Deep Multitask Learning for Semantic Dependency Parsing

Hao Peng[*]   Sam Thomson[†]   Noah A. Smith[*]

[*]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
{hapeng,nasmith}@cs.washington.edu, sthomson@cs.cmu.edu

Abstract

We present a deep neural architecture that parses sentences into three semantic dependency graph formalisms. By using efficient, nearly arc-factored inference and a bidirectional-LSTM composed with a multi-layer perceptron, our base system is able to significantly improve the state of the art for semantic dependency parsing, without using hand-engineered features or syntax. We then explore two multitask learning approaches—one that shares parameters across formalisms, and one that uses higher-order structures to predict the graphs jointly. We find that both approaches improve performance across formalisms on average, achieving a new state of the art. Our code is open-source and available at https://github.com/Noahs-ARK/NeurboParser.

1 Introduction

Labeled directed graphs are a natural and flexible representation for semantics (Copestake et al., 2005; Baker et al., 2007; Surdeanu et al., 2008; Banarescu et al., 2013, *inter alia*). Their generality over trees, for instance, allows them to represent relational semantics while handling phenomena like coreference and coordination. Even syntactic formalisms are moving toward graphs (de Marneffe et al., 2014). However, full semantic graphs can be expensive to annotate, and efforts are fragmented across competing semantic theories, leading to a limited number of annotations in any one formalism. This makes learning to parse more difficult, especially for powerful but data-hungry machine learning techniques like neural networks.

In this work, we hypothesize that the overlap among theories and their corresponding represen-
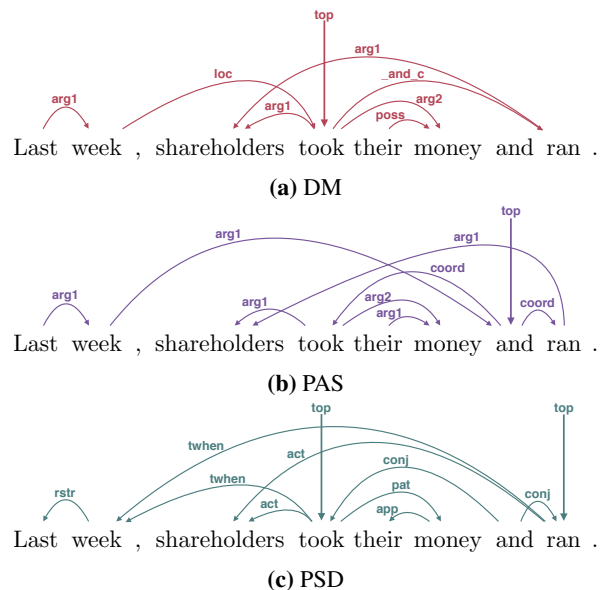

Figure 1: An example sentence annotated with the three semantic formalisms of the broad-coverage semantic dependency parsing shared tasks.

tations can be exploited using multitask learning (Caruana, 1997), allowing us to learn from more data. We use the 2015 SemEval shared task on Broad-Coverage Semantic Dependency Parsing (**SDP**; Oepen et al., 2015) as our testbed. The shared task provides an English-language corpus with parallel annotations for three semantic graph representations, described in §2. Though the shared task was designed in part to encourage comparison between the formalisms, we are the first to treat SDP as a multitask learning problem.

As a strong baseline, we introduce a new system that parses each formalism separately (§3). It uses a bidirectional-LSTM composed with a multi-layer perceptron to score arcs and predicates, and has efficient, nearly arc-factored inference. Experiments show it significantly improves on state-of-the-art methods (§3.4).

We then present two multitask extensions (§4.2

2037

	DM		PAS		PSD	
	id	ood	id	ood	id	ood
# labels	59	47	42	41	91	74
% trees	2.3	9.7	1.2	2.4	42.2	51.4
% projective	2.9	8.8	1.6	3.5	41.9	54.4

Table 1: Graph statistics for in-domain (WSJ, "id") and out-of-domain (Brown corpus, "ood") data. Numbers taken from Oepen et al. (2015).

and §4.3), with a parameterization and factorization that implicitly models the relationship between multiple formalisms. Experiments show that both techniques improve over our basic model, with an additional (but smaller) improvement when they are combined (§4.5). Our analysis shows that the improvement in unlabeled F_1 is greater for the two formalisms that are more structurally similar, and suggests directions for future work. Finally, we survey related work (§5), and summarize our contributions and findings (§6).

2 Broad-Coverage Semantic Dependency Parsing (SDP)

First defined in a SemEval 2014 shared task (Oepen et al., 2014), and then extended by Oepen et al. (2015), the broad-coverage semantic dependency parsing (**SDP**) task is centered around three semantic formalisms whose annotations have been converted into bilexical dependencies. See Figure 1 for an example. The formalisms come from varied linguistic traditions, but all three aim to capture predicate-argument relations between content-bearing words in a sentence.

While at first glance similar to syntactic dependencies, semantic dependencies have distinct goals and characteristics, more akin to semantic role labeling (SRL; Gildea and Jurafsky, 2002) or the abstract meaning representation (AMR; Banarescu et al., 2013). They abstract over different syntactic realizations of the same or similar meaning (e.g., *"She gave me the ball."* vs. *"She gave the ball to me."*). Conversely, they attempt to distinguish between different senses even when realized in similar syntactic forms (e.g., *"I baked in the kitchen."* vs. *"I baked in the sun."*).

