Deep Semantic Role Labeling: What Works and What's Next

Luheng He[†], Kenton Lee[†], Mike Lewis[‡], and Luke Zettlemoyer[†*]

[†] Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
{luheng, kentonl, lsz}@cs.washington.edu

[‡] Facebook AI Research, Menlo Park, CA
mikelewis0@fb.com

[*]Allen Institute for Artificial Intelligence, Seattle, WA
lukez@allenai.org

Abstract

We introduce a new deep learning model for semantic role labeling (SRL) that significantly improves the state of the art, along with detailed analyses to reveal its strengths and limitations. We use a deep highway BiLSTM architecture with constrained decoding, while observing a number of recent best practices for initialization and regularization. Our 8-layer ensemble model achieves 83.2 F1 on the CoNLL 2005 test set and 83.4 F1 on CoNLL 2012, roughly a 10% relative error reduction over the previous state of the art. Extensive empirical analysis of these gains show that (1) deep models excel at recovering long-distance dependencies but can still make surprisingly obvious errors, and (2) that there is still room for syntactic parsers to improve these results.

1 Introduction

Semantic role labeling (SRL) systems aim to recover the predicate-argument structure of a sentence, to determine essentially "who did what to whom", "when", and "where." Recently breakthroughs involving end-to-end deep models for SRL without syntactic input (Zhou and Xu, 2015; Marcheggiani et al., 2017) seem to overturn the long-held belief that syntactic parsing is a prerequisite for this task (Punyakanok et al., 2008). In this paper, we show that this result can be pushed further using deep highway bidirectional LSTMs with constrained decoding, again significantly moving the state of the art (another 2 points on CoNLL 2005). We also present a careful empirical analysis to determine what works well and what might be done to progress even further.

Our model combines a number of best practices in the recent deep learning literature. Following Zhou and Xu (2015), we treat SRL as a BIO tagging problem and use deep bidirectional LSTMs. However, we differ by (1) simplifying the input and output layers, (2) introducing highway connections (Srivastava et al., 2015; Zhang et al., 2016), (3) using recurrent dropout (Gal and Ghahramani, 2016), (4) decoding with BIO-constraints, and (5) ensembling with a product of experts. Our model gives a 10% relative error reduction over previous state of the art on the test sets of CoNLL 2005 and 2012. We also report performance with predicted predicates to encourage future exploration of end-to-end SRL systems.

We present detailed error analyses to better understand the performance gains, including (1) design choices on architecture, initialization, and regularization that have a surprisingly large impact on model performance; (2) different types of prediction errors showing, e.g., that deep models excel at predicting long-distance dependencies but still struggle with known challenges such as PP-attachment errors and adjunct-argument distinctions; (3) the role of syntax, showing that there is significant room for improvement given oracle syntax but errors from existing automatic parsers prevent effective use in SRL.

In summary, our main contributions incluede:

- A new state-of-the-art deep network for end-to-end SRL, supported by publicly available code and models.[1]
- An in-depth error analysis indicating where the model works well and where it still struggles, including discussion of structural consistency and long-distance dependencies.
- Experiments that point toward directions for future improvements, including a detailed discussion of how and when syntactic parsers could be used to improve these results.

[1] https://github.com/luheng/deep_srl

2 Model

Two major factors contribute to the success of our deep SRL model: (1) applying recent advances in training deep recurrent neural networks such as highway connections (Srivastava et al., 2015) and RNN-dropouts (Gal and Ghahramani, 2016),[2] and (2) using an A* decoding algorithm (Lewis and Steedman, 2014; Lee et al., 2016) to enforce structural consistency at prediction time without adding more complexity to the training process.

Formally, our task is to predict a sequence $\boldsymbol{y}$ given a sentence-predicate pair $(\boldsymbol{w}, v)$ as input. Each $y_i \in \boldsymbol{y}$ belongs to a discrete set of BIO tags $\boldsymbol{T}$. Words outside argument spans have the tag O, and words at the beginning and inside of argument spans with role r have the tags B_r and I_r respectively. Let $n = |\boldsymbol{w}| = |\boldsymbol{y}|$ be the length of the sequence.

Predicting an SRL structure under our model involves finding the highest-scoring tag sequence over the space of all possibilities $\mathcal{Y}$:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{w}, \boldsymbol{y}) \tag{1}$$

We use a deep bidirectional LSTM (BiLSTM) to learn a locally decomposed scoring function conditioned on the input: $\sum_{t=1}^{n} \log p(y_t \mid \boldsymbol{w})$.

To incorporate additional information (e.g., structural consistency, syntactic input), we augment the scoring function with penalization terms:

$$f(\boldsymbol{w}, \boldsymbol{y}) = \sum_{t=1}^{n} \log p(y_t \mid \boldsymbol{w}) - \sum_{c \in \mathcal{C}} c(\boldsymbol{w}, y_{1:t}) \tag{2}$$

Each constraint function c applies a *non-negative* penalty given the input $\boldsymbol{w}$ and a length-t prefix $y_{1:t}$. These constraints can be hard or soft depending on whether the penalties are finite.

