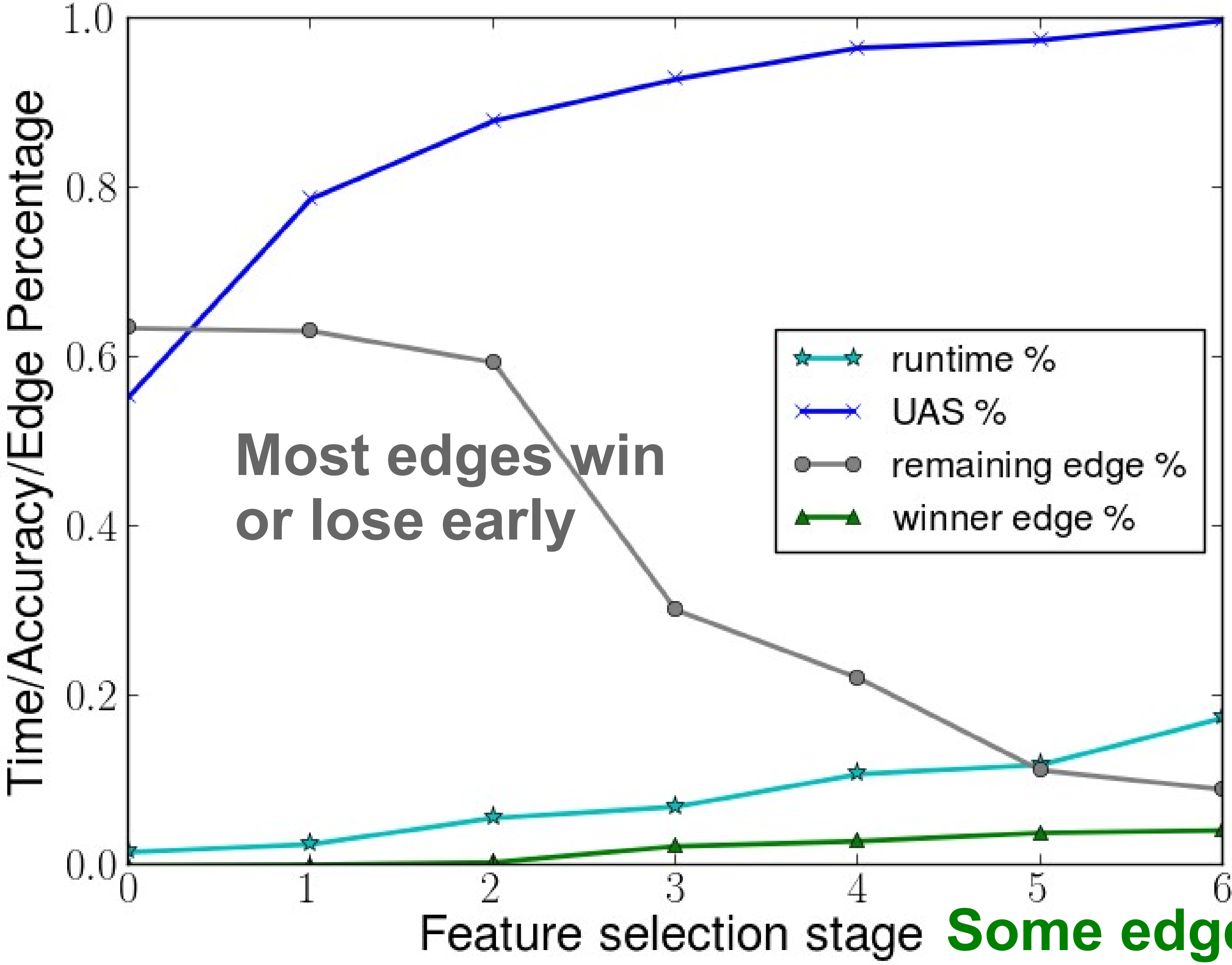
What Happens During the Average Parse?



What Happens During the Average Parse?

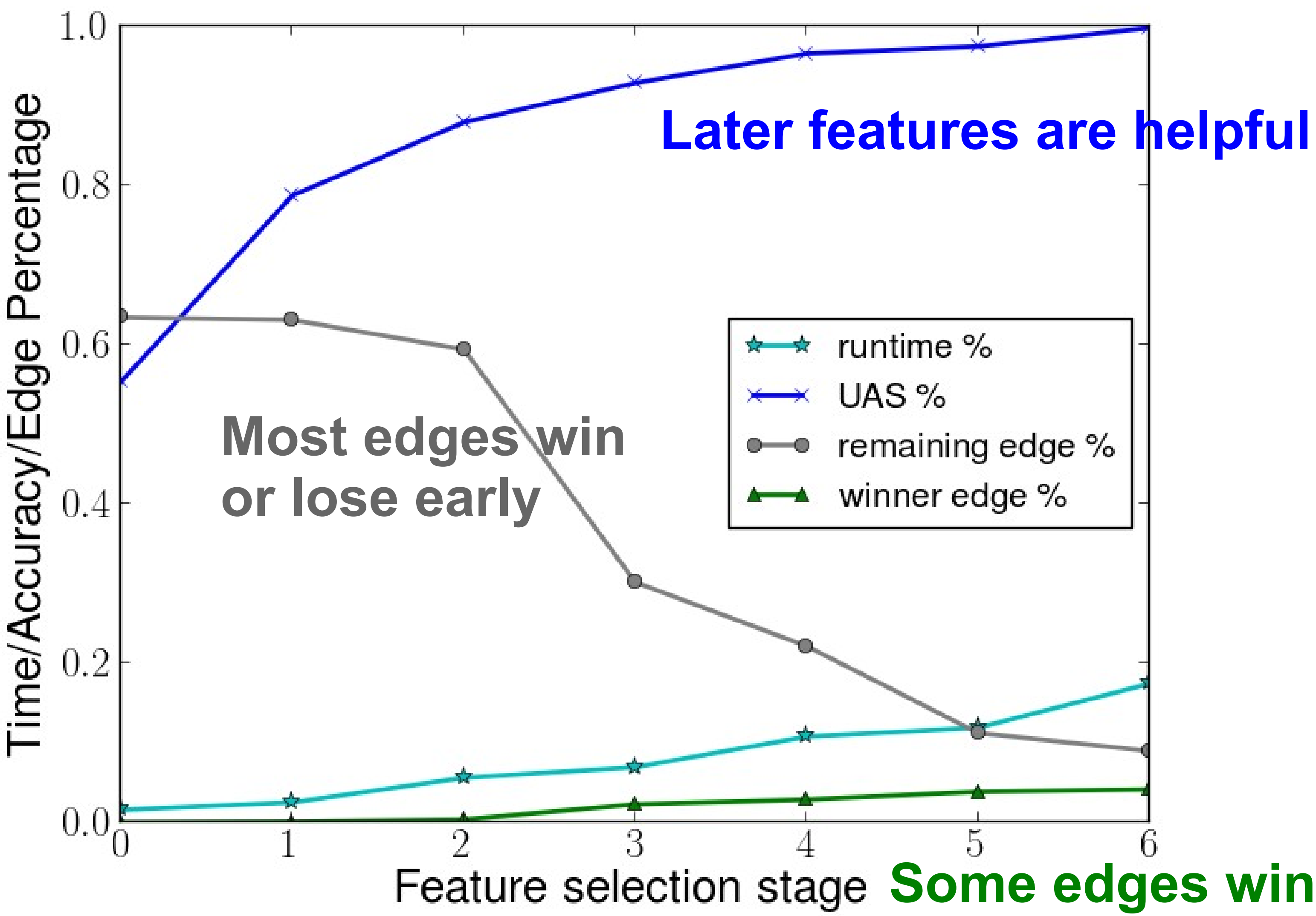


What Happens During the Average Parse?



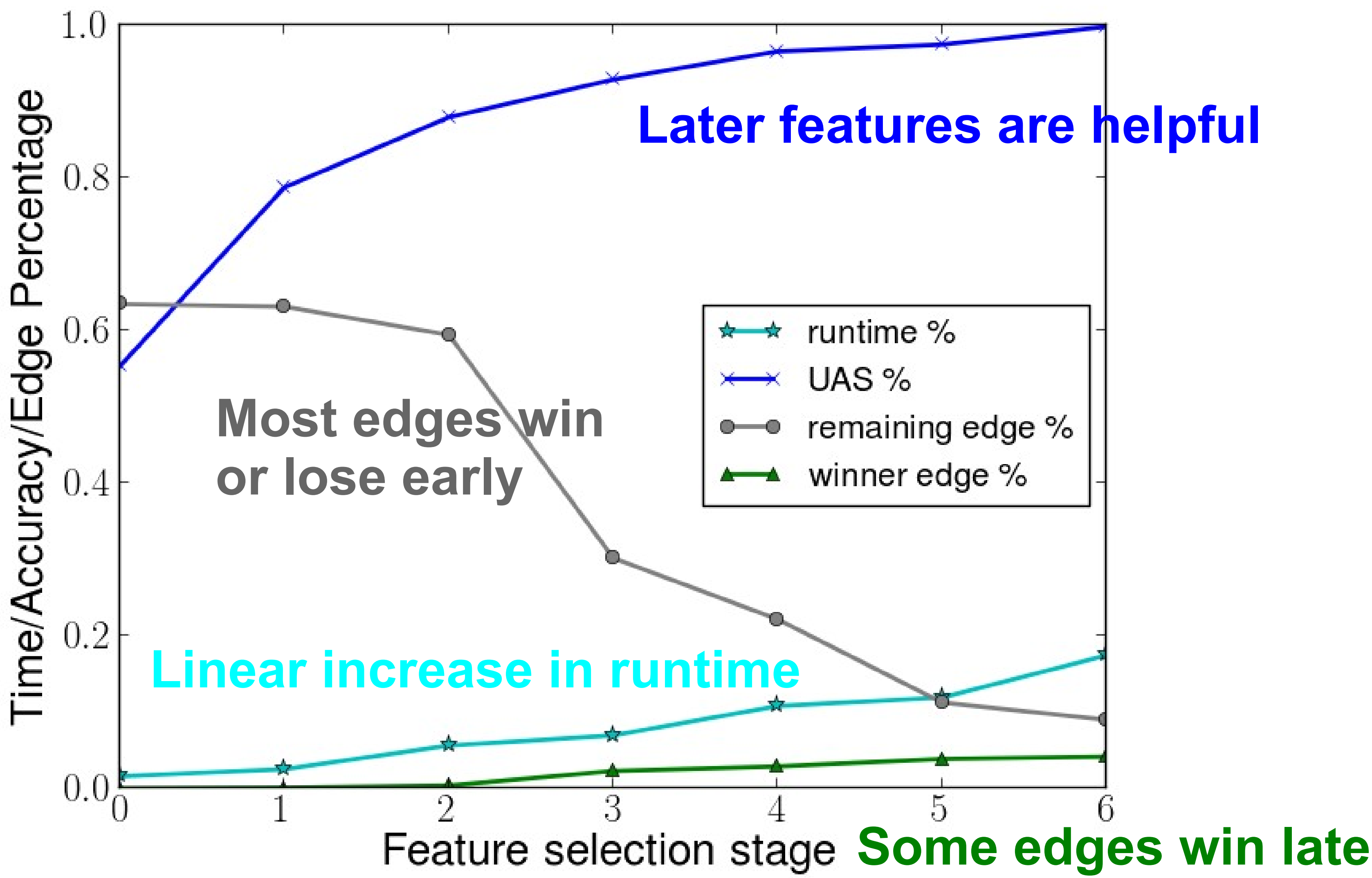
Some edges win late

What Happens During the Average Parse?



