# High-Order Low-Rank Tensors for Semantic Role Labeling

**Tao Lei**[1], **Yuan Zhang**[1], **Lluís Màrquez**[2], **Alessandro Moschitti**[2], **and Regina Barzilay**[1]
[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology
[2]ALT Research Group, Qatar Computing Research Institute
[1]{taolei, yuanzh, regina}@csail.mit.edu
[2]{lmarquez, amoschitti}@qf.org.qa

## Abstract

This paper introduces a tensor-based approach to semantic role labeling (SRL). The motivation behind the approach is to automatically induce a compact feature representation for words and their relations, tailoring them to the task. In this sense, our dimensionality reduction method provides a clear alternative to the traditional feature engineering approach used in SRL. To capture meaningful interactions between the argument, predicate, their syntactic path and the corresponding role label, we compress each feature representation first to a lower dimensional space prior to assessing their interactions. This corresponds to using an overall cross-product feature representation and maintaining associated parameters as a four-way low-rank tensor. The tensor parameters are optimized for the SRL performance using standard online algorithms. Our tensor-based approach rivals the best performing system on the CoNLL-2009 shared task. In addition, we demonstrate that adding the representation tensor to a competitive tensor-free model yields 2% absolute increase in F-score.[1]

## 1 Introduction

The accuracy of Semantic Role Labeling (SRL) systems depends strongly on the features used by the underlying classifiers. For instance, the top performing system on the CoNLL–2009 shared task employs over 50 language-specific templates for feature generation (Che et al., 2009). The templates are manually created and thus offer specific means of incorporating prior knowledge into the method. However, finding compact, informative templates is difficult since the relevant signal may be spread over many correlated features. Moreover, the use of lexicalized features, which are inevitably sparse, leads to overfitting. In this case it is advantageous to try to automatically compress the feature set to use a small number of underlying co-varying dimensions. Dimensionality reduction of this kind can be incorporated into the classifier directly by utilizing tensor calculus. In this paper, we adopt this strategy.

We start by building high-dimensional feature vectors that are subsequently mapped into a low-dimensional representation. Since this high-dimensional representation has to reflect the interaction between different indicators of semantic relations, we construct it as a cross-product of smaller feature vectors that capture distinct facets of semantic dependence: *predicate*, *argument*, *syntactic path* and *role label*. By compressing this sparse representation into lower dimensions, we obtain dense representations for words (predicate, argument) and their connecting paths, uncovering meaningful interactions. The associated parameters are maintained as a four-way low-rank tensor, and optimized for SRL performance. Tensor modularity enables us to employ standard online algorithms for training.

Our approach to SRL is inspired by recent success of our tensor-based approaches in dependency parsing (Lei et al., 2014). Applying analogous techniques to SRL brings about new challenges, however. The scoring function needs to reflect the high-order interactions between the predicate, argument,

[1]Our code is available at `https://github.com/taolei87/SRLParser`.

their syntactic path and the corresponding role label. Therefore, we parametrize the scoring function as a four-way tensor. Generalization to high-order tensors also requires new initialization and update procedures. For instance, the SVD initialization used in our dependency parsing work results in memory explosion when extending to our 4-way tensor. Instead, we employ the power method (De Lathauwer et al., 1995) to build the initial tensor from smaller pieces, one rank-1 component at a time. For learning, in order to optimize an overall non-convex objective function with respect to the tensor parameters, we modify the passive-aggressive algorithm to update all the low-rank components in one step. The update strategy readily generalizes to any high-order tensor.

We evaluate our tensor-based approach for SRL on the CoNLL–2009 shared task benchmark datasets of five languages: English, German, Chinese, Catalan and Spanish (Surdeanu et al., 2008). As a baseline, we use a simple SRL model that relies on a minimal set of standard features. Our results demonstrate that the tensor-based model outperforms the original SRL model by a significant margin, yielding absolute improvements of 2.1% $\mathrm{F}_1$ score. We also compare our results against the best performing system on this task (Zhao et al., 2009a). On three out of five languages, the tensor-based model outperforms this system. These results are particularly notable because the system of Zhao et al. (2009a) employs a rich set of language-specific features carefully engineered for this task. Finally, we demonstrate that using four-way tensor yields better performance than its three-way counterpart, highlighting the importance of modeling the relation between role labels and properties of the path.

## 2 Related Work

A great deal of SRL research has been dedicated to designing rich, expressive features. The initial work by Gildea and Jurafsky (2002) already identified a compact core set of features, which were widely adopted by the SRL community. These features describe the predicate, the candidate argument, and the syntactic relation between them (*path*). Early systems primarily extended this core set by including local context lexicalized patterns (e.g., $n$-grams), several extended representations of the *path* features, and some linguistically motivated syntactic patterns, as the *syntactic frame* (Surdeanu et al., 2003; Xue and Palmer, 2004; Pradhan et al., 2005).

