Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling

Diego Marcheggiani **Ivan Titov**
ILLC, University of Amsterdam
{marcheggiani|titov}@uva.nl

Abstract

Semantic role labeling (SRL) is the task of identifying the predicate-argument structure of a sentence. It is typically regarded as an important step in the standard natural language processing pipeline, providing information to downstream tasks such as information extraction and question answering. As the semantic representations are closely related to syntactic ones, we exploit syntactic information in our model. We propose a version of graph convolutional networks (GCNs), a recent class of multilayer neural networks operating on graphs, suited to modeling syntactic dependency graphs. GCNs over syntactic dependency trees are used as sentence encoders, producing latent feature representations of words in a sentence and capturing information relevant to predicting the semantic representations. We observe that GCN layers are complementary to LSTM ones: when we stack both GCN and LSTM layers, we obtain a substantial improvement over an already state-of-the-art LSTM SRL model, resulting in the best reported scores on the standard benchmark (CoNLL-2009) both for Chinese and English.

1 Introduction

Semantic role labeling (SRL) (Gildea and Jurafsky, 2002) can be informally described as the task of discovering *who* did *what* to *whom*. For example, consider an SRL dependency graph shown above the sentence in Figure 1. Formally, the task includes (1) detection of predicates (e.g., *makes*); (2) labeling the predicates with a sense from a sense inventory (e.g., make.01); (3) identifying and assigning arguments to *semantic roles* (e.g., *Sequa* is A0, i.e., an agent / 'doer' for the corresponding predicate, and *engines* is A1, i.e., a patient / 'an affected entity'). SRL is often regarded as an important step in the standard NLP pipeline, providing information to downstream tasks such as information extraction and question answering.

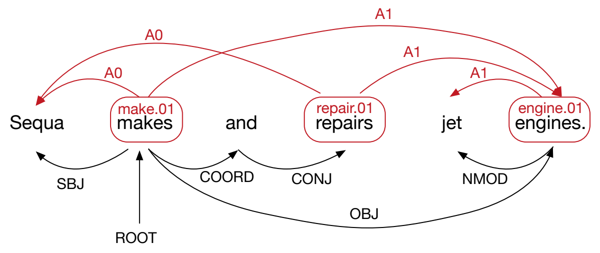

Figure 1: An example sentence annotated with semantic (top) and syntactic dependencies (bottom).

The semantic representations are closely related to syntactic ones, even though the syntax-semantics interface is far from trivial (Levin, 1993). For example, one can observe that many arcs in the syntactic dependency graph (shown in black below the sentence in Figure 1) are mirrored in the semantic dependency graph. Given these similarities and also because of availability of accurate syntactic parsers for many languages, it seems natural to exploit syntactic information when predicting semantics. Though historically most SRL approaches did rely on syntax (Thompson et al., 2003; Pradhan et al., 2005; Punyakanok et al., 2008; Johansson and Nugues, 2008), the last generation of SRL models put syntax aside in favour of neural sequence models, namely LSTMs (Zhou and Xu, 2015; Marcheggiani et al., 2017), and outperformed syntactically-driven methods on standard benchmarks. We believe that one of the reasons for this radical choice is the lack of simple and effective ways for neural networks to encode syntactic information at the level of words. In this work we aim to fill this gap.

Specifically, we rely on graph convolutional networks (GCNs) (Duvenaud et al., 2015; Kipf and Welling, 2016), a recent class of multilayer neural networks operating on graphs. For every node in the graph (in our case a word in a sentence), GCN encodes relevant information about its neighborhood as a real-valued feature vector. GCNs have been studied largely in the context of undirected unlabeled graphs and we are not aware of any previous application of GCNs to NLP. We introduce a generalization of GCNs of Kipf and Welling (2016) applicable to labeled directed graphs, as necessary for modeling syntactic dependency structures.

One layer GCN encodes only information about immediate neighbors and j layers are needed to encode j-order neighborhoods (i.e., information about nodes at most j hops aways). This contrasts with recurrent and recursive neural networks (Elman, 1990; Socher et al., 2013) which, at least in theory, can capture statistical dependencies across unbounded paths in a trees or in a sequence. However, as we will further discuss in Section 3.3, this is not a serous limitation when GCNs are used in combination with encoders based on recurrent networks (LSTMs). When we stack GCNs on top of LSTM layers, we obtain a substantial improvement over an already state-of-the-art LSTM SRL model, resulting in the best reported scores on the standard benchmark (CoNLL-2009), both for English and Chinese.

Interestingly, again unlike recursive neural networks, GCNs do not constrain the graph to be a tree. We believe that there are many applications in NLP, where GCN-based encoders of sentences or even documents can be used to incorporate knowledge about linguistic structures (e.g., representations of syntax, semantics or discourse). For example, GCNs can take as input combined syntactic-semantic graphs (e.g., the entire graph from Figure 1), and be used within downstream tasks such as machine translation or question answering. However, we leave this for future work and here solely focus on SRL.