Some edges win late

What Happens During the Average Parse?

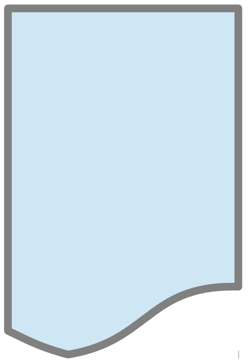


Summary: How Early Decisions Are Made

- Winners
  - Will definitely appear in the 1-best tree
- Losers
  - Have the same child as a winning edge
  - Form cycle with winning edges
  - Cross a winning edge *(optional)*
  - Share root ($) with a winning edge *(optional)*
- Undetermined
  - Add the next feature group to the remaining gray edges

Feature Template Ranking

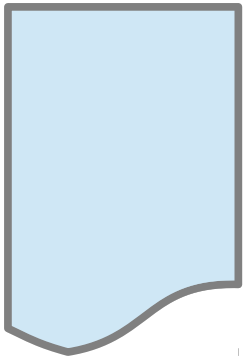
- Forward selection



Feature Template Ranking

- Forward selection

A 0.60
B 0.49
C 0.55

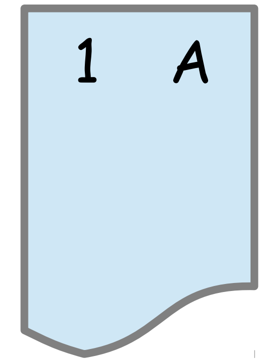


Feature Template Ranking

- Forward selection

A 0.60
B 0.49
C 0.55

$\xrightarrow{A}$



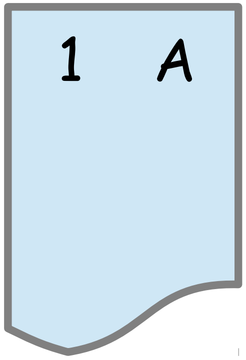
Feature Template Ranking

- Forward selection

A	0.60
B	0.49
C	0.55

$\xrightarrow{A}$

A&B	0.80
A&C	0.85



Feature Template Ranking

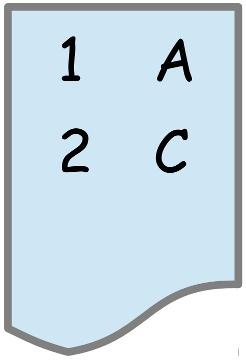
- Forward selection

A 0.60
B 0.49
C 0.55

$\xrightarrow{\text{A}}$

A&B 0.80
A&C 0.85

$\xrightarrow{\text{C}}$



Feature Template Ranking

- Forward selection

A	0.60
B	0.49
C	0.55

$\xrightarrow{\text{A}}$

A&B	0.80
A&C	0.85

$\xrightarrow{\text{C}}$

A&C&B 0.9

1	A
2	C
3	B

Feature Template Ranking

- Forward selection

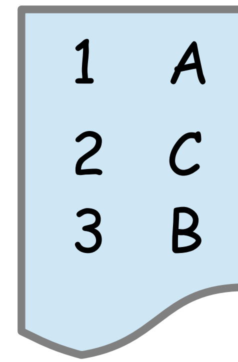
A 0.60
B 0.49
C 0.55

$\xrightarrow{A}$

A&B 0.80
A&C 0.85

$\xrightarrow{C}$

A&C&B 0.9



- Grouping

head cPOS+ mod cPOS + in-between punct #	0.49
in-between cPOS	0.59
⋮	
head POS + mod POS + in-between conj #	0.71
head POS + mod POS + in-between POS + dist	0.72
⋮	
head token + mod cPOS + dist	0.80
⋮	

Feature Template Ranking

- Forward selection

A 0.60
B 0.49
C 0.55

$\xrightarrow{A}$

A&B 0.80
A&C 0.85

$\xrightarrow{C}$

A&C&B 0.9

1	A
2	C
3	B

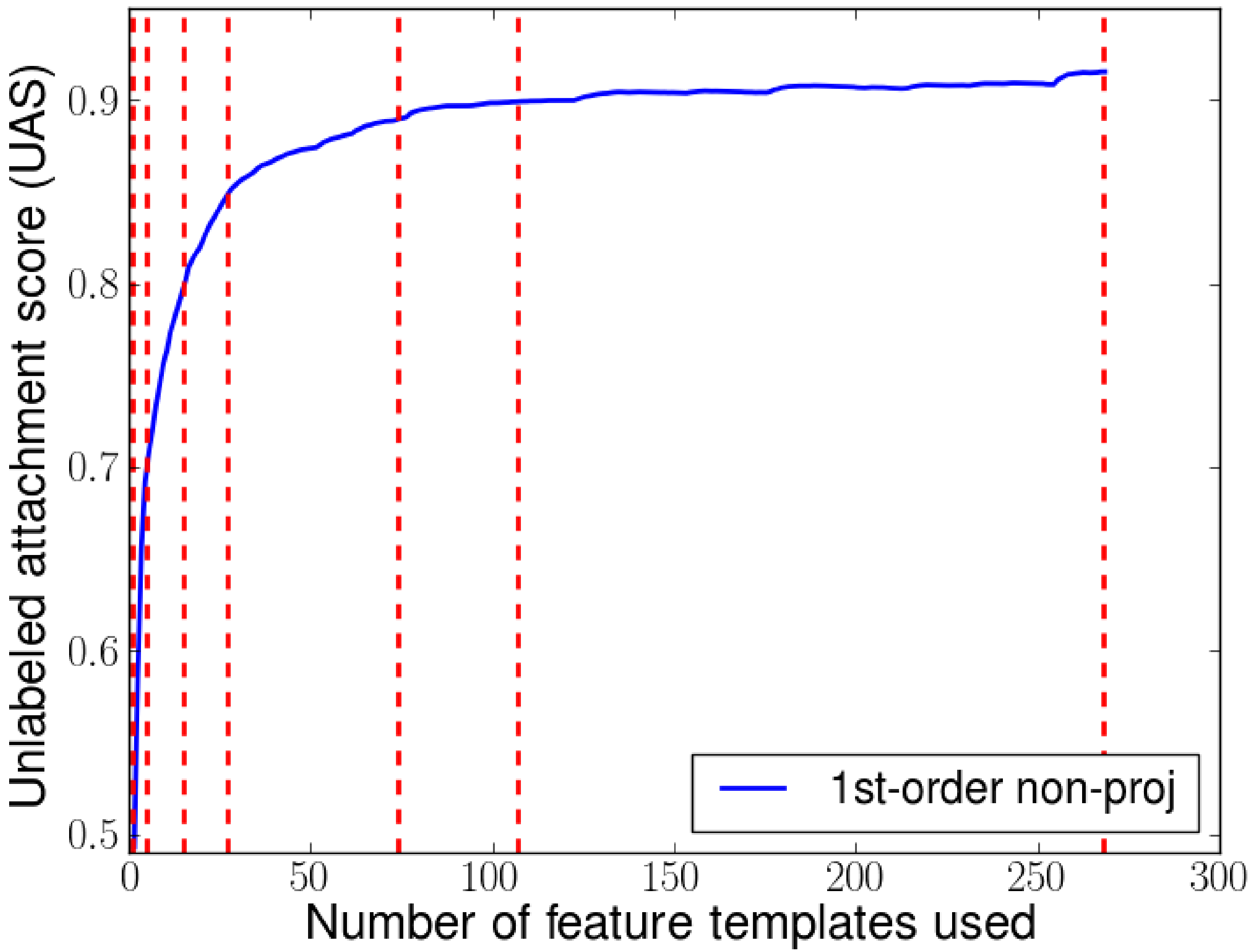
- Grouping

head cPOS+ mod cPOS + in-between punct #	0.49
in-between cPOS	0.59
⋮	
head POS + mod POS + in-between conj #	0.71
head POS + mod POS + in-between POS + dist	0.72
⋮	
head token + mod cPOS + dist	0.80
⋮	

+ ~0.1

+ ~0.1

Partition Feature List Into Groups



How to pick the winners?

How to pick the winners?

- Learn a classifier

How to pick the winners?

- Learn a classifier
- Features
  - Currently added parsing features
  - Meta-features -- confidence of a prediction

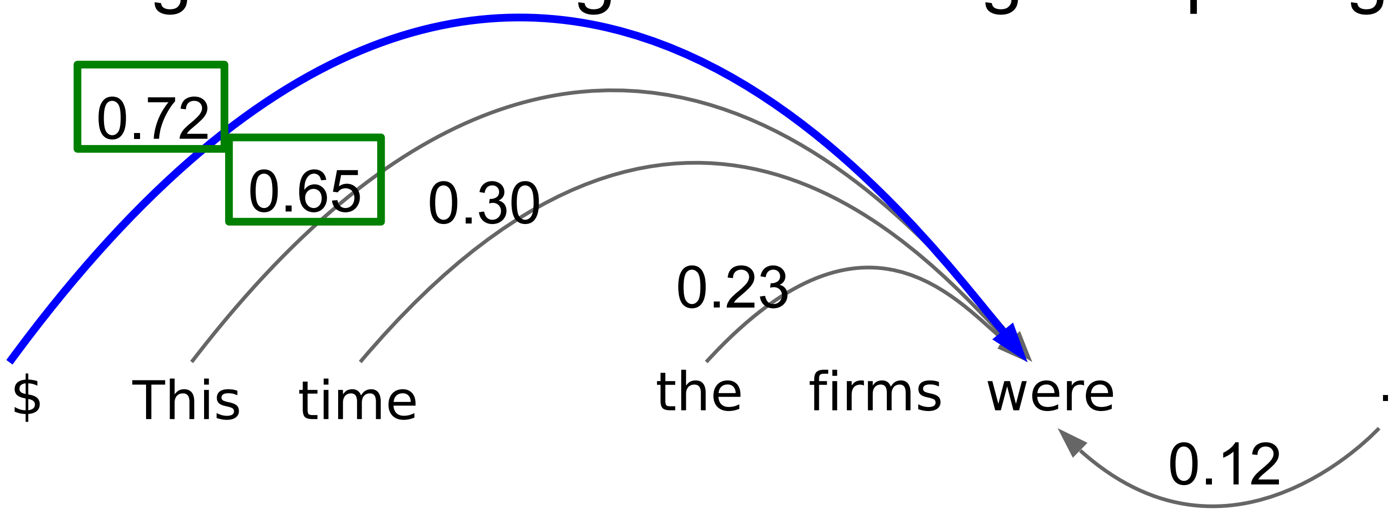
How to pick the winners?