More recent approaches explored a broader range of features. Among others, Toutanova et al. (2008), Martins and Almeida (2014) and Yang and Zong (2014) have explored high-order features involving several arguments and even pairs of sentence predicates. Other approaches have focused on semantic generalizations of lexical features using selectional preferences, neural network embeddings or latent word language models (Zapirain et al., 2013; Collobert et al., 2011; Deschacht and Moens, 2009; Roth and Woodsend, 2014). To avoid the intensive feature engineering inherent in SRL, Moschitti et al. (2008) employ kernel learning. Although attractive from this perspective, the kernel-based approach comes with a high computational cost. In contrast to prior work, our approach effectively learns low-dimensional representation of words and their roles, eliminating the need for heavy manual feature engineering. Finally, system combination approaches such as reranking typically outperform individual systems (Björkelund et al., 2010). Our method can be easily integrated as a component in one of those systems.

In technical terms, our work builds on our recent tensor-based approach for dependency parsing (Lei et al., 2014). In that work, we use a three-way tensor to score candidate dependency relations within a first-order scoring function. The tensor captures the interaction between words and their syntactic (head-modifier) relations. In contrast, the scoring function in SRL involves higher-order interactions between the path, argument, predicate and their associated role label. Therefore, we parametrized the scoring function with a four-way low-rank tensor. To help with this extension, we developed a new initialization and update strategy. Our experimental results demonstrate that the new representation tailored to SRL outperforms previous approaches.

## 3 Problem Formulation

Our setup follows the CoNLL–2009 shared task (Hajič et al., 2009). Each token in sentence $\mathbf{x}$ is annotated with a predicted POS tag and predicted

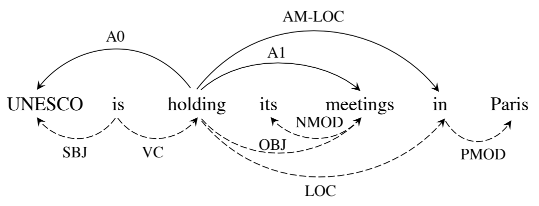

Figure 1: Example sentence from the CoNLL–2009 dataset annotated with syntactic and semantic dependencies. The lower graph is the syntactic dependency tree for the sentence. The upper part contains the semantic dependencies for the predicate "holding".

word lemma. Some tokens are also marked as predicates, i.e., argument-bearing tokens. The goal is to determine the semantic dependencies for each predicate $p_i$ (cf. upper part of Figure 1). These dependencies identify the arguments of each predicate and their role labels. In this work, we focus only on the semantic side – that is, identification and classification of predicate arguments. To this end, our system takes as input a syntactic dependency tree $\mathbf{y}_{\text{syn}}$ derived from a state-of-the-art parser (bottom part of Figure 1).

More formally, let $\{p_i\} \subset \mathbf{x}$ be the set of verbal and nominal predicates in the sentence. For each predicate $p_i$ (e.g., "holding"), our goal is to predict tuples $(p_i, a_{ij}, r_{ij})$ specifying the semantic dependency arcs, where $a_{ij} \in \mathbf{x}$ is one argument (e.g., "meetings"), and $r_{ij}$ is the corresponding semantic role label (e.g., A1). The semantic parse is then the collection of predicted arcs $\mathbf{z}_{\text{sem}} = \{(p_i, a_{ij}, r_{ij})\}$.

We decouple syntactic and semantic inference problems into two separate steps. We first run our syntactic dependency parser RBGParser[2] to obtain the syntactic dependency tree $\mathbf{y}_{\text{syn}}$. The semantic parse $\mathbf{z}_{\text{sem}}$ is then found conditionally on the syntactic part:

$$\mathbf{z}^*_{\text{sem}} = \arg\max_{\mathbf{z}_{\text{sem}}} S_{\text{sem}}(\mathbf{x}, \mathbf{y}_{\text{syn}}, \mathbf{z}_{\text{sem}}), \qquad (1)$$

Here $S_{\text{sem}}(\cdot)$ is the parametrized scoring function to be learned. We build our scoring function by combining a traditional feature scoring function with a tensor-based scoring function.

[2]https://github.com/taolei87/RBGParser

| | |
|---|---|
| Predicate word | Path |
| Predicate POS | Path + arg. POS |
| Argument word | Path + pred. POS |
| Argument POS | Path + arg. word |
| Pred. + arg. words | Path + pred. word |
| Pred. + arg. POS | Voice + pred. + arg. POS |
| Voice + pred. word | Voice + pred. POS |

Table 1: Templates for first-order semantic features. These features are also (optionally) combined with role labels.

