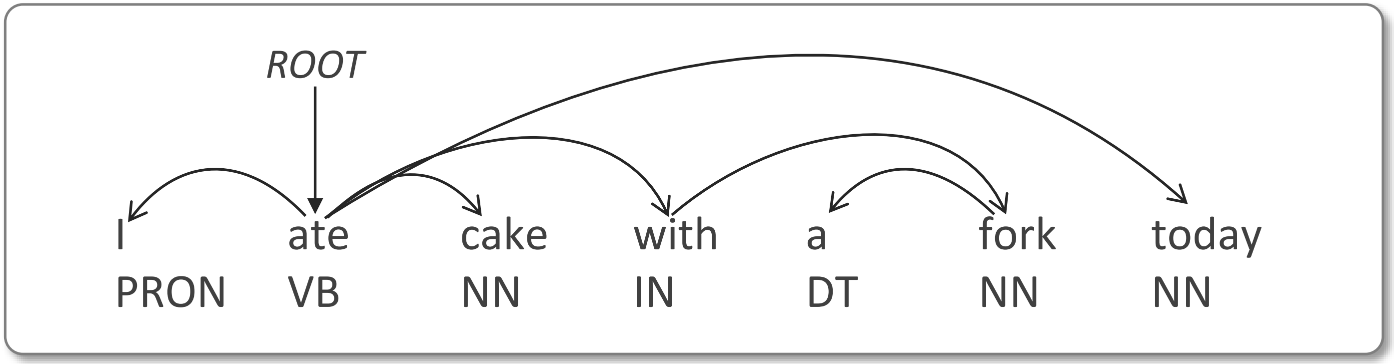
Low-Rank Tensors for Scoring Dependency Structures

Tao Lei

Yu Xin, Yuan Zhang, Regina Barzilay, Tommi Jaakkola

CSAIL, MIT

Dependency Parsing



- Dependency parsing as maximization problem:

$$y^* = \underset{y \in T(x)}{\operatorname{argmax}} \; S(x, y; \theta)$$

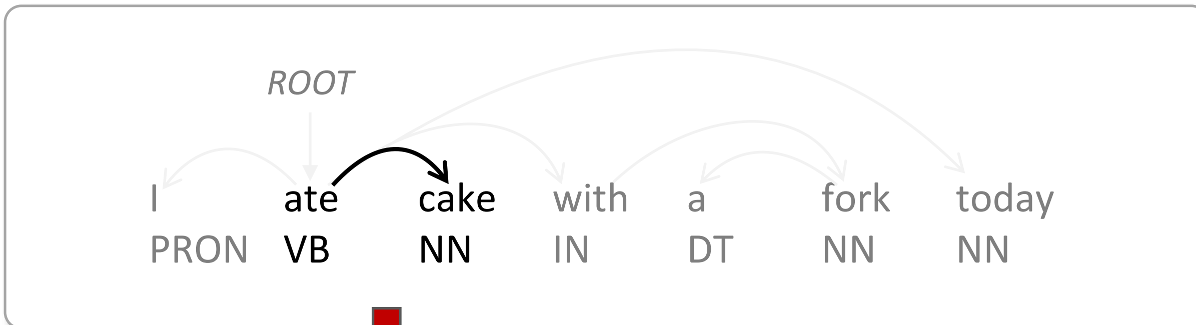
- Key aspects of a parsing system:

1. Accurate scoring function $S(x, y; \theta)$ → *Our Goal*
2. Efficient decoding procedure argmax

Finding Expressive Feature Set

Traditional view:

requires a rich, expressive set of manually-crafted feature templates



High-dim. sparse vector $\phi(x, y) \in \mathbb{R}^L$

1	0	1	1	0	0	0	… …	0

Feature Template:

`head POS, modifier POS and length`

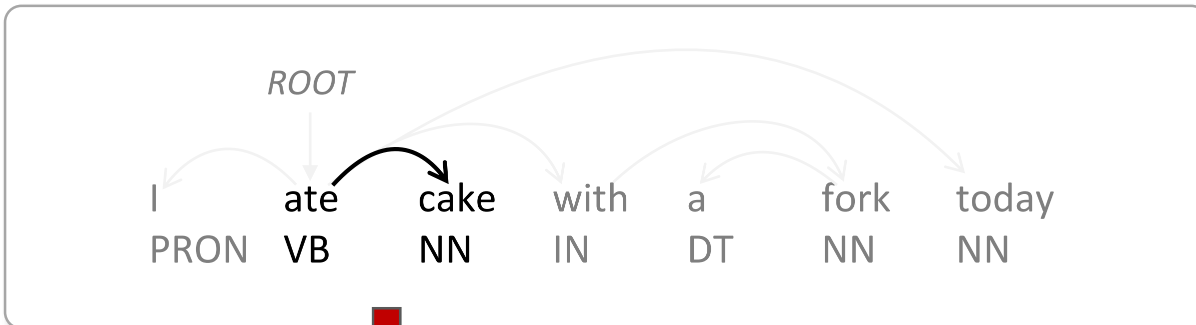
Feature Example:

“VB⊕NN⊕2”

Finding Expressive Feature Set

Traditional view:

requires a rich, expressive set of manually-crafted feature templates



High-dim. sparse vector $\phi(x, y) \in \mathbb{R}^L$

1	0	1	1	0	0	0	… …	0

Feature Template:

`head word and modifier word`

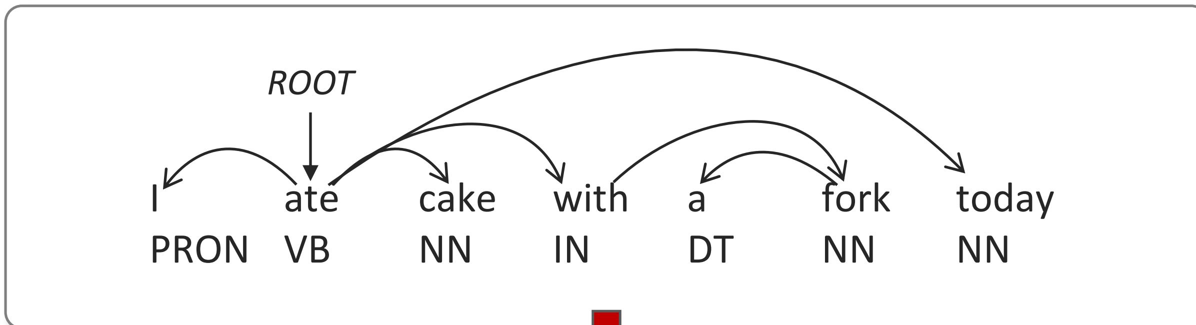
Feature Example:

“ate⊕cake”

Finding Expressive Feature Set

Traditional view:

requires a rich, expressive set of manually-crafted feature templates



High-dim. sparse vector $\phi(x, y) \in \mathbb{R}^L$

1	0	2	1	2	0	0	… …	0

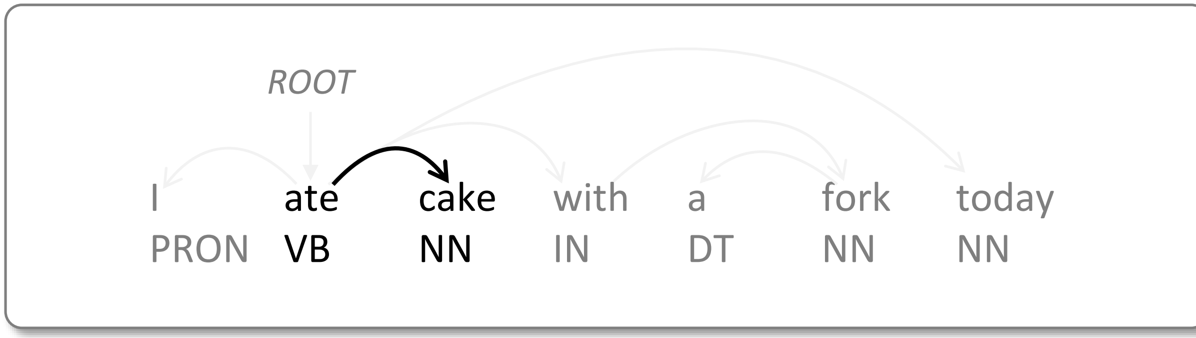
Parameter vector $\theta \in \mathbb{R}^L$ •