2.1 Deep BiLSTM Model

Our model computes the distribution over tags using stacked BiLSTMs, which we define as follows:

$$\boldsymbol{i}_{l,t} = \sigma(\mathbf{W}^l_{\mathrm{i}}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}^l_{\mathrm{i}}) \tag{3}$$
$$\boldsymbol{o}_{l,t} = \sigma(\mathbf{W}^l_{\mathrm{o}}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}^l_{\mathrm{o}}) \tag{4}$$
$$\boldsymbol{f}_{l,t} = \sigma(\mathbf{W}^l_{\mathrm{f}}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}^l_{\mathrm{f}} + 1) \tag{5}$$
$$\tilde{\boldsymbol{c}}_{l,t} = \tanh(\mathbf{W}^l_{\mathrm{c}}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}^l_{\mathrm{c}}) \tag{6}$$
$$\boldsymbol{c}_{l,t} = \boldsymbol{i}_{l,t} \circ \tilde{\boldsymbol{c}}_{l,t} + \boldsymbol{f}_{l,t} \circ \boldsymbol{c}_{t+\delta_l} \tag{7}$$
$$\boldsymbol{h}_{l,t} = \boldsymbol{o}_{l,t} \circ \tanh(\boldsymbol{c}_{l,t}) \tag{8}$$

[2] We thank Mingxuan Wang for suggesting highway connections with simplified inputs and outputs. Part of our model is extended from his unpublished implementation.

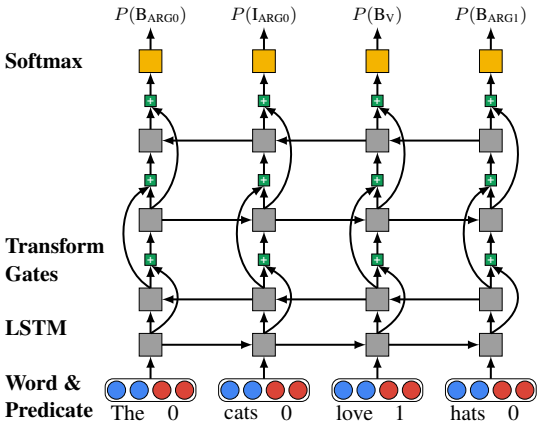

Figure 1: Highway LSTM with four layers. The curved connections represent highway connections, and the plus symbols represent transform gates that control inter-layer information flow.

where $x_{l,t}$ is the input to the LSTM at layer l and timestep t. δ_l is either 1 or -1, indicating the directionality of the LSTM at layer l.

To stack the LSTMs in an interleaving pattern, as proposed by Zhou and Xu (2015), the layer-specific inputs $x_{l,t}$ and directionality δ_l are arranged in the following manner:

$$\boldsymbol{x}_{l,t} = \begin{cases} [\mathbf{W}_{\text{emb}}(w_t), \mathbf{W}_{\text{mask}}(t = v)] & l = 1 \\ \boldsymbol{h}_{l-1,t} & l > 1 \end{cases} \tag{9}$$

$$\delta_l = \begin{cases} 1 & \text{if } l \text{ is even} \\ -1 & \text{otherwise} \end{cases} \tag{10}$$

The input vector $\boldsymbol{x}_{1,t}$ is the concatenation of token w_t's word embedding and an embedding of the binary feature $(t = v)$ indicating whether w_t word is the given predicate.

Finally, the locally normalized distribution over output tags is computed via a softmax layer:

$$p(y_t \mid \boldsymbol{x}) \propto \exp(\mathbf{W}^y_{\text{tag}} \boldsymbol{h}_{L,t} + \boldsymbol{b}_{\text{tag}}) \tag{11}$$

Highway Connections To alleviate the vanishing gradient problem when training deep BiLSTMs, we use gated highway connections (Zhang et al., 2016; Srivastava et al., 2015). We include *transform gates* $\boldsymbol{r}_t$ to control the weight of linear and non-linear transformations between layers (See Figure 1). The output $\boldsymbol{h}_{l,t}$ is changed to:

$$\boldsymbol{r}_{l,t} = \sigma(\mathbf{W}^l_{\mathrm{r}}[\boldsymbol{h}_{l,t-1}, \boldsymbol{x}_t] + \boldsymbol{b}^l_{\mathrm{r}}) \tag{12}$$
$$\boldsymbol{h}'_{l,t} = \boldsymbol{o}_{l,t} \circ \tanh(\boldsymbol{c}_{l,t}) \tag{13}$$
$$\boldsymbol{h}_{l,t} = \boldsymbol{r}_{l,t} \circ \boldsymbol{h}'_{l,t} + (1 - \boldsymbol{r}_{l,t}) \circ \mathbf{W}^l_{\mathrm{h}} \boldsymbol{x}_{l,t} \tag{14}$$

Recurrent Dropout To reduce over-fitting, we use dropout as described in Gal and Ghahramani (2016). A shared dropout mask $\boldsymbol{z}_l$ is applied to the hidden state:

$$\widetilde{\boldsymbol{h}}_{l,t} = \boldsymbol{r}_{l,t} \circ \boldsymbol{h}'_{l,t} + (1 - \boldsymbol{r}_{l,t}) \circ \mathbf{W}^l_{\mathrm{h}} \boldsymbol{x}_{l,t} \quad (15)$$

$$\boldsymbol{h}_{l,t} = \boldsymbol{z}_l \circ \widetilde{\boldsymbol{h}}_{l,t} \quad (16)$$

$\boldsymbol{z}_l$ is shared across timesteps at layer l to avoid amplifying the dropout noise along the sequence.

