Linguistically-Informed Self-Attention for Semantic Role Labeling

Anonymous ACL submission

Abstract

The current state-of-the-art end-to-end semantic role labeling (SRL) model is a deep neural network architecture with no explicit linguistic features. However, prior work has shown that gold syntax trees can dramatically improve SRL, suggesting that neural network models could see great improvements from explicit modeling of syntax. In this work, we present linguistically-informed self-attention (LISA): a new neural network model that combines multi-head self-attention with multi-task learning across dependency parsing, part-of-speech, predicate detection and SRL. For example, syntax is incorporated by training one of the attention heads to attend to syntactic parents for each token. Our model can predict all of the above tasks, but it is also trained such that if a high-quality syntactic parse is already available, it can be beneficially injected at test time without re-training our SRL model. In experiments on the CoNLL-2005 SRL dataset LISA achieves an increase of 2.5 F1 absolute over the previous state-of-the-art on newswire with predicted predicates and more than 2.0 F1 on out-of-domain data. On ConLL-2012 English SRL we also show an improvement of more than 3.0 F1, a 13% reduction in error.

1 Introduction

Semantic role labeling (SRL) extracts a high-level representation of meaning from a sentence, labeling e.g. *who* did *what* to *whom*. Explicit representations of such semantic information have been shown to improve results in challenging downstream tasks such as dialog systems (Tur et al., 2005; Chen et al., 2013), machine reading (Berant et al., 2014; Wang et al., 2015) and machine translation (Liu and Gildea, 2010; Bazrafshan and Gildea, 2013).

Though syntax was long considered an obvious prerequisite for SRL systems (Levin, 1993; Punyakanok et al., 2008), recently deep neural network architectures have surpassed syntactically-informed models (Zhou and Xu, 2015; Marcheggiani et al., 2017; He et al., 2017; Tan et al., 2018), achieving state-of-the art SRL performance with no explicit modeling of syntax.

Still, recent work (Roth and Lapata, 2016; He et al., 2017; Marcheggiani and Titov, 2017) indicates that neural network models could see even higher performance gains by leveraging syntactic information rather than ignoring it. He et al. (2017) indicate that many of the errors made by a strong syntax-free neural-network on SRL are tied to certain syntactic confusions such as prepositional phrase attachment, and show that while constrained inference using a relatively low-accuracy predicted parse can provide small improvements in SRL accuracy, providing a gold-quality parse leads to very significant gains. Marcheggiani and Titov (2017) incorporate syntax from a high-quality parser (Kiperwasser and Goldberg, 2016) using graph convolutional neural networks (Kipf and Welling, 2017), but like He et al. (2017) they attain only small increases over a model with no syntactic parse, and even perform worse than a syntax-free model on out-of-domain data. These works suggest that though syntax has the potential to improve neural network SRL models, we have not yet designed an architecture which maximizes the benefits of auxiliary syntactic information.

In response, we propose *linguistically-informed self-attention* (LISA): a model which combines multi-task learning (Caruana, 1993) with stacked layers of multi-head self-attention (Vaswani et al., 2017) trained to act as an oracle providing syntactic parses to downstream parameters tasked with predicting semantic role labels. Our model is end-to-end: earlier layers are trained to predict prerequisite parts-of-speech and predicates, which are supplied to later layers for scoring. The model is trained such that, as syntactic parsing models improve, providing high-quality parses at test time can only improve its performance, allowing the model to benefit maximally from improved parsing models without requiring re-training. Unlike previous work, we encode each sentence only once, predict its predicates, part-of-speech tags and syntactic parse, then predict the semantic roles for all predicates in the sentence in parallel, leading to exceptionally fast training and decoding speeds: our model matches state-of-the art accuracy in less than one quarter the training time.

In extensive experiments on the CoNLL-2005 and CoNLL-2012 datasets, we show that our linguistically-informed models consistently outperform the syntax-free state-of-the-art for SRL models with predicted predicates. On CoNLL-2005, our single model out-performs the previous state-of-the-art single model on the WSJ test set by nearly 1.5 F1 points absolute using its own predicted parses, and by 2.5 points using a state-of-the-art parse (Dozat and Manning, 2017). On the challenging out-of-domain Brown test set, our model also improves over the previous state-of-the-art by more than 2.0 F1. On CoNLL-2012, our model gains 1.4 points with its own parses and more than 3.0 points absolute over previous work: 13% reduction in error. Our single models also out-perform state-of-the-art ensembles across all datasets, up to more than 1.4 F1 over a strong five-model ensemble on CoNLL-2012.

2 Model

Our goal is to design an efficient neural network model which makes use of linguistic information as effectively as possible in order to perform end-to-end SRL. LISA achieves this by combining: (1) Multi-task learning across four related tasks; (2) a new technique of supervising neural attention to predict syntactic dependencies; and (3) careful conditioning of different parts of the model on gold versus predicted annotations during training.

Figure 1 depicts the overall architecture of our model. To first encode rich token-level representations, our neural network model takes word embeddings as input, which are passed through stacked convolutional, feed-forward and multi-head self-attention layers (Vaswani et al., 2017) to efficiently produce contextually encoded token embeddings (Eqns. 1-4). We choose this combination of network components because we found it to perform better than LSTM, CNN or self-attention layers alone in terms of speed-accuracy Pareto efficiency in initial experiments.

To predict semantic role labels, the contextually encoded tokens are projected to distinct *predicate* and *role* embeddings (§2.4), and each predicted predicate is scored with the sequence's role representations using a bilinear model (Eqn. 5), producing per-label scores for BIO-encoded semantic role labels for each token and each semantic frame in the sequence entirely in parallel.