Structurally, they are labeled directed graphs whose vertices are tokens in the sentence. This is in contrast to AMR whose vertices are abstract concepts, with no explicit alignment to tokens, which makes parsing more difficult (Flanigan et al., 2014). Their arc labels encode broadly-

applicable semantic relations rather than being tailored to any specific downstream application or ontology.[1] They are not necessarily trees, because a token may be an argument of more than one predicate (e.g., in *"John wants to eat,"* John is both the wanter and the would-be eater). Their analyses may optionally leave out non–content-bearing tokens, such as punctuation or the infinitival *"to,"* or prepositions that simply mark the type of relation holding between other words. But when restricted to content-bearing tokens (including adjectives, adverbs, etc.), the subgraph is connected. In this sense, SDP provides a *whole-sentence* analysis. This is in contrast to PropBank-style SRL, which gives an analysis of only verbal and nominal predicates (Palmer et al., 2005). Semantic dependency graphs also tend to have higher levels of nonprojectivity than syntactic trees (Oepen et al., 2014). Sentences with graphs containing cycles have been removed from the dataset by the organizers, so all remaining graphs are directed acyclic graphs. Table 1 summarizes some of the dataset's high-level statistics.

Formalisms. Following the SemEval shared tasks, we consider three formalisms. The **DM** (DELPH-IN MRS) representation comes from DeepBank (Flickinger et al., 2012), which are manually-corrected parses from the LinGO English Resource Grammar (Copestake and Flickinger, 2000). LinGO is a head-driven phrase structure grammar (HPSG; Pollard and Sag, 1994) with minimal recursion semantics (Copestake et al., 2005). The **PAS** (Predicate-Argument Structures) representation is extracted from the Enju Treebank, which consists of automatic parses from the Enju HPSG parser (Miyao, 2006). PAS annotations are also available for the Penn Chinese Treebank (Xue et al., 2005). The **PSD** (Prague Semantic Dependencies) representation is extracted from the tectogrammatical layer of the Prague Czech-English Dependency Treebank (Hajič et al., 2012). PSD annotations are also available for a Czech translation of the WSJ Corpus. In this work, we train and evaluate only on English annotations.

Of the three, PAS follows syntax most closely, and prior work has found it the easiest to predict. PSD has the largest set of labels, and parsers

---

[1]This may make another disambiguation step necessary to use these representations in a downstream task, but there is evidence that modeling semantic composition separately from grounding in any ontology is an effective way to achieve broad coverage (Kwiatkowski et al., 2013).
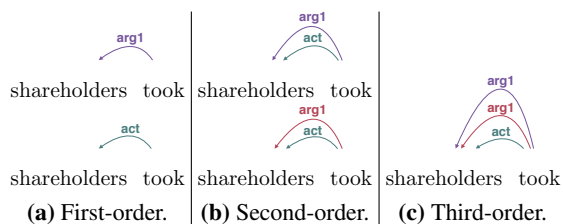
Figure 2: Examples of local structures. We refer to the number of arcs that a structure contains as its **order**.

have significantly lower performance on it (Oepen et al., 2015).

3   Single-Task SDP

Here we introduce our basic model, in which training and prediction for each formalism is kept completely separate. We also lay out basic notation, which will be reused for our multitask extensions.

3.1   Problem Formulation

The output of semantic dependency parsing is a labeled directed graph (see Figure 1). Each arc has a label from a predefined set $\mathcal{L}$, indicating the semantic relation of the child to the head. Given input sentence x, let $\mathcal{Y}(x)$ be the set of possible semantic graphs over x. The graph we seek maximizes a score function S:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}(x)} S(x, y), \tag{1}$$

We decompose S into a sum of local scores s for **local structures** (or "parts") p in the graph:

$$S(x, y) = \sum_{p \in y} s(p). \tag{2}$$

For notational simplicity, we omit the dependence of s on x. See Figure 2a for examples of local structures. s is a parameterized function, whose parameters (denoted Θ and suppressed here for clarity) will be learned from the training data (§3.3). Since we search over every possible labeled graph (i.e., considering each labeled arc for each pair of words), our approach can be considered a **graph-based** (or **all-pairs**) method. The models presented in this work all share this common graph-based approach, differing only in the set of structures they score and in the parameterization of the scoring function s. This approach also underlies state-of-the-art approaches to SDP (Martins and Almeida, 2014).

3.2   Basic Model

Our basic model is inspired by recent successes in neural arc-factored graph-based dependency parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017; Kuncoro et al., 2016). It borrows heavily from the neural arc-scoring architectures in those works, but decodes with a different algorithm under slightly different constraints.

3.2.1   Basic Structures

Our basic model factors over three types of structures (p in Equation 2):

- predicate, indicating a predicate word, denoted $i \rightarrow \cdot$;
- unlabeled arc, representing the existence of an arc from a predicate to an argument, denoted $i \rightarrow j$;
- labeled arc, an arc labeled with a semantic role, denoted $i \xrightarrow{\ell} j$.

Here i and j are word indices in a given sentence, and ℓ indicates the arc label. This list corresponds to the most basic structures used by Martins and Almeida (2014). Selecting an output y corresponds precisely to selecting which instantiations of these structures are included.