2.2 Constrained A* Decoding

The approach described so far does not model any dependencies between the output tags. To incorporate constraints on the output structure at decoding time, we use A* search over tag prefixes for decoding. Starting with an empty sequence, the tag sequence is built from left to right. The score for a partial sequence with length t is defined as:

$$f(\boldsymbol{w}, y_{1:t}) = \sum_{i=1}^{t} \log p(y_i \mid \boldsymbol{w}) - \sum_{c \in \mathcal{C}} c(\boldsymbol{w}, y_{1:i}) \quad (17)$$

An admissible A* heuristic can be computed efficiently by summing over the best possible tags for all timesteps after t:

$$g(\boldsymbol{w}, y_{1:t}) = \sum_{i=t+1}^{n} \max_{y_i \in \boldsymbol{T}} \log p(y_i \mid \boldsymbol{w}) \quad (18)$$

Exploration of the prefixes is determined by an agenda $\mathcal{A}$ which is sorted by $f(\boldsymbol{w}, y_{1:t}) + g(\boldsymbol{w}, y_{1:t})$. In the worst case, A* explores exponentially many prefixes, but because the distribution $p(y_t \mid \boldsymbol{w})$ learned by the BiLSTM models is very peaked, the algorithm is efficient in practice. We list some example constraints as follows:

BIO Constraints These constraints reject any sequence that does not produce valid BIO transitions, such as $\mathrm{B}_{\mathrm{ARG0}}$ followed by $\mathrm{I}_{\mathrm{ARG1}}$.

SRL Constraints Punyakanok et al. (2008); Täckström et al. (2015) described a list of SRL-specific global constraints:

- Unique core roles (U): Each core role (ARG0-ARG5, ARGA) should appear at most once for each predicate.
- Continuation roles (C): A continuation role C-X can exist only when its base role X is realized before it.
- Reference roles (R): A reference role R-X can exist only when its base role X is realized (not necessarily before R-X).

We only enforce U and C constraints, since the R constraints are more commonly violated in gold data and enforcing them results in worse performance (see discussions in Section 4.3).

Syntactic Constraints We can enforce consistency with a given parse tree by rejecting or penalizing arguments that are not constituents. In Section 4.4, we will discuss the motivation behind using syntactic constraints and experimental results using both predicted and gold syntax.

2.3 Predicate Detection

While the CoNLL 2005 shared task assumes gold predicates as input (Carreras and Màrquez, 2005), this information is not available in many downstream applications. We propose a simple model for end-to-end SRL, where the system first predicts a set of predicate words $\boldsymbol{v}$ from the input sentence $\boldsymbol{w}$. Then each predicate in $\boldsymbol{v}$ is used as an input to argument prediction. We independently predict whether each word in the sentence is a predicate, using a binary softmax over the outputs of a bidirectional LSTM trained to maximize the likelihood of the gold labels.

3 Experiments

3.1 Datasets

We measure the performance of our SRL system on two PropBank-style, span-based SRL datasets: CoNLL-2005 (Carreras and Màrquez, 2005) and CoNLL-2012 (Pradhan et al., 2013)[3]. Both datasets provide gold predicates (their index in the sentence) as part of the input. Therefore, each provided predicate corresponds to one training/test tag sequence. We follow the train-development-test split for both datasets and use the official evaluation script from the CoNLL 2005 shared task for evaluation on both datasets.

3.2 Model Setup

Our network consists of 8 BiLSTM layers (4 forward LSTMs and 4 reversed LSTMs) with 300-dimensional hidden units, and a softmax layer for predicting the output distribution.

Initialization All the weight matrices in BiLSTMs are initialized with random orthonormal matrices as described in Saxe et al. (2013).

[3]We used the version of OntoNotes downloaded at: http://cemantix.org/data/ontonotes.html.

Method	Development				WSJ Test				Brown Test				Combined
	P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.	F1
Ours (PoE)	**83.1**	**82.4**	**82.7**	**64.1**	**85.0**	**84.3**	**84.6**	**66.5**	**74.9**	**72.4**	**73.6**	**46.5**	**83.2**
Ours	81.6	81.6	81.6	62.3	83.1	83.0	83.1	64.3	72.9	71.4	72.1	44.8	81.6
Zhou	79.7	79.4	79.6	-	82.9	82.8	82.8	-	70.7	68.2	69.4	-	81.1
FitzGerald (Struct.,PoE)	81.2	76.7	78.9	55.1	82.5	78.2	80.3	57.3	74.5	70.0	72.2	41.3	-
Täckström (Struct.)	81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	74.3	68.6	71.3	39.8	-
Toutanova (Ensemble)	-	-	78.6	58.7	81.9	78.8	80.3	60.1	-	-	68.8	40.8	-
Punyakanok (Ensemble)	80.1	74.8	77.4	50.7	82.3	76.8	79.4	53.8	73.4	62.9	67.8	32.3	77.9

Table 1: Experimental results on CoNLL 2005, in terms of precision (P), recall (R), F1 and percentage of completely correct predicates (Comp.). We report results of our best single and ensemble (PoE) model. The comparison models are Zhou and Xu (2015), FitzGerald et al. (2015), Täckström et al. (2015), Toutanova et al. (2008) and Punyakanok et al. (2008).