To incorporate syntax, one self-attention head is trained to attend to each token's syntactic parent, allowing the model to use this attention head as an oracle for syntactic dependencies. We encourage the model to use this syntactic information as much as possible by giving subsequent layers access to a gold parse oracle during training, allowing either the predicted parse attention or an externally predicted parse to be used at test time. We introduce this *syntactically-informed self-attention* in more detail in §2.2.

We integrate part-of-speech and predicate information into earlier layers by re-purposing representations closer to the input to predict predicates and part-of-speech (POS) tags (§2.3). We simplify optimization and benefit from shared statistical strength derived from highly correlated POS and predicates by treating tagging and predicate detection as a single task, performing multi-class classification into the joint Cartesian product space of POS and predicate labels.

The model is trained end-to-end by maximum likelihood using stochastic gradient descent (§2.5).

2.1 Neural network token encoder

The input to the network is a sequence $\mathcal{X}$ of T token representations x_t. Each token representation is the sum of a fixed (pre-trained) and learned (randomly initialized) word embedding. In the case where we feed a predicate indicator embedding p_t as input to the network, we concatenate that representation with the word embedding to give the final token embedding.

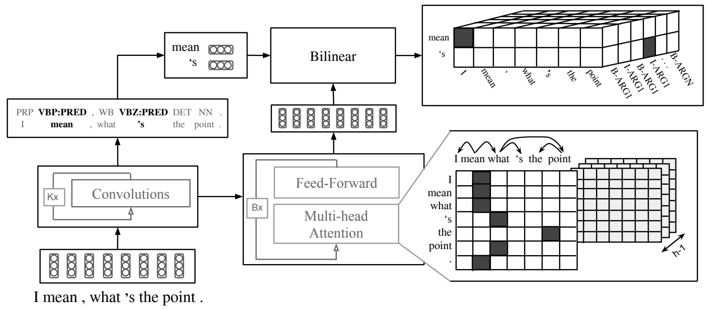

Figure 1: Word embeddings are input to k CNN layers. This output is passed to (1) a joint POS/predicate classifier and (2) j layers of multi-head self-attention. One attention head is trained to attend to parse parents. A bilinear operation scores distinct predicate and role representations to produce BIO-encoded SRL predictions.

These token representations are then the input to a series of width-3 stacked convolutional layers with residual connections (He et al., 2016), producing contextually embedded token representations $c_t^{(k)}$ at each layer k. We denote the kth convolutional layer as $C^{(k)}$. Let $r(\cdot)$ denote the leaky ReLU activation function (Maas et al., 2013), and let $LN(\cdot)$ denote layer normalization (Ba et al., 2016), then starting with input x_t, the final CNN output is given by the recurrence:

$$c_t^{(k)} = LN(c_t^{(k-1)} + r(C^{(k)} c_t^{(k-1)})) \qquad (1)$$

We use leaky ReLU activations to avoid dead activations and vanishing gradients (Hochreiter, 1998), whereas layer normalization reduces covariate shift between layers (Ioffe and Szegedy, 2015) without requiring distinct train- and test-time operations.

We then feed this representation as input to a series of residual multi-head self-attention layers with feed-forward connections in the style of the encoder portion of the Transformer architecture of Vaswani et al. (2017). This architecture allows each token to observe long-distance context from the entire sentence like an LSTM, but unlike an LSTM, representations for all tokens can be computed in parallel at each layer.

We first project[1] the output of the convolutional layers to a representation $c_t^{(p)}$ that is the same size as the output of the self-attention layers and add a positional encoding vector computed as a deterministic sinusoidal function of t, following Vaswani et al. (2017). We then apply the self-attention layers to this projected representation, applying layer normalization after each residual connection. Denoting the jth self-attention layer as $T^{(j)}(\cdot)$, the output of that layer $s_t^{(j)}$, and h as the number of attention heads at each layer, the following recurrence applied to initial input $c_t^{(p)}$:

$$s_t^{(j)} = LN(s_t^{(j-1)} + T^{(j)}(s_t^{(j-1)})) \qquad (2)$$

gives our final token representations $s_t^{(j)}$.

Each self-attention layer is made up of three stages: First, an initial projection of each input to H key, value and query representations a_{th}^{key}, a_{th}^{query} and a_{th}^{value}, with dimensions d_k, d_q and d_v respectively. Next, we take the product of the vector a_t^{key} with the matrix Q_h of query vectors of each other token in the sequence with respect to head h, scale the result by $d_k^{-0.5}$, and normalize with the softmax function. This gives us a vector of T attention weights with respect to token t:

$$a_{th} = \text{softmax}(d_k^{-0.5} Q_h a_{th}^{key}) \qquad (3)$$

with which we perform a weighted sum of the value vectors a_{vh}^{value} for each other token v to compose a new token representation for each attention head. The representations for each attention head are concatenated into a single vector a_t. We feed this representation through a multi-layer perception, add it to the initial representation and apply layer normalization to give the final output of self-attention layer j:

$$s_t^{(j)} = LN(s_t^{(j-1)} + r(W_3 r(W_2 r(W_1 a_t)))) \quad (4)$$

2.2 Syntactically-informed self-attention

Typically, neural attention mechanisms are left on their own to learn to attend to relevant inputs. Instead, we propose training the self-attention to attend to specific tokens corresponding to the syntactic structure of the sentence as a mechanism for passing linguistic knowledge to later layers.

Specifically, we train with an auxiliary objective on one attention head which encourages that head to attend to each token's parent in a syntactic dependency tree. We use the attention weights a_{th} between token t and each other token q in the sequence as the distribution over possible heads for

[1] All of our linear projections include bias terms, which we omit in this exposition for the sake of clarity.

token t: $P(q = \text{head}(t) \mid \mathcal{X}) = a_{thq}$, where we define the root token as having a self-loop. This attention head thus emits a directed graph[2] where each token's head is the token to which the attention assigns the highest weight.

