Syntactic Scaffolds for Semantic Structures

Swabha Swayamdipta[♠]   **Sam Thomson**[♠]
Kenton Lee[◇]   **Luke Zettlemoyer**[♥]   **Chris Dyer**[♡]   **Noah A. Smith**[♥♣]

[♠]Language Technologies Institute, Carnegie Mellon University
[◇]Google AI Language
[♥]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♡]Google DeepMind
[♣]Allen Institute for Artificial Intelligence
{swabha,sammthomson}@cs.cmu.edu {kentonl,cdyer}@google.com
{lsz,nasmith}@cs.washington.edu

Abstract

We introduce the **syntactic scaffold**, an approach to incorporating syntactic information into semantic tasks. Syntactic scaffolds avoid expensive syntactic processing at runtime, only making use of a treebank during training, through a multitask objective. We improve over strong baselines on PropBank semantics, frame semantics, and coreference resolution, achieving competitive performance on all three tasks.

1 Introduction

As algorithms for the semantic analysis of natural language sentences have developed, the role of *syntax* has been repeatedly revisited. Linguistic theories have argued for a very tight integration of syntactic and semantic processing (Steedman, 2000; Copestake and Flickinger, 2000), and many systems have used syntactic dependency or phrase-based parsers as preprocessing for semantic analysis (Gildea and Palmer, 2002; Punyakanok et al., 2008; Das et al., 2014). Meanwhile, some recent methods forgo explicit syntactic processing altogether (Zhou and Xu, 2015; He et al., 2017; Lee et al., 2017; Peng et al., 2017).

Because annotated training datasets for semantics will always be limited, we expect that syntax—which offers an incomplete but potentially useful view of semantic structure—will continue to offer useful inductive bias, encouraging semantic models toward better generalization. We address the central question: is there a way for semantic analyzers to benefit from syntax without the computational cost of syntactic parsing?

We propose a multitask learning approach to incorporating syntactic information into learned representations of neural semantics models (§2). Our approach, the **syntactic scaffold**, minimizes an auxiliary supervised loss function, derived from a syntactic treebank. The goal is to steer the distributed, contextualized representations of words and spans toward accurate semantic *and* syntactic labeling. We avoid the cost of training or executing a full syntactic parser, and at test time (i.e., runtime in applications) the semantic analyzer has no additional cost over a syntax-free baseline. Further, the method does not assume that the syntactic treebank overlaps the dataset for the primary task.

Many semantic tasks involve labeling spans, including semantic role labeling (SRL; Gildea and Jurafsky, 2002) and coreference resolution (Ng, 2010) (tasks we consider in this paper), as well as named entity recognition and some reading comprehension and question answering tasks (Rajpurkar et al., 2016). These spans are usually syntactic constituents (cf. PropBank; Palmer et al., 2005), making phrase-based syntax a natural choice for a scaffold. See Figure 1 for an example sentence with syntactic and semantic annotations. Since the scaffold task is not an end in itself, we relax the syntactic parsing problem to a collection of independent span-level predictions, with no constraint that they form a valid parse tree. This means we never need to run a syntactic parsing algorithm.

Our experiments demonstrate that the syntactic scaffold offers a substantial boost to state-of-the-art baselines for two SRL tasks (§5) and coreference resolution (§6). Our models use the strongest available neural network architectures for these tasks, integrating deep representation learning (He et al., 2017) and structured prediction at the level of spans (Kong et al., 2016). For SRL, the base-
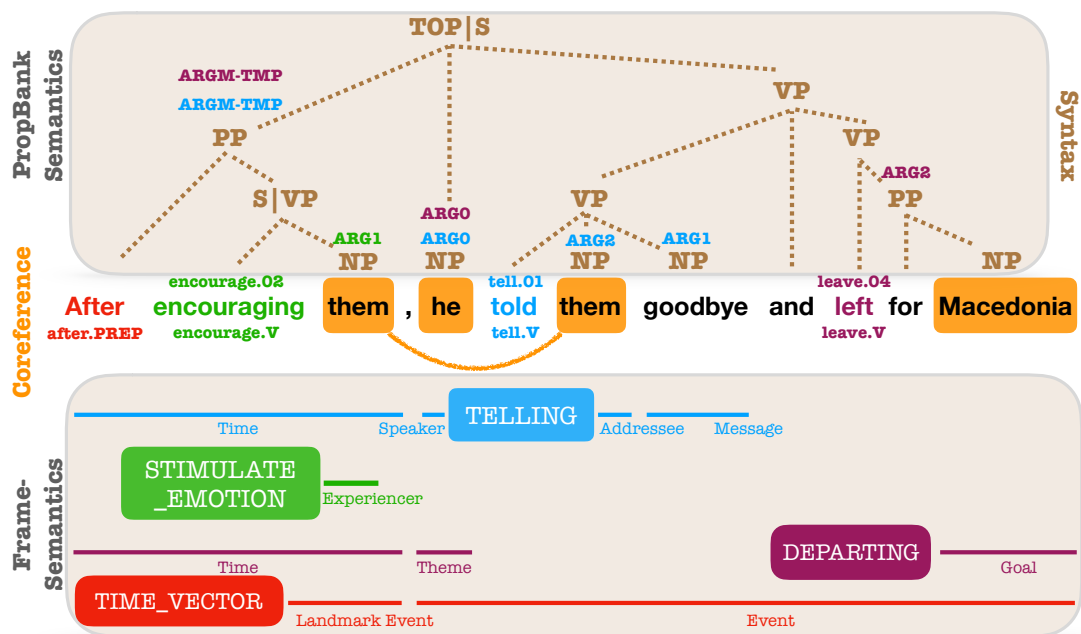
Figure 1: An example sentence with syntactic, PropBank and coreference annotations from OntoNotes, and author-annotated frame-semantic structures. PropBank SRL arguments and coreference mentions are annotated on top of syntactic constituents. All but one frame-semantic argument (EVENT) is a syntactic constituent. Targets evoke frames shown in the color-coded layers.