To ensure the internal consistency of predictions, the following constraints are enforced during decoding:

- $i \rightarrow \cdot$ if and only if there exists at least one j such that $i \rightarrow j$;
- If $i \rightarrow j$, then there must be exactly one label ℓ such that $i \xrightarrow{\ell} j$. Conversely, if not $i \rightarrow j$, then there must not exist any $i \xrightarrow{\ell} j$;

We also enforce a **determinism** constraint (Flanigan et al., 2014): certain labels must not appear on more than one arc emanating from the same token. The set of deterministic labels is decided based on their appearance in the training set. Notably, we do not enforce that the predicted graph is connected or spanning. If not for the predicate and determinism constraints, our model would be **arc-factored**, and decoding could be done for each i, j pair independently. Our structures do overlap though, and we employ AD[3] (Martins et al., 2011) to find the highest-scoring internally consistent semantic graph. AD[3] is an approximate discrete optimization algorithm based on dual decomposition. It can be used to decode factor graphs over discrete variables when scored structures overlap, as is the case here.

3.2.2 Basic Scoring

Similarly to Kiperwasser and Goldberg (2016), our model learns representations of tokens in a sentence using a bi-directional LSTM (BiLSTM). Each different type of structure (predicate, unlabeled arc, labeled arc) then shares these same BiLSTM representations, feeding them into a multi-layer perceptron (MLP) which is specific to the structure type. We present the architecture slightly differently from prior work, to make the transition to the multitask scenario (§4) smoother. In our presentation, we separate the model into a function ϕ that represents the input (corresponding to the BiLSTM and the initial layers of the MLPs), and a function ψ that represents the output (corresponding to the final layers of the MLPs), with the scores given by their inner product.[2]

Distributed input representations. Long short-term memory networks (LSTMs) are a variant of recurrent neural networks (RNNs) designed to alleviate the vanishing gradient problem in RNNs (Hochreiter and Schmidhuber, 1997). A bi-directional LSTM (BiLSTM) runs over the sequence in both directions (Schuster and Paliwal, 1997; Graves, 2012).

Given an input sentence x and its corresponding part-of-speech tag sequence, each token is mapped to a concatenation of its word embedding vector and POS tag vector. Two LSTMs are then run in opposite directions over the input vector sequence, outputting the concatenation of the two hidden vectors at each position i: $\mathbf{h}_i = \left[\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i\right]$ (we omit $\mathbf{h}_i$'s dependence on x and its own parameters). $\mathbf{h}_i$ can be thought of as an encoder that contextualizes each token conditioning on all of its context, without any Markov assumption. $\mathbf{h}$'s parameters are learned jointly with the rest of the model (§3.3); we refer the readers to Cho (2015) for technical details.

The input representation ϕ of a predicate structure depends on the representation of one word:

$$\phi(i{\rightarrow}\cdot) = \tanh\left(\mathbf{C}_{\text{pred}}\mathbf{h}_i + \mathbf{b}_{\text{pred}}\right). \quad (3a)$$
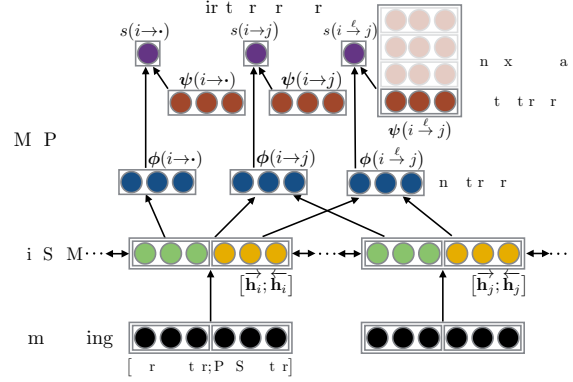

Figure 3: Illustration of the architecture of the basic model. i and j denote the indices of tokens in the given sentence. The figure depicts single-layer BiLSTM and MLPs, while in practice we use two layers for both.

For unlabeled arc and labeled arc structures, it depends on both the head and the modifier (but not the label, which is captured in the distributed output representation):

$$\phi(i{\rightarrow}j) = \tanh\left(\mathbf{C}_{\text{UA}}\left[\mathbf{h}_i; \mathbf{h}_j\right] + \mathbf{b}_{\text{UA}}\right), \quad (3b)$$

$$\phi(i \xrightarrow{\ell} j) = \tanh\left(\mathbf{C}_{\text{LA}}\left[\mathbf{h}_i; \mathbf{h}_j\right] + \mathbf{b}_{\text{LA}}\right). \quad (3c)$$

Distributed output representations. NLP researchers have found that embedding discrete output labels into a low dimensional real space is an effective way to capture commonalities among them (Srikumar and Manning, 2014; Hermann et al., 2014; FitzGerald et al., 2015, *inter alia*). In neural language models (Bengio et al., 2003; Mnih and Hinton, 2007, *inter alia*) the weights of the output layer could also be regarded as an output embedding.