The contributions of this paper can be summarized as follows:

- we are the first to apply GCNs to NLP (i.e., to encode sentences);
- we propose a generalization of GCNs suited to encoding syntactic information at word level;
- we propose a GCN-based SRL model and obtain state-of-the-art results on English and Chinese portions of the CoNLL-2009 dataset;
- we show that bidirectional LSTMs and syntax-based GCNs have complementary modeling power.

2 Graph Convolutional Networks

In this section we describe GCNs introduced by Kipf and Welling (2016). GCNs are neural networks operating on graphs and inducing features of nodes (i.e., real-valued vectors / embeddings) based on properties of their neighborhoods. In Kipf and Welling (2016), they were shown very effective for the node classification task: the classifier was estimated jointly with a GCN, so that the induced node features were informative for the node classification problem. Depending on how many layers of convolution are used, GCNs can capture information only about immediate neighbors (with one layer of convolution) or any nodes at most k hops aways (if k layers are stacked on top of each other).

More formally, consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ ($|V| = n$) and $\mathcal{E}$ are sets of nodes and edges, respectively. Kipf and Welling (2016) assume that edges contain all the self-loops, i.e., $(v, v) \in \mathcal{E}$ for any v. We can define a matrix $X \in \mathbb{R}^{m \times n}$ with each its column $x_v \in \mathbb{R}^m$ ($v \in \mathcal{V}$) encoding node features. The vectors can either encode genuine features (e.g., this vector can encode the title of a paper if citation graphs are considered) or be a one-hot vector. The node representation, encoding information about its immediate neighbors, is computed as

$$h_v = \rho\left(c_v \sum_{u \in \mathcal{N}(v)} (W x_u + b)\right), \qquad (1)$$

where $W \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$ are a weight matrix and a bias, respectively; $\mathcal{N}(v)$ are neighbors of v; c_v is a normalization factor; ρ is an activation function (e.g., that of linear rectifier units, ReLU). Note that $v \in \mathcal{N}(v)$ (because of self-loops), so the input feature representation of v (i.e. x_v) affects its induced representation h_v. The normalization factor is needed to ensure that representations of all nodes are on the same scale. It is crucial for graphs with very large degree variations (e.g., citation graphs), and c_v can be defined simply as $\frac{1}{|\mathcal{N}(v)|}$.[1]

As in standard convolutional networks (LeCun et al., 2001), by stacking GCN layers one can incorporate higher degree neighborhoods:

$$h_v^{(j+1)} = \rho\left(c_v \sum_{u \in \mathcal{N}(v)} W^{(j)} h_u^{(j)} + b^{(j)}\right) \qquad (2)$$

where j denotes the layer number and $h_v^{(1)} = x_v$.

3 Syntactic GCNs

As GCNs were developed for undirected unlabeled graphs, whereas syntactic dependency trees are directed and labeled (we refer to the dependency labels as *syntactic functions*), we first need to modify the computation in order to incorporate label information (Section 3.1). In the subsequent section, we incorporate gates in GCNs, so that the model can decide which edges are more relevant to the task in question. Having gates is also important as we rely on automatically predicted syntactic representations, and the gates can detect and downweight potentially erroneous edges. We also discuss shortcomings of GCNs and how they are addressed when GCNs are used in tandem with LSTMs (Section 3.3).

3.1 Incorporating directions and labels

Now we introduce a generalization of GCNs appropriate for directed labeled graphs, such as syntactic dependency trees. First note that there is no reason to assume that information flows only along the syntactic dependency arcs (e.g., from *makes* to *Sequa*), so we allow it to flow in the opposite direction as

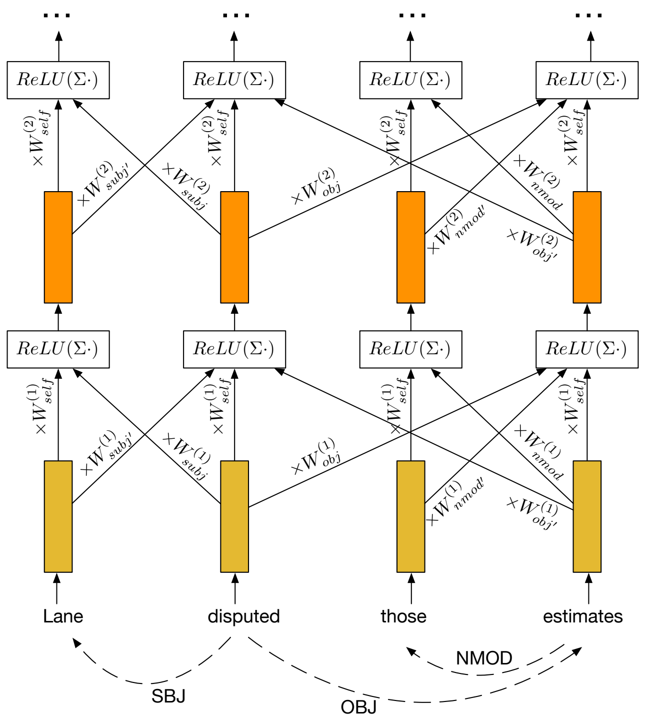

Figure 2: A simplified syntactic GCN (bias terms and gates are omitted); the syntactic graph of the sentence is shown with dashed lines at the bottom. Parameter matrices are sub-indexed with syntactic functions, and apostrophes (e.g., *subj'*) signify that the the information flows in the direction opposite of the dependency arcs (i.e., from dependents to heads).