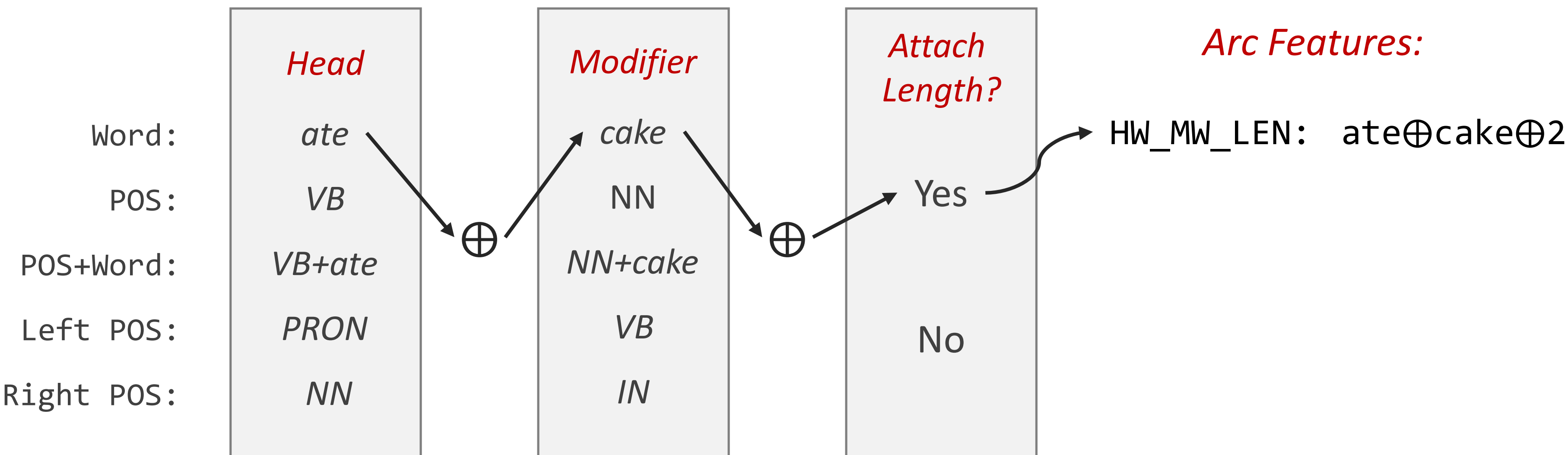
0.1	0.3	2.2	1.1	0	0.1	0.9	… …	0

$$S_\theta(x, y) = \langle \theta, \phi(x, y) \rangle$$

Traditional Scoring Revisited

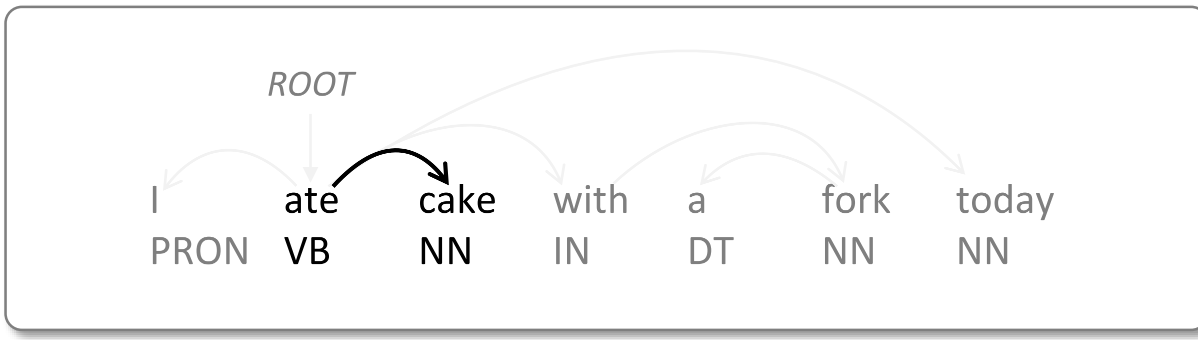
- Features and templates are ***manually-selected*** concatenations of atomic features, in traditional vector-based scoring:

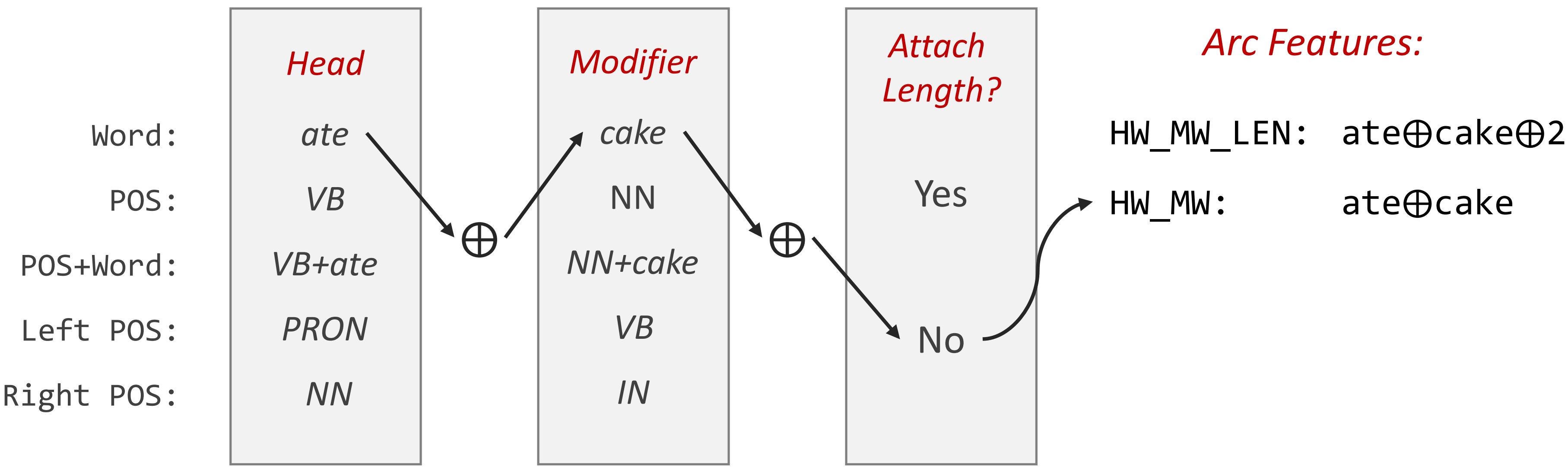




Traditional Scoring Revisited

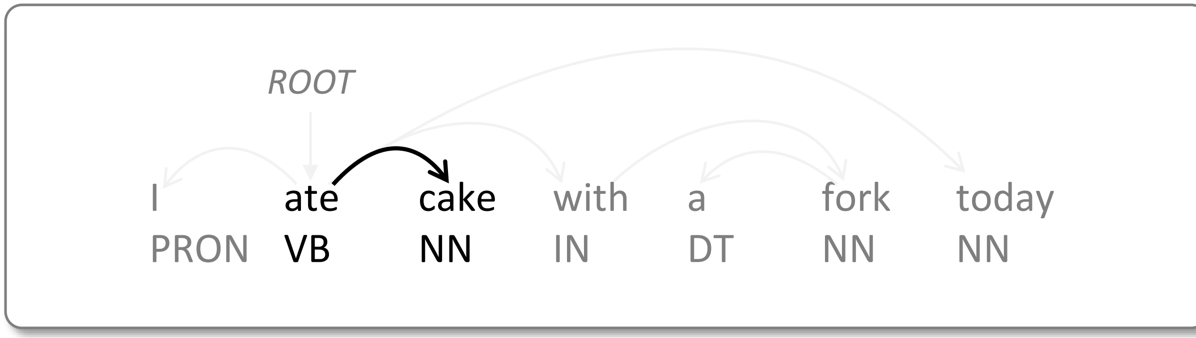
- Features and templates are ***manually-selected*** concatenations of atomic features, in traditional vector-based scoring:

