Low-Rank Tensors for Scoring Dependency Structures

Tao Lei

Yu Xin, Yuan Zhang, Regina Barzilay, Tommi Jaakkola

CSAIL, MIT
Our Goal

Dependency Parsing

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

• Dependency parsing as maximization problem:

• Key aspects of a parsing system:

1. Accurate scoring function $S(x, y; \theta)$

2. Efficient decoding procedure $\text{argmax}$
Finding Expressive Feature Set

*Traditional view:* 
requires a rich, expressive set of manually-crafted feature templates

![Diagram showing POS tagging and feature example](image)

*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\(\oplus\)NN\(\oplus\)2”
Finding Expressive Feature Set

**Traditional view:**

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

```
Traditional view:

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 \( \oplus \) 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:

```
I ate cake with a fork today
```

**Arc Features:**

\[ HW\_MW\_LEN: \text{ate} \oplus \text{cake} \oplus 2 \]
Traditional Scoring Revisited

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

- **Word:** ate
  - **POS:** VB
  - **POS+Word:** VB+ate
  - **Left POS:** PRON
  - **Right POS:** NN

- **Head**
  - ate
  - PB

- **Modifier**
  - cake
  - NN

- **Attach Length?**
  - Yes

- **Arc Features:**
  - HW_MW_LEN: ate ⊕ cake ⊕ 2
  - HW_MW: ate ⊕ cake
Traditional Scoring Revisited

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

```
<table>
<thead>
<tr>
<th>Head</th>
<th>Modifier</th>
<th>Attach Length?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word: ate</td>
<td>POS: VB</td>
<td>POS+Word: VB+ate</td>
</tr>
<tr>
<td>PRON</td>
<td>cake</td>
<td>NN+cake</td>
</tr>
<tr>
<td>NN</td>
<td>with</td>
<td>VB</td>
</tr>
<tr>
<td>IN</td>
<td>a</td>
<td>NN</td>
</tr>
<tr>
<td>DT</td>
<td>fork</td>
<td>NN</td>
</tr>
<tr>
<td>NN</td>
<td>today</td>
<td></td>
</tr>
</tbody>
</table>
```

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