well (i.e., from dependents to heads[2]). We use a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the edge set contains all pairs of nodes (i.e., words) adjacent in the dependency tree. In our example, both (*Sequa*, *makes*) and (*makes*, *Sequa*) belong to the edge set. The graph is labeled, and the label $L(u, v)$ for $(u, v) \in E$ contains both information about the syntactic function and indicates whether the edge is in the same or opposite direction as the syntactic dependency arc. For example, the label for (*makes*, *Sequa*) is $subj$, whereas the label for (*Sequa*, *makes*) is $subj'$, with the apostrophe indicating that the edge is in the direction opposite to the corresponding syntactic arc. Similarly, self-loops will have label $self$. Consequently, we can simply assume that the GCN pa-

[1]We refer the reader to Kipf and Welling (2016) for details and potential alternatives, including other normalization choices.

[2]For syntactic dependency graphs we use the notation where edges point from syntactic heads to their dependents (see Figure 1), though other notations are used sometimes in literature.

rameters are label-specific, resulting in the following computation, also illustrated in Figure 2:

$$h_v^{(j+1)} = \rho \left(\sum_{u \in \mathcal{N}(v)} W_{L(u,v)}^{(j)} h_u^{(j)} + b_{L(u,v)}^{(j)} \right).$$

Note that we also dropped the normalization because we believe that unlike previous GCN applications, in syntax, degree (e.g., verb valency) carries important information and the normalization makes the model largely agnostic to it. This model is over-parameterized,[3] especially given that SRL datasets are moderately sized, by deep learning standards. So instead of learning the GCN parameters directly, we define them as

$$W_{L(u,v)}^{(j)} = V_{dir(u,v)}^{(j)}, \tag{3}$$

where $dir(u, v)$ indicates whether the edge (u, v) is directed (1) along, (2) in the opposite direction to the syntactic dependency arc, or (3) is a self-loop; $V_{dir(u,v)}^{(j)} \in \mathbb{R}^{m \times m}$. Our simplification captures the intuition that information should be propagated differently along edges depending whether this is a head-to-dependent or dependent-to-head edge (i.e., along or opposite the corresponding syntactic arc) and whether it is a self-loop. So we do not share any parameters between these three very different edge types. Syntactic functions are important, but perhaps less crucial, so they are encoded only in the feature vectors $b_{L(u,v)}$.

3.2 Edge-wise gating

Both uniform normalization, as in Kipf and Welling (2016), and dropping normalization ($c_v = 1$), as we suggested in the preceding section, are problematic. Besides not capturing the degree information properly (as we discussed above), uniform normalization would dump activations of nodes with a high degree, and these nodes (e.g., many of them are verbs, central for SRL) carry important information. Overall, uniformly accepting information from all neighboring nodes may not be appropriate for the SRL setting. For example, we see in Figure 1 that many semantic arcs just mirror their syntactic counter-parts, so they may need to be up-weighted.

[3]Chinese and English CoNLL 2009 datasets used 41 and 48 different syntactic functions, which would result in having 83 and 97 different matrices in every layer, respectively.

Moreover, we rely on automatically predicted syntactic structures, and, even for English, syntactic parsers are far from being perfect, especially when used out-of-domain. It is risky for a downstream application to rely on a potentially wrong syntactic edge, so the corresponding message in the neural network may need to be down-weighted. We would like the model to automatically detect risky cases: for example, preposition attachment disambiguation is a hard problem for a parser, so it may need to learn that if u is a preposition then it should not trust an edge connecting it to its syntactic head.

In order to address the above issues, inspired by recent literature (van den Oord et al., 2016; Dauphin et al., 2016), we calculate for each edge node pair a scalar gate of the form

$$g_{u,v}^{(j)} = \sigma \left(h_u^{(j)} \cdot \hat{v}_{dir(u,v)}^{(j)} + \hat{b}_{L(u,v)}^{(j)} \right), \tag{4}$$

where σ is the logistic sigmoid function, $\hat{v}_{dir(u,v)}^{(j)} \in \mathbb{R}^m$ and $\hat{b}_{L(u,v)}^{(j)} \in \mathbb{R}$ are weights and a bias for the gate. With this additional gating mechanism, the final syntactic GCN computation is formulated as

$$h_v^{(j+1)} = \rho(\sum_{u \in \mathcal{N}(v)} g_{v,u}^{(j)} (V_{dir(u,v)}^{(j)} h_u^{(j)} + b_{L(u,v)}^{(j)})). \tag{5}$$

3.3 Complementarity of GCNs and LSTMs

The inability of GCNs to capture dependencies between nodes far away from each other in the graph, together with no parameter sharing across convolution layers, may seem like a serious problem, especially in the context of SRL: paths between predicates and arguments often include many dependency arcs (Roth and Lapata, 2016). However, when graph convolution is performed on top of LSTM states (i.e., LSTM states serves as input $\mathbf{x}_v = \mathbf{h}_v^{(0)}$ to GCN) rather than static word embeddings, GCN may not need to capture more than a couple of hops.