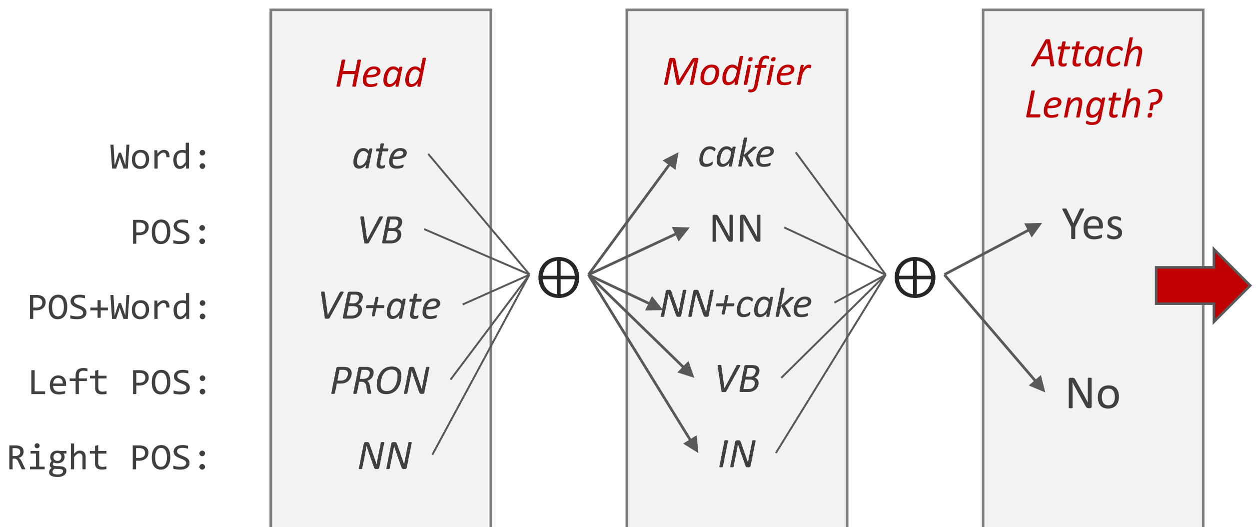




Traditional Scoring Revisited

- Features and templates are ***manually-selected*** concatenations of atomic features, in traditional vector-based scoring:





Arc Features:

```
HW_MW_LEN:  ate⊕cake⊕2
HW_MW:      ate⊕cake
HP_MP_LEN:  VB⊕NN⊕2
HP_MP:      VB⊕NN
 …           …
```

Traditional Scoring Revisited

- **Problem:** very difficult to pick the best subset of concatenations

Too few templates	⇨	Lose performance
Too many templates	⇨	Too many parameters to estimate
Searching the best set?	⇨	Features are correlated Choices are exponential

- **Our approach:** use low-rank tensor (i.e. multi-way array)
  - Capture a whole range of feature combinations
  - Keep the parameter estimation problem in control

Low-Rank Tensor Scoring: Formulation

- Formulate ***ALL possible*** concatenations as a rank-1 tensor

ϕ_h	ϕ_m	$\phi_{h,m}$
atomic head feature vector	atomic modifier feature vector	atomic arc feature vector

Head	*Modifier*	*Attach Length?*
ate	*cake*	Yes
VB	NN	
VB+ate	*NN+cake*	
PRON	*VB*	No
NN	*IN*	

Low-Rank Tensor Scoring: Formulation

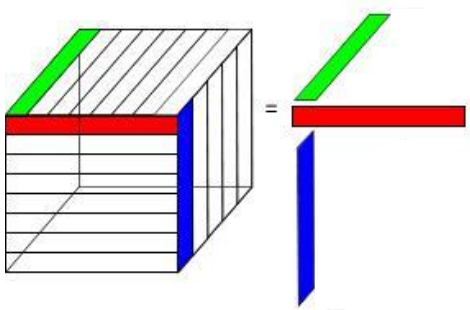
- Formulate ***ALL possible*** concatenations as a rank-1 tensor

$$\phi_h \quad \otimes \quad \phi_m \quad \otimes \quad \phi_{h,m} \quad \in \mathbb{R}^{n \times n \times d}$$

atomic head feature vector

atomic modifier feature vector

atomic arc feature vector



$$(x \otimes y \otimes z)_{ijk} = x_i y_j z_k$$

tensor product

Each entry indicates the occurrence of one feature concatenation

Low-Rank Tensor Scoring: Formulation

- Formulate ***ALL possible*** concatenations as a rank-1 tensor

$$\phi_h \otimes \phi_m \otimes \phi_{h,m} \in \mathbb{R}^{n \times n \times d}$$

atomic head feature vector — atomic modifier feature vector — atomic arc feature vector

- Formulate the parameters as a tensor as well

$\theta \in \mathbb{R}^L$: $S_\theta(h \to m) = \langle \theta, \phi_{h \to m} \rangle$ (vector-based)

$A \in \mathbb{R}^{n \times n \times d}$: $S_{tensor}(h \to m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle$ (tensor-based)

Involves features not in θ

Can be huge. On English: $n \times n \times d \approx 10^{11}$

Low-Rank Tensor Scoring: Formulation

- Formulate ***ALL possible*** concatenations as a rank-1 tensor

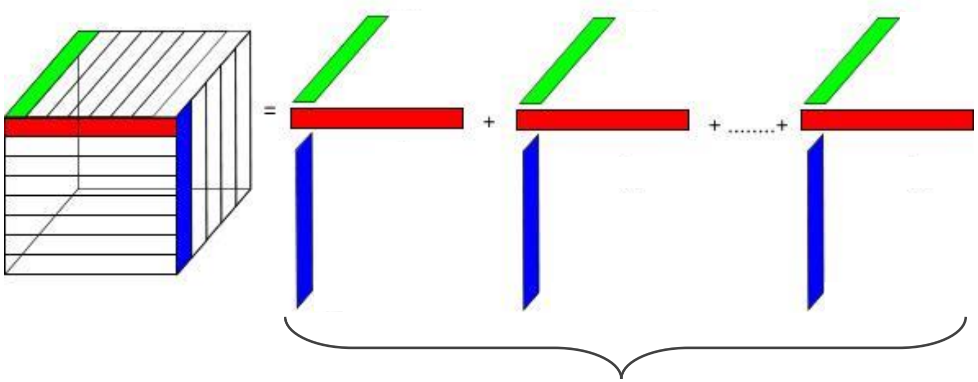
$$\phi_h \otimes \phi_m \otimes \phi_{h,m} \in \mathbb{R}^{n \times n \times d}$$

atomic head feature vector	atomic modifier feature vector	atomic arc feature vector

- Formulate the parameters as a ***low-rank tensor***

$\theta \in \mathbb{R}^L$: $\quad S_\theta(h \to m) = \langle \theta, \phi_{h\to m} \rangle$ (vector-based)

$A \in \mathbb{R}^{n \times n \times d}$: $\quad S_{tensor}(h \to m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle$ (tensor-based)



r rank-1 tensors

$U, V \in \mathbb{R}^{r \times n}, W \in \mathbb{R}^{r \times d}$:

$$A = \sum U(i) \otimes V(i) \otimes W(i)$$

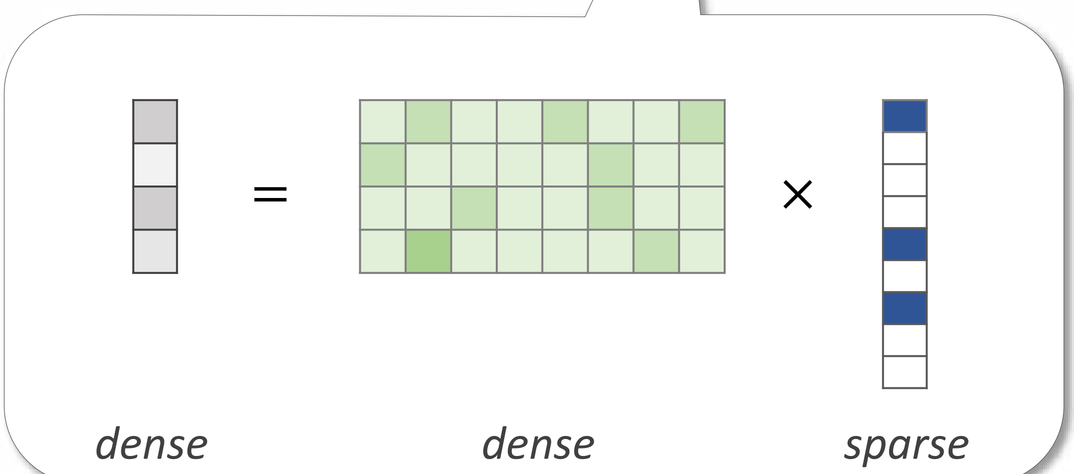
Low-rank tensor

Low-Rank Tensor Scoring: Formulation

$$A = \sum U(i) \otimes V(i) \otimes W(i) \quad \Longrightarrow$$

$$S_{tensor}(h \to m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle$$
$$= \sum_{i=1}^{r} [U\phi_h]_i [V\phi_m]_i [W\phi_{h,m}]_i$$