This attention head now becomes an oracle for syntax, denoted $\mathcal{P}$, providing a dependency parse to downstream layers. This model not only predicts its own dependency arcs, but allows for the injection of auxiliary parse information at test time by simply swapping out the oracle given by a_{th} to one produced by e.g. a state-of-the-art parser. In this way, our model can benefit from improved, external parsing models without re-training. Unlike typical multi-task models, ours maintains the ability to leverage external syntactic information.

Unfortunately, this parsing objective does not maximize the model's ability to use the syntactic information for predicting semantic role labels in later layers. Though one would expect model accuracy to increase significantly if injecting e.g. gold dependency arcs into the learned attention head at test time, we find that without specialized training this is not the case: Without the training described below, fixing $\mathcal{P}$ to gold parses at test time improves SRL F1 over predicted parses by 0.3 points, whereas the F1 increases by 7.0 when the model is trained with our technique.[3] Injecting high-accuracy predicted parses follows the same trend.

We hypothesize that the model is limited by the poor representations to which it has access during early training. When training begins, the model observes randomly initialized attention rather than strong syntactic information, even in the head which will be trained to provide it with such information. Thus rather than learning to look to this head for syntax, the model learns to encode that information itself, like a model which was trained with no explicit syntax at all. Prior work (Zhang and Weiss, 2016), has alleviated this problem by pre-training the parameters of earlier tasks before initiating the training of later tasks. However, optimization in this setting becomes computationally expensive and complicated, especially as the number of auxiliary tasks increases, and when using adaptive techniques for stochastic gradient descent such as Adam (Kingma and Ba, 2015).

To alleviate this problem, during training we clamp $\mathcal{P}$ to the gold parse ($\mathcal{P}_G$) when using its representation for later layers, while still training a_{th} to predict syntactic heads. We find that this vastly improves the model's ability to leverage the parse information encoded in $\mathcal{P}$ at test time. Our approach is essentially an extension of teacher forcing (Williams and Zipser, 1989) to MTL. Though a large body of work suggests that, by closing the gap between observed data distributions during train and test, training on predicted rather than gold labels leads to improved test-time accuracy (Daumé III et al., 2009; Ross et al., 2011; Choi and Palmer, 2011; Goldberg and Nivre, 2012; Chang et al., 2015; Bengio et al., 2015; Ballesteros et al., 2016), our simple approach works surprisingly well; we leave more advanced scheduled sampling techniques to future work.

2.3 Multi-task learning

We also share the parameters of lower layers in our model to predict POS tags and predicates. Following He et al. (2017), we focus on the end-to-end setting, where predicates must be predicted on-the-fly. Since we also train our model to predict syntactic dependencies, it is beneficial to give the model some knowledge of POS information. While much previous work employs a pipelined approach to both POS tagging for dependency parsing and predicate detection for SRL, we take a multi-task learning (MTL) approach (Caruana, 1993), sharing the parameters of earlier layers in our SRL model with a joint POS and predicate detection objective. Since POS is a strong predictor of predicates,[4] and the complexity of training a multi-task model increases with the number of tasks, we combine POS tagging and predicate detection into a joint label space: for each POS tag TAG in the training data which co-occurs with a predicate, we add a label of the form TAG:PREDICATE.

Specifically, we experiment with feeding a lower-level representation, r_t, which may be either $c_t^{(k)}$, the output of the convolutional layers, or $s_t^{(1)}$, the output of the first self-attention layer, to a linear classifier. We compute locally-normalized probabilities using the softmax function: $P(z_t \mid \mathcal{X}) \propto \exp(r_t)$, where z_t is a label in the joint space. We apply this supervision at earlier lay-

[2] In most but not all cases, the head emits a tree, but we do not currently enforce it.

[3] CoNLL-2012. CoNLL-2005 yields similar results.

[4] All of the predicates in the CoNLL-2005 dataset are verbs, whereas the CoNLL-2012 dataset includes some nominal predicates.

ers following prior work (Søgaard and Goldberg, 2016; Hashimoto et al., 2017).

2.4 Predicting semantic roles

Our final goal is to predict semantic roles for each predicate in the sequence. We score each predicate[5] against each token in the sequence using a bilinear operation, producing per-label scores for each token for each predicate, with predicates and syntax determined by oracles $\mathcal{V}$ and $\mathcal{P}$.

First, we project each token representation $s_t^{(j)}$ to a predicate-specific representation s_t^{pred} and a role-specific representation s_t^{role}. We then provide these representations to a bilinear transformation U for scoring. So, the role label scores s_{ft} for the token at index t with respect to the predicate at index f (i.e. token t and frame f) are given by:

$$s_{ft} = (s_f^{pred})^T U s_t^{role} \tag{5}$$

which can be computed in parallel across all semantic frames in an entire minibatch. We calculate a locally normalized distribution over role labels for token t in frame f using the softmax function: $P(y_{ft} \mid \mathcal{P}, \mathcal{V}, \mathcal{X}) \propto \exp(s_{ft})$.

At test time, we perform constrained decoding using the Viterbi algorithm to emit valid sequences of BIO tags, using unary scores s_{ft} and the transition probabilities given by the training data.