Method	Development				Test			
	P	R	F1	Comp.	P	R	F1	Comp.
Ours (PoE)	**83.5**	**83.2**	**83.4**	**67.5**	**83.5**	**83.3**	**83.4**	**68.5**
Ours	81.8	81.4	81.5	64.6	81.7	81.6	81.7	66.0
Zhou	-	-	81.1	-	-	-	81.3	-
FitzGerald (Struct.,PoE)	81.0	78.5	79.7	60.9	81.2	79.0	80.1	62.6
Täckström (Struct.)	80.5	77.8	79.1	60.1	80.6	78.2	79.4	61.8
Pradhan (revised)	-	-	-	-	78.5	76.6	77.5	55.8

Table 2: Experimental results on CoNLL 2012 in the same metrics as above. We compare our best single and ensemble (PoE) models against Zhou and Xu (2015), FitzGerald et al. (2015), Täckström et al. (2015) and Pradhan et al. (2013).

All tokens are lower-cased and initialized with 100-dimensional GloVe embeddings pre-trained on 6B tokens (Pennington et al., 2014) and updated during training. Tokens that are not covered by GloVe are replaced with a randomly initialized UNK embedding.

Training We use Adadelta (Zeiler, 2012) with $\epsilon = 1e^{-6}$ and $\rho = 0.95$ and mini-batches of size 80. We set RNN-dropout probability to 0.1 and clip gradients with norm larger than 1. All the models are trained for 500 epochs with early stopping based on development results. [4]

Ensembling We use a product of experts (Hinton, 2002) to combine predictions of 5 models, each trained on 80% of the training corpus and validated on the remaining 20%. For the CoNLL 2012 corpus, we split the training data from each sub-genre into 5 folds, such that the training data will have similar genre distributions.

Constrained Decoding We experimented with different types of constraints on the CoNLL 2005 and CoNLL 2012 development sets. Only the BIO hard constraints significantly improve over the ensemble model. Therefore, in our final results, we only use BIO hard constraints during decoding. [5]

3.3 Results

In Table 1 and 2, we compare our best single and ensemble model with previous work. Our ensemble (PoE) has an absolute improvement of 2.1 F1 on both CoNLL 2005 and CoNLL 2012 over the previous state of the art. Our single model also achieves more than a 0.4 improvement on both datasets. In comparison with the best reported results, our percentage of completely correct predicates improves by 5.9 points. While the continuing trend of improving SRL without syntax seems to suggest that neural end-to-end systems no longer needs parsers, our analysis in Section 4.4 will show that accurate syntactic information can improve these deep models.

[4] Training the full model on CoNLL 2005 takes about 5 days on a single Titan X Pascal GPU.

[5] A* search in this setting finds the optimal sequence for all sentences and is therefore equivalent to Viterbi decoding.

	Predicate Detection			End-to-end SRL (Single)			End-to-end SRL (PoE)			
Dataset	P	R	F1	P	R	F1	P	R	F1	Δ F1
CoNLL 2005 Dev.	97.4	97.4	97.4	80.3	80.4	80.3	81.8	81.2	81.5	-1.2
WSJ Test	94.5	98.5	96.4	80.2	82.3	81.2	82.0	83.4	82.7	-1.9
Brown Test	89.3	95.7	92.4	67.6	69.6	68.5	69.7	70.5	70.1	-3.5
CoNLL 2012 Dev.	88.7	90.6	89.7	74.9	76.2	75.5	76.5	77.8	77.2	-6.2
CoNLL 2012 Test	93.7	87.9	90.7	78.6	75.1	76.8	80.2	76.6	78.4	-5.0

Table 3: Predicate detection performance and end-to-end SRL results using predicted predicates. Δ F1 shows the absolute performance drop compared to our best ensemble model with gold predicates.

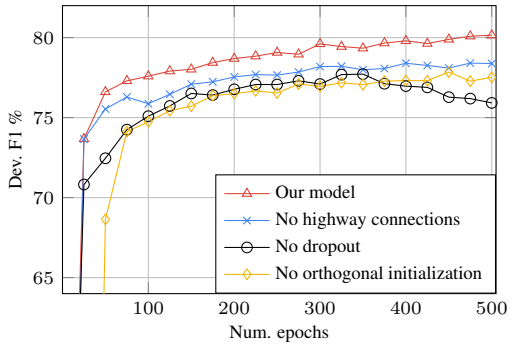

Figure 2: Smoothed learning curve of various ablations. The combination of highway layers, orthonormal parameter initialization and recurrent dropout is crucial to achieving strong performance. The numbers shown here are without constrained decoding.

