Graph Convolutions over Constituent Trees for Syntax-Aware Semantic Role Labeling

Diego Marcheggiani[1]* **Ivan Titov**[2,3]
[1]Amazon
[2]ILCC, School of Informatics, University of Edinburgh
[3]ILLC, University of Amsterdam
marchegg@amazon.es ititov@inf.ed.ac.uk

Abstract

Semantic role labeling (SRL) is the task of identifying predicates and labeling argument spans with semantic roles. Even though most semantic-role formalisms are built upon constituent syntax, and only syntactic constituents can be labeled as arguments (e.g., FrameNet and PropBank), all the recent work on syntax-aware SRL relies on dependency representations of syntax. In contrast, we show how graph convolutional networks (GCNs) can be used to encode constituent structures and inform an SRL system. Nodes in our SpanGCN correspond to constituents. The computation is done in 3 stages. First, initial node representations are produced by 'composing' word representations of the first and last words in the constituent. Second, graph convolutions relying on the constituent tree are performed, yielding syntactically-informed constituent representations. Finally, the constituent representations are 'decomposed' back into word representations, which are used as input to the SRL classifier. We evaluate SpanGCN against alternatives, including a model using GCNs over dependency trees, and show its effectiveness on standard English SRL benchmarks CoNLL-2005, CoNLL-2012, and FrameNet.

1 Introduction

The task of semantic role labeling (SRL) involves predicting the predicate-argument structure of a sentence. More formally, for every predicate, the SRL model must identify all argument spans and label them with their semantic roles (see Figure 1). The most popular resources for estimating SRL models are PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998). In both cases, annotations are made on top of syntactic constituent structures. Earlier work on SRL hinged

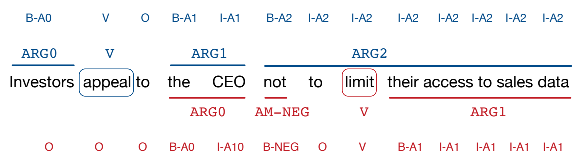

Figure 1: An example with semantic-role annotation and its reduction to the sequence labeling problem (BIO labels): the argument structure for predicates *appeal* and *limit* are shown in blue and red, respectively.

on constituent syntactic structure, using the trees to derive features and constraints on role assignments (Gildea and Jurafsky, 2002; Pradhan et al., 2005; Punyakanok et al., 2008). In contrast, modern SRL systems largely ignore treebank syntax (He et al., 2018a, 2017; Marcheggiani et al., 2017; Zhou and Xu, 2015) and instead use powerful feature extractors, for example, LSTM sentence encoders.

There have been recent successful attempts to improve neural SRL models using syntax (Marcheggiani and Titov, 2017; Strubell et al., 2018; He et al., 2018b). Nevertheless, they have relied on syntactic dependency representations rather than constituent trees. In these methods, information from dependency trees is injected into word representations using graph convolutional networks (GCN) (Kipf and Welling, 2017) or self-attention mechanisms (Vaswani et al., 2017). Since SRL annotations are done on top of syntactic constituents,[1] we argue that exploiting constituency syntax, rather than dependency one, is more natural and may yield more predictive features for semantic roles. For example, even though constituent boundaries could be derived from dependency structures, this would

*Research conducted when the author was at the University of Amsterdam.

[1]There exists another formulation of SRL, where the focus is on predicting semantic dependency graphs (Surdeanu et al., 2008). For English, however, these dependency annotations are automatically derived from span-based PropBank.

require an unbounded number of hops over the dependency structure in GCNs or self-attention. This would be impractical: both Strubell et al. (2018) and Marcheggiani and Titov (2017) use only one hop in their best systems.

Neural models typically treat SRL as a sequence labeling problem, and hence predictions are made for individual words. Though injecting dependency syntax into word representations is relatively straightforward, it is less clear how to incorporate constituency syntax.[2] This work shows how GCNs can be directly applied to span-based structures. We propose a multi-stage architecture based on GCNs to inject constituency syntax into word representations. Nodes in our SpanGCN correspond to constituents. The computation is done in 3 stages. First, initial span representations are produced by 'composing' word representations of the first and last words in the constituent. Second, graph convolutions relying on the constituent tree are performed, yielding syntactically-informed constituent representations. Finally, the constituent representations are 'decomposed' back into word representations, which are used as input to the SRL classifier. This approach directly injects information about boundaries and syntactic labels of constituents into word representations and also provides information about the word's neighbourhood in the constituent structure.