line itself is a novel globally normalized structured conditional random field, which outperforms the previous state of the art.[1] Syntactic scaffolds result in further improvements over prior work—3.6 absolute F_1 in FrameNet SRL, 1.1 absolute F_1 in PropBank SRL, and 0.6 F_1 in coreference resolution (averaged across three standard scores). Our code is open source and available at `https://github.com/swabhs/scaffolding`.

2 Syntactic Scaffolds

Multitask learning (Caruana, 1997) is a collection of techniques in which two or more tasks are learned from data with at least some parameters shared. We assume there is only one task about whose performance we are concerned, denoted T_1 (in this paper, T_1 is either SRL or coreference resolution). We use the term "scaffold" to refer to a second task, T_2, that can be combined with T_1 during multitask learning. A scaffold task is *only* used during training; it holds no intrinsic interest beyond biasing the learning of T_1, and after learning is completed, the scaffold is discarded.

A **syntactic scaffold** is a task designed to steer the (shared) model toward awareness of syntactic

structure. It could be defined through a syntactic parser that shares some parameters with T_1's model. Since syntactic parsing is costly, we use simpler syntactic prediction problems (discussed below) that do not produce whole trees.

As with multitask learning in general, we do not assume that the same data are annotated with outputs for T_1 and T_2. In this work, T_2 is defined using phrase-structure syntactic annotations from OntoNotes 5.0 (Weischedel et al., 2013; Pradhan et al., 2013). We experiment with three settings: one where the corpus for T_2 does not overlap with the training datasets for T_1 (frame-SRL) and two where there is a complete overlap (PropBank SRL and coreference). Compared to approaches which require multiple output labels over the same data, we offer the major advantage of not requiring any assumptions about, or specification of, the relationship between T_1 and T_2 output.

3 Related Work

We briefly contrast the syntactic scaffold with existing alternatives.

Pipelines. In a typical pipeline, T_1 and T_2 are separately trained, with the output of T_2 used to define the inputs to T_1 (Wolpert, 1992). Using syntax as T_2 in a pipeline is perhaps the most

---

[1]This excludes models initialized with deep, contextualized embeddings (Peters et al., 2018), an approach orthogonal to ours.

common approach for semantic structure prediction (Toutanova et al., 2008; Yang and Mitchell, 2017; Wiseman et al., 2016).[2] However, pipelines introduce the problem of cascading errors (T_2's mistakes affect the performance, and perhaps the training, of T_1; He et al., 2013). To date, remedies to cascading errors are so computationally expensive as to be impractical (e.g., Finkel et al., 2006). A syntactic scaffold is quite different from a pipeline since the output of T_2 is never explicitly used.

Latent variables. Another solution is to treat the output of T_2 as a (perhaps structured) latent variable. This approach obviates the need of supervision for T_2 and requires marginalization (or some approximation to it) in order to reason about the outputs of T_1. Syntax as a latent variable for semantics was explored by Zettlemoyer and Collins (2005) and Naradowsky et al. (2012). Apart from avoiding marginalization, the syntactic scaffold offers a way to use auxiliary syntactically-annotated data as direct supervision for T_2, and it need not overlap the T_1 training data.

Joint learning of syntax and semantics. The motivation behind joint learning of syntactic and semantic representations is that any one task is helpful in predicting the other (Lluís and Màrquez, 2008; Lluís et al., 2013; Henderson et al., 2013; Swayamdipta et al., 2016). This typically requires joint prediction of the outputs of T_1 and T_2, which tends to be computationally expensive at both training and test time.

Part of speech scaffolds. Similar to our work, there have been multitask models that use part-of-speech tagging as T_2, with transition-based dependency parsing (Zhang and Weiss, 2016) and CCG supertagging (Søgaard and Goldberg, 2016) as T_1. Both of the above approaches assumed parallel input data and used both tasks as supervision. Notably, we simplify our T_2, throwing away the structured aspects of syntactic parsing, whereas part-of-speech tagging has very little structure to begin with. While their approach results in improved token-level representations learned via supervision from POS tags, these must still be composed to obtain span representations. In-

stead, our approach learns span-level representations from phrase-type supervision directly, for semantic tasks. Additionally, these methods explore architectural variations in RNN layers for including supervision, whereas we focus on incorporating supervision with minimal changes to the baseline architecture. To the best of our knowledge, such simplified syntactic scaffolds have not been tried before.