3.4 Ablations

Figure 2 shows learning curves of our model ablations on the CoNLL 2005 development set. We ablate our full model by removing highway connections, RNN-dropout, and orthonormal initialization independently. Without dropout, the model overfits at around 300 epochs at 78 F1. Orthonormal parameter initialization is surprisingly important—without this, the model achieves only 65 F1 within the first 50 epochs. All 8 layer ablations suffer a loss of more than 1.7 in absolute F1 compared to the full model.

3.5 End-to-end SRL

The network for predicate detection (Section 2.3) contains 2 BiLSTM layers with 100-dimensional hidden units, and is trained for 30 epochs. For end-to-end evaluation, all arguments predicted for the false positive predicates are counted as precision loss, and all arguments for the false negative predicates are considered as recall loss.

Table 3 shows the predicate detection F1 as well as end-to-end SRL results with predicted predicates.[6] On CoNLL 2005, the predicate detector achieved over 96 F1, and the final SRL results only drop 1.2-3.5 F1 compared to using the gold predicates. However, on CoNLL 2012, the predicate detector has only about 90 F1, and the final SRL results decrease by up to 6.2 F1. This is at least in part due to the fact that CoNLL 2012 contains some nominal and copula predicates (Weischedel et al., 2013), making predicate identification a more challenging problem.

4 Analysis

To better understand our deep SRL model and its relation to previous work, we address the following questions with a suite of empirical analyses:

- What is the model good at and what kinds of mistakes does it make?
- How well do LSTMs model global structural consistency, despite conditionally independent tagging decisions?
- Is our model implicitly learning syntax, and could explicitly modeling syntax still help?

All the analysis in this section is done on the CoNLL 2005 development set with gold predicates, unless otherwise stated. We are also able to compare to previous systems whose model predictions are available online (Punyakanok et al., 2005; Pradhan et al., 2005).[7]

4.1 Error Types Breakdown

Inspired by Kummerfeld et al. (2012), we define a set of oracle transformations that fix various prediction errors *sequentially* and observe the relative improvement after each operation (see Table 4). Figure 3 shows how our work compares to the pre-

[6]The frame identification numbers reported in Pradhan et al. (2013) are not comparable, due to errors in the original release of the data, as mentioned in Täckström et al. (2015).

[7]Model predictions of CoNLL 2005 systems: http://www.cs.upc.edu/~srlconll/st05/st05.html

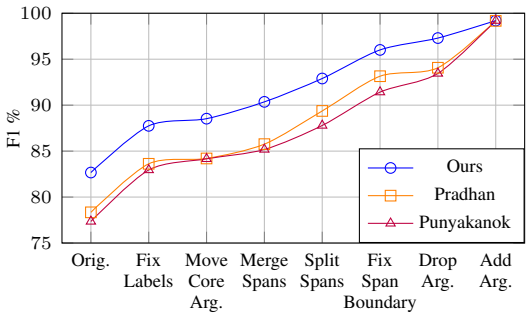

Figure 3: Performance after doing each type of oracle transformation in sequence, compared to two strong non-neural baselines. The gap is closed after the *Add Arg.* transformation, showing how our approach is gaining from predicting more arguments than traditional systems.

vious systems in terms of different types of mistakes. While our model makes a similar number of labeling errors to traditional syntax-based systems, it has far fewer missing arguments (perhaps due to parser errors making some arguments difficult to recover for syntax-based systems).

Label Confusion As shown in Table 4, our system most commonly makes labeling errors, where the predicted span is an argument but the role was incorrectly labeled. Table 5 shows a confusion matrix for the most frequent labels. The model often confuses ARG2 with AM-DIR, AM-LOC and AM-MNR. These confusions can arise due to the use of ARG2 in many verb frames to represent semantic relations such as direction or location. For example, ARG2 in the frame *move.01* is defined as *Arg2-GOL: destination*. [8] This type of argument-adjunct distinction is known to be difficult (Kingsbury et al., 2002), and it is not surprising that our neural model has many such failure cases.

Attachment Mistakes A second common source of error is reflected by the *Merge Spans* transformation (10.6%) and the *Split Spans* transformation (14.7%). These errors are closely tied to prepositional phrase (PP) attachment errors, which are also known to be some of the biggest challenges for linguistic analysis (Kummerfeld et al., 2012). Figure 4 shows the distribution of syntactic span labels involved in an attachment mistake, where 62% of the syntactic spans are prepositional phrases. For example, in *Sumitomo*

[8] Source: Unified verb index: http://verbs.colorado.edu.