## 3.1 Traditional Scoring Using Manually-designed Features

In a typical feature-based approach (Johansson, 2009; Che et al., 2009), feature templates give rise to rich feature descriptions of the semantic structure. The score $S_{\text{sem}}(\mathbf{x}, \mathbf{y}_{\text{syn}}, \mathbf{z}_{\text{sem}})$ is then defined as the inner product between the parameter vector and the feature vector. In the first-order arc-factored case,

$$\begin{aligned} S_{\text{sem}}(\mathbf{x}, \mathbf{y}_{\text{syn}}, \mathbf{z}_{\text{sem}}) &= \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}_{\text{syn}}, \mathbf{z}_{\text{sem}}) \\ &= \sum_{(p,a,r) \in \mathbf{z}_{\text{sem}}} \mathbf{w} \cdot \boldsymbol{\phi}(p, a, r), \end{aligned}$$

where $\mathbf{w}$ are the model parameters and $\boldsymbol{\phi}(p, a, r)$ is the feature vector representing a single semantic arc $(p, a, r)$ (we suppress its dependence on $\mathbf{x}$ and $\mathbf{y}_{\text{syn}}$). We also experiment with second order features, i.e., considering two arguments associated with the same predicate, or two predicates sharing the same token as argument.

For the arc-factored model, there are mainly four types of atomic information that define the arc features in $\boldsymbol{\phi}(p, a, r)$:

(a) the predicate token $p$ (and its local context);
(b) the argument token $a$ (and its local context);
(c) the dependency label path that connects $p$ and $a$ in the syntactic tree;
(d) the semantic role label $r$ of the arc.

These pieces of atomic information are either used directly or combined as unigram up to 4-gram features into traditional models. To avoid heavy feature engineering and overfitting, we use a light and compact feature set derived from the information in (a)–(d). Table 1 shows the complete list of feature templates, used as our first-order semantic baseline in the experiments.

## 3.2 Low-rank Scoring via Projected Representations

Now, we describe the tensor-based scoring function. We characterize each semantic arc $(p, a, r)$ using the cross-product of atomic feature vectors associated with the four types of information described above: the predicate vector $\boldsymbol{\phi}(p)$, the argument vector $\boldsymbol{\phi}(a)$, the dependency path vector $\boldsymbol{\phi}(path)$ and the semantic role label vector $\boldsymbol{\phi}(r)$. For example, in the simplest case $\boldsymbol{\phi}(p), \boldsymbol{\phi}(a) \in [0,1]^n$ are one-hot indicator vectors, where $n$ is the size of the vocabulary. Similarly, $\boldsymbol{\phi}(path) \in [0,1]^m$ and $\boldsymbol{\phi}(r) \in [0,1]^l$ are indicator vectors where $m$ is the number of unique paths (seen in the training set) and $l$ is the number of semantic role labels. Of course, we can add other atomic information into these atomic vectors. For example, $\boldsymbol{\phi}(p)$ will not only indicate the word form of the current predicate $p$, but also the word lemma, POS tag and surrounding tokens as well. The cross-product of these four vectors is an extremely high-dimensional rank-1 tensor,

$$\boldsymbol{\phi}(p) \otimes \boldsymbol{\phi}(a) \otimes \boldsymbol{\phi}(path) \otimes \boldsymbol{\phi}(r) \;\in\; \mathbb{R}^{n \times n \times m \times l}$$

in which each entry indicates the combination of four atomic features appearing in the semantic arc $(p, a, r)$[3]. The rank-1 tensor (cross-product) captures all possible combinations over atomic units, and therefore it is a full feature expansion over the manually selected feature set in Table 1. Similar to the traditional scoring, the semantic arc score is the inner product between a 4-way parameter tensor $A$ and this feature tensor:

$$\begin{aligned} & A \in \mathbb{R}^{n \times n \times m \times l} : \\ & vec(A) \cdot vec\left(\boldsymbol{\phi}(p) \otimes \boldsymbol{\phi}(a) \otimes \boldsymbol{\phi}(path) \otimes \boldsymbol{\phi}(r)\right), \end{aligned} \tag{2}$$

where $vec(\cdot)$ denotes the vector representation of a matrix / tensor.

Instead of reducing and pruning possible feature concatenations (e.g., by manual feature template construction as in the traditional approach), this tensor scoring method avoids parameter explosion and overfitting by assuming a low-rank factorization of the parameters $A$. Specifically, A is decomposed into the sum of $k$ simple rank-1 components,

$$A = \sum_{i=1}^{k} P(i) \otimes Q(i) \otimes R(i) \otimes S(i). \tag{3}$$

Here $k$ is a small constant, $P, Q \in \mathbb{R}^{k \times n}$, $R \in \mathbb{R}^{k \times m}$ and $S \in \mathbb{R}^{k \times l}$ are parameter matrices, and $P(i)$ (and similarly $Q(i)$, $R(i)$ and $S(i)$) represents the $i$-th row vector of matrix $P$.