- Learn a classifier
- Features
  - Currently added parsing features
  - Meta-features -- confidence of a prediction
- Training examples
  - Input: each blue edge in current 1-best tree
  - Output: is the edge in the gold tree? If so, we want it to win!

Classifier Features

- Currently added parsing features
- Meta-features
  - the firms : …, 0.5, $\boxed{0.8, 0.85}$

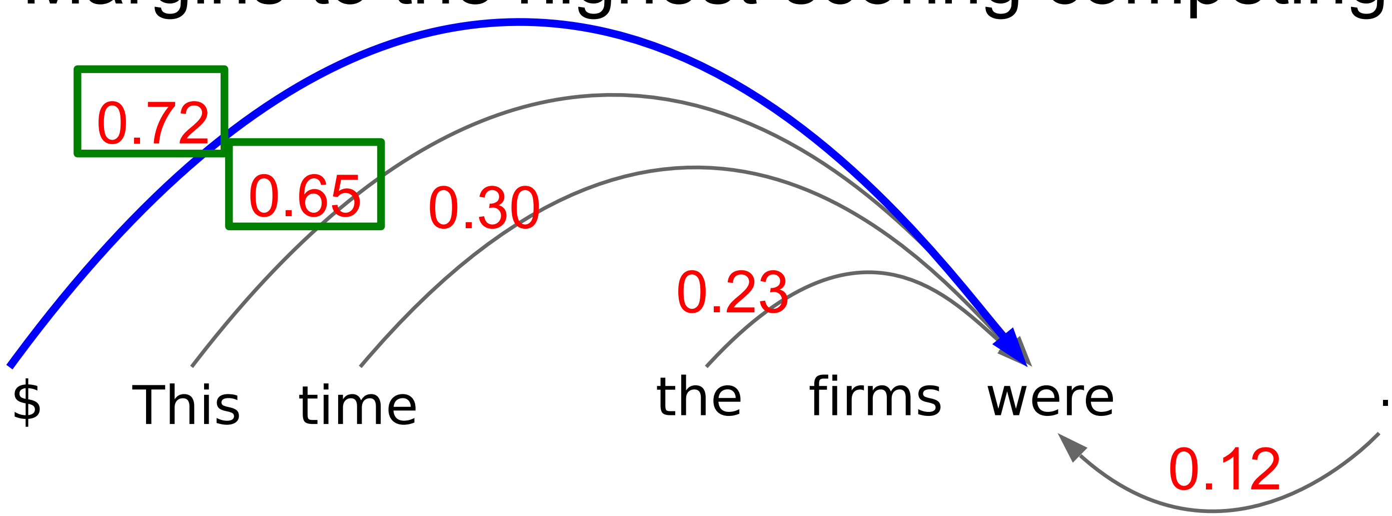
    (scores are normalized by the sigmoid function)
  - Margins to the highest-scoring competing edge



  - Index of the next feature group

Classifier Features

- Currently added parsing features
- Meta-features
  - the firms : …, 0.5, $\boxed{0.8,\ 0.85}$
    (scores are normalized by the sigmoid function)
  - Margins to the highest-scoring competing edge



  - Index of the next feature group

Classifier Features

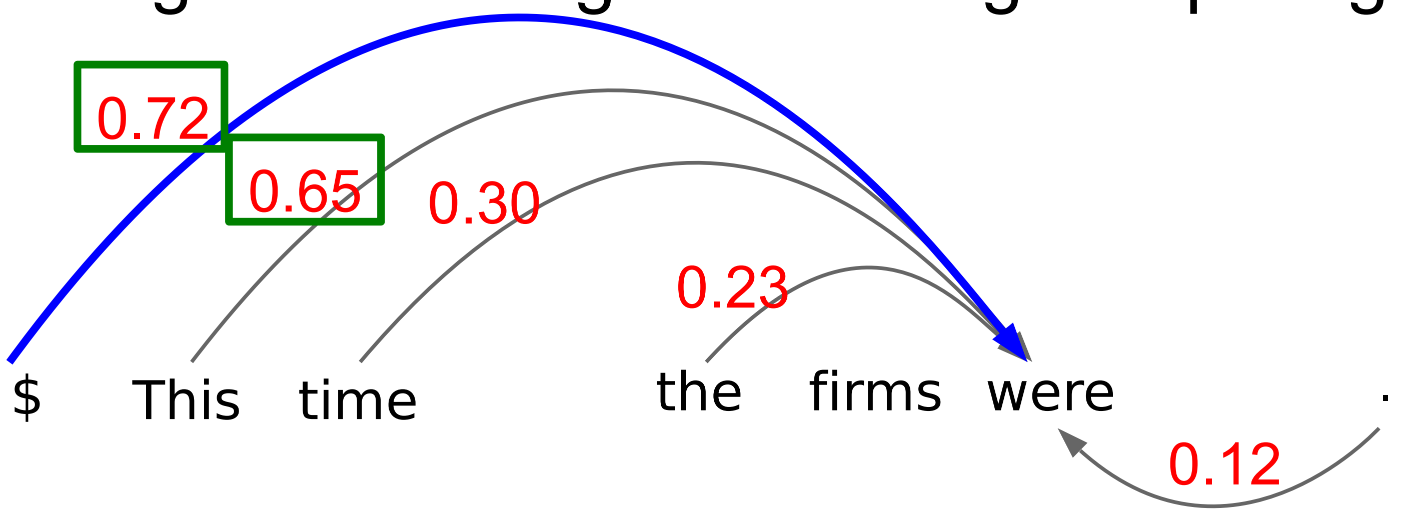
- Currently added parsing features
- Meta-features

Dynamic Features

  - the firms : …, 0.5, $\boxed{0.8, 0.85}$

    (scores are normalized by the sigmoid function)

  - Margins to the highest-scoring competing edge



  - Index of the next feature group

How To Train With Dynamic Features

- Training examples are not fixed in advance!
- Winners/losers from stages < k affect:
  - *Set* of edges to classify at stage k
  - The dynamic *features* of those edges at stage k
- Bad decisions can cause future errors

How To Train With Dynamic Features

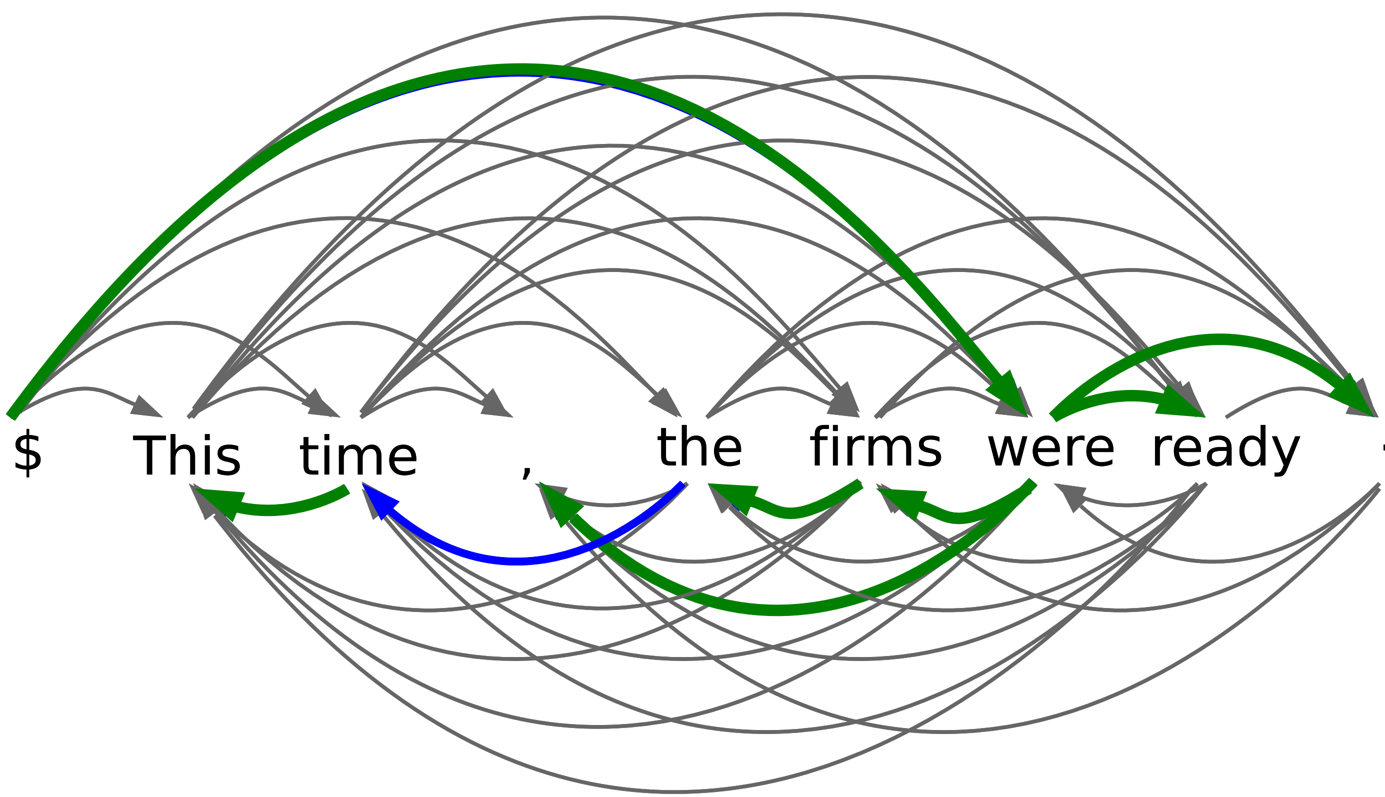
- Training examples are not fixed in advance!!
- Winners/losers from stages < k affect:
  - *Set* of edges to classify at stage k
  - The dynamic *features* of those edges at stage k
- Bad decisions can cause future errors