To elaborate on this, let us speculate what role GCNs would play when used in combinations with LSTMs, given that LSTMs have already been shown very effective for SRL (Zhou and Xu, 2015; Marcheggiani et al., 2017). Though LSTMs are capable of capturing at least some degree of syntax (Linzen et al., 2016) without explicit syntactic

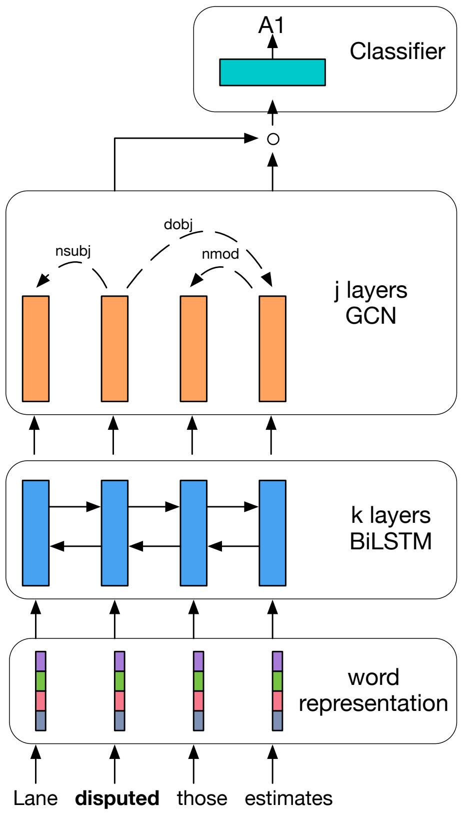

Figure 3: Predicting an argument and its label with an LSTM + GCN encoder.

supervision, SRL datasets are moderately sized, so LSTM models may still struggle with harder cases. Typically, harder cases for SRL involve arguments far away from their predicates. In fact, 20% and 30% of arguments are more than 5 tokens away from their predicate, in our English and Chinese collections, respectively. However, if we imagine that we can 'teleport' even over a single (longest) syntactic dependency edge, the 'distance' would shrink: only 9% and 13% arguments will now be more than 5 LSTM steps away (again for English and Chinese, respectively). GCNs are providing this 'teleportation' capability. These observations suggest that LSTMs and GCNs may be complementary, and we will see that empirical results support this intuition.

4 Syntax-Aware Neural SRL Encoder

In this work, we build our semantic role labeler on top of the syntax-agnostic LSTM-based SRL model of Marcheggiani et al. (2017), which already achieves state-of-the-art results on the CoNLL-2009 English dataset. Following their approach we employ the same bidirectional (BiLSTM) encoder and enrich it with a syntactic GCN.

The CoNLL 2009 benchmark assumes that predicate positions are already marked in the test set (e.g., we would know that *makes*, *repairs* and *engines* in Figure 1 are predicates), so no predicate identification is needed. Also, as we focus here solely on identifying arguments and labeling them with semantic roles, for predicate disambiguation (i.e., marking *makes* as *make.01*) we use of an off-the-shelf disambiguation model (Roth and Lapata, 2016; Björkelund et al., 2009). As in Marcheggiani et al. (2017) and in most previous work, we process individual predicates in isolation, so for each predicate, our tasks reduces to a sequence labeling problem. In other words, given a predicate (e.g., *disputed* in Figure 3) one needs to identify and label all its arguments (e.g., label *estimates* as A1).

The semantic role labeler we propose is composed of four components (see Figure 3):

- a word representation component where the word representation is created (look-ups of embeddings of the word and its features);
- a BiLSTM encoder that takes as input the word representation of each word in a sentence;
- a syntax-based GCN encoder that re-encodes the BiLSTM representation based on the automatically predicted syntactic structure of the sentence;
- a role classifier that take as input the GCN representation of the candidate argument and the representation of the predicate to predict the role associated with the candidate word (including 'NULL' to indicate that a word is not an argument of the predicate).

4.1 Word representations

For each word w_i in the considered sentence, we create a sentence-specific word representation x_i. We represent each word w as the concatenation of four vectors:[4] a randomly initialized word embedding $x^{re} \in \mathbb{R}^{d_w}$, a pre-trained word embedding

[4]We drop the index i from the notation for the sake of brevity.