The advantages of this low-rank assumption are as follows. First, computing the score no longer requires maintaining and constructing extremely large tensors. Instead, we can project atomic vectors via $P$, $Q$, $R$ and $S$ obtaining small dense vectors, and subsequently calculating the arc score by

$$\sum_{i=1}^{k} [P\boldsymbol{\phi}(p)]_i \, [Q\boldsymbol{\phi}(a)]_i \, [R\boldsymbol{\phi}(path)]_i \, [S\boldsymbol{\phi}(r)]_i \,.$$

Second, projecting atomic units such as words, POS tags and labels into dense, low-dimensional vectors can effectively alleviate the sparsity problem, and it enables the model to capture high-order feature interactions between atomic units, while avoiding the parameter explosion problem.

## 3.3 Combined System

Similar to our low-rank syntactic dependency parsing model (Lei et al., 2014), our final scoring function $S_{\text{sem}}(\mathbf{x}, \mathbf{y}_{\text{syn}}, \mathbf{z}_{\text{sem}})$ is the combination of the traditional scoring and the low-rank scoring,

$$\begin{aligned} S_{\text{sem}}(\mathbf{x}, \mathbf{y}_{\text{syn}}, \mathbf{z}_{\text{sem}}) \quad &= \\ \gamma\, \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}_{\text{syn}}, \mathbf{z}_{\text{sem}}) + (1-\gamma) & \sum_{(p,a,r) \in \mathbf{z}_{\text{sem}}} \sum_{i=1}^{k} \\ [P\boldsymbol{\phi}(p)]_i \, [Q\boldsymbol{\phi}(a)]_i \, [R\boldsymbol{\phi}(path)]_i & \, [S\boldsymbol{\phi}(r)]_i \,. \end{aligned}$$

where $\gamma \in [0,1]$ is a hyper-parameter balancing the two scoring terms. We tune this value on the development set. Finally, the set of parameters of our model is denoted as $\boldsymbol{\theta} = \{\mathbf{w}, P, Q, R, S\}$. Our goal is to optimize the weight vector $\mathbf{w}$ as well as the four projection matrices given the training set.

[3] We always add a bias term into these atomic vectors (e.g., a fixed "1" attached to the beginning of every vector). Therefore, their cross-product will contain all unigram to 4-gram concatenations, not just 4-gram concatenations.

## 4 Learning

We now describe the learning method for our SRL model. Let $D = \{(\hat{\mathbf{x}}^{(i)}, \hat{\mathbf{y}_{\text{syn}}}^{(i)}, \hat{\mathbf{z}_{\text{sem}}}^{(i)})\}_{i=1}^{N}$ be the collection of $N$ training samples. The values of the set of parameters $\boldsymbol{\theta} = \{\mathbf{w}, P, Q, R, S\}$ are estimated on the basis of this training set. Following standard practice, we optimize the parameter values in a maximum soft-margin framework. That is, for the given sentence $\hat{\mathbf{x}}$ and the corresponding syntactic tree $\hat{\mathbf{y}_{\text{syn}}}$, we adjust parameter values to separate gold semantic parse and other incorrect alternatives:

$$\begin{aligned} &\forall \mathbf{z}_{\text{sem}} \in \mathcal{Z}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}}) : \\ &\quad S_{sem}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}}, \hat{\mathbf{z}_{\text{sem}}}) \geq S_{sem}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}}, \mathbf{z}_{\text{sem}}) \\ &\qquad + cost(\hat{\mathbf{z}_{\text{sem}}}, \mathbf{z}_{\text{sem}}) \end{aligned} \tag{4}$$

where $\mathcal{Z}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}})$ represent the set of all possible semantic parses, and $cost(\hat{\mathbf{z}_{\text{sem}}}, \mathbf{z}_{\text{sem}})$ is a non-negative function representing the structural difference between $\hat{\mathbf{z}_{\text{sem}}}$ and $\mathbf{z}_{\text{sem}}$. The cost is zero when $\mathbf{z}_{\text{sem}} = \hat{\mathbf{z}_{\text{sem}}}$, otherwise it becomes positive and therefore is the "margin" to separate the two parses. Following previous work (Johansson, 2009; Martins and Almeida, 2014), this cost function is defined as the sum of arc errors – we add 1.0 for each false-positive arc, 2.0 for each false-negative arc (a missing arc) and 0.5 if the predicate-argument pair $(p, a)$ is in both parses but the semantic role label $r$ is incorrect.