Reinforcement / Imitation Learning

- Dataset Aggregation (DAgger) (Ross et al., 2011)
  - Iterates between training and running a model
  - Learns to recover from past mistakes

Upper Bound of Our Performance

- "Labels"
  - Gold edges always win
  - 96.47% UAS with 2.9% first-order features



How To Train Our Parser

1. Train parsers (non-projective, projective) *using all features*

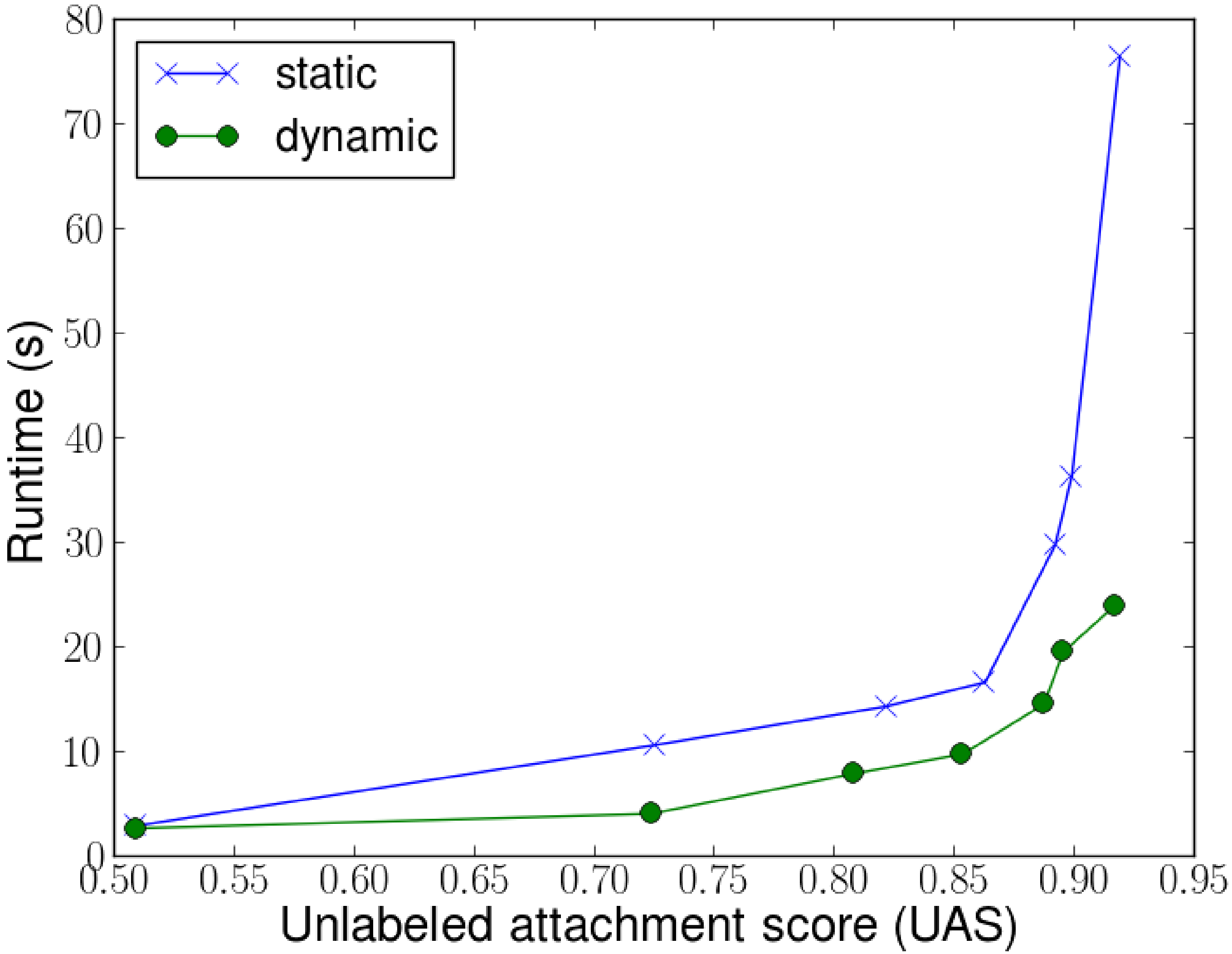
2. Rank and group feature templates

3. Iteratively train a classifier to decide winners/losers

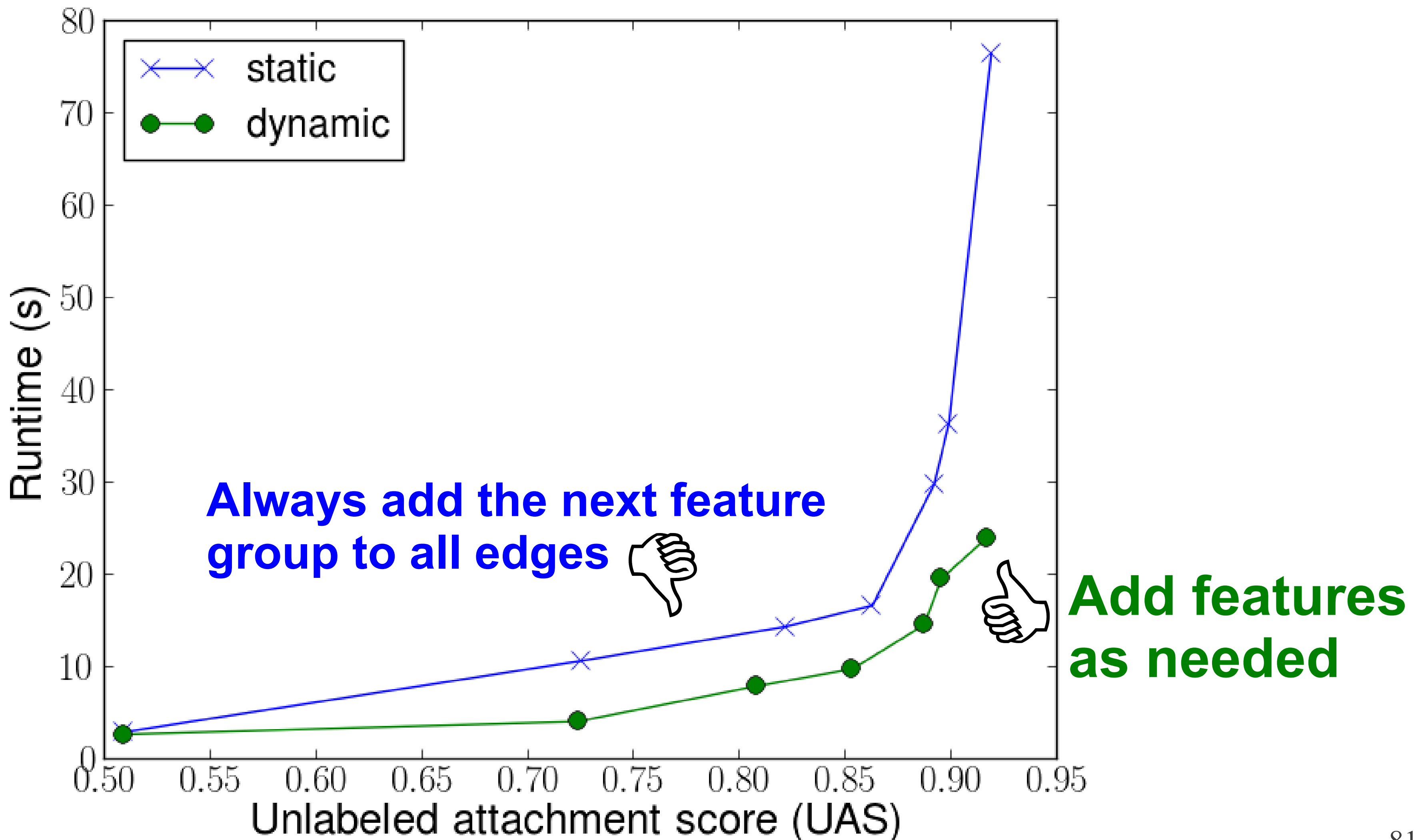
Experiment

- Data
  - Penn Treebank: English
  - CoNLL-X: Bulgarian, Chinese, German, Japanese, Portuguese, Swedish
- Parser
  - MSTParser (McDonald et al., 2006)
- Dynamically-trained Classifier
  - LibLinear (Fan et al., 2008)

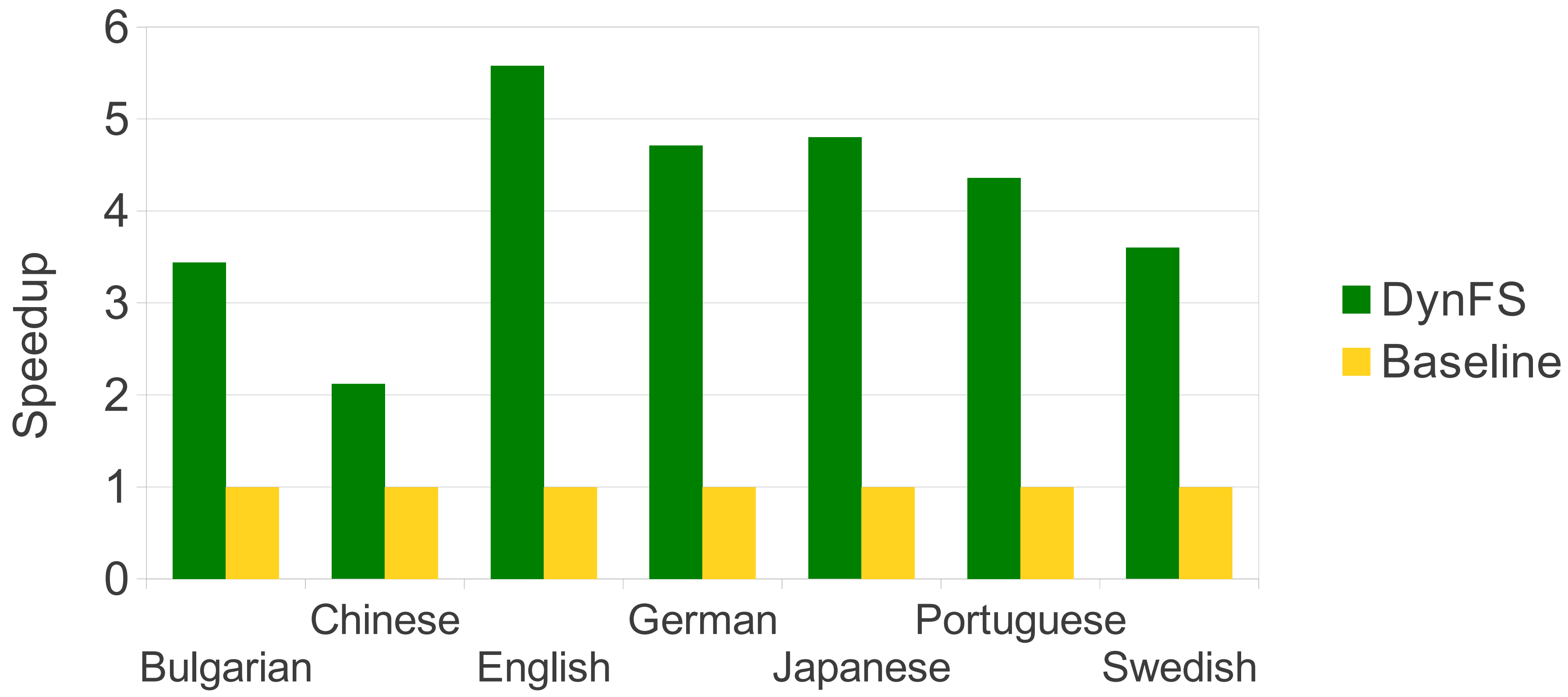
Dynamic Feature Selection Beats Static Forward Selection