$x^{pe} \in \mathbb{R}^{d_w}$ estimated on an external text collection, a randomly initialized part-of-speech tag embedding $x^{pos} \in \mathbb{R}^{d_p}$ and a randomly initialized lemma embedding $x^{le} \in \mathbb{R}^{d_l}$ (active only if the word is a predicate). The randomly initialized embeddings x^{re}, x^{pos}, and x^{le} are fine-tuned during training, while the pre-trained ones are kept fixed. The final word representation is given by $x = x^{re} \circ x^{pe} \circ x^{pos} \circ x^{le}$, where $\circ$ represents the concatenation operator.

4.2 Bidirectional LSTM layer

One of the most popular and effective ways to represent sequences, such as sentences (Mikolov et al., 2010), is to use recurrent neural networks (RNN) (Elman, 1990). In particular their gated versions, Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014), have proven effective in modeling long sequences (Chiu and Nichols, 2016; Sutskever et al., 2014).

Formally, an LSTM can be defined as a function $LSTM_\theta(x_{1:i})$ that takes as input the sequence $x_{1:i}$ and returns a hidden state $h_i \in \mathbb{R}^{d_h}$. This state can be regarded as a representation of the sentence from the start to the position i, or, in other words, it encodes the word at position i along with its left context. However, the right context is also important, so Bidirectional LSTMs (Graves, 2008) use two LSTMs: one for the forward pass, and another for the backward pass, $LSTM_F$ and $LSTM_B$, respectively. By concatenating the states of both LSTMs, we create a complete context-aware representation of a word $BiLSTM(x_{1:n}, i) = LSTM_F(x_{1:i}) \circ LSTM_B(x_{n:i})$. We follow Marcheggiani et al. (2017) and stack k layers of bidirectional LSTMs, where each layer takes the lower layer as its input.

4.3 Graph convolutional layer

The representation calculated with the BiLSTM encoder is fed as input to a GCN of the form defined in Equation (5). The neighboring nodes of a node v, namely $\mathcal{N}(v)$, are predicted by an external syntactic dependency parser. GCNs let us incorporate syntactic information in our neural SRL model in a straightforward way, and yield an effective and fast method.

4.4 Semantic role classifier

The classifier predicts semantic roles of words given the predicate while relying on word representations provided by the GCN; we concatenate hidden states of the candidate argument word and the predicate word and use them as input to a classifier (Figure 3, top). The softmax classifier computes the probability of the role (including special 'NULL' role, encoding that the word is not an argument of the predicate):

$$p(r|t_i, t_p, l) \propto \exp(W_{l,r}(t_i \circ t_p)), \tag{6}$$

where t_i and t_p are representations produced by the graph convolutional encoder, l is the lemma of predicate p and the symbol $\propto$ signifies proportionality.[5] As FitzGerald et al. (2015) and Marcheggiani et al. (2017), instead of using a fixed matrix $W_{l,r}$ or simply assuming that $W_{l,r} = W_r$, we jointly embed the role r and predicate lemma l using a non-linear transformation:

$$W_{l,r} = ReLU(U(q_l \circ q_r)), \tag{7}$$

where U is a parameter matrix, whereas $q_l \in \mathbb{R}^{d'_l}$ and $q_r \in \mathbb{R}^{d_r}$ are randomly initialized embeddings of predicate lemmas and roles. In this way each role prediction is predicate-specific, and, at the same time, we expect to learn a good representation for roles associated with infrequent predicates. As our training objective we use the categorical cross-entropy.

5 Experiments

5.1 Datasets and parameters

We tested the proposed SRL model on the English and Chinese CoNLL-2009 dataset with standard splits into training, test and development sets. For English, we used external embeddings of Dyer et al. (2015), learned using the structured skip n-gram approach of Ling et al. (2015). For Chinese we used external embeddings produced with the neural language model of Bengio et al. (2003). As in (Kiperwasser and Goldberg, 2016), we applied word dropout at the word representation level (Iyyer et al., 2015): a word is replaced with a special unknown

[5]We abuse the notation and define by p both the predicate word and its position in the sentence.

token *UNK* with probability $\frac{\alpha}{fr(w)+\alpha}$, where α is an hyper-parameter and $fr(w)$ is the frequency of the word w. The predicted POS tags for both languages were provided by the CoNLL-2009 shared-task organizers. For the predicate disambiguator we used the ones from Roth and Lapata (2016) for English and from Björkelund et al. (2009) for Chinese. We parsed English sentences with the BIST Parser (Kiperwasser and Goldberg, 2016), whereas for Chinese we used automatically predicted parses provided by the CoNLL-2009 shared-task organizers.

We used *edge dropout* in GCN: when computing $h_v^{(j)}$, we ignore each node $v \in \mathcal{N}(v)$ with probability β. Adam (Kingma and Ba, 2015) was used as an optimizer. The hyperparameter tuning and all model selection were performed on the English development set; the chosen values are shown in Table 1 and are the same for English and Chinese experiments (except for differences indicated below).