Word embeddings. Our definition of a scaffold task *almost* includes stand-alone methods for estimating word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Peters et al., 2018). After training word embeddings, the tasks implied by models like the skip-gram or ELMo's language model become irrelevant to the downstream use of the embeddings. A noteworthy difference is that, rather than pre-training, a scaffold is integrated directly into the training of T_1 through a multitask objective.

Multitask learning. Neural architectures have often yielded performance gains when trained for multiple tasks together (Collobert et al., 2011; Luong et al., 2015; Chen et al., 2017; Hashimoto et al., 2017). In particular, performance of semantic role labeling tasks improves when done jointly with other semantic tasks (FitzGerald et al., 2015; Peng et al., 2017, 2018). Contemporaneously with this work, Hershcovich et al. (2018) proposed a multitask learning setting for universal syntactic dependencies and UCCA semantics (Abend and Rappoport, 2013). Syntactic scaffolds focus on a primary semantic task, treating syntax as an auxillary, eventually forgettable prediction task.

4 Syntactic Scaffold Model

We assume two sources of supervision: a corpus $\mathcal{D}_1$ with instances $\mathbf{x}$ annotated for the primary task's outputs $\mathbf{y}$ (semantic role labeling or coreference resolution), and a treebank $\mathcal{D}_2$ with sentences $\mathbf{x}$, each with a phrase-structure tree $\mathbf{z}$.

4.1 Loss

Each task has an associated loss, and we seek to minimize the combination of task losses,

$$\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_1} \mathcal{L}_1(\mathbf{x},\mathbf{y}) + \delta \sum_{(\mathbf{x},\mathbf{z})\in\mathcal{D}_2} \mathcal{L}_2(\mathbf{x},\mathbf{z}) \qquad (1)$$

with respect to parameters, which are partially shared, where δ is a tunable hyperparameter. In

---

[2] There has been some recent work on SRL which completely forgoes syntactic processing (Zhou and Xu, 2015), however it has been shown that incorporating syntactic information still remains useful (He et al., 2017).

the rest of this section, we describe the scaffold task. We define the primary tasks in Sections 5–6.

Each input is a sequence of tokens, $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$, for some n. We refer to a **span** of contiguous tokens in the sentence as $\mathbf{x}_{i:j} = \langle x_i, x_{i+1}, \ldots, x_j \rangle$, for any $1 \leqslant i \leqslant j \leqslant n$. In our experiments we consider only spans up to a maximum length D, resulting in $O(nD)$ spans.

Supervision comes from a phrase-syntactic tree $\mathbf{z}$ for the sentence, comprising a syntactic category $z_{i:j} \in C$ for every span $\mathbf{x}_{i:j}$ in $\mathbf{x}$ (many spans are given a NULL label). We experiment with different sets of labels C (§4.2).

In our model, every span $\mathbf{x}_{i:j}$ is represented by an embedding vector $\mathbf{v}_{i:j}$ (see details in §5.3). A distribution over the category assigned to $z_{i:j}$ is derived from $\mathbf{v}_{i:j}$:

$$p(z_{i:j} = c \mid \mathbf{x}_{i:j}) = \operatorname*{softmax}_c \mathbf{w}_c \cdot \mathbf{v}_{i:j} \qquad (2)$$

where $\mathbf{w}_c$ is a parameter vector associated with category c. We sum the log loss terms for all the spans in a sentence to give its loss:

$$\mathcal{L}_2(\mathbf{x}, \mathbf{z}) = - \sum_{\substack{1 \leqslant i \leqslant j \leqslant n \\ j-i \leqslant D}} \log p(z_{i:j} \mid \mathbf{x}_{i:j}). \qquad (3)$$

4.2 Labels for the Syntactic Scaffold Task

Different kinds of syntactic labels can be used for learning syntactically-aware span representations:

- **Constituent identity**: $C = \{0, 1\}$; is a span a constituent, or not?
- **Non-terminal**: c is the category of a span, including a NULL for non-constituents.
- **Non-terminal and parent**: c is the category of a span, concatenated with the category of its immediate ancestor. NULL is used for non-constituents, and for empty ancestors.
- **Common non-terminals**: Since a majority of semantic arguments and entity mentions are labeled with a small number of syntactic categories,[3] we experiment with a three-way classification among (i) noun phrase (or prepositional phrase, for frame SRL); (ii) any other category; and (iii) NULL.

In Figure 1, for the span "*encouraging them*", the constituent identity scaffold label is 1, the non-terminal label is S|VP, the non-terminal and parent label is S|VP+par=PP, and the common non-terminals label is set to OTHER.

---

[3] In the OntoNotes corpus, which includes both syntactic and semantic annotations, 44% of semantic arguments are noun phrases and 13% are prepositional phrases.

5 Semantic Role Labeling

We contribute a new SRL model which contributes a strong baseline for experiments with syntactic scaffolds. The performance of this baseline itself is competitive with state-of-the-art methods (§7).