Dense low-dim representations: $U\phi_h \quad V\phi_m \quad W\phi_{h,m} \quad \in \mathbb{R}^r$



Low-Rank Tensor Scoring: Formulation

$$A = \sum U(i) \otimes V(i) \otimes W(i) \quad \Longrightarrow$$

$$S_{tensor}(h \to m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle$$
$$= \sum_{i=1}^{r} [U\phi_h]_i [V\phi_m]_i [W\phi_{h,m}]_i$$

Dense low-dim representations: $U\phi_h \quad V\phi_m \quad W\phi_{h,m} \quad \in \mathbb{R}^r$

Element-wise products: $[U\phi_h]_i [V\phi_m]_i [W\phi_{h,m}]_i$

Sum over these products: $\sum_{i=1}^{r} [U\phi_h]_i [V\phi_m]_i [W\phi_{h,m}]_i$

Intuition and Explanations

Example: Collaborative Filtering

Approximate user-ratings via low-rank

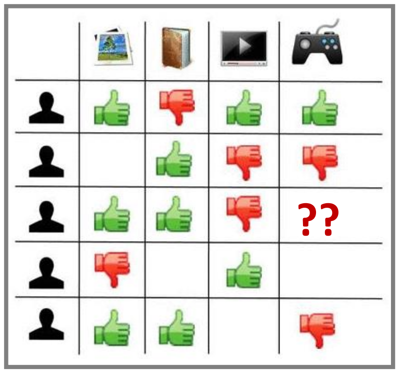
user-rating sparse matrix A

- Ratings not completely independent
- Items share **hidden** properties ("price" and "quality")
- Users have **hidden** preferences over properties

Intuition and Explanations

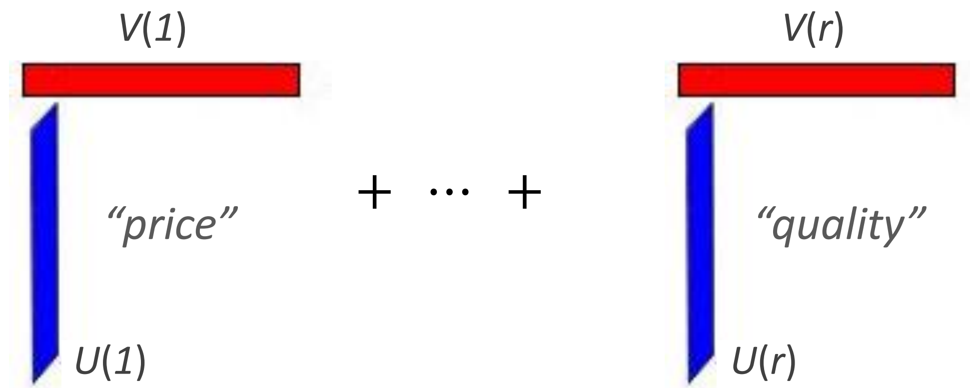
Example: Collaborative Filtering

Approximate user-ratings via low-rank



user-rating sparse matrix A

$\approx$



$$A = U^{\mathrm{T}} V = \sum U(i) \otimes V(i)$$

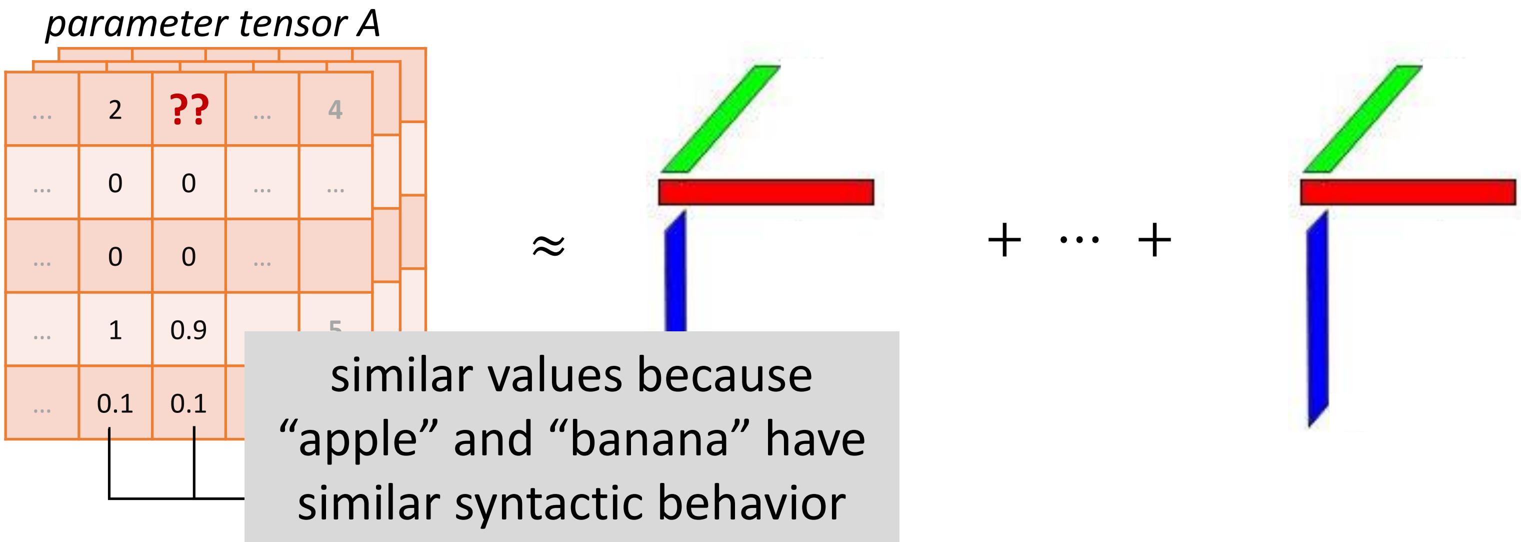
# of parameters: $n \times m$ $(n+m)r$

Intuition: Data and parameters can be approximately characterized by a small number of hidden factors

Intuition and Explanations

Our Case:

Approximate parameters (feature weights) via low-rank



$$A = \sum U(i) \otimes V(i) \otimes W(i)$$

- Hidden properties associated with each word
- Share parameter values via the hidden properties

Low-Rank Tensor Scoring: Summary

- Naturally captures full feature expansion (concatenations)

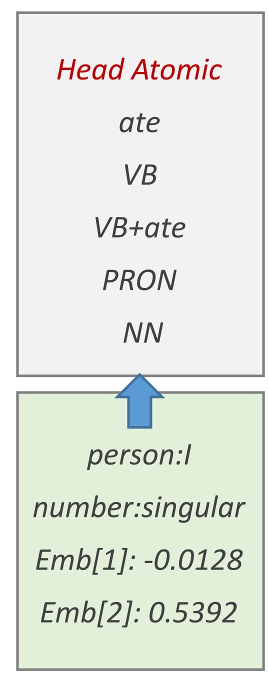
  -- Without mannually specifying a bunch of feature templates

- Controlled feature expansion by low-rank (small *r*)

  -- better feature tuning and optimization

- Easily add and utilize new, auxiliary features

  -- Simply append them as atomic features



Combined Scoring

- Combining traditional and tensor scoring in $S_\gamma(x,y)$:

$$\gamma \cdot \boxed{S_\theta(x,y)} + (1-\gamma) \cdot \boxed{S_{tensor}(x,y)} \qquad \gamma \in [0,1]$$

Set of manual selected features

Full feature expansion controlled by low-rank

Similar "sparse+low-rank" idea for matrix decomposition:
Tao and Yuan, 2011; Zhou and Tao, 2011;
Waters et al., 2011; Chandrasekaran et al., 2011

- Final maximization problem given parameters θ, U, V, W:

$$y^* = \underset{y \in T(x)}{\operatorname{argmax}} \; S_\gamma(x, y; \theta, U, V, W)$$

Learning Problem

- Given training set $\mathrm{D} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{N}$
- Search for parameter values that score the gold trees higher than others:

$$\forall y \in \textbf{Tree}\,(x_i): \qquad S(\hat{x}_i, \widehat{y_i}) \geq S(\hat{x}_i, y) + |\widehat{y_i} - y| - \xi_i$$

Non-negative loss

unsatisfied constraints are penalized against

- The training objective:

$$\min_{\theta, U, V, W, \xi_i \geq 0} \; C \underbrace{\sum_i \xi_i}_{\text{Training loss}} + \underbrace{\|U\|^2 + \|V\|^2 + \|W\|^2 + \|\theta\|^2}_{\text{Regularization}}$$

Calculating the loss requires to solve the expensive maximization problem;

Following common practices, adopt online learning framework.