We associate each first-order structure p with a d-dimensional real vector $\psi(p)$ which does not depend on particular words in p. Predicates and unlabeled arcs are each mapped to a single vector:

$$\psi(i{\rightarrow}\cdot) = \psi_{\text{pred}}, \quad (4a)$$

$$\psi(i{\rightarrow}j) = \psi_{\text{UA}}, \quad (4b)$$

and each label gets a vector:

$$\psi(i \xrightarrow{\ell} j) = \psi_{\text{LA}}(\ell). \quad (4c)$$

Scoring. Finally, we use an inner product to score first-order structures:

$$s(p) = \phi(p) \cdot \psi(p). \quad (5)$$

Figure 3 illustrates our basic model's architecture.

3.3 Learning

The parameters of the model are learned using a max-margin objective. Informally, the goal is to learn parameters for the score function so that the gold parse is scored over every incorrect parse with a margin proportional to the cost of the incorrect parse. More formally, let $\mathcal{D} = \left\{ (x_i, y_i) \right\}_{i=1}^{N}$ be the training set consisting of N pairs of sentence x_i and its gold parse y_i. Training is then the following ℓ_2-regularized empirical risk minimization problem:

$$\min_{\Theta} \frac{\lambda}{2} \|\Theta\|^2 + \frac{1}{N} \sum_{i=1}^{N} L(x_i, y_i; \Theta), \quad (6)$$

where Θ is all parameters in the model, and L is the structured hinge loss:

$$L(x_i, y_i; \Theta) = \max_{y \in \mathcal{Y}(x_i)} \left\{ S(x_i, y) + c(y, y_i) \right\} \\ - S(x_i, y_i). \quad (7)$$

c is a weighted Hamming distance that trades off between precision and recall (Taskar et al., 2004). Following Martins and Almeida (2014), we encourage recall over precision by using the costs 0.6 for false negative arc predictions and 0.4 for false positives.

3.4 Experiments

We evaluate our basic model on the English dataset from SemEval 2015 Task 18 closed track.[3] We split as in previous work (Almeida and Martins, 2015; Du et al., 2015), resulting in 33,964 training sentences from §00–19 of the WSJ corpus, 1,692 development sentences from §20, 1,410 sentences from §21 as in-domain test data, and 1,849 sentences sampled from the Brown Corpus as out-of-domain test data.

The *closed* track differs from the *open* and *gold* tracks in that it does not allow access to any syntactic analyses. In the open track, additional machine generated syntactic parses are provided, while the gold-track gives access to various gold-standard syntactic analyses. Our model is evaluated with closed track data; it does not have access to any syntactic analyses during training or test.

We refer the readers to §4.4 for implementation details, including training procedures, hyperparameters, pruning techniques, etc..

	Model	DM	PAS	PSD	Avg.
	Du et al., 2015	89.1	91.3	75.7	86.3
id	A&M, 2015	88.2	90.9	76.4	86.0
	BASIC	**89.4**	**92.2**	**77.6**	**87.4**
	Du et al., 2015	81.8	87.2	73.3	81.7
ood	A&M, 2015	81.8	86.9	74.8	82.0
	BASIC	**84.5**	**88.3**	**75.3**	**83.6**

Table 2: Labeled parsing performance (F_1 score) on both in-domain (id) and out-of-domain (ood) test data. The last column shows the micro-average over the three tasks. Bold font indicates best performance without syntax. Underlines indicate statistical significance with Bonferroni (1936) correction compared to the best baseline system.[4]

Empirical results. As our model uses no explicit syntactic information, the most comparable models to ours are two state-of-the-art closed track systems due to Du et al. (2015) and Almeida and Martins (2015). Du et al. (2015) rely on graph-tree transformation techniques proposed by Du et al. (2014), and apply a voting ensemble to well-studied tree-oriented parsers. Closely related to ours is Almeida and Martins (2015), who used rich, hand-engineered second-order features and AD³ for inference.

Table 2 compares our basic model to both baseline systems (labeled F_1 score) on SemEval 2015 Task 18 test data. Scores of those systems are repeated from the official evaluation results. Our basic model significantly outperforms the best published results with a 1.1% absolute improvement on the in-domain test set and 1.6% on the out-of-domain test set.

4 Multitask SDP

We introduce two extensions to our single-task model, both of which use training data for all three formalisms to improve performance on each formalism's parsing task. We describe a first-order model, where representation functions are enhanced by parameter sharing while inference is kept separate for each task (§4.2). We then introduce a model with cross-task higher-order structures that uses joint inference *across* different tasks (§4.3). Both multitask models use AD³ for decoding, and are trained with the same margin-based objective, as in our single-task model.

4.1 Problem Formulation

We will use an additional superscript $t \in \mathcal{T}$ to distinguish the three tasks (e.g., $y^{(t)}$, $\phi^{(t)}$), where $\mathcal{T} = \{\mathrm{DM}, \mathrm{PAS}, \mathrm{PSD}\}$. Our task is now to predict three graphs $\{y^{(t)}\}_{t \in \mathcal{T}}$ for a given input sentence x. Multitask SDP can also be understood as parsing x into a single unified *multigraph* $y = \bigcup_{t \in \mathcal{T}} y^{(t)}$. Similarly to Equations 1–2, we decompose y's score $S(x, y)$ into a sum of local scores for local structures in y, and we seek a multigraph $\hat{y}$ that maximizes $S(x, y)$.