FrameNet. In the FrameNet lexicon (Baker et al., 1998), a *frame* represents a type of event, situation, or relationship, and is associated with a set of semantic roles, called *frame elements*. A frame can be evoked by a word or phrase in a sentence, called a *target*. Each frame element of an evoked frame can then be realized in the sentence as a sentential span, called an *argument* (or it can be unrealized). Arguments for a given frame do not overlap.

PropBank. PropBank similarly disambiguates predicates and identifies argument spans. Targets are disambiguated to lexically specific *senses* rather than shared frames, and a set of generic roles is used for all targets, reducing the argument label space by a factor of 17. Most importantly, the arguments were annotated on top of syntactic constituents, directly coupling syntax and semantics. A detailed example for both formalisms is provided in Figure 1.

Semantic structure prediction is the task of identifying targets, labeling their frames or senses, and labeling all their argument spans in a sentence. Here we assume gold targets and frames, and consider only the SRL task.

Formally, a single input instance for argument identification consists of: an n-word sentence $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$, a single target span $t = \langle t_{\text{start}}, t_{\text{end}} \rangle$, and its evoked frame, or sense, f. The argument labeling task is to produce a **segmentation** of the sentence: $\mathbf{s} = \langle s_1, s_2, \ldots, s_m \rangle$ for each input $\mathbf{x}$. A **segment** $s = \langle i, j, y_{i:j} \rangle$ corresponds to a labeled span of the sentence, where the label $y_{i:j} \in \mathcal{Y}_f \cup \{\text{NULL}\}$ is either a role that the span fills, or NULL if the span does not fill any role. In the case of PropBank, $\mathcal{Y}_f$ consists of all possible roles. The segmentation is constrained so that argument spans cover the sentence and do not overlap ($i_{k+1} = 1 + j_k$ for s_k; $i_1 = 1$; $j_m = n$). Segments of length 1 such that $i = j$ are allowed. A separate segmentation is predicted for each target annotation in a sentence.

5.1 Semi-Markov CRF

In order to model the non-overlapping arguments of a given target, we use a semi-Markov conditional random field (**semi-CRF**; Sarawagi et al., 2004). Semi-CRFs define a conditional distribution over labeled segmentations of an input sequence, and are globally normalized. A single target's arguments can be neatly encoded as a labeled segmentation by giving the spans in between arguments a reserved NULL label. Semi-Markov models are more powerful than BIO tagging schemes, which have been used successfully for PropBank SRL (Collobert et al., 2011; Zhou and Xu, 2015, *inter alia*), because the semi-Markov assumption allows scoring variable-length segments, rather than fixed-length label n-grams as under an $(n-1)$-order Markov assumption. Computing the marginal likelihood with a semi-CRF can be done using dynamic programming in $O(n^2)$ time (§5.2). By filtering out segments longer than D tokens, this is reduced to $O(nD)$.

Given an input $\mathbf{x}$, a semi-CRF defines a conditional distribution $p(\mathbf{s} \mid \mathbf{x})$. Every segment $s = \langle i, j, y_{i:j} \rangle$ is given a real-valued score, $\psi(\langle i, j, y_{i:j} = r \rangle, \mathbf{x}_{i:j}) = \mathbf{w}_r \cdot \mathbf{v}_{i:j}$, where $\mathbf{v}_{i:j}$ is an embedding of the span (§5.3) and $\mathbf{w}_r$ is a parameter vector corresponding to its label. The score of the entire segmentation $\mathbf{s}$ is the sum of the scores of its segments: $\Psi(\mathbf{x}, \mathbf{s}) = \sum_{k=1}^{m} \psi(s_k, \mathbf{x}_{i_k:j_k})$. These scores are exponentiated and normalized to define the probability distribution. The sum-product variant of the semi-Markov dynamic programming algorithm is used to calculate the normalization term (required during learning). At test time, the max-product variant returns the most probable segmentation, $\hat{\mathbf{s}} = \arg\max_{\mathbf{s}} \Psi(\mathbf{s}, \mathbf{x})$.

The parameters of the semi-CRF are learned to maximize a criterion related to the conditional log-likelihood of the gold-standard segments in the training corpus (§5.2). The learner evaluates and adjusts segment scores $\psi(s_k, \mathbf{x})$ for every span in the sentence, which in turn involves learning embedded representations for all spans (§5.3).

5.2 Softmax-Margin Objective

Typically CRF and semi-CRF models are trained to maximize a conditional log-likelihood objective. In early experiments, we found that incorporating a structured cost was beneficial; we do so by using a softmax-margin training objective (Gimpel and Smith, 2010), a "cost-aware" variant

of log-likelihood:

$$\mathcal{L}_1 = -\sum_{(\mathbf{x}, \mathbf{s}^*) \in \mathcal{D}_1} \log \frac{\exp \Psi(\mathbf{s}^*, \mathbf{x})}{Z(\mathbf{x}, \mathbf{s}^*)}, \qquad (4)$$

$$Z(\mathbf{x}, \mathbf{s}^*) = \sum_{\mathbf{s}} \exp\{\Psi(\mathbf{s}, \mathbf{x}) + \text{cost}(\mathbf{s}, \mathbf{s}^*)\}. \qquad (5)$$