```
<table>
<thead>
<tr>
<th>Arc Features:</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW_MW_LEN: ate@cake@2</td>
</tr>
<tr>
<td>HW_MW: ate@cake</td>
</tr>
<tr>
<td>HP_MP_LEN: VB@NN@2</td>
</tr>
<tr>
<td>HP_MP: VB@NN</td>
</tr>
<tr>
<td>... ...</td>
</tr>
</tbody>
</table>
```
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

\[ \phi_h \]
atomic head feature vector

\[ \phi_m \]
atomic modifier feature vector

\[ \phi_{h,m} \]
atomic arc feature vector

- Head
  - *ate*
  - *VB*
  - *VB+ate*
  - *PRON*
  - *NN*

- Modifier
  - *cake*
  - *NN*
  - *NN+cake*
  - *VB*
  - *IN*

- Attach Length?
  - *Yes*
  - *No*
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

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

*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 \hspace{1cm} atomic modifier feature vector \hspace{1cm} atomic arc feature vector

• Formulate the parameters as a tensor as well

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

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

Involves features not in \( \theta \)

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**

\[ \theta \in \mathbb{R}^L: \quad S_{\theta}(h \rightarrow m) = \langle \theta, \phi_{h \rightarrow m} \rangle \]  

(vector-based)

\[ A \in \mathbb{R}^{n \times n \times d}: \quad S_{\text{tensor}}(h \rightarrow m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle \]  

(tensor-based)

\[ U, V \in \mathbb{R}^{r \times n}, W \in \mathbb{R}^{r \times d}: \quad A = \sum U(i) \otimes V(i) \otimes W(i) \]

**Low-rank tensor**
Low-Rank Tensor Scoring: Formulation

\[ A = \sum U(i) \otimes V(i) \otimes W(i) \quad \Rightarrow \quad S_{\text{tensor}}(h \rightarrow m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle \]

\[ = \sum_{i=1}^{r} [U \phi_h]_i [V \phi_m]_i [W \phi_{h,m}]_i \]

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

\[ A = \sum U(i) \otimes V(i) \otimes W(i) \quad \Rightarrow \quad S_{\text{tensor}}(h \to m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle \]

\[ = \sum_{i=1}^{r} [U \phi_h]_i [V \phi_m]_i [W \phi_{h,m}]_i \]

**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

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 $A$

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

# of parameters: $n \times m$ \hspace{1cm} $(n + m)r$

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 manually 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 $S_\gamma(x, y)$:

$$\gamma \cdot S_\theta(x, y) + (1 - \gamma) \cdot S_{\text{tensor}}(x, y)$$

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 $\theta, U, V, W$:

$$y^* = \arg\max_{y \in T(x)} S_\gamma(x, y; \theta, U, V, W)$$
Learning Problem

• Given training set $D = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^N$:

• Search for parameter values that score the gold trees higher than others:

$$\forall y \in \text{Tree} (x_i): \quad S(\hat{x}_i, \hat{y}_i) \geq S(\hat{x}_i, y) + |\hat{y}_i - y| - \xi_i$$

• The training objective:

$$\min_{\theta, U, V, W, \xi_i \geq 0} \sum_i \xi_i + \|U\|^2 + \|V\|^2 + \|W\|^2 + \|\theta\|^2$$

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) \textbf{tailored} to our tensor setting

\textbf{(i) Iterate over training samples successively:}

\[
\sum_{i=1}^{r} [U \phi_h]_i [V \phi_m]_i [W \phi_{h,m}]_i
\]

is not linear nor convex

\textbf{(ii) choose to update a pair of sets (}\(\theta, U\), (\(\theta, V\)) or (\(\theta, W\)):}

\[
\text{Increments: } \quad \theta^{(t+1)} = \theta^{(t)} + \Delta \theta, \quad U^{(t+1)} = U^{(t)} + \Delta U
\]

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

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

<table>
<thead>
<tr>
<th>Unigram features:</th>
<th>form</th>
<th>lemma</th>
<th>form-p</th>
<th>lemma-p</th>
<th>form-n</th>
<th>lemma-n</th>
<th>pos</th>
<th>pos-p</th>
<th>bias</th>
<th>pos-n</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>morph</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bigram features:</th>
<th>pos-p, pos</th>
<th>pos, pos-n</th>
<th>pos, lemma</th>
<th>morph, lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigram features:</td>
<td>pos-p, pos, pos-n</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram showing structured dependencies among features.
Experiment Setup

**Datasets**
- 14 languages in CoNLL 2006 & 2008 shared tasks

**Features**
- Only 16 atomic word features for tensor
- Combine with 1\textsuperscript{st}-order (single arc) and up to 3\textsuperscript{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\textsuperscript{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

![Graph showing the impact of tensor component with iterations]

- No tensor \((\gamma = 1)\)
Impact of Tensor Component

- Tensor component achieves better generalization on test data

\[ \gamma = 1 \quad \text{No tensor} \]
\[ \gamma = 0 \quad \text{Tensor only} \]
Impact of Tensor Component

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

![Graph showing the impact of tensor component with iterations]

- No tensor \((\gamma = 1)\)
- Tensor only \((\gamma = 0)\)
- Combined \((\gamma = 0.3)\)
Our traditional scoring component is just as good as the state-of-the-art system.
Overall 3\textsuperscript{rd}-order Results

- The 1\textsuperscript{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 \( \phi_h, \phi_m \)
  
  \( \phi_h \otimes \phi_m \) involves more than \((50 \times 3)^2\) values for 50-dimensional embeddings!

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

![Graph showing the rank of the tensor for different languages: Japanese, English, Chinese, and Slovene. The graph plots the rank on the y-axis and a numeric scale on the x-axis.]}
Choices of Gamma

![Graph showing choices of gamma values](image-url)