2.5 Training

We maximize the sum of the likelihoods of the individual tasks, entrusting the network to learn parameters which model the complex coupling between tasks, rather than explicitly modeling structure in the output labels:

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T}\Big[\sum_{f=1}^{F} & \log P(y_{ft} \mid \mathcal{P}_G, \mathcal{V}_G, \mathcal{X}) \\ & + \log P(z_t \mid \mathcal{X}) \\ & + \lambda \log P(\mathrm{head}(t) \mid \mathcal{X})\Big] \end{aligned} \tag{6}$$

where λ is a penalty on the syntactic attention loss. Note that as described in §2.2, the terms for the syntactically-informed attention and joint predicate/POS prediction are conditioned only on the input sequence $\mathcal{X}$, whereas the SRL component is conditioned on gold predicates $\mathcal{V}_G$ and gold parse structure $\mathcal{P}_G$ during training.

[5] CoNLL-2012 contains only single-word predicates. In CoNLL-2005, some predicates are multi-word verbs, such as "sign up." In this case, we drop the particle.

We train the model using Nadam (Dozat, 2016) SGD combined with the learning rate schedule in Vaswani et al. (2017). In addition to MTL, we regularize our model using element-wise and word dropout (Srivastava et al., 2014; Dai and Le, 2015) and parameter averaging. We use gradient clipping to avoid exploding gradients (Bengio et al., 1994; Pascanu et al., 2013). Our models are implemented in TensorFlow (Abadi et al., 2015) with source code and models to be released upon publication. Additional details on optimization and hyperparameters are included in Appendix A.

3 Related work

Early approaches to SRL (Pradhan et al., 2005; Surdeanu et al., 2007; Johansson and Nugues, 2008; Toutanova et al., 2008) focused on developing rich sets of linguistic features as input to a linear model, often combined with complex constrained inference e.g. with an ILP (Punyakanok et al., 2008). Täckström et al. (2015) showed that constraints could be enforced more efficiently using a clever dynamic program for exact inference. Sutton and McCallum (2005) modeled syntactic parsing and SRL jointly, and Lewis et al. (2015) jointly modeled SRL and CCG parsing.

Collobert et al. (2011) were among the first to use a neural network model for SRL, a CNN over word embeddings which failed to out-perform non-neural models. FitzGerald et al. (2015) successfully employed neural networks by embedding lexicalized features and providing them as factors in the model of Täckström et al. (2015).

More recent neural models are syntax-free. Zhou and Xu (2015), Marcheggiani et al. (2017) and He et al. (2017) all use variants of deep LSTMs with constrained decoding in e.g. a linear-chain CRF (Lafferty et al., 2001), while Tan et al. (2018) alternate self-attention and LSTM layers. Like this work, He et al. (2017) present end-to-end experiments where they predict predicates using an LSTM. Concurrent to this work, Peters et al. (2018) report significant gains on CoNLL-2012 with gold predicates by jointly training a wide and deep LSTM language model and using its hidden representations as token embeddings for He et al. (2017). Future work could explore synergies and speed-accuracy trade-offs between LISA and Peters et al. (2018), analyzing the extent to which the deep LSTM of Peters et al. (2018) models syntax efficiently and effectively for SRL.

Some work has incorporated syntax into neural models for SRL. Roth and Lapata (2016) incorporate syntax by embedding dependency paths, and similarly Marcheggiani and Titov (2017) encode syntax using a graph CNN over a predicted syntax tree, out-performing models without syntax on CoNLL-2009. However, both models are at risk of over-fitting to or otherwise inheriting the flaws of the predictions upon which they are trained. Indeed, Marcheggiani and Titov (2017) report that their model does not out-perform a similar syntax-free model on out-of-domain data.

Syntactically-informed self-attention is similar to the concurrent work of Liu and Lapata (2018), who use edge marginals produced by the matrix-tree algorithm as attention weights for document classification and natural language inference.

MTL (Caruana, 1993) is popular in NLP. Collobert et al. (2011) multi-task part-of-speech, chunking, NER and SRL. Zhang and Weiss (2016) jointly train a dependency parser and POS tagger. Søgaard and Goldberg (2016) train a multi-task model for POS, chunking and CCG tagging. Hashimoto et al. (2017) built a single, jointly trained model for POS, chunking, parsing, semantic relatedness and entailment, using a special regularization scheme to facilitate training. Bingel and Søgaard (2017) and Alonso and Plank (2017) investigate different combinations of NLP tagging tasks including POS, chunking and FrameNet semantics (Baker, 2014). Luong et al. (2016) enhance a machine translation model by multi-tasking with parsing. MTL has also been applied to semantic dependency parsing: Swayamdipta et al. (2017) multi-task with a syntax-based tagging objective while Peng et al. (2017) train on three semantic dependency frameworks.

The question of training on gold versus predicted labels is closely related to learning to search (Daumé III et al., 2009; Ross et al., 2011; Chang et al., 2015) and scheduled sampling (Bengio et al., 2015), with applications in NLP to sequence labeling and transition-based parsing (Choi and Palmer, 2011; Goldberg and Nivre, 2012; Ballesteros et al., 2016). We believe more sophisticated approaches extending these techniques to MTL could improve LISA in future work.

4 Experimental results

We present results on the CoNLL-2005 shared task (Carreras and Marquèz, 2005) and the CoNLL-2012 English subset of OntoNotes 5.0 (Pradhan et al., 2006), achieving state-of-the-art results for a single model with predicted predicates on both corpora. In all experiments, we initialize with pre-trained GloVe word embeddings (Pennington et al., 2014), hyperparameters that resulted in the best performance on the validation set were selected via a small grid search, and models were trained for a maximum of 7 days on one TitanX GPU using early stopping on the validation set.[6] For CoNLL-2005 we convert constituencies to dependencies using the Stanford head rules v3.5 (de Marneffe and Manning, 2008) and for CoNLL-2012 we use ClearNLP (Choi and Palmer, 2012b), following previous work. A detailed description of hyperparameter settings and data pre-processing can be found in Appendix A.