We design the cost function so that it factors by predicted span, in the same way Ψ does:

$$\text{cost}(\mathbf{s}, \mathbf{s}^*) = \sum_{s \in \mathbf{s}} \text{cost}(s, \mathbf{s}^*) = \sum_{s \in \mathbf{s}} \mathbb{I}(s \notin \mathbf{s}^*). \qquad (6)$$

The softmax-margin criterion, like log-likelihood, is globally normalized over all of the exponentially many possible labeled segmentations. The following zeroth-order semi-Markov dynamic program (Sarawagi et al., 2004) efficiently computes the new partition function:

$$\alpha_j = \sum_{\substack{s = \langle i, j, y_{i:j} \rangle \\ j - i \leq D}} \alpha_{i-1} \exp\{\Psi(s, \mathbf{x}) + \text{cost}(s, \mathbf{s}^*)\}, \qquad (7)$$

where $Z = \alpha_n$, under the base case $\alpha_0 = 1$.

The prediction under the model can be calculated using a similar dynamic program with the following recurrence where $\gamma_0 = 1$:

$$\gamma_j = \max_{\substack{s = \langle i, j, y_{i:j} \rangle \\ j - i \leq D}} \gamma_{i-1} \exp \Psi(s, \mathbf{x}). \qquad (8)$$

Our model formulation enforces that arguments do not overlap. We do not enforce any other SRL constraints, such as non-repetition of core frame elements (Das et al., 2012).

5.3 Input Span Representation

This section describes the neural architecture used to obtain the span embedding, $\mathbf{v}_{i:j}$, corresponding to a span $\mathbf{x}_{i:j}$ and the target in consideration, $t = \langle t_{\text{start}}, t_{\text{end}} \rangle$. For the scaffold task, since the syntactic treebank does not contain annotations for semantic targets, we use the last verb in the sentence as a placeholder target, wherever target features are used. If there are no verbs, we use the first token in the sentence as a placeholder target. The parameters used to learn $\mathbf{v}$ are shared between the tasks.

We construct an embedding for the span using

- $\mathbf{h}_i$ and $\mathbf{h}_j$: contextualized embeddings for the words at the span boundary (§5.3.1),
- $\mathbf{u}_{i:j}$: a span summary that pools over the contents of the span (§5.3.2), and

3776

- $\mathbf{a}_{i:j}$: and a hand-engineered feature vector for the span (§5.3.3).

This embedding is then passed to a feedforward layer to compute the span representation, $\mathbf{v}_{i:j}$.

5.3.1 Contextualized Token Embeddings

To obtain contextualized embeddings of each token in the input sequence, we run a bidirectional LSTM (Graves, 2012) with ℓ layers over the full input sequence. To indicate which token is a predicate, a linearly transformed one-hot embedding $\mathbf{v}$ is used, following Zhou and Xu (2015) and He et al. (2017). The input vector representing the token at position q in the sentence is the concatenation of a fixed pretrained embedding $\mathbf{x}_q$ and $\mathbf{v}_q$. When given as input to the bidirectional LSTM, this yields a hidden state vector $\mathbf{h}_q$ representing the qth token in the context of the sentence.

5.3.2 Span Summary

Tokens within a span might convey different amounts of information necessary to label the span as a semantic argument. Following Lee et al. (2017), we use an attention mechanism (Bahdanau et al., 2014) to summarize each span. Each contextualized token in the span is passed through a feed-forward network to obtain a weight, normalized to give $\sigma_k = \underset{i \leqslant k \leqslant j}{\mathrm{softmax}}\ \mathbf{w}^{\mathrm{head}} \cdot \mathbf{h}_k$, where $\mathbf{w}^{\mathrm{head}}$ is a learned parameter. The weights σ are then used to obtain a vector that summarizes the span, $\mathbf{u}_{i:j} = \sum_{i \leqslant k \leqslant j;\, j-i<D} \sigma_k \cdot \mathbf{h}_k$.

5.3.3 Span Features

We use the following three features for each span:
- width of the span in tokens (Das et al., 2014)
- distance (in tokens) of the span from the target (Täckström et al., 2015)
- position of the span with respect to the target (*before*, *after*, *overlap*) (Täckström et al., 2015)

Each of these features is encoded as a one-hot-embedding and then linearly transformed to yield a feature vector, $\mathbf{a}_{i:j}$.

6 Coreference Resolution

Coreference resolution is the task of determining clusters of mentions that refer to the same entity. Formally, the input is a document $\mathbf{x} = x_1, x_2, \ldots, x_n$ consisting of n words. The goal is to predict a set of clusters $\mathbf{c} = \{c_1, c_2, \ldots\}$, where each cluster $c = \{s_1, s_2, \ldots\}$ is a set of spans and

each span $s = \langle i, j \rangle$ is a pair of indices such that $1 \leqslant i \leqslant j \leqslant n$.