Online Learning

- Use passive-aggressive algorithm (Crammer et al. 2006) ***tailored*** to our tensor setting

(i) Iterate over training samples successively:

$$(\hat{x}_1, \hat{y}_1) \quad \cdots \longrightarrow (\hat{x}_i, \hat{y}_i) \longrightarrow \cdots \longrightarrow \cdots \quad (\hat{x}_N, \hat{y}_N)$$

revise parameter values
for i-th training sample

$\sum_{i=1}^{r} [U\phi_h]_i [V\phi_m]_i [W\phi_{h,m}]_i$
is not linear nor convex

(ii) choose to update a pair of sets (θ, U), (θ, V) or (θ, W):

Increments: $\theta^{(t+1)} = \theta^{(t)} + \Delta\theta, \qquad U^{(t+1)} = U^{(t)} + \Delta U$

Sub-problem: $\min_{\Delta\theta, \Delta U} \frac{1}{2}\|\Delta\theta\|^2 + \frac{1}{2}\|\Delta U\|^2 + C\xi_i$

Efficient parameter update via closed-form solution

Experiment Setup

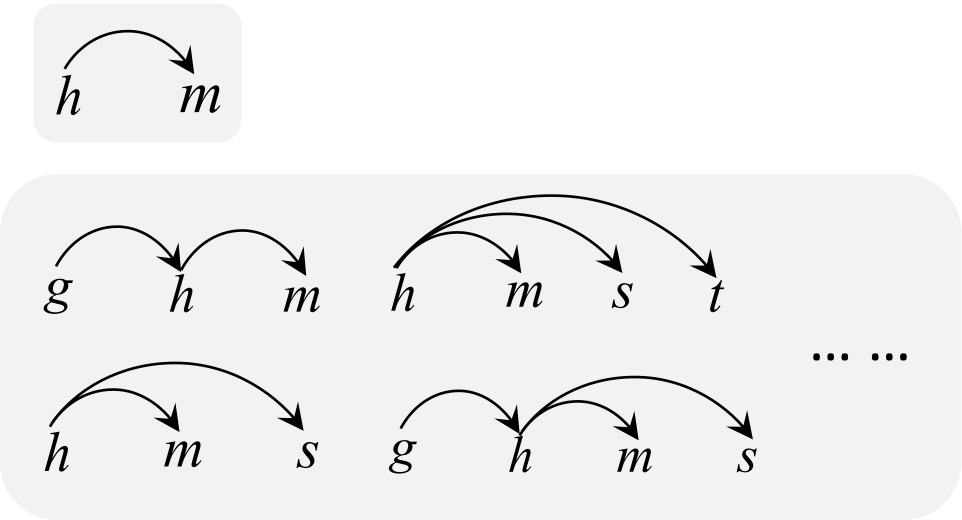
Datasets

- 14 languages in CoNLL 2006 & 2008 shared tasks

Features

- Only 16 atomic word features for tensor
- Combine with 1st-order (single arc) and up to 3rd-order (three arcs) features used in MST/Turbo parsers

Unigram features:		
form	form-p	form-n
lemma	lemma-p	lemma-n
pos	pos-p	pos-n
morph	bias	
Bigram features:		
pos-p, pos		
pos, pos-n		
pos, lemma		
morph, lemma		
Trigram features:		
pos-p, pos, pos-n		



Experiment Setup

Datasets

- 14 languages in CoNLL 2006 & 2008 shared tasks

Features

- Only 16 atomic word features for tensor
- Combine with 1^{st}-order (single arc) and up to 3^{rd}-order (three arcs) features used in MST/Turbo parsers

Implementation

- By default, rank of the tensor r=50
- 3-way tensor captures only 1^{st}-order arc-based features
- Train 10 iterations for all 14 languages

Baselines and Evaluation Measure

MST and Turbo Parsers

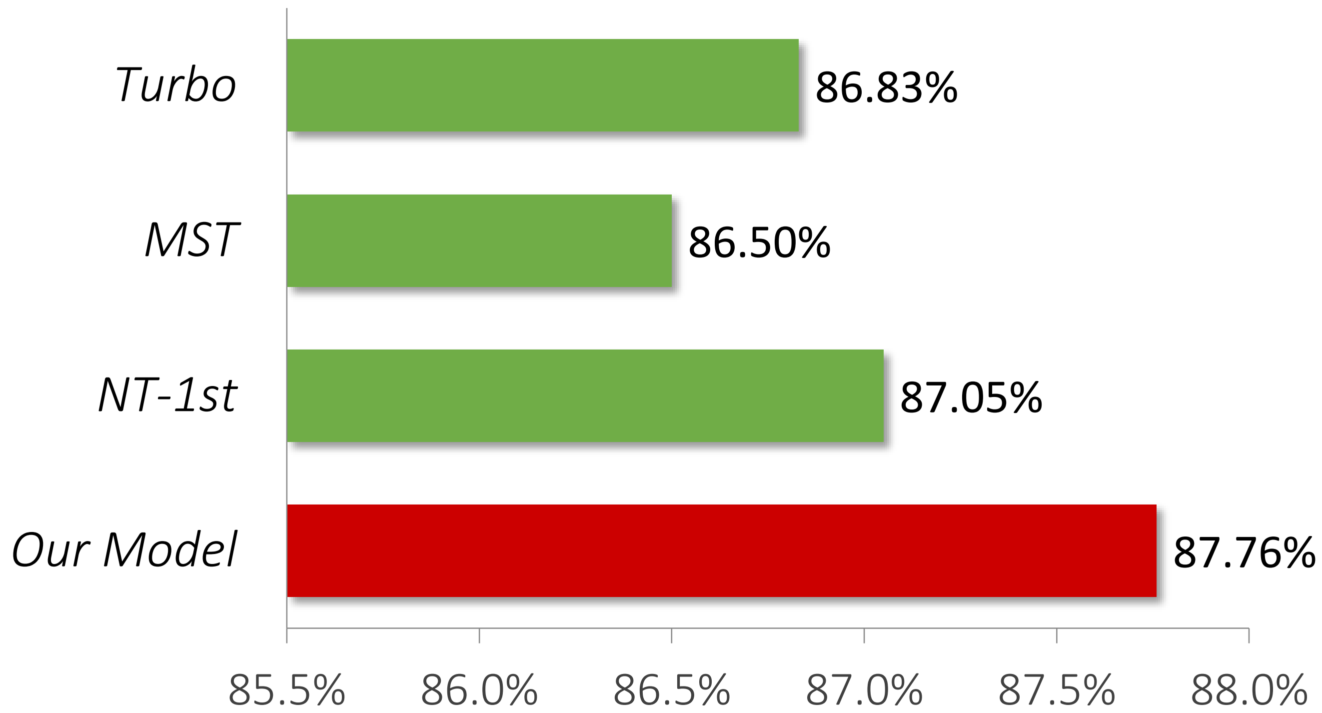
representative graph-based parsers;
use similar set of features

NT-1st and NT-3rd

variants of our model by removing the tensor component;
reimplementation of MST and Turbo Parser features