Semantic role labeler	
d_w (word embeddings EN)	100
d_w (word embeddings CH)	128
d_{pos} (POS embeddings)	16
d_l (lemma embeddings)	100
d_h (LSTM hidden states)	512
d_r (role representation)	128
d'_l (output lemma representation)	128
k (BiLSTM depth)	3
j (GCN depth)	1
α (word dropout)	.25
β (edge dropout)	.3
learning rate	.01

Table 1: Hyperparameter values.

5.2 Results and discussion

In order to show that GCN layers are effective, we first compare our model against its version which lacks GCN layers. Importantly, to measure the genuine contribution of GCNs, we first tuned this syntax-agnostic model (e.g., the number of LSTM layers) to get best possible performance on the development set.[6]

System (English)	P	R	F_1
SRL without syntax	84.3	81.1	82.7
SRL with syntax GCN (j=1)	85.2	81.6	83.3
SRL with syntax GCN (j=2)	84.1	81.4	82.7

Table 2: Semantic role labeling results without predicate disambiguation on the English development set.

System (Chinese)	P	R	F_1
SRL without syntax	78.3	72.3	75.2
SRL with syntax GCN (j=1)	79.9	74.4	77.1
SRL with syntax GCN (j=2)	78.7	74.0	76.2

Table 3: Semantic role labeling results without predicate disambiguation on the Chinese development set.

We compare the syntax-agnostic model with the syntax-aware one, with one layer of graph convolution over syntax ($j = 1$) and with two layers of graph convolution ($j = 2$). As we rely on the same off-the-shelf disambiguator for all versions of the model, in Table 2 and 3 we report SRL-only scores (i.e., predicate disambiguation is not evaluated) on the English and Chinese development sets. On the both datasets, the syntax-aware model with one GCN layers ($j = 1$) performs the best, outperforming the LSTM version by 1.9% and 0.6% for Chinese and English, respectively. Stacking two graph convolutional layers does not give any benefit, we will get back to this point in section 5.3, where we will see why and when having multiple layers is beneficial. Though the reasons for why the improvements on Chinese are much larger are not entirely clear (e.g., both languages are relative fixed word order ones, and the syntactic parses for Chinese are considerably less accurate), this may be attributed to a higher proportion of long-distance dependencies between predicates and arguments in Chinese (see Section 3.3).

In Figure 4, we show the F_1 scores as a function of the distance, in terms of tokens, between a candidate argument and its predicate. As expected, GCNs appears to be more beneficial for long distance dependencies, as shorter ones are already accurately

[6]For example, if we would have used only one layer of LSTMs, gains from using GCNs would be even larger.

System	P	R	F_1
Lei et al. (2015) (local)	-	-	86.6
FitzGerald et al. (2015) (local)	-	-	86.7
Roth and Lapata (2016) (local)	88.1	85.3	86.7
Marcheggiani et al. (2017) (local)	88.6	86.7	87.6
Ours (local)	**89.1**	**86.8**	**88.0**
Björkelund et al. (2010) (global)	88.6	85.2	86.9
FitzGerald et al. (2015) (global)	-	-	87.3
Foland and Martin (2015) (global)	-	-	86.0
Swayamdipta et al. (2016) (global)	-	-	85.0
Roth and Lapata (2016) (global)	90.0	85.5	87.7
FitzGerald et al. (2015) (ensemble)	-	-	87.7
Roth and Lapata (2016) (ensemble)	90.3	85.7	87.9
Ours (ensemble 3x)	**90.5**	**87.7**	**89.1**

Table 4: Results on the test set for English.

System	P	R	F_1
Zhao et al. (2009) (global)	80.4	75.2	77.7
Björkelund et al. (2009) (global)	82.4	75.1	78.6
Roth and Lapata (2016) (global)	83.2	75.9	79.4
Ours (local)	**84.6**	**80.4**	**82.5**

Table 5: Results on the Chinese test set.

captured by the LSTM encoder.

In order to compare to previous work, in Table 4 we report test results on the English in-domain (WSJ) evaluation data. Our model is *local*, as all the argument detection and labeling decisions are conditionally independent: their interaction is captured solely by the LSTM+GCN encoder. This makes our model fast and simple, though, as shown in previous work, *global* modeling of the structured output is beneficial.[7] We leave this extension for future work. Interestingly, we outperform even the best joint model and the best ensemble of joint models, without using joint modeling or ensembles. When we create an ensemble of 3 models with the product-of-expert combination rule, we improve by 1.2% over the best previous result, achieving 89.1% F_1.

When we study the Chinese results (Table 5), we can see that our best model outperforms the state-of-the-art model of Roth and Lapata (2016) by even larger margin of 3.1%.

[7]As seen in Table 4, labelers of FitzGerald et al. (2015) and Roth and Lapata (2016) gained 0.6-1.0% from using global modeling.