For both datasets, we compare our best models (**LISA**$_G$) to three strong sets of baselines: the syntax-free deep LSTM model of He et al. (2017) which was the previous state-of-the-art model for SRL with predicted predicates, both as an ensemble of five models (**PoE**) and as a single model (**single**); an ablation of our own self-attention model where we don't incorporate any syntactic information (**SA**), and another ablation where we do train with syntactically-informed self-attention, but where downstream layers in the model are conditioned on the predicted attention weights (i.e. dynamic oracle, **D**) rather than the gold parse (**G**) during training (**LISA**$_D$).

We demonstrate that our models can benefit from injecting state-of-the-art predicted parses at test time (**+D&M**) by setting the attention oracle to parses predicted by Dozat and Manning (2017), the state-of-the-art dependency parser for English PTB and winner of the 2017 CoNLL shared task (Zeman et al., 2017). In all cases, using these parses at test time improves performance.

We also evaluate our model using the gold syntactic parse at test time (**+Gold**), to provide an upper bound for the benefit that syntax could have for SRL using LISA. These experiments show that despite LISA's strong performance, there remains substantial room for improvement. In §4.4 we perform detailed analysis comparing SRL models us-

[6]Our best reported CoNLL-2012 model was trained for just under 6 days, though it matched He et al. (2017) after 1 day 4 hours. Our best ConLL-2005 model was trained for 3.5 days, though we match He et al. (2017) after 1.5 days. He et al. (2017) report training their CoNLL-2005 model for 5 days on the same hardware.

Test set	D&M POS/UAS	LISA_D POS/UAS	LISA_G POS/UAS
WSJ	97.0/96.1	97.0/92.4	97.0/93.0
Brown	94.5/92.0	94.9/87.8	95.0/88.8
CoNLL-12	97.5/94.4	96.5/90.6	96.6/91.6

Table 1: Parsing (UAS) and POS accuracies of the models used in SRL experiments on both datasets.

Model	P	R	F1
He et al. (2017) single	78.6	75.1	76.8
He et al. (2017) PoE	80.2	76.6	78.4
SA	78.54	76.90	77.71
LISA_D	79.28	76.73	77.98
+D&M	79.43	76.83	78.11
+Gold	*79.62*	*77.03*	*78.31*
LISA_G	79.06	77.36	78.20
+D&M	**80.71**	**79.07**	**79.88**
+Gold	*86.20*	*84.28*	*85.23*

Table 2: Precision, recall and F1 on the CoNLL-2012 test set. Italics indicate a synthetic upper bound obtained by providing a gold test parse.

ing gold and predicted parses to better understand where syntax provides the most benefit to SRL, and what remains to be improved.

4.1 Dependency parsing

We first report the unlabeled attachment scores (UAS) of our parsing models on the CoNLL-2005 and 2012 SRL test sets (Table 1). Dozat and Manning (2017) achieves the best scores, obtaining state-of-the-art results on the CoNLL-2012 split of OntoNotes in terms of UAS, followed by LISA_G then LISA_D.[7] We still see SRL accuracy improvements despite our relatively low parser UAS from LISA's predicted parses, but the difference in accuracy likely explains the large increase in SRL we see from decoding with D&M parses.

4.2 CoNLL-2012 SRL

Table 2 reports precision, recall and F1 on the CoNLL-2012 test set. Our SA model already performs strongly without access to syntax, out-performing the single model of He et al. (2017) but under-performing their ensemble. Adding syntactically-informed training to the self-attention increases over the model without syntax, achieving about the same score using dynamic versus gold parse oracles for downstream layers during training. When evaluating using an injected parse, we see that (1) a large increase of more than 1.5 F1 absolute for LISA_G and (2) this increase is markedly larger than for LISA_D. With the injected D&M parse, our single models impressively outperform the ensemble.

[7] The previous best score we know of is 92.5 attained by Mate (Bohnet, 2010), as reported in Choi et al. (2015).

Model	P	R	F1
He et al. (2017)	93.7	87.9	90.7
SA	98.67	**89.39**	**93.80**
LISA_G	**99.12**	88.66	93.60

Table 3: Predicate detection precision, recall and F1 on the CoNLL-2012 test set.

We also report predicate detection precision, recall and F1 (Table 3). Our models obtain much higher scores than He et al. (2017) on this task, likely explaining improvements of our basic SA model over theirs. Like He et al. (2017), our model achieves much higher precision than recall, indicative of the model memorizing predicate words from the training data. Interestingly, our SA model out-performs syntax-infused models by a small margin. We hypothesize that this could be due to asking the LISA models to learn to predict more tasks, taking some model capacity away from predicate detection.

4.3 CoNLL-2005 SRL

Table 4 lists precision, recall and F1 on the CoNLL-2005 test sets. Unlike on CoNLL-2012, our SA baseline does not out-perform He et al. (2017). This is likely due to their predicate detection scores being closer to ours on this data (Table 5). Interestingly, unlike on CoNLL-2012 we see a distinct improvement between LISA_G and LISA_D in models which use LISA parses: LISA_G training leads to improved SRL scores by more than 1 F1 absolute using LISA-predicted parses. Similar to CoNLL-2012, we see very little improvement from adding D&M parses at test-time with the dynamic oracle, whereas we obtain the highest score of all when using D&M parses combined with LISA_G, demonstrating that our training technique markedly improves LISA's ability to leverage improved parses at test time. Our best single models out-perform the ensemble of

WSJ Test	P	R	F1
He et al. (2017) single	80.2	82.3	81.2
He et al. (2017) PoE	82.0	83.4	82.7
SA	81.43	80.69	81.06
LISA_D	81.95	81.14	81.54
+D&M	82.12	81.30	81.70
LISA_G	82.78	82.57	82.68
+D&M	**83.71**	**83.69**	**83.70**
+Gold	*87.50*	*87.47*	*87.48*

Brown Test	P	R	F1
He et al. (2017) single	67.6	69.6	68.5
He et al. (2017) PoE	69.7	**70.5**	70.1
SA	70.10	66.01	67.99
LISA_D	70.55	66.56	68.49
+D&M	70.73	66.93	68.78
LISA_G	71.93	69.45	70.67
+D&M	**72.60**	69.73	**71.13**
+Gold	*77.32*	*74.69*	*75.98*

Table 4: Precision, recall and F1 on the CoNLL-2005 WSJ and Brown (out-of-domain) test sets.