4.2 Multitask SDP with Parameter Sharing

A common approach when using BiLSTMs for multitask learning is to share the BiLSTM part of the model across tasks, while training specialized classifiers for each task (Søgaard and Goldberg, 2016). In this spirit, we let each task keep its own specialized MLPs, and explore two variants of our model that share parameters at the BiLSTM level.

The first variant consists of a set of task-specific BiLSTM encoders as well as a common one that is shared across all tasks. We denote it FREDA. FREDA uses a neural generalization of "frustratingly easy" domain adaptation (Daumé III, 2007; Kim et al., 2016), where one augments domain-specific features with a shared set of features to capture global patterns. Formally, let $\{\mathbf{h}^{(t)}\}_{t \in \mathcal{T}}$ denote the three task-specific encoders. We introduce another encoder $\widetilde{\mathbf{h}}$ that is shared across all tasks. Then a new set of input functions $\{\phi^{(t)}\}_{t \in \mathcal{T}}$ can be defined as in Equations 3a–3c, for example:

$$\phi^{(t)}(i \xrightarrow{\ell} j) = \tanh\big(\mathbf{C}_{\mathrm{LA}}^{(t)}\big[\mathbf{h}_i^{(t)}; \mathbf{h}_j^{(t)}; \\ \widetilde{\mathbf{h}}_i; \widetilde{\mathbf{h}}_j\big] + \mathbf{b}_{\mathrm{LA}}^{(t)}\big). \quad (8)$$

The predicate and unlabeled arc versions are analogous. The output representations $\{\psi^{(t)}\}$ remain task-specific, and the score is still the inner product between the input representation and the output representation.

The second variant, which we call SHARED, uses *only* the shared encoder $\widetilde{\mathbf{h}}$, and doesn't use task-specific encoders $\{\mathbf{h}^{(t)}\}$. It can be understood as a special case of FREDA where the dimensions of the task-specific encoders are 0.

4.3 Multitask SDP with Cross-Task Structures

In syntactic parsing, higher-order structures have commonly been used to model interactions be-

tween multiple adjacent arcs in the same dependency tree (Carreras, 2007; Smith and Eisner, 2008; Martins et al., 2009; Zhang et al., 2014, *inter alia*). Lluís et al. (2013), in contrast, used second-order structures to jointly model syntactic dependencies and semantic roles. Similarly, we use higher-order structures *across* tasks instead of *within* tasks. In this work, we look at interactions between arcs that share the same head and modifier.[5] See Figures 2b and 2c for examples of higher-order cross-task structures.

Higher-order structure scoring. Borrowing from Lei et al. (2014), we introduce a low-rank tensor scoring strategy that, given a higher-order structure p, models interactions between the first-order structures (i.e., arcs) p is made up of. This approach builds on and extends the parameter sharing techniques in §4.2. It can either follow FREDA or SHARED to get the input representations for first-order structures.

We first introduce basic tensor notation. The **order** of a tensor is the number of its dimensions. The **outer product** of two vectors forms a second-order tensor (matrix) where $[\mathbf{u} \otimes \mathbf{v}]_{i,j} = u_i v_j$. We denote the **inner product** of two tensors of the same dimensions by $\langle \cdot, \cdot \rangle$, which first takes their element-wise product, then sums all the elements in the resulting tensor.

For example, let p be a labeled third-order structure, including one labeled arc from each of the three different tasks: $p = \{p^{(t)}\}_{t \in \mathcal{T}}$. Intuitively, $s(p)$ should capture every pairwise interaction between the three input and three output representations of p. Formally, we want the score function to include a parameter for each term in the outer product of the representation vectors: $s(p) =$

$$\left\langle \mathcal{W}, \bigotimes_{t \in \mathcal{T}} \left(\phi^{(t)}\left(p^{(t)}\right) \otimes \psi^{(t)}\left(p^{(t)}\right) \right) \right\rangle, \quad (9)$$

where $\mathcal{W}$ is a sixth-order tensor of parameters.[6]

With typical dimensions of representation vectors, this leads to an unreasonably large number of

---

[5]In the future we hope to model structures over larger motifs, both across and within tasks, to potentially capture when an arc in one formalism corresponds to a path in another formalism, for example.

[6]This is, of course, not the only way to model interactions between several representations. For instance, one could concatenate them and feed them into another MLP. Our preliminary experiments in this direction suggested that it may be less effective given a similar number of parameters, but we did not run full experiments.

parameters. Following Lei et al. (2014), we upper-bound the rank of $\mathcal{W}$ by r to limit the number of parameters (r is a hyperparameter, decided empirically). Using the fact that a tensor of rank at most r can be decomposed into a sum of r rank-1 tensors (Hitchcock, 1927), we reparameterize $\mathcal{W}$ to enforce the low-rank constraint by construction:

$$\mathcal{W} = \sum_{j=1}^{r} \bigotimes_{t \in \mathcal{T}} \left(\left[\mathbf{U}_{\text{LA}}^{(t)} \right]_{j,:} \otimes \left[\mathbf{V}_{\text{LA}}^{(t)} \right]_{j,:} \right), \quad (10)$$

where $\mathbf{U}_{\text{LA}}^{(t)}, \mathbf{V}_{\text{LA}}^{(t)} \in \mathbb{R}^{r \times d}$ are now our parameters. $[\cdot]_{j,:}$ denotes the jth row of a matrix. Substituting this back into Equation 9 and rearranging, the score function $s(p)$ can then be rewritten as:

$$\sum_{j=1}^{r} \prod_{t \in \mathcal{T}} \left[\mathbf{U}_{\text{LA}}^{(t)} \boldsymbol{\phi}^{(t)} \left(p^{(t)} \right) \right]_j \left[\mathbf{V}_{\text{LA}}^{(t)} \boldsymbol{\psi}^{(t)} \left(p^{(t)} \right) \right]_j.$$
$$(11)$$

We refer readers to Kolda and Bader (2009) for mathematical details.