### 4.1 Online Update

The parameters are updated successively after each training sentence. Each update first checks whether the constraint (4) is violated. This requires "cost-augmented decoding" to find the maximum violation with respect to the gold semantic parse:

$$\begin{aligned} \tilde{\mathbf{z}_{\text{sem}}} = \arg\max_{\mathbf{z}_{\text{sem}}} \; & S_{\text{sem}}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}}, \mathbf{z}_{\text{sem}}) \\ & + cost(\hat{\mathbf{z}_{\text{sem}}}, \mathbf{z}_{\text{sem}}) \end{aligned}$$

When the constraint (4) is violated (i.e. $\tilde{\mathbf{z}_{\text{sem}}} \neq \hat{\mathbf{z}_{\text{sem}}}$), we seek a parameter update $\Delta\boldsymbol{\theta}$ to fix this violation. In other words, we define the hinge loss for this example as follows,

$$\begin{aligned} loss(\boldsymbol{\theta}) \quad = \quad & \max\{\, 0,\; S_{\text{sem}}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}}, \tilde{\mathbf{z}_{\text{sem}}}) \\ & + cost(\hat{\mathbf{z}_{\text{sem}}}, \tilde{\mathbf{z}_{\text{sem}}}) - S_{\text{sem}}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}}, \hat{\mathbf{z}_{\text{sem}}}) \,\} \end{aligned}$$

and we revise the parameter values to minimize this loss function.

Since this loss function is neither linear nor convex with respect to the parameters $\boldsymbol{\theta}$ (more precisely the low-rank component matrices $P$, $Q$, $R$ and $S$), we can use the same alternating passive-aggressive (PA) update strategy in our previous work (Lei et al., 2014) to update one parameter matrix at one time while fixing the other matrices. However, as we demonstrated later, modifying the passive-aggressive algorithm slightly can give us a **joint** update over all components in $\boldsymbol{\theta}$. Our preliminary experiment shows this modified version achieves better results compared to the alternating PA.

### 4.2 Joint PA Update for Tensor

The original passive-aggressive parameter update $\Delta\theta$ is derived for a linear, convex loss function by solving a quadatic optimization problem. Although our scoring function $S_{\text{sem}}(\cdot)$ is not linear, we can simply approximate it with its first-order Taylor expansion:

$$S(\mathbf{x}, \mathbf{y}, \mathbf{z}; \boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx S(\mathbf{x}, \mathbf{y}, \mathbf{z}; \boldsymbol{\theta}) + \frac{dS}{d\boldsymbol{\theta}} \cdot \Delta\boldsymbol{\theta}$$

In fact, by plugging this into the hinge loss function and the quadratic optimization problem, we get a joint closed-form update which can be simply described as,

$$\Delta\boldsymbol{\theta} = \max\left\{ C,\; \frac{loss(\boldsymbol{\theta})}{\|g\boldsymbol{\theta}\|^2} \right\} g\boldsymbol{\theta}$$

where

$$g\boldsymbol{\theta} = \frac{dS}{d\boldsymbol{\theta}}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}}, \hat{\mathbf{z}_{\text{sem}}}) - \frac{dS}{d\boldsymbol{\theta}}(\hat{\mathbf{x}}, \hat{\mathbf{y}_{\text{syn}}}, \tilde{\mathbf{z}_{\text{sem}}}),$$

and $C$ is a regularization hyper-parameter controlling the maximum step size of each update. Note that $\boldsymbol{\theta}$ is the set of all parameters, the update jointly adjusts all low-rank matrices and the traditional weight vector. The PA update is "adaptive" in the sense that its step size is propotional to the $loss(\boldsymbol{\theta})$ of the current training sample. Therefore the step size is adaptively decreased as the model fits the training data.

### 4.3 Tensor Initialization

Since the scoring and loss function with high-order tensor components is highly non-convex, our model performance can be impacted by the initialization of the matrices $P$, $Q$, $R$ and $S$. In addition to intializing these low-rank components randomly, we also experiment with a strategy to provide a good guess of the low-rank tensor.

First, note that the traditional manually-selected feature set (i.e., $\phi(p, a, r)$ in our notation) is an expressive and informative subset of the huge feature expansion covered in the feature tensor. We can train our model using only the manual feature set and then use the corresponding feature weights $\mathbf{w}$ to intialize the tensor. Specifically, we create a sparse tensor $T \in \mathbb{R}^{n \times n \times m \times l}$ by putting each parameter weight in $\mathbf{w}$ into its corresponding entry in $T$. We then try to find a low-rank approximation of sparse tensor $T$ by approximately minimizing the squared error:

$$\min_{P,Q,R,S} \|T - \sum_i P(i) \otimes Q(i) \otimes R(i) \otimes S(i)\|_2^2$$

In the low-rank dependency parsing work (Lei et al., 2014), this is achieved by unfolding the sparse tensor $T$ into a $n \times nml$ matrix and taking the SVD to get the top low-rank components. Unfortunately this strategy does not apply in our case (and other high-order tensor cases) because even the number of columns in the unfolded matrix is huge, $nml > 10^{11}$, and simply taking the SVD would fail because of memory limits.