	WSJ Test			Brown Test		
	P	R	F1	P	R	F1
He et al.	94.5	**98.5**	96.4	89.3	**95.7**	92.4
LISA_G	**98.3**	98.0	**98.2**	**94.3**	92.7	**93.5**

Table 5: Predicate detection precision, recall and F1 on CoNLL-2005.

He et al. (2017) on both the in-domain and out-of-domain test sets by about 1.0 F1 absolute.

Our CoNLL-2005 predicate detection performance follows similar trends to those of CoNLL-2012 (Table 5): all our models out-perform the baseline in terms of F1. In the case of CoNLL-2005, He et al. (2017) attains higher recall, especially on the Brown test set, while our model achieves higher precision. We report only LISA_G since there is little difference across *SA models.

LISA in its current form does not perform as well when gold predicates are given at test time. Table 6 presents LISA_G performance with predicate indicator embeddings provided on the input. On neither test set does our model using LISA parses out-perform the state-of-the-art. With D&M parses, our models out-perform He et al. (2017), but not Tan et al. (2018).

We attribute this behavior to two factors. First, the models of He et al. (2017) and Tan et al. (2018)

WSJ Test	P	R	F1
He et al. (2017)	83.1	83.0	83.1
Tan et al. (2018)	84.5	85.2	84.8
LISA_G	82.61	82.83	82.72
+D&M	83.88	83.90	83.89

Brown Test	P	R	F1
He et al. (2017)	72.9	71.4	72.1
Tan et al. (2018)	73.5	74.6	74.1
LISA_G	71.36	70.05	70.70
+D&M	73.31	71.80	72.55

Table 6: Precision, recall and F1 on CoNLL-2005 with gold predicates.

are larger than our models.[8]. Our models were designed to predict predicates, and we found the current model size sufficient for good performance in this setting. Second, our model encodes each sequence only once, while the works to which we compare re-encode the sequence anew for each predicate. Since our model predicts its own predicates using a shared sentence encoding, it is impossible to encode sequences in this way. We also do not enforce that the model assign the correct predicate label during decoding, leading to incorrect predicate predictions despite gold predicate inputs. For example, in a challenging sentence which contains two distinct semantic frames with the identical predicate *continued*, our model incorrectly predicts both tokens as predicates in one of the frames. With more careful modeling toward gold predicates, our technique could be improved for this setting. Still, LISA shows impressive performance when gold predicates are not available, as when using SRL in the wild.

4.4 Analysis

In §4.2 and §4.3 we observed that while LISA performs well with state-of-the-art predicted syntax, it still sees a large gain across all datasets of 4-5 F1 points absolute when provided with gold syntax trees. In order to better understand the nature of these improvements, we perform a detailed model analysis based on that of He et al. (2017). All experiments in this section are performed on CoNLL-2005 development data.

[8]Our models have 6 total layers (2 CNN and 4 self-attention) whereas He et al. (2017) use 8 layers of bi-directional LSTM and Tan et al. (2018) use 10 layers of self-attention

	Greedy F1	Viterbi F1	Δ F1
LISA$_G$	79.53	80.64	+1.11
+D&M	80.60	81.34	+0.74
+Gold	*84.77*	*85.56*	*+0.79*

Table 7: Comparison of development F1 scores with and without Viterbi decoding at test time.

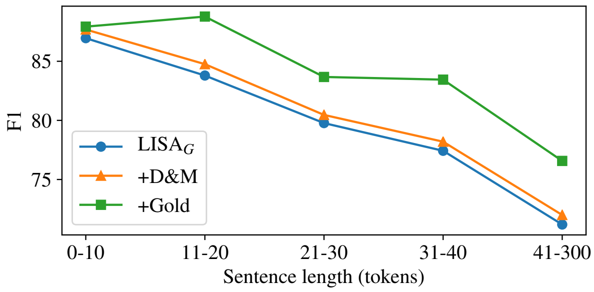

Figure 2: F1 score as a function of sentence length.

First, we compare the impact of Viterbi decoding with LISA, D&M, and gold syntax trees (Table 7), finding the same trends across both datasets. While Viterbi decoding makes a larger difference over greedy decoding with LISA parses than with D&M, we find that Viterbi has the exact same impact for D&M and gold parses: Gold parses provide no improvement over state-of-the-art predicted parses in terms of BIO label consistency.

We also assess SRL F1 as a function of sentence length. In Figure 2 we see that providing LISA with gold parses is particularly helpful for sentences longer than 10 tokens. This likely directly follows from the tendency of syntactic parsers to perform worse on longer sentences.

Next, we compare SRL error types. Following He et al. (2017), we apply a series of corrections to model predictions in order to understand which error types the gold parse resolves: e.g. *Fix Labels* fixes labels on spans which match gold boundaries, whereas *Merge Spans* merges adjacent predicted spans into a gold span.[9]

In Figure 3 we see that much of the performance gap between the gold and predicted parses is due to span boundary errors (*Merge Spans*, *Split Spans* and *Fix Span Boundary*), which supports the hypothesis proposed by He et al. (2017) that incorporating syntax could be particularly helpful for resolving these errors. He et al. (2017) also point out

[9]Refer to He et al. (2017) for a detailed explanation of the different error types.