Operation	Description	%
Fix Labels	Correct the span label if its boundary matches gold.	29.3
Move Arg.	Move a unique core argument to its correct position.	4.5
Merge Spans	Combine two predicted spans into a gold span if they are separated by at most one word.	10.6
Split Spans	Split a predicted span into two gold spans that are separated by at most one word.	14.7
Fix Boundary	Correct the boundary of a span if its label matches an overlapping gold span.	18.0
Drop Arg.	Drop a predicted argument that does not overlap with any gold span.	7.4
Add Arg.	Add a gold argument that does not overlap with any predicted span.	11.0

Table 4: Oracle transformations paired with the relative error reduction after each operation. All the operations are permitted only if they do not cause any overlapping arguments.

pred. \ gold	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	55	11	13	4	0	0	0	0	0
A1	78	-	46	0	0	22	11	10	25	14
A2	11	23	-	48	15	56	33	41	25	0
A3	3	2	2	-	4	0	0	0	25	14
ADV	0	0	0	4	-	0	15	29	25	36
DIR	0	0	5	4	0	-	11	2	0	0
LOC	5	9	12	0	4	0	-	10	0	14
MNR	3	0	12	26	33	0	0	-	0	21
PNC	0	3	5	4	0	11	4	2	-	0
TMP	0	8	5	0	41	11	26	6	0	-

Table 5: Confusion matrix for labeling errors, showing the percentage of predicted labels for each gold label. We only count predicted arguments that match gold span boundaries.

financed the acquisition from Sears, our model mistakenly labels the prepositional phrase *from Sears* as the ARG2 of *financed*, whereas it should instead attach to *acquisition*.

4.2 Long-range Dependencies

To analyze the model's ability to capture long-range dependencies, we compute the F1 of our model on arguments with various distances to the predicate. Figure 5 shows that performance tends to degrade, for all models, for arguments further from the predicate. Interestingly, the gap between shallow and deep models becomes much larger for the long-distance predicate-argument structures. The absolute gap between our 2 layer and 8 layer models is 3-4 F1 for arguments that are within 2 words to the predicate, and 5-6 F1 for arguments that are farther away from the predicate. Surpris-

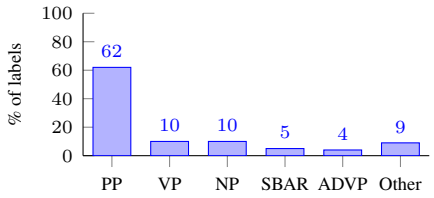

Figure 4: For cases where our model either splits a gold span into two ($Z \rightarrow XY$) or merges two gold constituents ($XY \rightarrow Z$), we show the distribution of syntactic labels for the Y span. Results show the major cause of these errors is inaccurate prepositional phrase attachment.

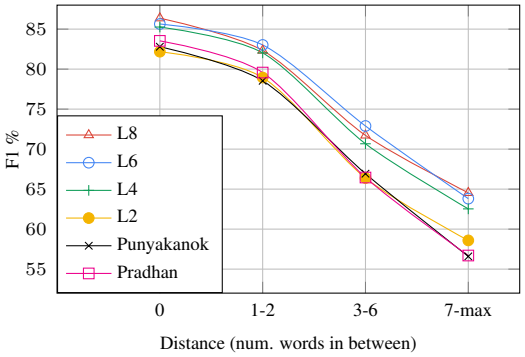

Figure 5: F1 by surface distance between predicates and arguments. Performance degrades least rapidly on long-range arguments for the deeper neural models.

ingly, the neural model performance deteriorates less severely on long-range dependencies than traditional syntax-based models.

4.3 Structural Consistency

We can quantify two types of structural consistencies: the BIO constraints and the SRL-specific constraints. Via our ablation study, we show that deeper BiLSTMs are better at enforcing these structural consistencies, although not perfectly.

BIO Violations The BIO format requires argument spans to begin with a B tag. Any I tag directly following an O tag or a tag with different label is considered a violation. Table 6 shows the number of BIO violations per token for BiLSTMs with different depths. The number of BIO violations decreases when we use a deeper model. The gap is biggest between 2-layer and 4-layer models, and diminishes after that.

It is surprising that although the deeper models generate impressively accurate token-level predictions, they still make enough BIO errors to significantly hurt performance—when these constraints are simple enough to be enforced by trivial rules. We compare the average entropy between tokens involved in BIO violations with the averaged entropy of all tokens. For the 8-layer model, the average entropy on these tokens is 30 times higher than the averaged entropy on all tokens. This suggests that BIO inconsistencies occur when there is some ambiguity. For example, if the model is unsure whether two consecutive words should belong to an ARG0 or ARG1, it might generate inconsistent BIO sequences such as B_{ARG0}, I_{ARG1} when decoding at the token level. Using BIO-constrained decoding can resolve this ambiguity and result in a structurally consistent solution.

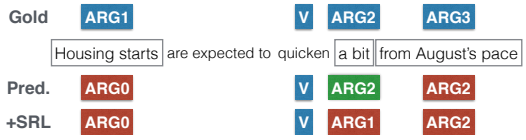

Figure 6: Example where performance is hurt by enforcing the constraint that core roles may only occur once (+SRL).

SRL Structure Violations The model predictions can also violate the SRL-specific constraints commonly used in prior work (Punyakanok et al., 2008; Täckström et al., 2015). As shown in Table 7, the model occasionally violates these SRL constraints. With our constrained decoding algorithm, it is straightforward to enforce the unique core roles (U) and continuation roles (C) constraints during decoding. The constrained decoding results are shown with the model named *L8+PoE+SRL* in Table 7.