Instead, we adopt the generalized high-order power method, a.k.a. power iteration (De Lathauwer et al., 1995), to incrementally obtain the most important rank-1 component one-by-one – $P(i)$, $Q(i)$, $R(i)$ and $S(i)$ for each $i = 1..k$. This method is a very simple iterative algorithm and is used to find the largest eigenvalues and eigenvectors (or singular values and vectors in SVD case) of a matrix. Its generalization directly applies to our high-order tensor case.

```
Input: sparse tensor $T$, rank number $i$
    and fixed rank-1 components $P(j)$, $Q(j)$,
    $R(j)$ and $S(j)$ for $j = 1..(i-1)$
Output: new component $P(i)$, $Q(i)$, $R(i)$ and
    $S(i)$.
-----------------------------------------------
1:  Randomly initialize four unit vectors $p$, $q$, $r$
    and $s$
2:  $T' = T - \sum_j P(j) \otimes Q(j) \otimes R(j) \otimes S(j)$
3:  repeat
4:      $p = \langle T', -, q, r, s \rangle$ and normalize it
5:      $q = \langle T', p, -, r, s \rangle$ and normalize it
6:      $r = \langle T', p, q, -, s \rangle$ and normalize it
7:      $s = \langle T', p, q, r, - \rangle$
8:      $norm = \|s\|_2^2$
9:  until $norm$ converges
10: $P(i) = p$ and $Q(i) = q$
11: $R(i) = r$ and $S(i) = s$
```

Figure 2: The iterative power method for high-order tensor initialization. The operator $p = \langle T', -, q, r, s \rangle$ is the multiplication between the tensor and three vectors, defined as $p_i = \sum_{jkl} T_{ijkl} q_j r_k s_l$. Similarly, $q_j = \sum_{ikl} T_{ijkl} p_i r_k s_l$ etc.

## 5 Implementation Details

**Decoding** Following Lluís et al. (2013), the decoding of SRL is formulated as a bipartite maximum assignment problem, where we assign arguments to semantic roles for each predicate. We use the maximum weighted assignment algorithm (Kuhn, 1955). For syntactic dependency parsing, we employ the randomized hill-climbing algorithm from our previous work (Zhang et al., 2014).

**Features** Table 1 summarizes the first-order feature templates. These features are mainly drawn from previous work (Johansson, 2009). In addition, we extend each template with the argument label.

Table 2 summarizes the atomic features used in $\phi(p)$ and $\phi(a)$ for the tensor component. For each predicate or argument, the feature vector includes its word form and POS tag, as well as the POS tags of the context words. We also add unsupervised word embeddings learned on raw corpus.[4] For atomic vectors $\phi(path)$ and $\phi(r)$ representing the path and the semantic role label, we use the indicator feature and a bias term.

## 6 Experimental Setup

**Dataset** We evaluate our model on the English dataset and other 4 datasets in the CoNLL-2009 shared task (Surdeanu et al., 2008). We use the

[4] `https://github.com/wolet/sprml13-word-embeddings`

| | | |
|---|---|---|
| word | word-l | word-r |
| pos | pos-l | pos-r |
| pos-l + pos | pos + pos-r | pos + word |
| pos-l + pos + pos-r | voice | embeddings |

Table 2: Predicate/argument atomic features used by our tensor for SRL. `word` stands for the word form (and also lemma), `pos` stands for the predicted POS tag and `voice` stands for the voice of the predicate. The suffixes `-l` and `-r` refer to the left and right of the current token respectively. For example, `pos-l` means the POS tag to the left of the current word in the sentence.

official split for training, development and testing. For English, the data is mainly drawn from the Wall Street Journal. In addition, a subset of the Brown corpus is used as the secondary out-of-domain test set, in order to evaluate how well the model generalizes to a different domain. Following the official practice, we use predicted POS tags, lemmas and morphological analysis provided in the dataset across all our experiments. The predicates in each sentence are also given during both training and testing. However, we neither predict nor use the sense for each predicate.

**Systems for Comparisons** We compare against three systems that achieve the top average performance in the joint syntactic and semantic parsing track of the CoNLL-2009 shared task (Che et al., 2009; Zhao et al., 2009a; Gesmundo et al., 2009). All approaches extensively explored rich features for the SRL task. We also compare with the state-of-the-art parser (Björkelund et al., 2010) for English, an improved version of systems participated in CoNLL-2009. This system combines the pipeline of dependency parser and semantic role labeler with a global reranker. Finally, we compare with the recent approach which employs distributional word representations for SRL (Roth and Woodsend, 2014). We directly obtain the outputs of all these systems from the CoNLL-2009 website[5] or the authors.

**Model Variants** Our full model utilizes 4-way tensor component and a standard feature set from (Johansson, 2009). We also compare against our model without the tensor component, as well as a variant with a 3-way tensor by combining the path and semantic role label parts into a single mode (dimension).