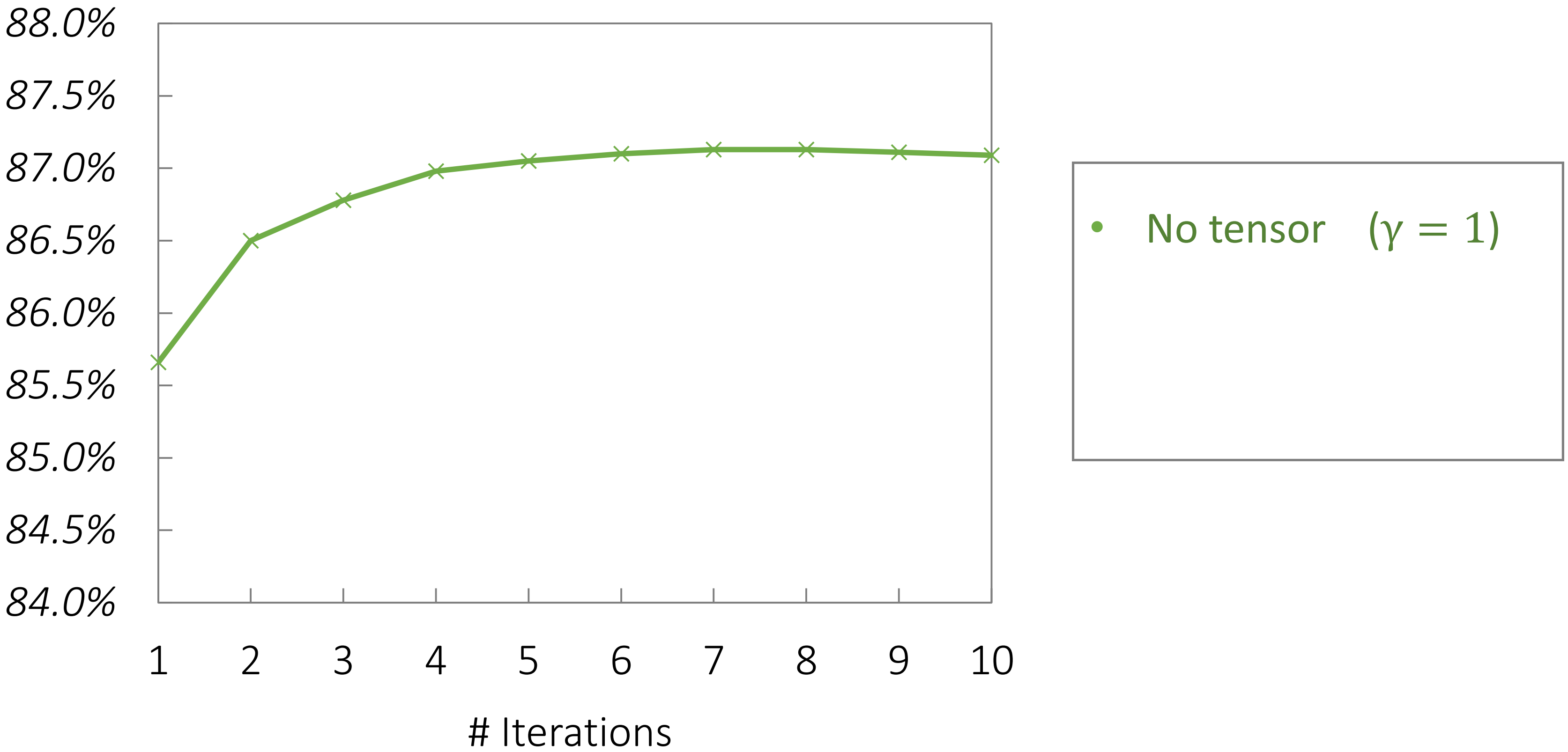
Unlabeled Attachment Score (UAS) evaluated without punctuations

Overall 1st-order Results



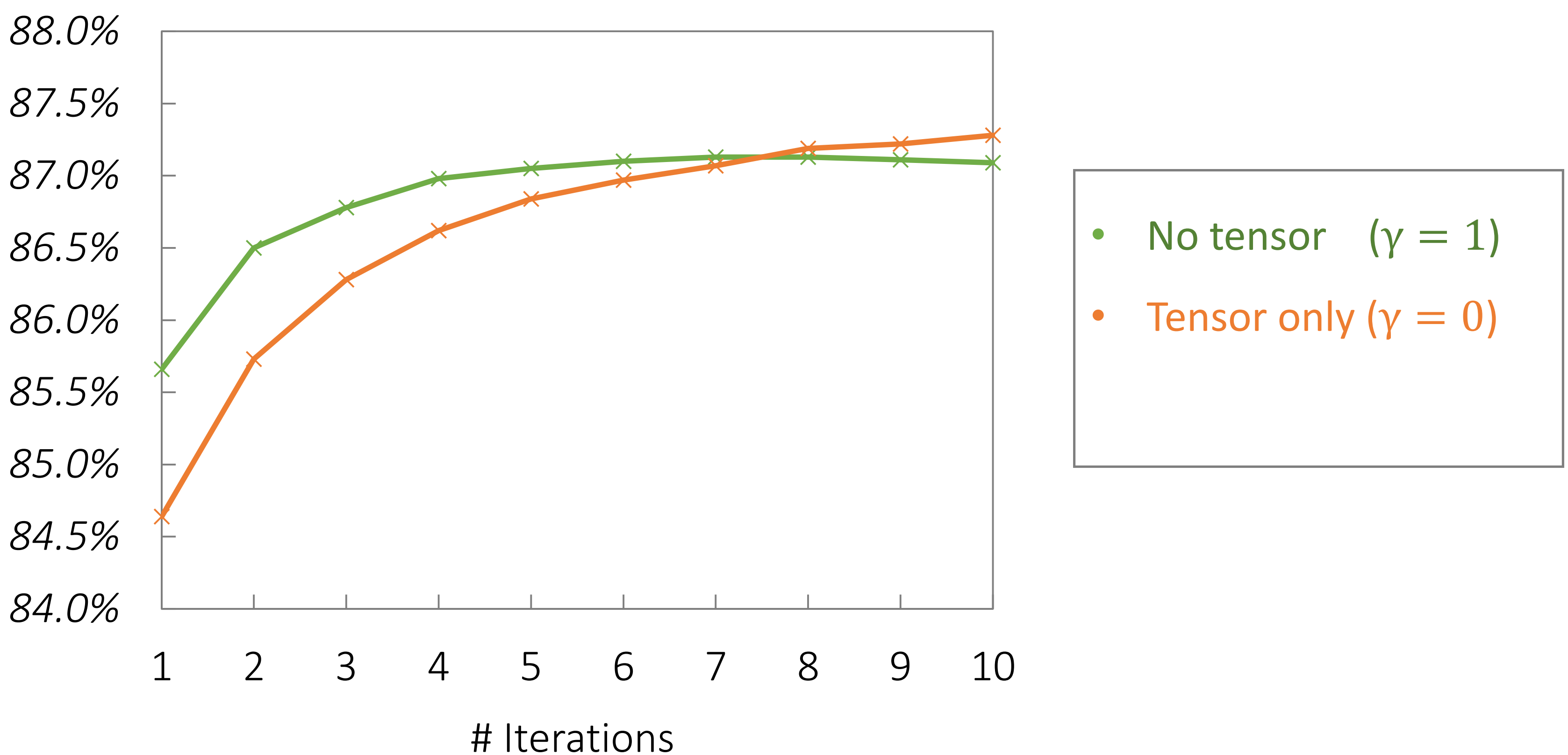
- > 0.7% average improvement
- Outperforms on 11 out of 14 languages