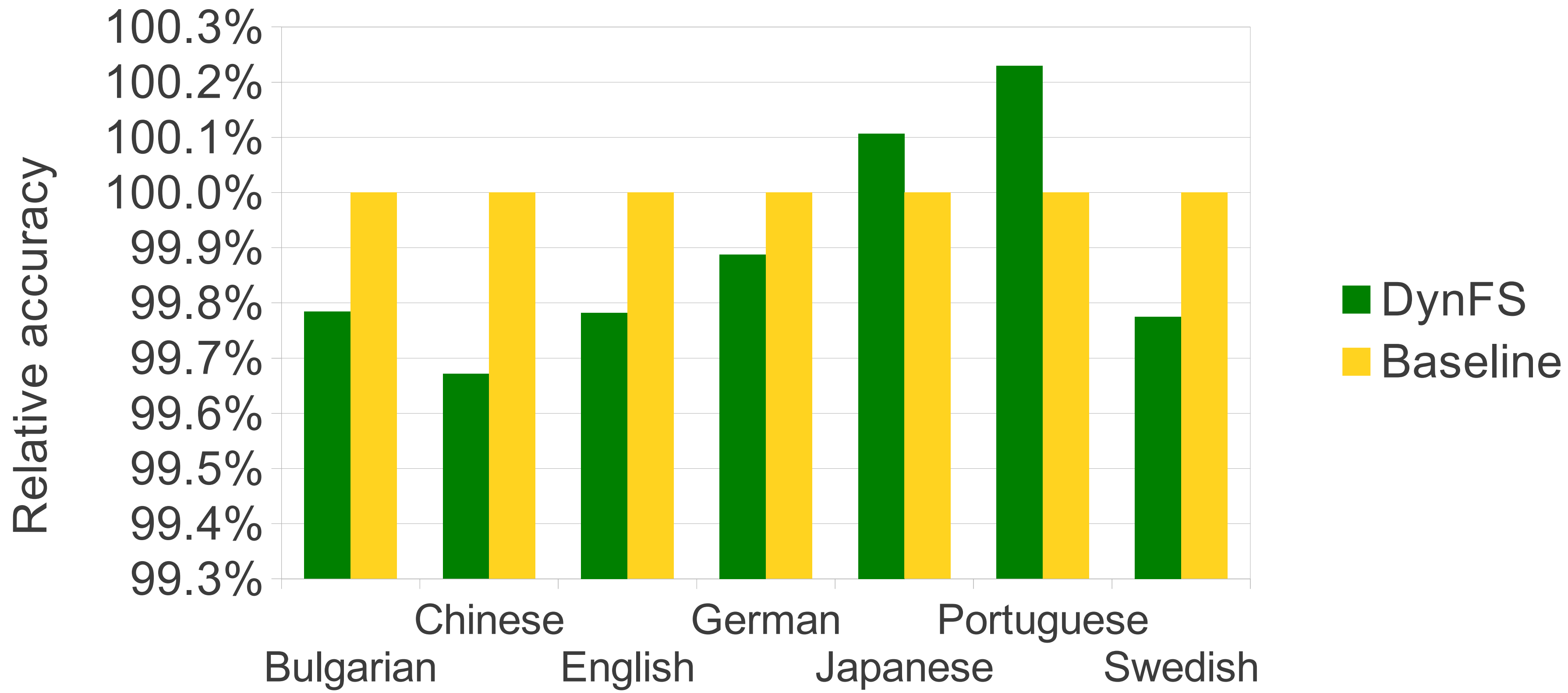
Dynamic Feature Selection Beats Static Forward Selection



Experiment: 1st-order
2x to 6x speedup

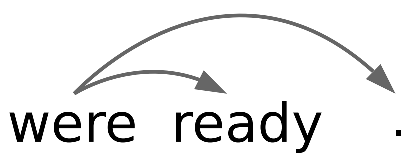


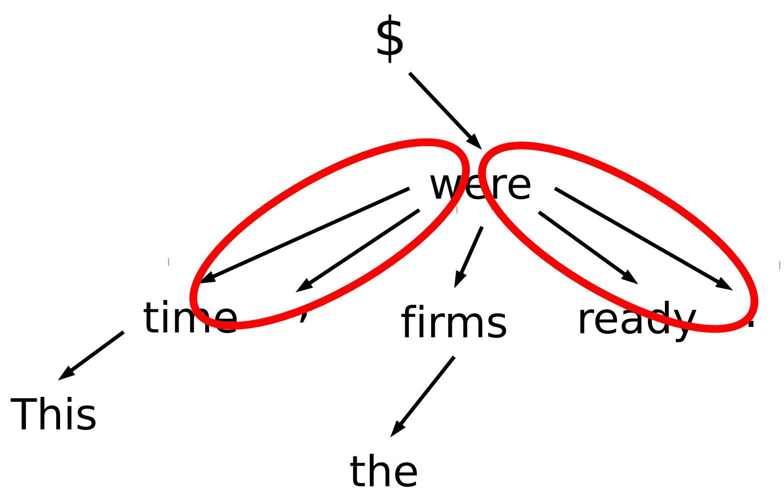
Experiment: 1st-order ~0.2% loss in accuracy



$$relative\ accuracy = \frac{\text{accuracy of the pruning parser}}{\text{accuracy of the full parser}}$$

Second-order Dependency Parsing





- Features depend on the siblings as well

- First-order:
  - $O(n^2)$ substructure to score

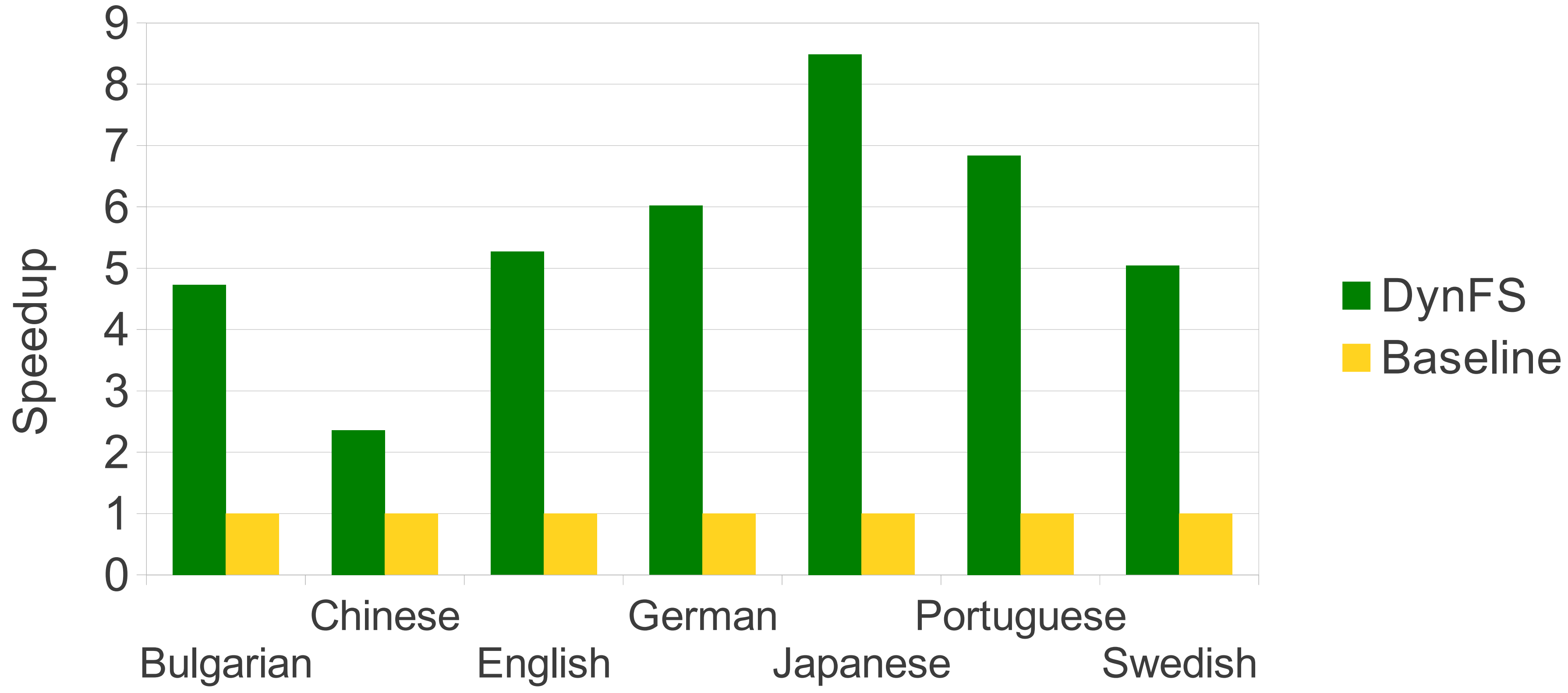
- Second-order:
  - $O(n^3)$ substructure to score