For labeled higher-order structures our parameters consist of the set of six matrices, $\{\mathbf{U}_{\text{LA}}^{(t)}\} \cup \{\mathbf{V}_{\text{LA}}^{(t)}\}$. These parameters are shared between second-order and third-order labeled structures. Labeled *second-order* structures are scored as Equation 11, but with the product extending over only the two relevant tasks. Concretely, only four of the representation functions are used rather than all six, along with the four corresponding matrices from $\{\mathbf{U}_{\text{LA}}^{(t)}\} \cup \{\mathbf{V}_{\text{LA}}^{(t)}\}$. *Unlabeled* cross-task structures are scored analogously, reusing the same representations, but with a separate set of parameter matrices $\{\mathbf{U}_{\text{UA}}^{(t)}\} \cup \{\mathbf{V}_{\text{UA}}^{(t)}\}$.

Note that we are not doing tensor factorization; we are learning $\mathbf{U}_{\text{LA}}^{(t)}$, $\mathbf{V}_{\text{LA}}^{(t)}$, $\mathbf{U}_{\text{UA}}^{(t)}$, and $\mathbf{V}_{\text{UA}}^{(t)}$ directly, and $\mathcal{W}$ is never explicitly instantiated.

Inference and learning. Given a sentence, we use AD^3 to jointly decode all three formalisms.[7] The training objective used for learning is the sum of the losses for individual tasks.

4.4 Implementation Details

Each input token is mapped to a concatenation of three real vectors: a pre-trained word vector; a randomly-initialized word vector; and a randomly-initialized POS tag vector.[8] All three are updated

Hyperparameter	Value
Pre-trained word embedding dimension	100
Randomly-initialized word embedding dimension	25
POS tag embedding dimension	25
Dimensions of representations ϕ and ψ	100
MLP layers	2
BiLSTM layers	2
BiLSTM dimensions	200
Rank of tensor r	100
α for word dropout	0.25

Table 3: Hyperparameters used in the experiments.

during training. We use 100-dimensional `GloVe` (Pennington et al., 2014) vectors trained over Wikipedia and Gigaword as pre-trained word embeddings. To deal with out-of-vocabulary words, we apply word dropout (Iyyer et al., 2015) and randomly replace a word w with a special unk-symbol with probability $\frac{\alpha}{1+\#(w)}$, where $\#(w)$ is the count of w in the training set.

Models are trained for up to 30 epochs with Adam (Kingma and Ba, 2015), with $\beta_1 = \beta_2 = 0.9$, and initial learning rate $\eta_0 = 10^{-3}$. The learning rate η is annealed at a rate of 0.5 every 10 epochs (Dozat and Manning, 2017). We apply early-stopping based on the labeled F_1 score on the development set.[9] We set the maximum number of iterations of AD^3 to 500 and round decisions when it doesn't converge. We clip the ℓ_2 norm of gradients to 1 (Graves, 2013; Sutskever et al., 2014), and we do not use mini-batches. Randomly initialized parameters are sampled from a uniform distribution over $\left[-\sqrt{6/(d_r + d_c)}, \sqrt{6/(d_r + d_c)} \right]$, where d_r and d_c are the number of the rows and columns in the matrix, respectively. An ℓ_2 penalty of $\lambda = 10^{-6}$ is applied to all weights. Other hyperparameters are summarized in Table 3.

We use the same pruner as Martins and Almeida (2014), where a first-order feature-rich unlabeled pruning model is trained for each task, and arcs with posterior probability below 10^{-4} are discarded. We further prune labeled structures that appear less than 30 times in the training set. In the development set, about 10% of the arcs remain after pruning, with a recall of around 99%.

---

[7]Joint inference comes at a cost; our third-order model is able to decode roughly 5.2 sentences (i.e., 15.5 task-specific graphs) per second on a single Xeon E5-2690 2.60GHz CPU.

[8]There are minor differences in the part-of-speech data provided with the three formalisms. For the basic models, we use the POS tags provided with the respective dataset; for the multitask models, we use the (automatic) POS tags provided with DM.

[9]Micro-averaged labeled F_1 for the multitask models.

4.5 Experiments

Experimental settings. We compare four multi-task variants to the basic model, as well as the two baseline systems introduced in §3.4.

- SHARED1 is a first-order model. It uses a single shared BiLSTM encoder, and keeps the inference separate for each task.
- FREDA1 is a first-order model based on "frustratingly easy" parameter sharing. It uses a shared encoder as well as task-specific ones. The inference is kept separate for each task.
- SHARED3 is a third-order model. It follows SHARED1 and uses a single shared BiLSTM encoder, but additionally employs cross-task structures and inference.
- FREDA3 is also a third-order model. It combines FREDA1 and SHARED3 by using both "frustratingly easy" parameter sharing and cross-task structures and inference.