As a baseline, we use the model of Lee et al. (2017), which we describe briefly in this section. This model decomposes the prediction of coreference clusters into a series of span classification decisions. Every span s predicts an antecedent $w_s \in \mathcal{Y}(s) = \{\text{NULL}, s_1, s_2, \ldots, s_m\}$. Labels s_1 to s_m indicate a coreference link between s and one of the m spans that precede it, and NULL indicates that s does not link to anything, either because it is not a mention or it is in a singleton cluster. The predicted clustering of the spans can be recovered by aggregating the predicted links.

Analogous to the SRL model (§5), every span s is represented by an embedding $\mathbf{v}_s$, which is central to the model. For each span s and a potential antecedent $a \in \mathcal{Y}(s)$, pairwise coreference scores $\Psi(\mathbf{v}_s, \mathbf{v}_a, \phi(s, a))$ are computed via feedforward networks with the span embeddings as input. $\phi(s, a)$ are pairwise discrete features encoding the distance between span s and span a and metadata, such as the genre and speaker information. We refer the reader to Lee et al. (2017) for the details of the scoring function.

The scores from Ψ are normalized over the possible antecedents $\mathcal{Y}(s)$ of each span to induce a probability distribution for every span:

$$p(w_s = a) = \underset{a \in \mathcal{Y}(s)}{\mathrm{softmax}}\ \Psi(\mathbf{v}_s, \mathbf{v}_a, \phi(s, a)) \quad (9)$$

In learning, we minimize the negative log-likelihood marginalized over the possibly correct antecedents:

$$\mathcal{L}_1 = -\sum_{s \in \mathcal{D}} \log \sum_{a^* \in G(s) \cap \mathcal{Y}(s)} p(w_s = a^*) \quad (10)$$

where $\mathcal{D}$ is the set of spans in the training dataset, and $G(s)$ indicates the gold cluster of s if it belongs to one and {NULL} otherwise.

To operate under reasonable computational requirements, inference under this model requires a two-stage beam search, which reduces the number of span pairs considered. We refer the reader to Lee et al. (2017) for details.

Input span representation. The input span embedding, $\mathbf{v}_s$ for coreference resolution and its syntactic scaffold follow the definition used in §5.3, with the key difference of using no target features. Since there is a complete overlap of input sentences between $\mathcal{D}_{\mathrm{sc}}$ and $\mathcal{D}_{\mathrm{pr}}$ as the coreference annotations are also from OntoNotes (Pradhan et al.,

2012), we reuse the **v** for the scaffold task. Additionally, instead of the entire document, each sentence in it is independently given as input to the bidirectional LSTMs.

7 Results

We evaluate our models on the test set of FrameNet 1.5 for frame SRL and on the test set of OntoNotes for both PropBank SRL and coreference. For the syntactic scaffold in each case, we use syntactic annotations from OntoNotes 5.0 (Weischedel et al., 2013; Pradhan et al., 2013).[4] Further details on experimental settings and datasets have been elaborated in the supplemental material.

Frame SRL. Table 1 shows the performance of all the scaffold models on frame SRL with respect to prior work and a semi-CRF baseline (§5.1) without a syntactic scaffold. We follow the official evaluation from the SemEval shared task for frame-semantic parsing (Baker et al., 2007).

Prior work for frame SRL has relied on predicted syntactic trees, in two different ways: by using syntax-based rules to prune out spans of text that are unlikely to contain any frame's argument; and by using syntactic features in their statistical model (Das et al., 2014; Täckström et al., 2015; FitzGerald et al., 2015; Kshirsagar et al., 2015).

The best published results on FrameNet 1.5 are due to Yang and Mitchell (2017). In their sequential model (SEQ), they treat argument identification as a sequence-labeling problem using a deep bidirectional LSTM with a CRF layer. In their relational model (REL), they treat the same problem as a span classification problem. Finally, they introduce an ensemble to integerate both models, and use an integer linear program for inference satisfying SRL constraints. Though their model does not do any syntactic pruning, it does use syntactic features for argument identification and labeling.[5]

Notably, all prior systems for frame SRL listed in Table 1 use a pipeline of syntax and semantics. Our semi-CRF baseline outperforms all prior work, without any syntax. This highlights the ben-

efits of modeling spans and of global normalization.

Turning to scaffolds, even the most coarse-grained constituent identity scaffold improves the performance of our syntax-agnostic baseline. The nonterminal and nonterminal and parent scaffolds, which use more detailed syntactic representations, improve over this. The greatest improvements come from the scaffold model predicting common nonterminal labels (NP and PP, which are the most common syntactic categories of semantic arguments, vs. others): 3.6% absolute improvement in F_1 measure over prior work.

Contemporaneously with this work, Peng et al. (2018) proposed a system for joint frame-semantic and semantic dependency parsing. They report results for joint frame and argument identification, and hence cannot be directly compared in Table 1. We evaluated their output for argument identification only; our semi-CRF baseline model exceeds their performance by 1 F_1, and our common nonterminal scaffold by 3.1 F_1.[6]

Model	Prec.	Rec.	F_1
Kshirsagar et al. (2015)	66.0	60.4	63.1
Yang and Mitchell (2017) (REL)	71.8	57.7	64.0
Yang and Mitchell (2017) (SEQ)	63.4	66.4	64.9
†Yang and Mitchell (2017) (ALL)	70.2	60.2	65.5
Semi-CRF baseline	67.8	66.2	67.0
+ constituent identity	68.1	67.4	67.7
+ nonterminal and parent	68.8	68.2	68.5
+ nonterminal	69.4	68.0	68.7
+ common nonterminals	69.2	69.0	**69.1**

Table 1: Frame SRL results on the test set of FrameNet 1.5., using gold frames. Ensembles are denoted by †.