  ~380 feature templates

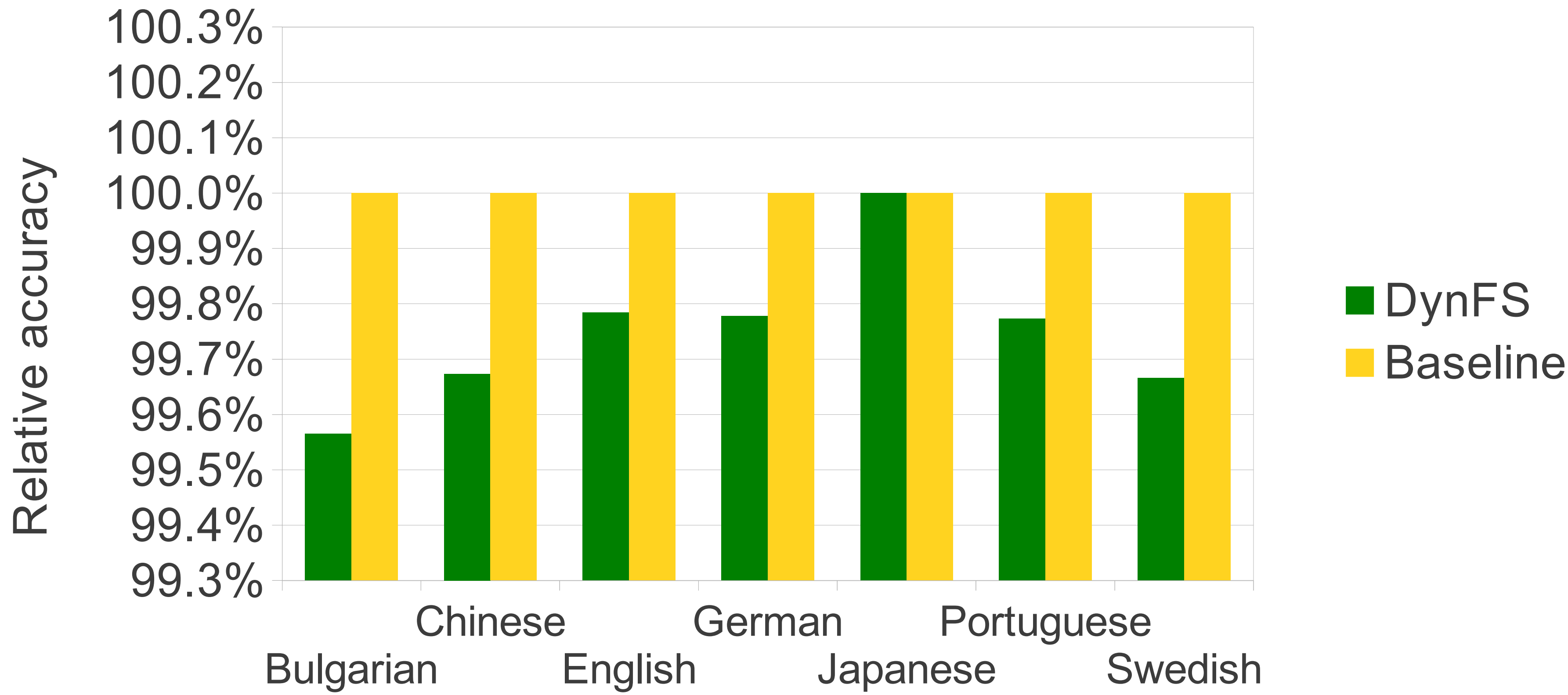
  ~96M features

- Decoding: still $O(n^3)$

Experiment: 2nd-order
2x to 8x speedup



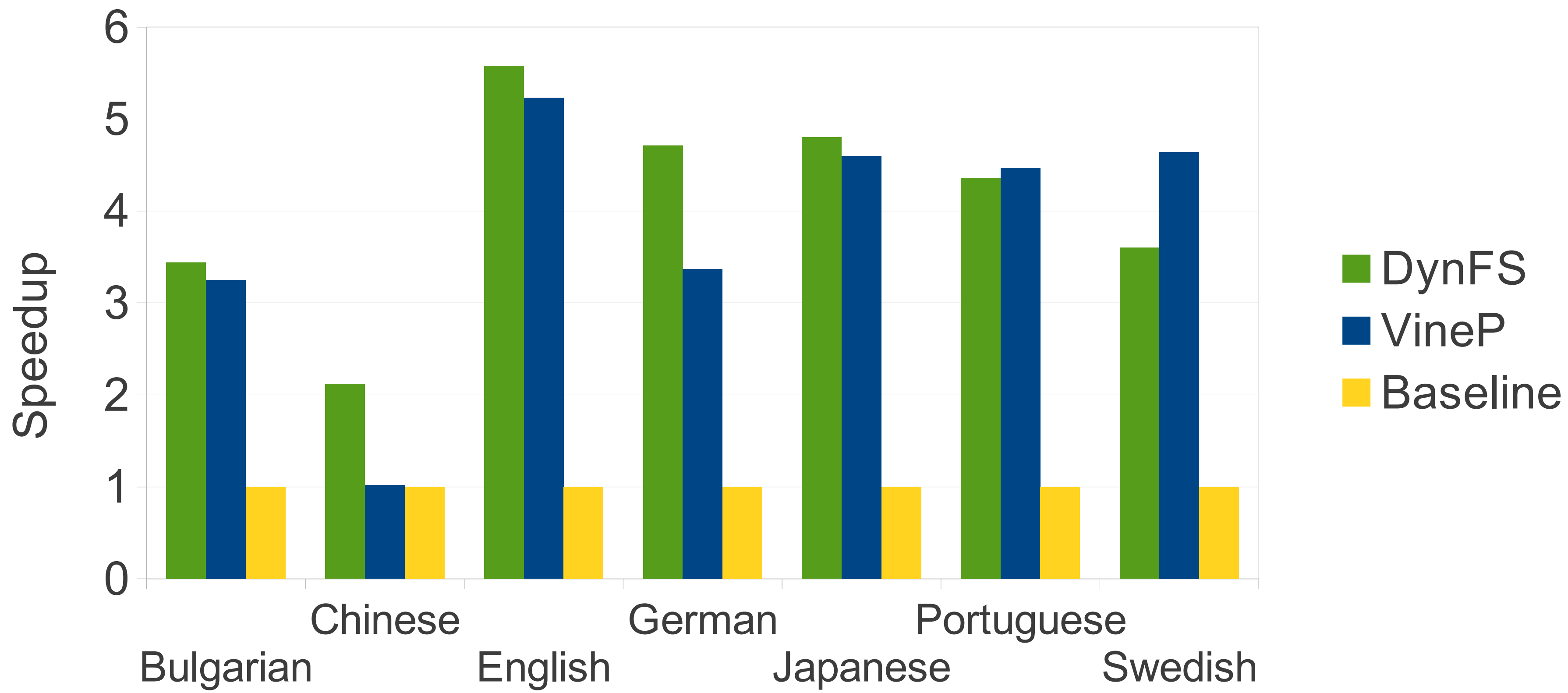
Experiment: 2nd-order ~0.3% loss in accuracy



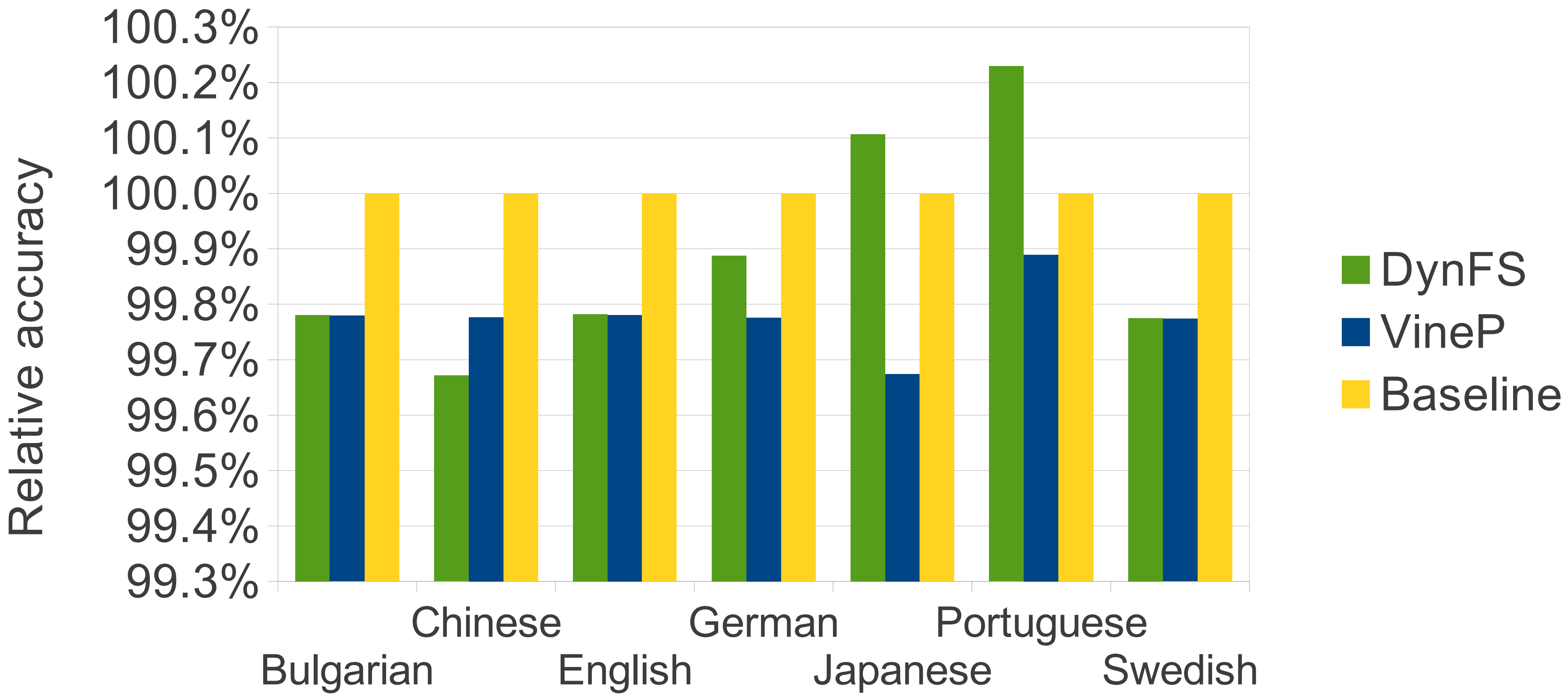
Ours vs Vine Pruning (Rush and Petrov, 2012)

- **Vine pruning**: a very fast parser that speeds up using orthogonal techniques
  - Start with short edges (*fully* scored)
  - Add long edges in if needed
- **Ours**
  - Start with all edges (*partially* scored)
  - Quickly remove unneeded edges
- Could be combined for further speedup!