In addition, we also examine the effects of syntax by comparing our models to the state-of-the-art open track system (Almeida and Martins, 2015).[10]

Main results overview. Table 4a compares our models to the best published results (labeled F_1 score) on SemEval 2015 Task 18 in-domain test set. Our basic model improves over all closed track entries in all formalisms. It is even with the best open track system for DM and PSD, but improves on PAS and on average, without making use of any syntax. Three of our four multitask variants further improve over our basic model; SHARED1's differences are statistically insignificant. Our best models (SHARED3, FREDA3) outperform the previous state-of-the-art closed track system by 1.7% absolute F_1, and the best open track system by 0.9%, without the use of syntax.

We observe similar trends on the out-of-domain test set (Table 4b), with the exception that, on PSD, our best-performing model's improvement over the open-track system of Almeida and Martins (2015) is not statistically significant.

The extent to which we might benefit from syntactic information remains unclear. With automatically generated syntactic parses, Almeida and Martins (2015) manage to obtain more than 1% absolute improvements over their closed track en-

	DM	PAS	PSD	Avg.
Du et al., 2015	89.1	91.3	75.7	86.3
A&M, 2015 (closed)	88.2	90.9	76.4	86.0
A&M, 2015 (open)†	89.4	91.7	77.6	87.1
BASIC	89.4	_92.2_	77.6	87.4
SHARED1	89.7	91.9	77.8	87.4
FREDA1	_90.0_	_92.3_	_78.1_	_87.7_
SHARED3	_90.3_	_92.5_	**_78.5_**	**_88.0_**
FREDA3	**_90.4_**	**_92.7_**	**_78.5_**	**_88.0_**

(a) Labeled F_1 score on the in-domain test set.

	DM	PAS	PSD	Avg.
Du et al., 2015	81.8	87.2	73.3	81.7
A&M, 2015 (closed)	81.8	86.9	74.8	82.0
A&M, 2015 (open)†	83.8	87.6	76.2	83.3
BASIC	_84.5_	_88.3_	75.3	83.6
SHARED1	_84.4_	_88.1_	75.4	83.5
FREDA1	_84.9_	_88.3_	75.8	_83.9_
SHARED3	**_85.3_**	_88.4_	76.1	_84.1_
FREDA3	**_85.3_**	**_89.0_**	76.4	**_84.4_**

(b) Labeled F_1 score on the out-of-domain test set.

Table 4: The last columns show the micro-average over the three tasks. † denotes the use of syntactic parses. Bold font indicates best performance among all systems, and underlines indicate statistical significance with Bonferroni correction compared to A&M, 2015 (open), the strongest baseline system.

try, which is consistent with the extensive evaluation by Zhang et al. (2016), but we leave the incorporation of syntactic trees to future work. Syntactic parsing could be treated as yet another output task, as explored in Lluís et al. (2013) and in the transition-based frameworks of Henderson et al. (2013) and Swayamdipta et al. (2016).

Effects of structural overlap. We hypothesized that the overlap between formalisms would enable multitask learning to be effective; in this section we investigate in more detail how structural overlap affected performance. By looking at *undirected* overlap between unlabeled arcs, we discover that modeling only arcs in the same direction may have been a design mistake.

DM and PAS are more structurally similar to each other than either is to PSD. Table 5 compares the structural similarities between the three for-

---
[10]Kanerva et al. (2015) was the winner of the gold track, which overall saw higher performance than the closed and open tracks. Since gold-standard syntactic analyses are not available in most realistic scenarios, we do not include it in this comparison.

| | Undirected | | | Directed | | |
	DM	PAS	PSD	DM	PAS	PSD
DM	-	67.2	56.8	-	64.2	26.1
PAS	70.0	-	54.9	66.9	-	26.1
PSD	57.4	56.3	-	26.4	29.6	-

Table 5: Pairwise structural similarities between the three formalisms in unlabeled F_1 score. Scores from Oepen et al. (2015).

| | DM | | PAS | | PSD | |
	UF	LF	UF	LF	UF	LF
FREDA1	91.7	90.4	93.1	91.6	89.0	79.8
FREDA3	91.9	90.8	93.4	92.0	88.6	80.4

Table 6: Unlabeled (UF) and labeled (LF) parsing performance of FREDA1 and FREDA3 on the development set of SemEval 2015 Task 18.

malisms in unlabeled F_1 score (each formalism's gold-standard unlabeled graph is used as a prediction of each other formalism's gold-standard unlabeled graph). All three formalisms have more than 50% overlap when ignoring arcs' directions, but considering direction, PSD is clearly different; PSD reverses the direction about half of the time it shares an edge with another formalism. A concrete example can be found in Figure 1, where DM and PAS both have an arc from *"Last"* to *"week,"* while PSD has an arc from *"week"* to *"Last."*

We can compare FREDA3 to FREDA1 to isolate the effect of modeling higher-order structures. Table 6 shows performance on the development data in both unlabeled and labeled F_1. We can see that FREDA3's unlabeled performance improves on DM and PAS, but *degrades* on PSD. This supports our hypothesis, and suggests that in future work, a more careful selection of structures to model might lead to further improvements.