Model	Prec.	Rec.	F_1
Zhou and Xu (2015)	-	-	81.3
He et al. (2017)	81.7	81.6	81.7
He et al. (2018a)	83.9	73.7	82.1
Tan et al. (2018)	81.9	83.6	82.7
Semi-CRF baseline	84.8	81.2	83.0
+ common nonterminals	85.1	82.6	**83.8**

Table 2: PropBank sSRL results, using gold predicates, on CoNLL 2012 test. For fair comparison, we show only non-ensembled models.

---

[4]http://cemantix.org/data/ontonotes.html

[5]Yang and Mitchell (2017) also evaluated on the full frame-semantic parsing task, which includes frame-SRL as well as identifying frames. Since our frame SRL performance improves over theirs, we expect that incorporation into a full system (e.g., using their frame identification module) would lead to overall benefits as well; this experiment is left to future work.

[6]This result is not reported in Table 1 since Peng et al. (2018) used a preprocessing which renders the test set slightly larger — the difference we report is calculated using their test set.

Model	MUC			B[3]			CEAF$_{\phi_4}$			Avg. F_1
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	
Wiseman et al. (2016)	77.5	69.8	73.4	66.8	57.0	61.5	62.1	53.9	57.7	64.2
Clark and Manning (2016b)	79.9	69.3	74.2	71.0	56.5	63.0	63.8	54.3	58.7	65.3
Clark and Manning (2016a)	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2	65.7
Lee et al. (2017)	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
+ common nonterminals	78.4	74.3	76.3	68.7	62.9	65.7	62.9	60.2	61.5	**67.8**

Table 3: Coreference resolution results on the test set on the English CoNLL-2012 shared task. The average F_1 of MUC, B[3], and CEAF$_{\phi_4}$ is the main evaluation metric. For fair comparison, we show only non-ensembled models.

PropBank SRL. We use the OntoNotes data from the CoNLL shared task in 2012 (Pradhan et al., 2013) for Propbank SRL. Table 2 reports results using gold predicates.

Recent competitive systems for PropBank SRL follow the approach of Zhou and Xu (2015), employing deep architectures, and forgoing the use of any syntax. He et al. (2017) improve on those results, and in analysis experiments, show that constraints derived using syntax may further improve performance. Tan et al. (2018) employ a similar approach but use feed-forward networks with self-attention. He et al. (2018a) use a span-based classification to jointly identify and label argument spans.

Our syntax-agnostic semi-CRF baseline model improves on prior work (excluding `ELMo`), showing again the value of global normalization in semantic structure prediction. We obtain further improvement of 0.8 absolute F_1 with the best syntactic scaffold from the frame SRL task. This indicates that a syntactic inductive bias is beneficial even when using sophisticated neural architectures.

He et al. (2018a) also provide a setup where initialization was done with deep contextualized embeddings, `ELMo` (Peters et al., 2018), resulting in 85.5 F_1 on the OntoNotes test set. The improvements from `ELMo` are methodologically orthogonal to syntactic scaffolds.

Since the datasets for learning PropBank semantics and syntactic scaffolds *completely* overlap, the performance improvement cannot be attributed to a larger training corpus (or, by extension, a larger vocabulary), though that might be a factor for frame SRL.

A syntactic scaffold can match the performance of a pipeline containing carefully extracted syntactic features for semantic prediction (Swayamdipta et al., 2017). This, along with other recent approaches (He et al., 2017, 2018b) show that syntax remains useful, even with strong neural models for SRL.

Coreference. We report the results on four standard scores from the CoNLL evaluation: MUC, B[3] and CEAF$_{\phi_4}$, and their average F_1 in Table 3. Prior competitive coreference resolution systems (Wiseman et al., 2016; Clark and Manning, 2016b,a) all incorporate synctactic information in a pipeline, using features and rules for mention proposals from predicted syntax.

Our baseline is the model from Lee et al. (2017), described in §6. Similar to the baseline model for frame SRL, and in contrast with prior work, this model does not use any syntax.

We experiment with the best syntactic scaffold from the frame SRL task. We used `NP`, `OTHER`, and `NULL` as the labels for the common nonterminals scaffold here, since coreferring mentions are rarely prepositional phrases. The syntactic scaffold outperforms the baseline by 0.6 absolute F_1. Contemporaneously, Lee et al. (2018) proposed a model which takes in account higher order inference and more aggressive pruning, as well as initialization with `ELMo` embeddings, resulting in 73.0 average F_1. All the above are orthogonal to our approach, and could be incorporated to yield higher gains.