Impact of Tensor Component



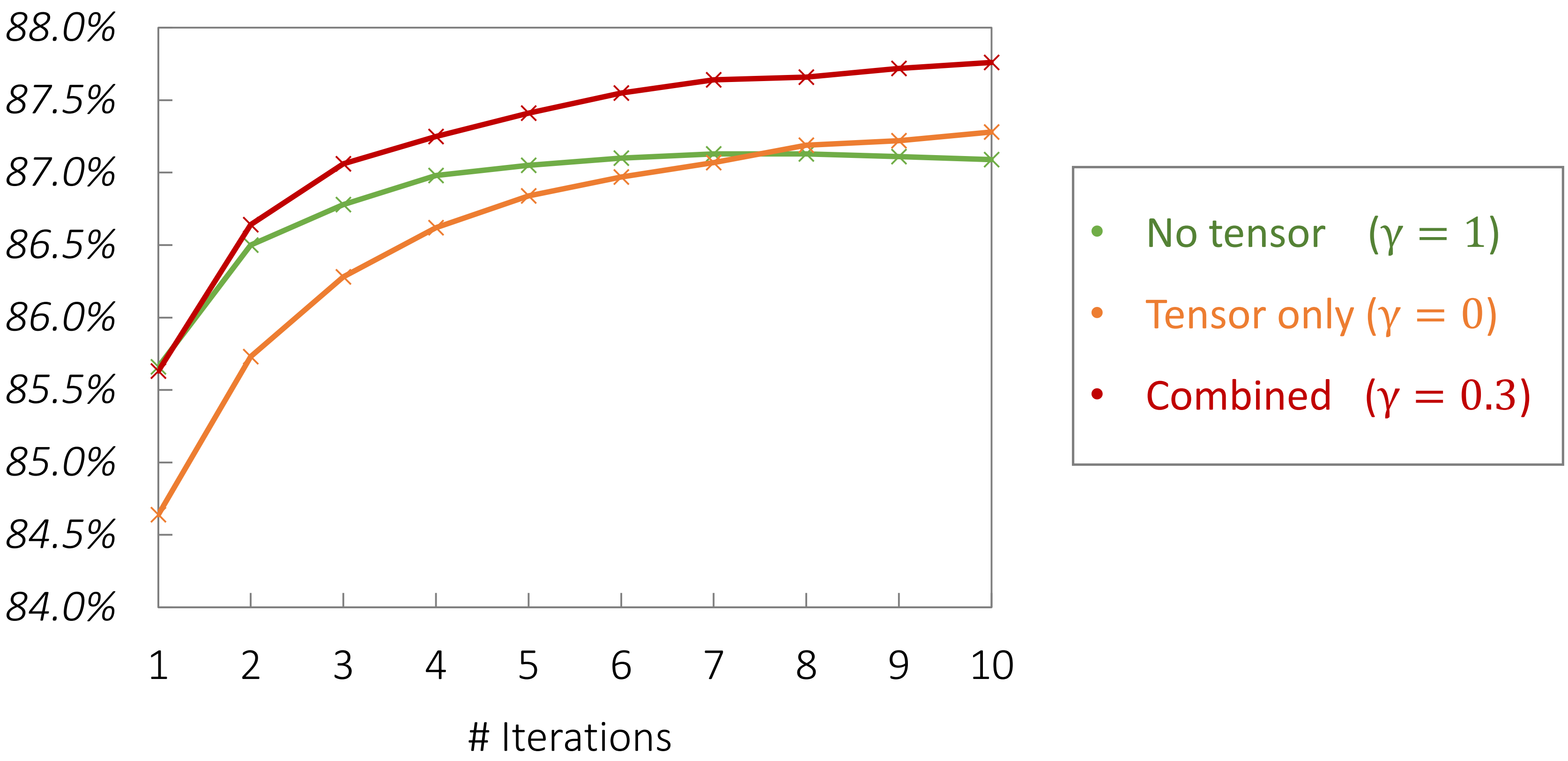
Impact of Tensor Component

- Tensor component achieves better generalization on test data

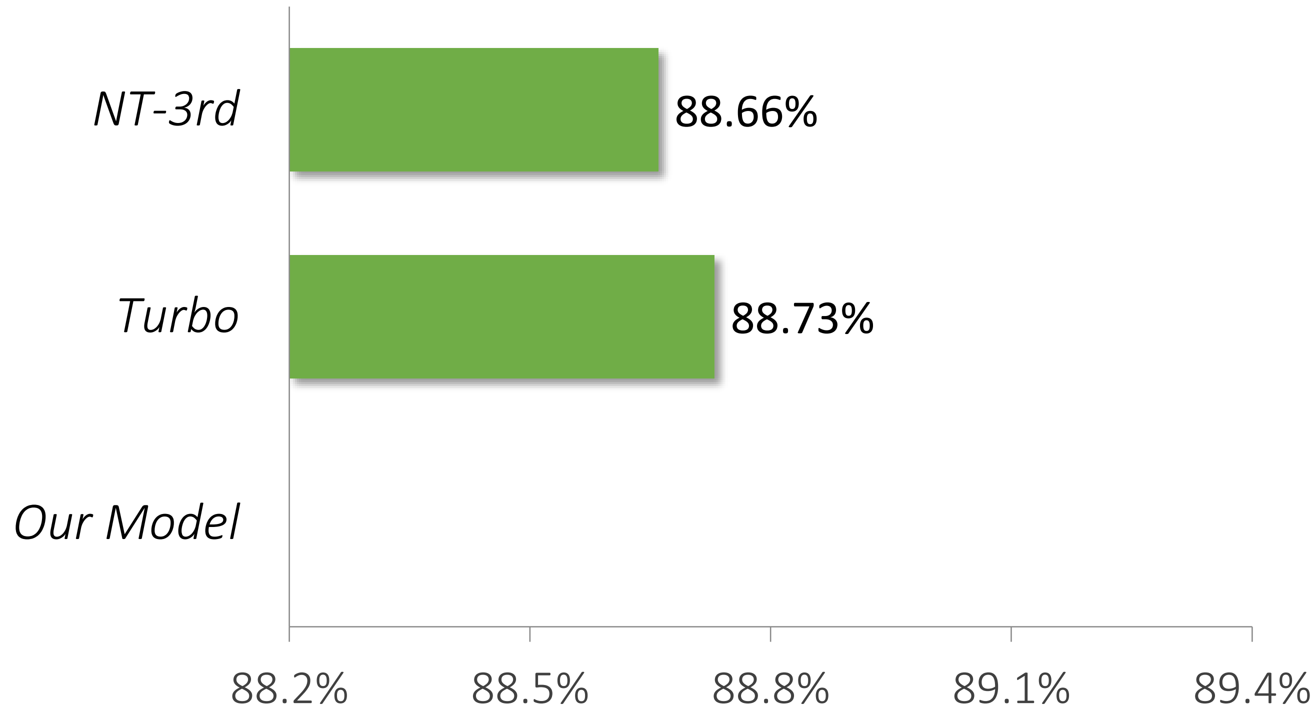


Impact of Tensor Component

- Tensor component achieves better generalization on test data
- Combined scoring outperforms single components

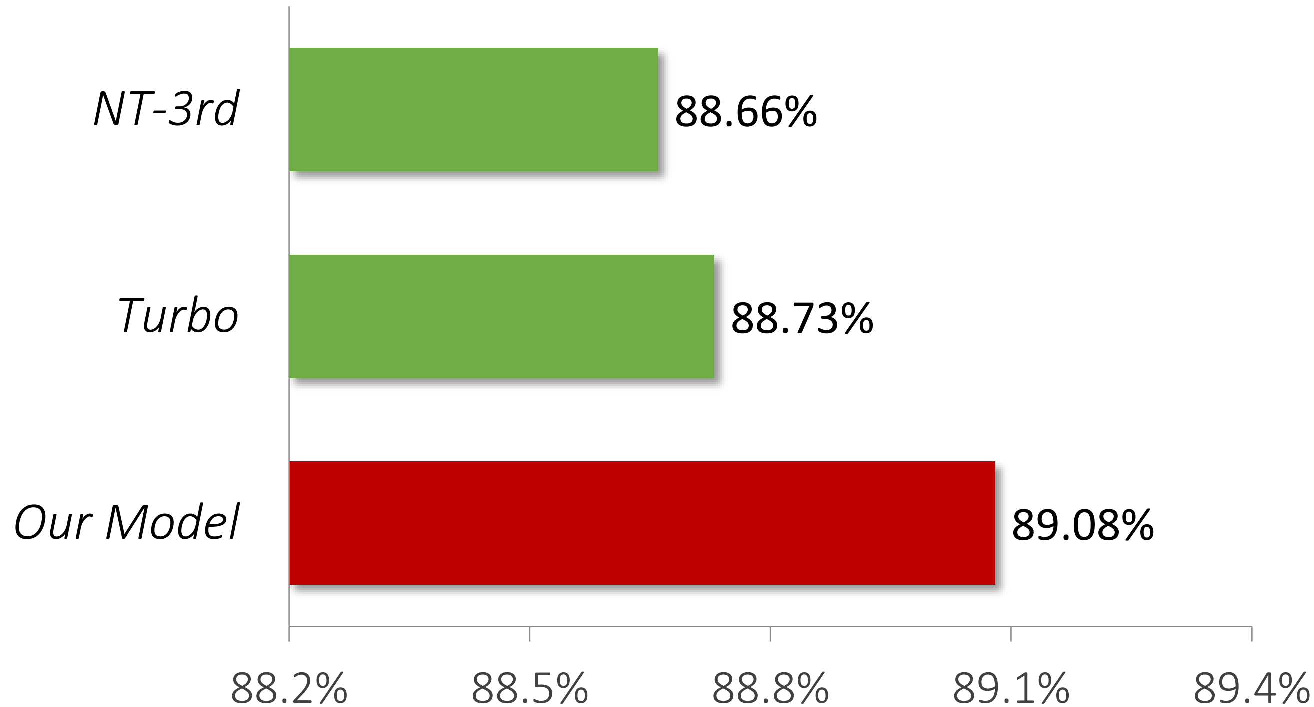


Overall 3rd-order Results



- Our traditional scoring component is just as good as the state-of-the-art system