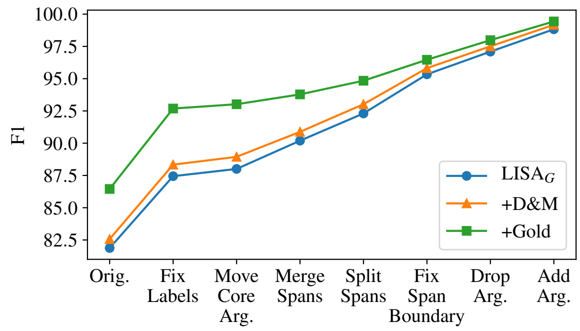

Figure 3: Performance of CoNLL-2005 models after performing corrections from He et al. (2017).

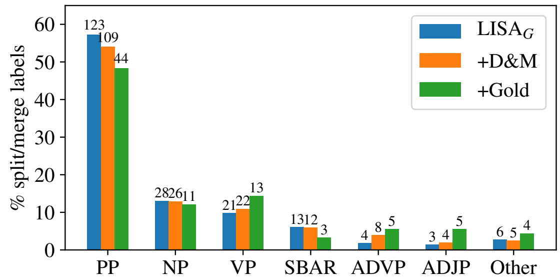

Figure 4: Percent and count of split/merge corrections performed in Figure 3, by phrase type.

that these errors are due mainly to prepositional phrase (PP) attachment mistakes. We also find this to be the case: Figure 4 shows a breakdown of split/merge corrections by phrase type. Though the number of corrections decreases substantially across phrase types, the proportion of corrections attributed to PPs remains the same (approx. 50%) even after providing the correct PP attachment to the model, indicating that PP span boundary mistakes are due not only to parse mistakes, but are a fundamental difficulty for SRL.

5 Conclusion

We present linguistically-informed self-attention: a new multi-task neural network model that effectively incorporates rich linguistic information for semantic role labeling. LISA out-performs the state-of-the-art on two benchmark SRL datasets, including out-of-domain, while training more than 4× faster. Future work will explore improving LISA's parsing accuracy, developing better training techniques and adapting to more tasks.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow.org* .

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *EACL*.

Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR* abs/1607.06450.

Collin F. Baker. 2014. Framenet: A knowledge base for natural language processing. In *Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore (19292014)*. pages 1–5.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 2005–2010.

Marzieh Bazrafshan and Daniel Gildea. 2013. Semantic roles for string to tree machine translation. In *ACL*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2):157–166.

Jonathan Berant, , Vivek Srikumar, Pei-Chun Chen, Brad Huang, Christopher D. Manning, Abby Vander Linden, Brittany Harding, and Peter Clark. 2014. Modeling biological processes for reading comprehension. In *EMNLP*.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*. pages 231–235.

Xavier Carreras and Lluís Marquèz. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL*.

Rich Caruana. 1993. Multitask learning: a knowledge-based source of inductive bias. In *ICML*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *ICML*.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Proc. of ASRU-IEEE*.

Jinho D. Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: short papers*. pages 687–692.

Jinho D. Choi and Martha Palmer. 2012a. Fast and robust part-of-speech tagging using dynamic model selection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. pages 363–367.

Jinho D. Choi and Martha Palmer. 2012b. Guidelines for the clear style constituent to dependency conversion. Technical Report 01-12, Institute of Cognitive Science, University of Colorado Boulder, Boulder, CO, USA.

Jinho D. Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *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(Aug):2493–2537.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28 (NIPS)*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning* 75(3):297–325.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam. In *ICLR Workshop track*.

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

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

W. N. Francis and H. Kučera. 1964. Manual of information to accompany a standard corpus of present-day edited american english, for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012: Technical Papers*. pages 959–976.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 770–778.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6(02):107–116.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. pages 57–60.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. pages 69–78.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations (ICLR)*. San Diego, California, USA.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Thomas N. Kipf and Max Welling. 2017. Semisupervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*. pages 282–289.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG Parsing and Semantic Role Labeling. In *EMNLP*.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics* 6:63–75.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*. volume 30.

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

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn TreeBank. *Computational Linguistics – Special issue on using large corpora: II* 19(2):313–330.

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. volume 27, pages 372–376.

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

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30 th International Conference on Machine Learning*.

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

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

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2006. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. pages 143–152.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL)*.

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.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 1192–1202.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 231–235.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research* 29:105–151.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *CoNLL*.

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

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

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

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

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 173–180.

Gokhan Tur, Dilek Hakkani-Tür, and Ananlada Chotimongkol. 2005. Semi-supervised learning for spoken language understanding using semantic role labeling. In *ASRU*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS)*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *ACL*.

R. J. Williams and D. Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 1–19. http://www.aclweb.org/anthology/K/K17/K17-3001.pdf.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1557–1566. https://doi.org/10.18653/v1/P16-1147.

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)*.

Model	Dev P	Dev R	Dev F1
He et al. (2017) single	74.9	76.2	75.5
He et al. (2017) PoE	76.5	77.8	77.2
SA	78.21	76.93	77.56
LISA$_D$	79.06	76.94	77.99
+D&M	79.17	77.02	78.08
+*Gold*	*79.41*	*77.27*	*78.32*
LISA$_G$	79.36	77.69	78.52
+D&M	**80.84**	**79.08**	**79.95**
+*Gold*	*86.32*	*84.41*	*85.35*

Table 8: Development precision, recall and F1 on CoNLL-2012.