**Evaluation Measures** Following standard practice in the SRL evaluation, we measure the performance using labeled F-score. To this end, we apply the evaluation script provided on the official website.[6] The standard evaluation script considers the predicate sense prediction as a special kind of semantic label.[7] Since we are neither predicting nor using the predicate sense information, we exclude this information in most of the evaluation. In addition, we combine the predicate sense classification output of (Björkelund et al., 2010) with our semantic role labeling output, to provide results directly comparable to previous reported numbers.

**Experimental Details** Across all experiments, we fix the rank of the tensor to 50 and train our model for a maximum of 20 epochs. Following common practice, we average parameters over all iterations. For each experimental setting, we tune the hyper-parameter $\gamma \in \{0.3, 0.5, 0.7, 0.9\}$ and $C \in \{0.01, 0.1, 1\}$ on the development set and apply the best model on the test set. Each model is evaluated on the development set after every epoch to pick the the best number of training epoch. For the experiments with random initialization on the tensor component, the vectors are initialized as random unit vectors. We combine our SRL model with our syntactic dependency parser, RBGParser v1.1 (Lei et al., 2014), for joint syntactic and semantic parsing. The labeled attachment score (LAS) of RBGParser is 90.4 on English, when we train the "standard" model type using the unsupervised word vectors.

## 7 Results

We first report the performance of our methods and other state-of-the-art SRL systems on English datasets (See Table 3). We single out performance

[5]`http://ufal.mff.cuni.cz/conll2009-st/results/results.php`

[6]`http://ufal.mff.cuni.cz/conll2009-st/scorer.html`

[7]Note that the original script includes such prediction in the F-score calculation, although the predicate sense is typically predicted in a separate step before semantic label classification.

| **Model** | **Excluding predicate senses** | | | **Including predicate senses** | |
|---|---|---|---|---|---|
| | WSJ-dev | WSJ-test | Brown-test | WSJ-test | Brown-test |
| 1st-order w/o tensor | 79.42 | 80.84 | 69.38 | 85.46 | 74.66 |
| + 3-way tensor | 80.77 | 82.19 | 69.76 | 86.34 | 74.94 |
| + 4-way tensor | **81.03** | **82.51*** | **70.77** | **86.58*** | **75.57** |
| CoNLL-2009 1st place | – | 82.08 | 69.84 | 86.15 | 74.58 |
| CoNLL-2009 2nd place | – | 81.20 | 68.86 | 85.51 | 73.82 |
| CoNLL-2009 3rd place | – | 78.66 | 65.89 | 83.24 | 70.65 |
| (Roth and Woodsend, 2014) | – | 80.87 | 69.33 | 85.50 | 74.67 |
| (Björkelund et al., 2010) | 78.85 | 81.35 | 68.34 | 85.80 | 73.92 |
| **Model + Reranker** | WSJ-dev | WSJ-test | Brown-test | WSJ-test | Brown-test |
| (Roth and Woodsend, 2014) + reranking | – | 82.10 | **71.12** | 86.34 | **75.88** |
| (Björkelund et al., 2010) + reranking | 80.50 | **82.87** | 70.91 | **86.86** | 75.71 |

Table 3: SRL labeled F-score of our model variants, and state-of-the-art systems on the CoNLL shared task. We consider a tensor-free variant of our model, and tensor-based variants that include first-order SRL features. For the latter, we consider implementations with 3-way and 4-way tensors. Winning systems (with and without a reranker) are marked in bold. Statistical significance with $p < 0.05$ is marked with $*$.

on English corpora because these datasets are most commonly used for system evalutation. As a single system without reranking, our model outperforms the five top performing systems (second block in Table 3) on both in-domain and out-of-domain datasets. The improvement from the F-score of 82.08% to our result 82.51% on the WSJ in-domain test set is significant with $p < 0.05$, which is computed using a randomized test tool[8] based on Yeh (2000). For comparison purposes, we also report F-score performance when predicate senses are included in evaluation. The relative performance between the systems is consistent independent of whether the predicate senses are included or excluded.

Table 4 shows the results of our system on other languages in the CoNLL-2009 shared task. Out of five languages, our model rivals the best performing system on three languages, achieving statistically significant gains on English and Chinese. Note that our model uses the same feature configuration for all the languages. In contrast, Zhao et al. (2009b) rely on language-specific configurations obtained via “huge feature engineering” (as noted by the authors).

Results in Table 3 and 4 also highlight the contribution of the tensor to the model performance, which is consistent across languages. Without the tensor component, our system trails the top two performing systems. However, adding the tensor component provides on average 2.1% absolute gain, resulting in competitive performance. The mode of the tensor also contributes to the performance – the 4-way tensor model performs better than the 3-way counterpart, demonstrating the importance of modeling the interactions between dependency paths and semantic role labels.