Although the violations are eliminated, the performance does not significantly improve. This is mainly due to two factors: (1) the model often already satisfies these constraints on its own, so the number of violations to be fixed are relatively small, and (2) the gold SRL structure sometimes violates the constraints and enforcing hard constraints can hurt performance. Figure 6 shows a sentence in the CoNLL 2005 development set. Our original model produces two ARG2s for the predicate *quicken*, and this violates the SRL constraints. When the A* decoder fixes this violation, it changes the first ARG1 into ARG2 because ARG0, ARG1, ARG2 is a more frequent pattern in the training data and has higher overall score.

	Accuracy		Violations	Avg. Entropy	
Model (no BIO)	F1	Token	BIO	All	BIO
L8+PoE	81.5	91.5	0.07	0.02	0.72
L8	80.5	90.9	0.07	0.02	0.73
L6	80.1	90.3	0.06	0.02	0.72
L4	79.1	90.2	0.08	0.02	0.70
L2	74.6	88.4	0.18	0.03	0.66

Table 6: Comparison of BiLSTM models without BIO decoding. We compare F1 and token-level accuracy (Token), averaged BIO violations per token (BIO), overall model entropy (All) model entropy at tokens involved in BIO violations (BIO). Increasing the depth of the model beyond 4 does not produce more structurally consistent output, emphasizing the need for constrained decoding.

4.4 Can Syntax Still Help SRL?

The Propbank-style SRL formalism is closely tied to syntax (Bonial et al., 2010; Weischedel et al., 2013). In Table 7, we show that 98.7% of the gold SRL arguments match an unlabeled constituent in the gold syntax tree. Similar to some recent work (Zhou and Xu, 2015), our model achieves strong performance without directly modeling syntax. A natural question follows: are neural SRL models implicitly learning syntax? Table 7 shows the trend of deeper models making predictions that are more consistent with the gold syntax in terms of span boundaries. With our best model (*L8+PoE*), 94.3% of the predicted arguments spans are part of the gold parse tree. This consistency is on par with previous CoNLL 2005 systems that directly model constituency and use predicted parse trees as features (Punyakanok, 95.3% and Pradhan, 93.0%).

Constrained Decoding with Syntax The above analysis raises a further question: would improving consistency with syntax provide improvements for SRL? Our constrained decoding algorithm described in Section 2.2 enables us to inject syntax as a decoding constraint without having to retrain the model. Specifically, if the decoded sequence contains k arguments that do not match any unlabeled syntactic constituent, it will receive a penalty of kC, where C is a single parameter dictating how much the model should trust the provided syntax. In Figure 7, we compare the SRL accuracy with syntactic constraints specified by gold parse or automatic parses. When using gold syntax, the predictions improve up to 2 F1 as the penalty increases. A state-of-the-art parser (Choe and Charniak, 2016) provides smaller gains, while using the Charniak parser (Charniak, 2000) hurts performance if the model places too much trust in it. These results suggest that high-quality syntax can still make a large impact on SRL.

			SRL-Violations		
Model or Oracle	F1	Syn %	U	C	R
Gold	100.0	98.7	24	0	61
L8+PoE	82.7	94.3	37	3	68
L8	81.6	94.0	48	4	73
L6	81.4	93.7	39	3	85
L4	80.5	93.2	51	3	84
L2	77.2	91.3	96	5	72
L8+PoE+SRL	82.8	94.2	5	1	68
L8+PoE+AutoSyn	83.2	96.1	113	3	68
L8+PoE+GoldSyn	85.0	97.6	102	3	68
Punyakanok	77.4	95.3	0	0	0
Pradhan	78.3	93.0	84	3	58

Table 7: Comparison of models with different depths and decoding constraints (in addition to BIO) as well as two previous systems. We compare F1, unlabeled agreement with gold constituency (Syn%) and each type of SRL-constraint violations (**U**nique core roles, **C**ontinuation roles and **R**eference roles). Our best model produces a similar number of constraint violations to the gold annotation, explaining why deterministically enforcing these constraints is not helpful.

A known challenge for syntactic parsers is robustness on out-of-domain data. Therefore we provide experimental results in Table 8 for both CoNLL 2005 and CoNLL 2012, which consists of 8 different genres. The penalties are tuned on the two development sets separately ($C = 10000$ on CoNLL 2005 and $C = 20$ on CoNLL 2012). On the CoNLL 2005 development set, the predicted syntax gives a 0.5 F1 improvement over our best model, while on in-domain test and out-of-domain development sets, the improvement is much smaller. As expected, on CoNLL 2012, syntax improves most on the newswire (NW) domain. These improvements suggest that while decoding with hard constraints is beneficial, joint training or multi-task learning could be even more effective by leveraging full, labeled syntactic structures.

5 Related Work

Traditional approaches to semantic role labeling have used syntactic parsers to identify constituents and model long-range dependencies, and enforced

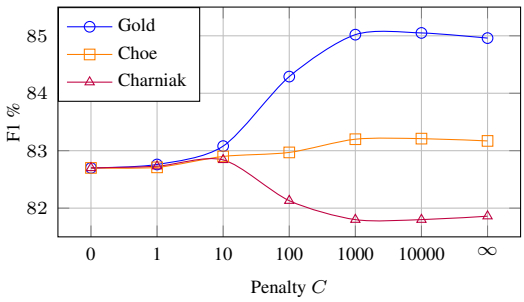

Figure 7: Performance of syntax-constrained decoding as the non-constituent penalty increases for syntax from two parsers (from Choe and Charniak (2016) and Charniak (2000)) and gold syntax. The best existing parser gives a small improvement, but the improvement from gold syntax shows that there is still potential for syntax to help SRL.