8 Discussion

To investigate the performance of the syntactic scaffold, we focus on the frame SRL results, where we observed the greatest improvement with respect to a non-syntactic baseline.

We consider a breakdown of the performance by the syntactic phrase types of the arguments, provided in FrameNet[7] in Figure 2. Not surpris-

---

[7]We used FrameNet syntactic phrase annotations for analysis only, and not in our models, since they are annotated only for the gold arguments.
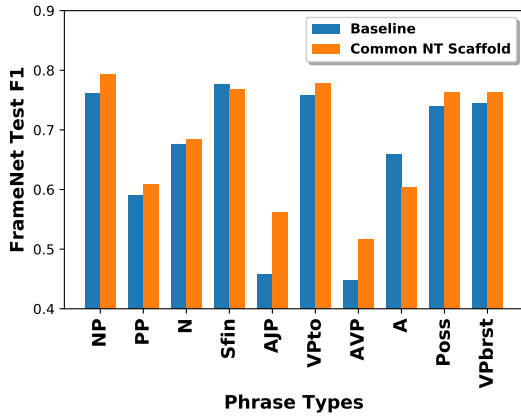
Figure 2: Performance breakdown by argument's phrase category, sorted left to right by frequency, for top ten phrase categories.
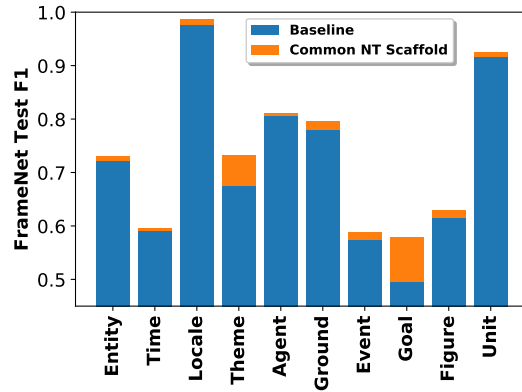

Figure 3: Performance breakdown by top ten frame element types, sorted left to right by frequency.

ingly, we observe large improvements in the common nonterminals used (`NP` and `PP`). However, the phrase type annotations in FrameNet do not correspond exactly to the OntoNotes phrase categories. For instance, FrameNet annotates non-maximal (`A`) and standard adjective phrases (`AJP`), while OntoNotes annotations for noun-phrases are flat, ignore the underlying adjective phrases. This explains why the syntax-agnostic baseline is able to recover the former while the scaffold is not.

Similarly, for frequent frame elements, scaffolding improves performance across the board, as shown in Fig. 3. The largest improvements come for `Theme` and `Goal`, which are predominantly realized as noun phrases and prepositional phrases.

9 Conclusion

We introduced syntactic scaffolds, a multitask learning approach to incorporate syntactic bias into semantic processing tasks. Unlike pipelines and approaches which jointly model syntax and semantics, no explicit syntactic processing is required at runtime. Our method improves the performance of competitive baselines for semantic role labeling on both FrameNet and PropBank, and for coreference resolution. While our focus was on span-based tasks, syntactic scaffolds could be applied in other settings (e.g., dependency and graph representations). Moreover, scaffolds need not be syntactic; we can imagine, for example, *semantic* scaffolds being used to improve NLP applications with limited annotated data. It remains an open empirical question to determine the relative merits of different kinds of scaffolds and multi-task learners, and how they can be most produc-

tively combined. Our code is publicly available at `https://github.com/swabhs/scaffolding`.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. ArXiv:1409.0473.

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

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of ACL*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1).

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. ArXiv:1704.07556.

Kevin Clark and Christopher D Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proc. of EMNLP*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proc. of ACL*.

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*, 12:2493–2537.

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

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proc. of *SEM*.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proc. of EMNLP*.

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

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

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proc. of ACL*.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*.

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.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proc. of EMNLP*.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *Proc. of EMNLP*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proc. of ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proc. of ACL*.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proc. of ACL*.

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.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*.

Diederik P. Kingma and Jimmy Ba. 2014. ADAM: A method for stochastic optimization. ArXiv:1412.6980.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Segmental Recurrent Neural Networks. In *Proc. of ICLR*.

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 NAACL*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proc. of EMNLP*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proc. of NAACL*.

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

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *Proc. of CoNLL*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. ArXiv:1511.06114.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. ArXiv:1301.3781.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proc. of ICML*.

Jason Naradowsky, Sebastian Riedel, and David A. Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proc. of EMNLP*.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proc. of ACL*.

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

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*.

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

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. ArXiv:1802.05365.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proc. of CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proc. of EMNLP*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. ArXiv:1606.05250.

Sunita Sarawagi, William W Cohen, et al. 2004. Semi-markov conditional random fields for information extraction. In *Proc. of NIPS*, volume 17.

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

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proc. of NIPS*.

Mark Steedman. 2000. Information structure and the syntax-phonology interface. *Linguistic Inquiry*, 31(4):649–689.

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

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. Arxiv:1706.09528.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the ACL*, 3:29–41.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proc. of AAAI*.

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

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. OntoNotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. In *Proc. of NAACL*.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proc. of EMNLP*.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI*.

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

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of ACL*.