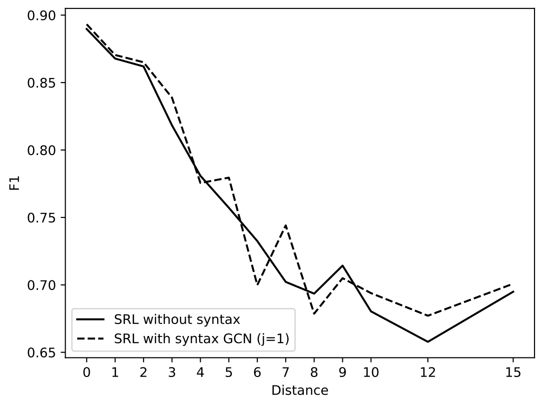

Figure 4: F_1 as function of word distance. The distance starts from zero, since nominal predicates can be arguments of themselves.

For the CoNLL shared task, beside using the standard test, on English systems are also tested out of domain. Statistical models are typically less accurate when they are applied to out-of-domain data, i.e. on data different from the one they have been trained on. The training portion of the English CoNLL data (originating from the WSJ portion of Penn Treebank (Marcus et al., 1993)) consists of newswire, whereas the out-of-domain test set (based on the Brown corpus portion) contains such genres as fiction and humor. Consequently, the predicted syntax for the out-of-domain test set is of much lower quality, which negatively affects the quality of GCN embeddings. However, our model works surprisingly well on out-of-domain data (Table 6), substantially outperforming all the previous syntax-aware models. This suggest that our model is fairly robust to mistakes in syntax. As expected though, our model does not outperform the syntax-agnostic model of Marcheggiani et al. (2017). Nevertheless, the ensemble version obtains achieves a substantial improvement over the best previous approach (+1.6 % F_1).

5.3 Analysis of syntactic GCNs

In this section we conduct an ablation study on the English CoNLL-2009 development set; we discuss the impact of predicted syntax and the behavior of different GCN architectures.

From Table 7, it is clear that the quality of a

System	P	R	F$_1$
Lei et al. (2015) (local)	-	-	75.6
FitzGerald et al. (2015) (local)	-	-	75.2
Roth and Lapata (2016) (local)	76.9	73.8	75.3
Marcheggiani et al. (2017) (local)	78.9	75.7	77.3
Ours (local)	**78.5**	**75.9**	**77.2**
Björkelund et al. (2010) (global)	77.9	73.6	75.7
FitzGerald et al. (2015) (global)	-	-	75.2
Foland and Martin (2015) (global)	-	-	75.9
Roth and Lapata (2016) (global)	78.6	73.8	76.1
FitzGerald et al. (2015) (ensemble)	-	-	75.5
Roth and Lapata (2016) (ensemble)	79.7	73.6	76.5
Ours (ensemble 3x)	**80.8**	**77.1**	**78.9**

Table 6: Results on the out-of-domain test set.

System	P	R	F$_1$
No Syntax k=3 j=0	84.3	81.1	82.7
Pred. Syntax k=3, j=1	85.2	81.6	83.3
Pred. Syntax k=3, j=2	84.1	81.4	82.7
Gold Syntax k=3, j=1	87.7	85.3	86.4
Gold Syntax k=3, j=2	87.3	85.2	86.2
Gold Syntax k=1, j=1	87.0	84.9	85.9
Gold Syntax k=1, j=2	86.7	85.5	86.1
Gold Syntax No-LSTM, j=1	83.8	73.6	78.4
Gold Syntax No-LSTM, j=2	86.6	78.9	82.6
Gold Syntax No-LSTM, j=3	87.3	80.3	83.7
Gold Syntax No-LSTM, j=4	86.0	80.8	83.3

Table 7: Semantic role labeling results with and without gold syntax on the development set.

parser translates into gains in performance from using GCNs. Whereas using predicted syntax resulted in an 0.6% improvement, relying on gold-standard (i.e., 'perfectly correct') syntax led to a further improvement of 3.1%.

Still, when combining LSTMs and GCNs, we do not observe improvements from deep GCNs, unlike the nodel labeled task considered in Kipf and Welling (2016). We hypothesized that this has to do with the expressive power of BiLSTMs. To validate this hypothesis, we first reduce the number of BiLSTM layers to 1 ($k = 1$), and then we even substitute it with a fully connected non-linear layer ($k = 0$). Adding the non-linear layer is necessary, as we need to project the word embeddings up to the dimensionality of the GCN layer.