[8]`http://www.nlpado.de/~sebastian/software/sigf.shtml`

| | | WSJ-test | Brown-test |
|---|---|---|---|
| 1st-order w/o tensor | | 80.84 | 69.38 |
| + 3-way tensor | Rnd. Init. | 81.87 | 69.82 |
| | PM. Init. | 82.19 | 69.76 |
| + 4-way tensor | Rnd. Init. | 81.63 | 70.63 |
| | PM. Init. | 82.51 | 70.77 |

Table 5: SRL labeled F-score for different initialization strategies of the first order model. Rnd stands for the random initialization, and PM for the power method initialization.

Table 5 shows the impact of initialization on the performance of the tensor-based model. The initialization based on the power method yields superior results compared to random initialization, for both

| Language | Test set | | | |
|---|---|---|---|---|
| | Ours (4-way tensor) | Ours (no tensor) | CoNLL 1st | CoNLL 2nd |
| English | **82.51*** | 80.84 | 82.08 | 81.20 |
| Catalan | 74.67 | 71.86 | **76.78*** | 74.02 |
| Chinese | **69.16*** | 68.43 | 68.52 | 68.71 |
| German | **76.94** | 74.03 | 74.65 | 76.27 |
| Spanish | 75.58 | 72.85 | **77.33*** | 74.01 |
| Average | 75.77 | 73.60 | 75.87 | 74.84 |

Table 4: Semantic labeled F-score excluding predicate senses on 5 languages in the CoNLL-2009 shared task. Statistical significance with $p < 0.05$ is marked with $*$. Adding the tensor leads to more than 2% absolute gain on average F-score. Our method with the same feature configuration (a standard set + 4-way tensor) rivals the best CoNLL-2009 system which explores much richer feature sets, language-specific feature engineering, and n-best parse combination (Zhao et al., 2009a).

| **Our method** | **WSJ-test** | **Gain** |
|---|---|---|
| 1st order w/o tensor | 80.84 | – |
| + 4-way tensor | 82.51 | +1.67 |
| + 3-way tensor | 82.19 | +1.35 |

| **(Roth and Woodsend)** | **WSJ-test** | **Gain** |
|---|---|---|
| original baseline | 80.38 | – |
| + pred & arg | 80.23 | -0.15 |
| + deppath | 80.63 | +0.25 |
| + span | 80.87 | +0.49 |

Table 6: Comparision between our low-rank tensor method and (Roth and Woodsend, 2014) for leveraing word compositions.

3-way and 4-way tensors. However, random initialization still delivers reasonable performance, outperforming the tensor-free model by more than 1% in F-score.

Finally, we compare our tensor-based approach against a simpler model that captures interactions between predicate, argument and syntactic path using word embeddings (Roth and Woodsend, 2014). Table 6 demonstrates that modeling feature interactions using tensor yields higher gains than using word embeddings alone. For instance, the highest gain achieved by Roth and Woodsend (2014) when the embeddings of the arguments are averaged is 0.5%, compared to 1.6% obtained by our model.

## 8 Conclusions

In this paper we introduce a tensor-based approach to SRL that induces a compact feature representation for words and their relations. In this sense, our dimensionality reduction method provides a clear alternative to a traditional feature engineering approach used in SRL. Augmenting a simple, yet competitive SRL model with the tensor component yields significant performance gains. We demonstrate that our full model outperforms the best performing systems on the CoNLL-2009 shared task.

## Acknowledgments

The authors acknowledge the support of the MURI program (W911NF-10-1-0533) and the DARPA BOLT program. This research is developed in a collaboration of MIT with the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the Interactive sYstems for Answer Search (IYAS) project. We are grateful to Anders Bjökelund and Michael Roth for providing the outputs of their systems. We thank Yu Xin, Tommi Jaakkola, the MIT NLP group and the ACL reviewers for their comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

## References

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*. Association for Computational Linguistics.

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 49–54, Boulder, Colorado, June. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 11(Aug):2493–2537.

Lieven De Lathauwer, Pierre Comon, Bart De Moor, and Joos Vandewalle. 1995. Higher-order power method. *Nonlinear Theory and its Applications, NOLTA95*, 1.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the Latent Words Language Model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 21–29, Singapore, August. Association for Computational Linguistics.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.

Richard Johansson. 2009. Statistical bistratal dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 561–569, Singapore.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland, June. Association for Computational Linguistics.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *Transactions of the Association for Computational Linguistics*, 1.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 471–476, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*, 60(1):11–39.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo, Japan.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 88–94, Barcelona, Spain, July. Association for Computational Linguistics.

Haitong Yang and Chengqing Zong. 2014. Multi-predicate semantic role labeling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 363–373, Doha, Qatar, October. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics.

Benat Zapirain, Eneko Agirre, Lluís Màrquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39(3):631–664.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 61–66. Association for Computational Linguistics.

Hai Zhao, Wenliang Chen, Chunyu Kity, and Guodong Zhou. 2009b. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Association for Computational Linguistics.