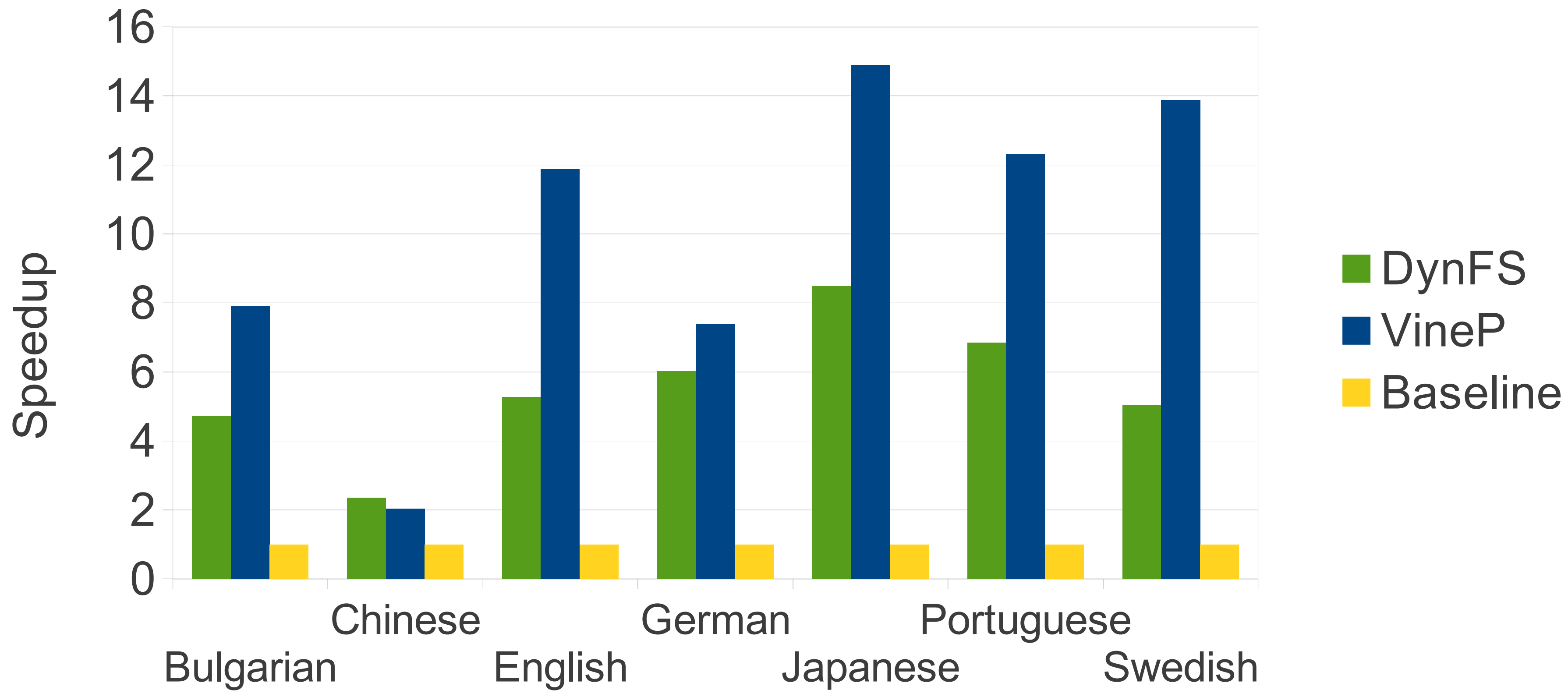
VS Vine Pruning: 1st-order comparable performance



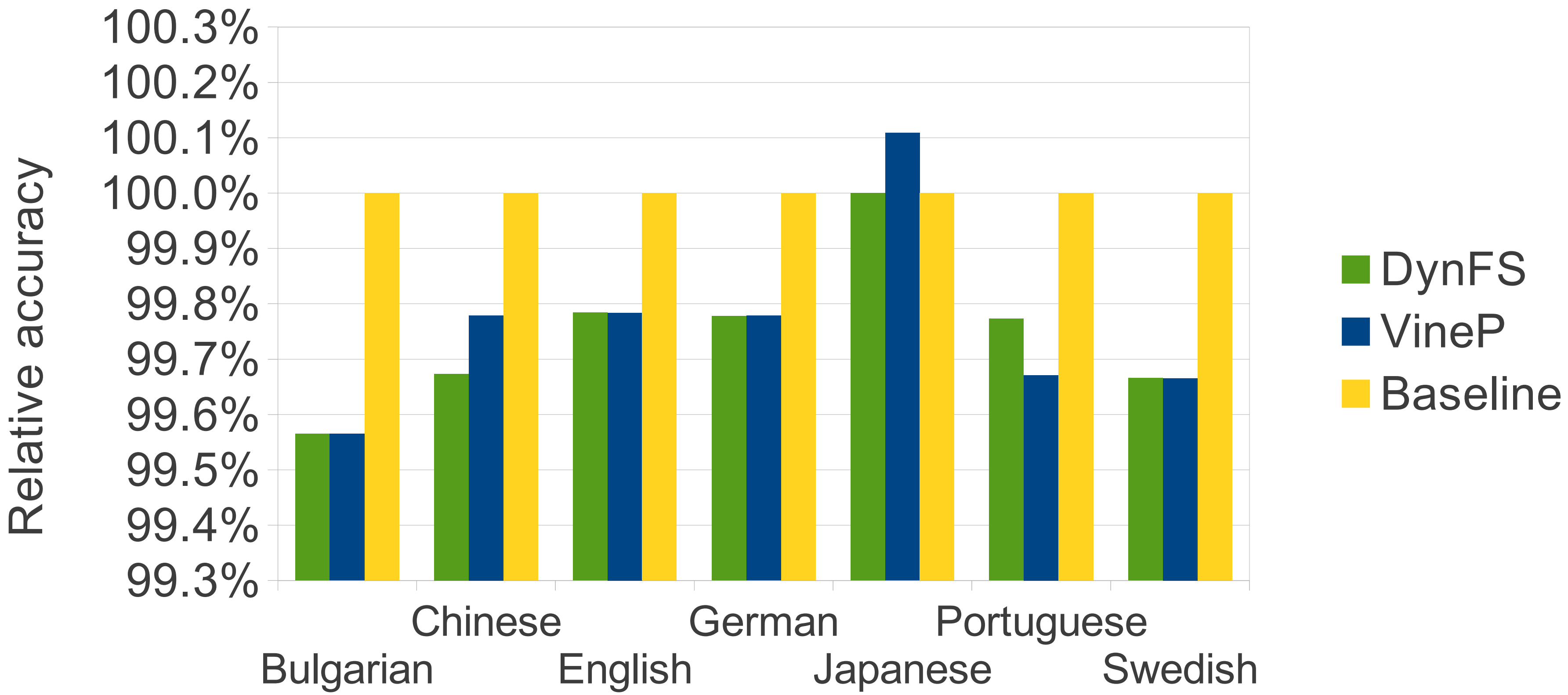
VS Vine Pruning: 1st-order



VS Vine Pruning: 2nd-order



VS Vine Pruning: 2nd-order



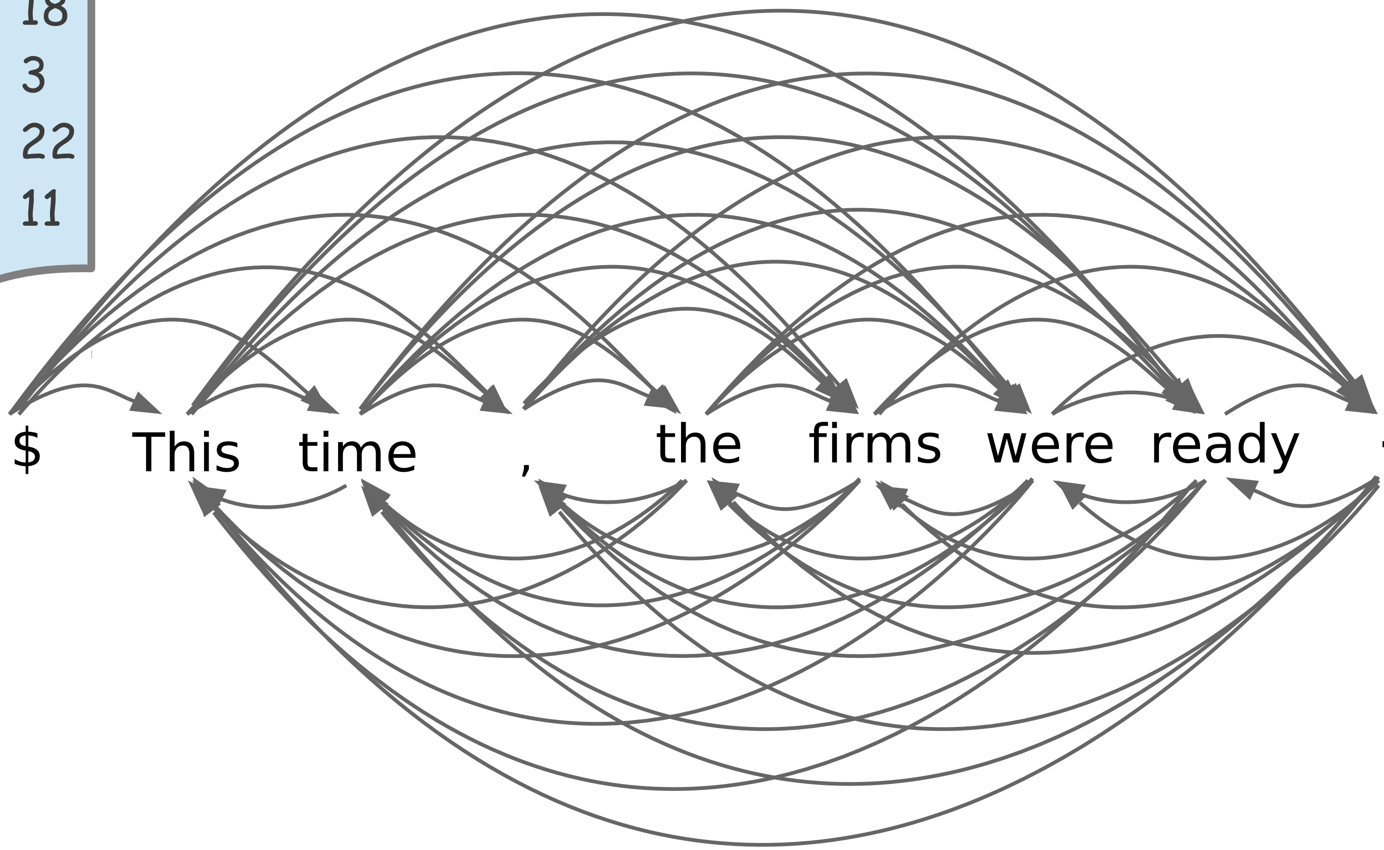
Conclusion

- **Feature computation** is expensive in structured prediction
- Commitment should be made **dynamically**
- Early commitment to edges reduce both searching and scoring time
- Can be used in other feature-rich models for structured prediction

Backup Slides

Static dictionary pruning (Rush and Petrov, 2012)

VB → CD: 18
VB ← CD: 3
NN → VBG: 22
NN ← VBG: 11
...