Overall 3^{rd}-order Results



- The 1^{st}-order tensor component remains useful on high-order parsing
- Outperforms state-of-the-art single system
- Achieves best published results on 5 languages

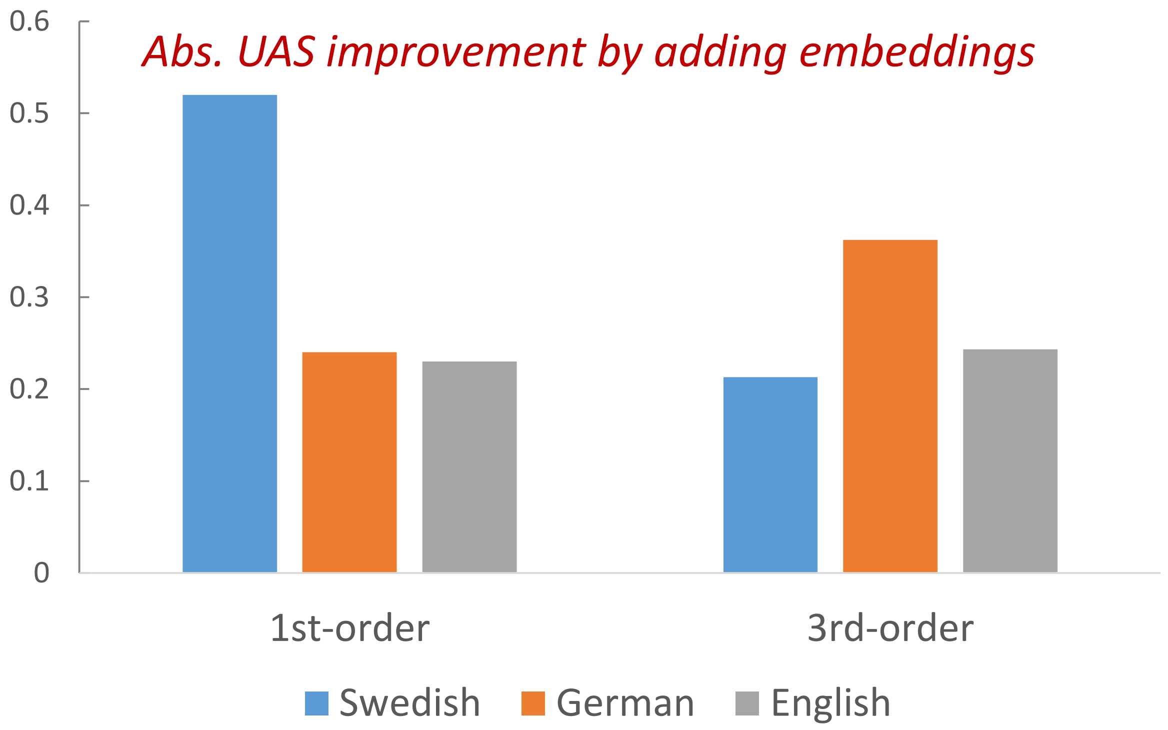
Leveraging Auxiliary Features

- Unsupervised word embeddings publicly available*

  English, German and Swedish have word embeddings in this dataset

- Append the embeddings of *current*, *previous* and *next* words into ϕ_h, ϕ_m

  $\phi_h \otimes \phi_m$ *involves more than* $(50 \times 3)^2$ *values for 50-dimensional embeddings!*



* https://github.com/wolet/sprml13-word-embeddings

Conclusion

- *Modeling:* we introduced a low-rank tensor factorization model for scoring dependency arcs

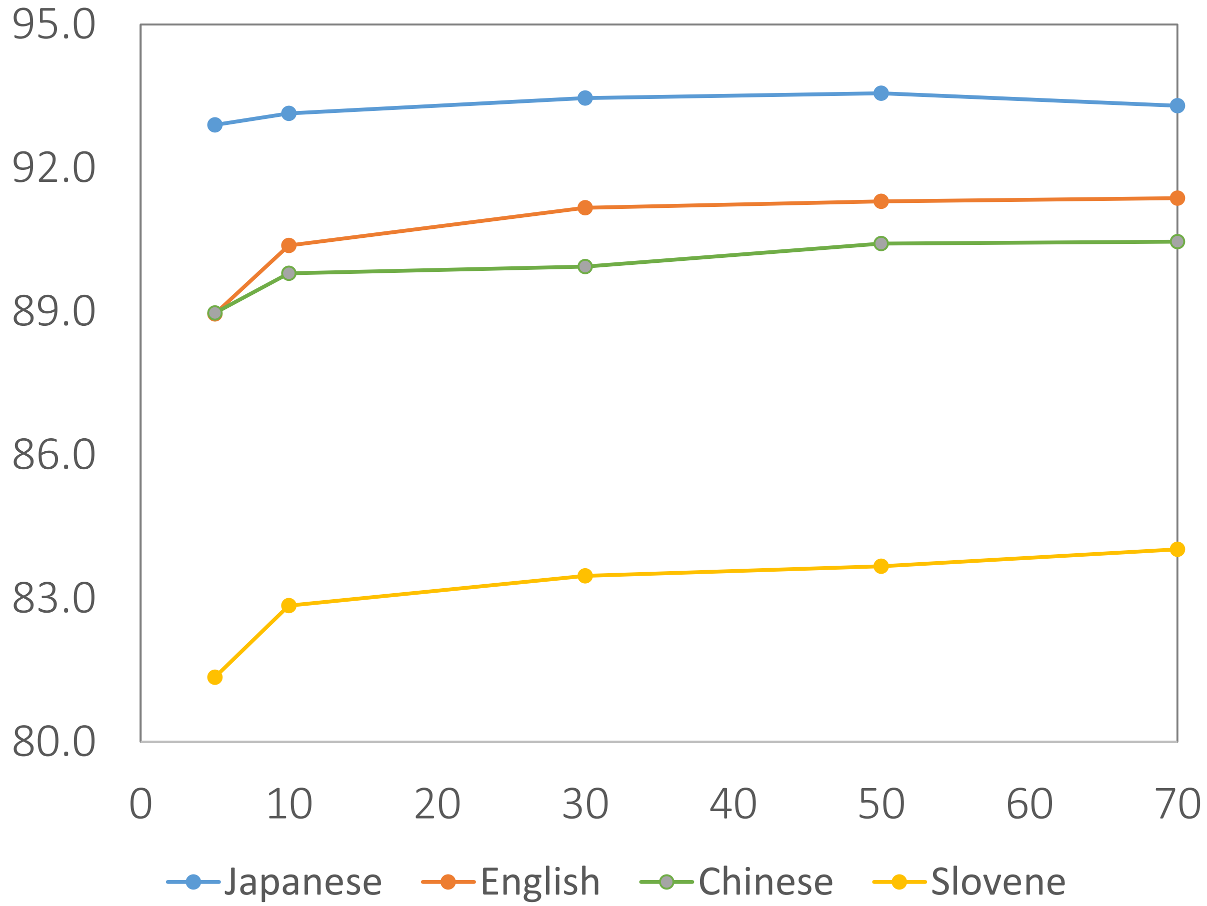
- *Learning:* we proposed an online learning method that directly optimizes the low-rank factorization for parsing performance, achieving state-of-the-art results

- *Opportunities & Challenges:* we hope to apply this idea to other structures and NLP problems.

Source code available at:

https://github.com/taolei87/RBGParser

Rank of the Tensor



Choices of Gamma