A Supplemental Material

A.1 CoNLL-2012 data and pre-processing

Following previous work (He et al., 2017), we evaluate our models on the CoNLL-2012 data split (Pradhan et al., 2006) of OntoNotes 5.0 (Hovy et al., 2006).[10] This dataset is drawn from seven domains: newswire, web, broadcast news and conversation, magazines, telephone conversations, and text from the bible. The text is annotated with gold part-of-speech, syntactic constituencies, named entities, word sense, speaker, co-reference and semantic role labels based on the PropBank guidelines (Palmer et al., 2005). Propositions may be verbal or nominal, and there are 41 distinct semantic role labels, excluding continuation roles and including the predicate.

We processed the data as follows: We convert the semantic proposition and role segmentations to BIO boundary-encoded tags, resulting in 129 distinct BIO-encoded tags (including continuation roles). We initialize word embeddings with 100d pre-trained GloVe embeddings trained on 6 billion tokens of Wikipedia and Gigaword (Pennington et al., 2014). Following the experimental setup for parsing from Choi et al. (2015), we convert constituency structure to dependencies using the ClearNLP dependency converter (Choi and Palmer, 2012b), use automatic part-of-speech tags assigned by the ClearNLP tagger (Choi and Palmer, 2012a), and exclude single-token sentences in our parsing evaluation.

[10]We constructed the data split following instructions at: http://cemantix.org/data/ontonotes.html

CoNLL-2012	Greedy F1	Viterbi F1	Δ F1
LISA$_G$	77.24	78.52	+1.28
+D&M	78.99	79.95	+0.96
+*Gold*	*84.44*	*85.35*	*+0.91*

Table 9: Comparison of CoNLL-2012 development F1 scores with and without Viterbi decoding at test time.

A.2 CoNLL-2005 data and pre-processing

The CoNLL-2005 data (Carreras and Marquèz, 2005) is based on the original PropBank corpus (Palmer et al., 2005), which labels the Wall Street Journal portion of the Penn TreeBank corpus (PTB) (Marcus et al., 1993) with predicate-argument structures, plus a challenging out-of-domain test set derived from the Brown corpus (Francis and Kučera, 1964). This dataset contains only verbal predicates, though some are multi-word verbs, and 28 distinct role label types. We obtain 105 SRL labels including continuations after encoding predicate argument segment boundaries with BIO tags.

We evaluate the SRL performance of our models using the `srl-eval.pl` script provided by the CoNLL-2005 shared task,[11] which computes segment-level precision, recall and F1 score. We also report the predicate detection scores output by this script.

For CoNLL-2005 we train the same parser as for CoNLL-2012 except on the typical split of the WSJ portion of the PTB using Stanford dependencies (de Marneffe and Manning, 2008) and POS tags from the Stanford CoreNLP `left3words` model (Toutanova et al., 2003). We train on WSJ sections 02-21, use section 22 for development and section 23 for test. This corresponds to the same train/test split used for propositions in the CoNLL-2005 dataset, except that section 24 is used for development rather than section 22.

A.3 Optimization and hyperparameters

We train the model using the Nadam (Dozat, 2016) algorithm for adaptive stochastic gradient descent (SGD), which combines Adam (Kingma and Ba, 2015) SGD with Nesterov momentum (Nesterov, 1983). We additionally vary the learning rate lr as a function of an initial learning rate lr_0 and the current training step $step$ as described in Vaswani

[11]http://www.lsi.upc.es/~srlconll/srl-eval.pl

Model	Dev		
	P	R	F
He et al. (2017) single	80.3	80.4	80.3
He et al. (2017) PoE	**81.8**	**81.2**	**81.5**
SA	79.70	78.59	79.14
LISA_D	79.23	79.21	79.22
+D&M	79.21	79.10	79.16
LISA_G	81.25	80.03	80.64
+D&M	81.71	80.97	81.34
+*Gold*	*86.02*	*85.11*	*85.56*

Table 10: Development precision, recall and F1 on CoNLL-2005.

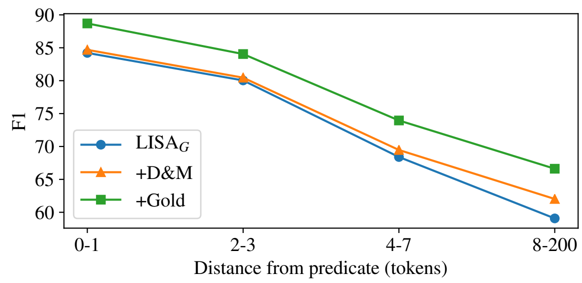

Figure 5: CoNLL-2005 F1 score as a function of the distance of the predicate from the argument span.

et al. (2017) using the following function:

$$lr = lr_0 \cdot \min(step^{-0.5}, step \cdot warm^{-1.5}) \quad (7)$$

which increases the learning rate linearly for the first $warm$ training steps, then decays it proportionally to the inverse square root of the step number. We found this learning rate schedule essential for training the self-attention model. We only update optimization moving-average accumulators for parameters which receive gradient updates at a given step.[12]

In all of our experiments we used initial learning rate 0.04, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1 \times 10^{-12}$ and dropout rates of 0.33 everywhere. We use four self-attention layers made up of 8 attention heads each with embedding dimension 64, and two CNN layers with filter size 1024. The size of all MLP projections: In the feed-forward portion of self-attention, $predicate$ and $role$ representations, and representation used for joint part-of-speech/predicate classification is 256. We train with $warm = 4000$ warmup steps and clip gradient norms to 5.

[12] Also known as *lazy* or *sparse* optimizer updates.