	CoNLL-05		CoNLL-2012 Dev.						
	Dev.	Test	BC	BN	NW	MZ	PT	TC	WB
L8+PoE	82.7	84.6	81.4	82.8	82.8	80.4	93.6	84.8	81.0
+AutoSyn	83.2	84.8	81.5	82.8	83.2	80.6	93.7	84.9	81.1

Table 8: F1 on CoNLL 2005, and the development set of CoNLL 2012, broken down by genres. Syntax-constrained decoding (+AutoSyn) shows bigger improvement on in-domain data (CoNLL 05 and CoNLL 2012 NW).

global consistency using integer linear programming (Punyakanok et al., 2008) or dynamic programs (Täckström et al., 2015). More recently, neural methods have been employed on top of syntactic features (FitzGerald et al., 2015; Roth and Lapata, 2016). Our experiments show that off-the-shelf neural methods have a remarkable ability to learn long-range dependencies, syntactic constituency structure, and global constraints without coding task-specific mechanisms for doing so.

An alternative line of work has attempted to reduce the dependency on syntactic input for semantic role labeling models. Collobert et al. (2011) first introduced an end-to-end neural-based approach with sequence-level training and uses a convolutional neural network to model the context window. However, their best system fell short of traditional feature-based systems. Neural methods have also been used as classifiers in transition-based SRL systems (Henderson et al., 2013; Swayamdipta et al., 2016). Most recently, several successful LSTM-based architectures have achieved state-of-the-art results in English span-based SRL (Zhou and Xu, 2015), Chinese SRL (Wang et al., 2015), and dependency-based SRL (Marcheggiani et al., 2017) with little to no syntactic input. Our techniques push results to more than 3 F1 over the best syntax-based models. However, we also show that there is potential for syntax to further improve performance.

6 Conclusion and Future Work

We presented a new deep learning model for span-based semantic role labeling with a 10% relative error reduction over the previous state of the art. Our ensemble of 8 layer BiLSTMs incorporated some of the recent best practices such as orthonormal initialization, RNN-dropout, and highway connections, and we have shown that they are crucial for getting good results with deep models.

Extensive error analysis sheds light on the strengths and limitations of our deep SRL model, with detailed comparison against shallower models and two strong non-neural systems. While our deep model is better at recovering long-distance predicate-argument relations, we still observe structural inconsistencies, which can be alleviated by constrained decoding.

Finally, we posed the question of whether deep SRL still needs syntactic supervision. Despite recent success without syntactic input, we found that our best neural model can still benefit from accurate syntactic parser output via straightforward constrained decoding. In our oracle experiment, we observed a 3 F1 improvement by leveraging gold syntax, showing the potential for high quality parsers to further improve deep SRL models.

Acknowledgments

The research was supported in part by DARPA under the DEFT program (FA8750-13-2-0019), the ARO (W911NF-16-1-0121), the NSF (IIS-1252835, IIS-1562364), gifts from Google and Tencent, and an Allen Distinguished Investigator Award. We are grateful to Mingxuan Wang for sharing his highway LSTM implementation and Sameer Pradhan for help with the CoNLL 2012 dataset. We thank Nicholas FitzGerald, Dan Garrette, Julian Michael, Hao Peng, and Swabha Swayamdipta for helpful comments, and the anonymous reviewers for valuable feedback.

References

Claire Bonial, Olga Babko-Malaya, Jinho D Choi, Jena Hwang, and Martha Palmer. 2010. Propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder* .

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 152–164.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of the First North American chapter of the Association for Computational Linguistics conference (NAACL)*. Association for Computational Linguistics, pages 132–139.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proc. of the 2016 Conference of Empirical Methods in Natural Language Processing (EMNLP)*.

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.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 960–970.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1019–1027.

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

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14(8):1771–1800.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the penn treebank. In *Proceedings of the human language technology conference*. pages 252–256.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proc. of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1048–1059.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural ccg parsing with optimality guarantees. In *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mike Lewis and Mark Steedman. 2014. A* ccg parsing with a supertag-factored model. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 990–1000.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. *arXiv preprint arXiv:1701.02593* .

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proc. of the 2005 Conference on Computational Natural Language Learning (CoNLL)*. pages 217–220.

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 the 2013 Conference on Computational Natural Language Learning (CoNLL)*. pages 143–152.

Vasin Punyakanok, Peter Koomen, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proc. of the 2005 Conference on Computational Natural Language Learning (CoNLL)*.

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.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* .

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*. pages 2377–2385.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. In *Proc. of the 2016 Conference on Computational Natural Language Learning (CoNLL)*. page 187.

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

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

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1626–1631.

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* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. 2016. Highway long short-term memory rnns for distant speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 5755–5759.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.