Reinforcement Learning 101

- Markov Decision Process (MDP)
  - State: all the information helping us to make decisions
  - Action: things we choose to do
  - Reward: criteria for evaluating actions
  - Policy: the “brain” that makes the decision
- Goal
  - Maximize the expected future reward

Policy Learning

- Markov Decision Process (MDP)

  π (the firms + context) = add / lock

  - reward = accuracy + λ·speed
- Reinforcement learning
  - Delayed reward
  - Long time to converge
- Imitation learning
  - Mimic the oracle
  - Reduced to supervised classification problem

Imitation Learning

- Oracle
  - (near) optimal performance
  - generate target action in any given state

π (the firms + context) = *lock*

π (time , the + context) = *add*

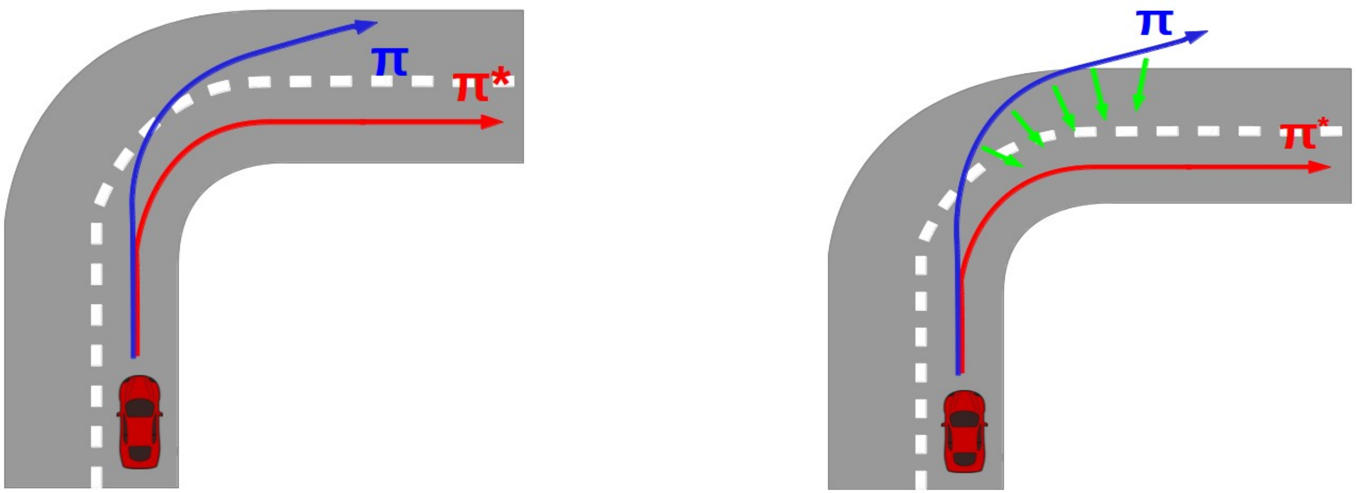
...

$$\{\psi(s), \pi^*(s)\}$$

Binary classifier

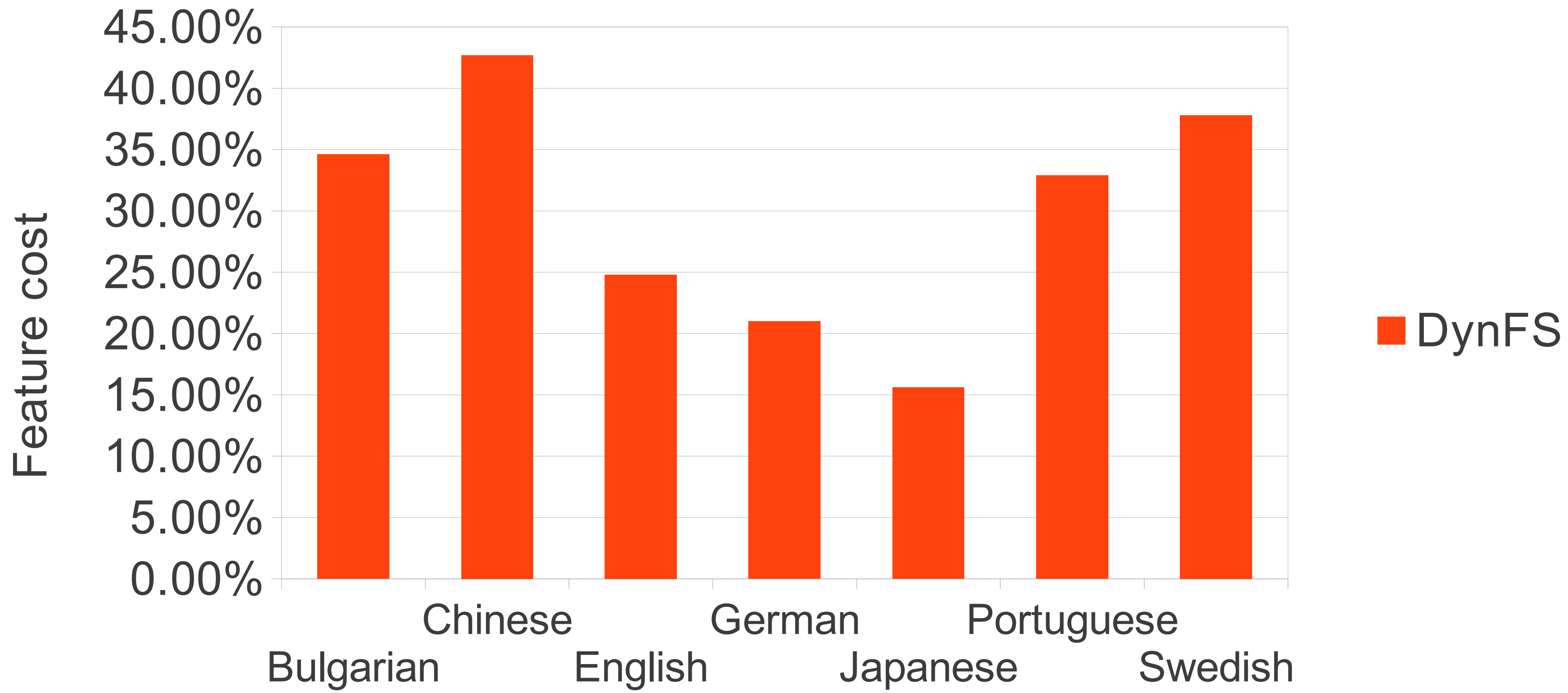
Dataset Aggregation (DAgger)

- Collect data from the oracle only
  - Different distribution at training and test time
- Iterative policy training



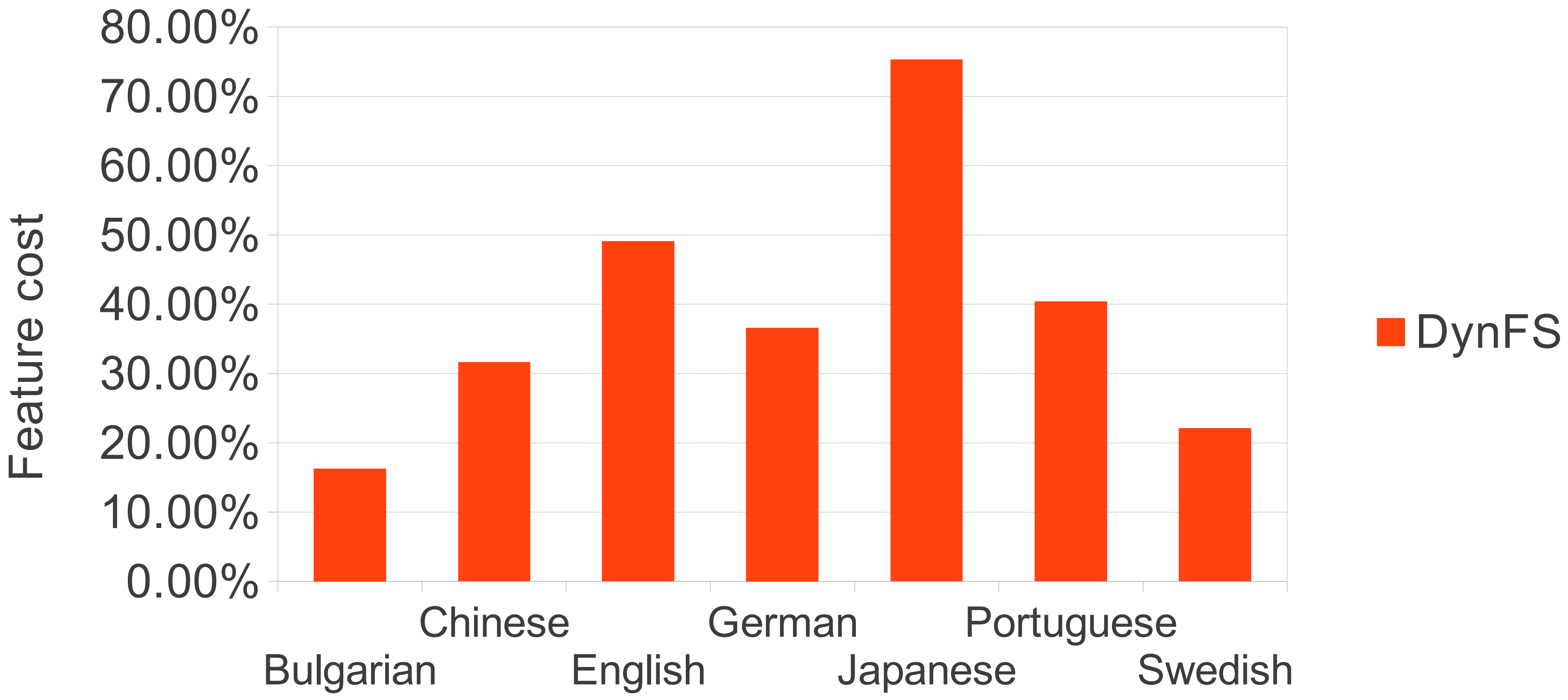
- Correct the learner's mistake
- Obtain a policy performs well under its own policy distribution

Experiment (1st-order)



$$cost = \frac{\text{\# feature templates used}}{\text{total \# feature templates on the statically pruned graph}}$$

Experiment (2nd-order)



Second-order Parsing

Second-order Parsing