5 Related Work

We note two important strands of related work.

Graph-based parsing. Graph-based parsing was originally invented to handle non-projective syntax (McDonald et al., 2005; Koo et al., 2010; Martins et al., 2013, *inter alia*), but has been adapted to semantic parsing (Flanigan et al., 2014; Martins and Almeida, 2014; Thomson et al., 2014; Kuhlmann, 2014, *inter alia*). Local structure scoring was traditionally done with linear models over hand-engineered features, but lately, various forms of representation learning

have been explored to learn feature combinations (Lei et al., 2014; Taub-Tabib et al., 2015; Pei et al., 2015, *inter alia*). Our work is perhaps closest to those who used BiLSTMs to encode inputs (Kiperwasser and Goldberg, 2016; Kuncoro et al., 2016; Wang and Chang, 2016; Dozat and Manning, 2017; Ma and Hovy, 2016).

Multitask learning in NLP. There have been many efforts in NLP to use joint learning to replace pipelines, motivated by concerns about cascading errors. Collobert and Weston (2008) proposed sharing the same word representation while solving multiple NLP tasks. Zhang and Weiss (2016) use a continuous stacking model for POS tagging and parsing. Ammar et al. (2016) and Guo et al. (2016) explored parameter sharing for multilingual parsing. Johansson (2013) and Kshirsagar et al. (2015) applied ideas from domain adaptation to multitask learning. Successes in multitask learning have been enabled by advances in representation learning as well as earlier explorations of parameter sharing (Ando and Zhang, 2005; Blitzer et al., 2006; Daumé III, 2007).

6 Conclusion

We showed two orthogonal ways to apply deep multitask learning to graph-based parsing. The first shares parameters when encoding tokens in the input with recurrent neural networks, and the second introduces interactions between output structures across formalisms. Without using syntactic parsing, these approaches outperform even state-of-the-art semantic dependency parsing systems that use syntax. Because our techniques apply to labeled directed graphs in general, they can easily be extended to incorporate more formalisms, semantic or otherwise. In future work we hope to explore cross-task scoring and inference for tasks where parallel annotations are not available. Our code is open-source and available at `https://github.com/Noahs-ARK/NeurboParser`.

Acknowledgements

References

Mariana S. C. Almeida and André F. T. Martins. 2015. Lisbon: Evaluating TurboSemanticParser on multiple languages and out-of-domain data. In *Proc. of SemEval*.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *TACL* 4:431–444.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR* 6:1817–1853.

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval'07 task 19: Frame semantic structure extraction. In *Proc. of SemEval*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. of LAW VII & ID*.

Yoshua Bengio, Rèjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *JMLR* 3:1137–1155.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*.

Carlo E. Bonferroni. 1936. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R. Istituto Superiore di Scienze Economiche e Commerciali di Firenze* 8:3–62.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of CoNLL*.

Rich Caruana. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Kyunghyun Cho. 2015. Natural language understanding with distributed representation. ArXiv:1511.07916.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proc. of LREC*.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language & Computation* 3(4):281–332.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proc. of LREC*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proc. of SemEval*.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proc. of SemEval*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of EMNLP*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: A dynamically annotated treebank of the Wall Street Journal. In *Proc. of TLT*.

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

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

Alex Graves. 2013. Generating sequences with recurrent neural networks. ArXiv 1308.0850.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2016. A universal framework for inductive transfer parsing across multi-typed treebanks. In *Proc. of COLING*.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proc. LREC*.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multi-lingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics* 39(4):949–998.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proc. of ACL*.

Frank L. Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematical Physics* 6(1):164–189.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proc. of NAACL*.

Jenna Kanerva, Juhani Luotolahti, and Filip Ginter. 2015. Turku: Semantic dependency parsing as a sequence classification. In *Proc. of SemEval*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proc. of COLING*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Review* 51(3):455–500.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proc. of ACL*.

Marco Kuhlmann. 2014. Linköping: Cubic-time graph parsing with a simple scoring scheme. In *Proc. of SemEval*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proc. of EMNLP*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proc. of EMNLP*.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proc. of ACL*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *TACL* 1:219–230.

Xuezhe Ma and Eduard Hovy. 2016. Neural probabilistic model for non-projective MST parsing. ArXiv 1701.00874.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proc. of SemEval*.

André F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

André F. T. Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL*.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Yusuke Miyao. 2006. From linguistic theory to syntactic analysis: Corpus-oriented grammar development and feature forest model.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proc. of ICML*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*.

Vivek Srikumar and Christopher D Manning. 2014. Learning distributed representations for structured output prediction. In *Proc. of NIPS*.

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 *Proc. of CoNLL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proc. of CoNLL*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*.

Hillel Taub-Tabib, Yoav Goldberg, and Amir Globerson. 2015. Template kernels for dependency parsing. In *Proc. of NAACL*.

Sam Thomson, Brendan O'Connor, Jeffrey Flanigan, David Bamman, Jesse Dodge, Swabha Swayamdipta, Nathan Schneider, Chris Dyer, and Noah A. Smith. 2014. CMU: Arc-factored, discriminative semantic dependency parsing. In *Proc. of SemEval*.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proc. of ACL*.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11(2):207–238.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proc. of EMNLP*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*.