Figure 7 shows that, when we reduce the number of BiLSTM layers to 1 and a second GCN layer is added, we obtain a small improvement of 0.1%. Interestingly, when BiLSTM layers are dropped altogether, stacking two layers ($j = 2$) of GCNs greatly improves the performance, resulting in a 4.2% jump in F$_1$. Adding a 3rd layer of GCN ($j = 3$) further improves the performance by 1.1% to 83.7% F$_1$. This confirms that, multilayer GCNs are in fact effective, but when they are used in a combination with LSTMs, only one layer is sufficent for SRL.[8]

6 Related Work

Perhaps the earliest methods modeling syntax-semantics interface with RNNs are due to (Henderson et al., 2008; Titov et al., 2009; Gesmundo et al., 2009), they used shift-reduce parsers for joint SRL and syntactic parsing, and relied on RNNs to model statistical dependencies across syntactic and semantic parsing actions. A more modern (e.g., based on LSTMs) and effective reincarnation of this line of research has been proposed in Swayamdipta et al. (2016). Other recent work which considered incorporation of syntactic information in neural SRL models include: FitzGerald et al. (2015) who use standard syntactic features within an MLP calculating potentials of a CRF model; Roth and Lapata (2016) who enriched standard features for SRL with LSTM representations of syntactic paths between arguments and predicates; Lei et al. (2015) who relied on low-rank tensor factorizations for modeling syntax. Also Foland and Martin (2015) used (non-graph) convolutional neural networks and provided syntactic features (e.g., dependency paths) as input. A very different line of research, but with similar goals to ours (i.e. integrating syntax while minimizing the amount of feature engineering), focused on tree kernels and their applications to SRL (Moschitti et al., 2008).

Beyond SRL, there have been many proposals on how to incorporate syntactic information in RNN models, for example, in the context of neural machine translation (Luong et al., 2015; Eriguchi et al., 2017; Sennrich and Haddow, 2016). One of

[8]Note though that GCN layers are computationally cheaper than LSTM ones, even in our non-optimized implementation.

the most popular and attractive approaches is to use tree-structured recursive neural networks (Socher et al., 2013), which, though originally introduced for constituent syntax, can also be applied in the dependency parsing context (Le and Zuidema, 2014; Dyer et al., 2015). An approach of Mou et al. (2015) to sentiment analysis and question classification, introduced even before GCNs became popular in the machine learning community (Duvenaud et al., 2015; Kipf and Welling, 2016), is related to graph convolution. However, it is inherently single-layer and tree-specific, uses bottom-up computations, does not share parameters across syntactic functions and does not use gates.

Previous approaches to integrating syntactic information in neural models are mainly designed to induce representations of sentences or syntactic constituents. In contrast, the approach we presented incorporates syntactic information at word level. This may be attractive from the engineering perspective, as it can be used, as we have shown, instead or along with RNN models.

GCNs or more generally applicable neural networks to graphs is now an active area of research in the machine learning community. Though most previous work have focused on undirected unlabeled graphs, the model of Pham et al. (2017) used directed graphs for the node classification task. Their model is different from ours in many respects. Importantly, they (as well as all previous GCN methods, as far as we are aware) focused on the set-up where training and testing is performed on the same (very large) graph. This contrasts with our setting, where we train and test on different small and automatically predicted graphs.

7 Conclusions and Future Work

We demonstrated how GCNs can be used to incorporate syntactic information in neural models and specifically to construct a syntax-aware SRL model, resulting in state-of-the-art results for Chinese and English. There are relatively straightforward steps which can further improve the SRL results. For example, we relied on labeling arguments independently, whereas using a joint model is likely to significantly improve the performance. Also, in this paper we consider the dependency version of the SRL task, however the model can be generalized to the span-based version (i.e. labeling argument spans with roles rather that syntactic heads of arguments) in a relatively straightforward fashion.

More generally, given simplicity of GCNs and their applicability to general graph structures (not necessarily trees), we believe that there are many NLP tasks, where GCNs can be used to incorporate linguistic structures (e.g., syntactic and semantic representations of sentences and discourse parses or co-reference graphs for documents).

Acknowledgements

We would to thank Anton Frolov, Michael Schlichtkrull, Thomas Kipf, Michael Roth, Max Welling and Wilker Aziz for their suggestions and comments. The project was supported by the European Research Council (ERC StG BroadSem 678254), the Dutch National Science Foundation (NWO VIDI 639.022.518) and an Amazon Web Services (AWS) grant.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of CoNLL Shared Task*.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of COLING: Demonstrations*.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*, 4:357–370.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525*.

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

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. Latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of CoNLL 2009 Shared Task*.

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

Alex Graves. 2008. *Supervised sequence labelling with recurrent neural networks*. Ph.D. thesis, München, Techn. Univ., Diss., 2008.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of CoNLL Shared Task*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, pages 393–400.

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

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. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *EMNLP*.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 2001. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of NAACL*.

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

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *NAACL*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *TACL*, 4:521–535.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114.

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

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

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

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

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of EMNLP*.

Trang Pham, Truyen Tran, Dinh Q. Phung, and Svetha Venkatesh. 2017. Column networks for collective classification. In *Proceedings AAAI*.

Sameer Pradhan, Kadri Hacioglu, Wayne H. Ward, James H. Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of 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 *Proceedings of ACL*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of WMT*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

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

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

Cynthia A. Thompson, Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labeling. In *Proceedings of ECML*.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online projectivisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of IJCAI*.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. 2016. Conditional image generation with pixelcnn decoders. In *NIPS*.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of CoNLL*.

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