We show the effectiveness of our approach on three English datasets: CoNLL-2005 (Carreras and Màrquez, 2005) and CoNLL-2012 (Pradhan et al., 2012) with PropBank-style (Palmer et al., 2005) annotation and on FrameNet 1.5 (Baker et al., 1998) [3]. By empirically comparing SpanGCN to GCNs over dependency structures, we confirm our intuition that constituents yield more informative features for the SRL task. [4]

SpanGCN may be beneficial in other NLP tasks, where neural sentence encoders are already effective and syntactic structure can provide an additional inductive bias, e.g., logical semantic parsing (Dong and Lapata, 2016) or sentence simplification (Chopra et al., 2016). Moreover, in principle, SpanGCN can be applied to other forms of span-based linguistic representations (e.g., co-reference, entity+relations graphs, semantic and discourse structures). However, we leave this for future work.

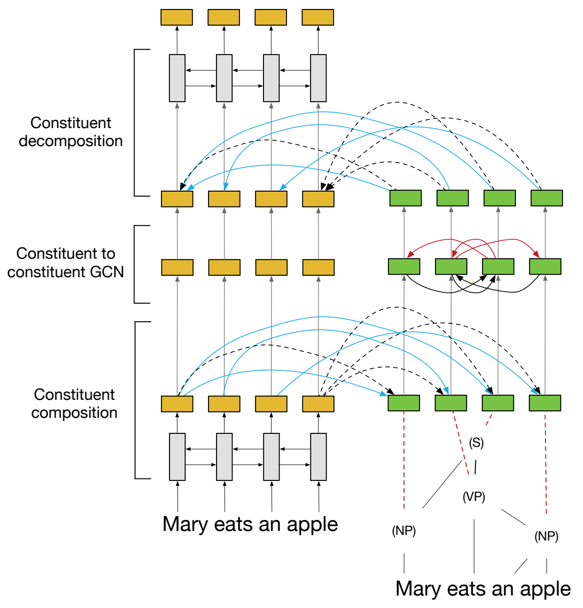

Figure 2: SpanGCN encoder. First, for each constituent, an initial representation is produced by composing the start and end tokens' BiLSTM states (cyan and black dashed arrows, respectively). This is followed by a constituent GCN: red and black arrows represent parent-to-children and children-to-parent messages, respectively. Finally, the constituent is decomposed back: each constituent sends messages to its start and end tokens.

2 Constituency Tree Encoding

The architecture for encoding constituency trees uses two building blocks, a bidirectional LSTM for encoding sequences and a graph convolutional network for encoding graph structures.

2.1 BiLSTM encoder

A bidirectional LSTM (BiLSTM) (Graves, 2013) consists of two LSTMs (Hochreiter and Schmidhuber, 1997), one that encodes the left context of a word and one that encodes the right context. In this paper, we use alternating-stack BiLSTMs as introduced by Zhou and Xu (2015), where the forward LSTM is used as input to the backward LSTM. As in He et al. (2017), we employ highway connections (Srivastava et al., 2015) between layers and recurrent dropout (Gal and Ghahramani, 2016) to avoid overfitting.

[2]Recently, Wang et al. (2019) proposed different ways of encoding dependency and constituency syntax based on the linearization approaches of Gómez-Rodríguez and Vilares (2018).

[3]Although we tested the model on English datasets, SpanGCN can be applied to constituent trees in any language.

[4]Code available at `https://github.com/diegma/span-gcn`.

2.2 GCN

The second building block we use is a graph convolutional network (Kipf and Welling, 2017). GCNs are neural networks that, given a graph, compute the representation of a node conditioned on the neighboring nodes. It can be seen as a message-passing algorithm where a node's representation is updated based on 'messages' sent by its neighboring nodes (Gilmer et al., 2017).

The input to GCN is 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 (2017) assume that the set of edges $\mathcal{E}$ contains also a self-loop, i.e., $(v, v) \in \mathcal{E}$ for any v. We refer to the initial representation of nodes with a matrix $X \in \mathbb{R}^{m \times n}$, with each of its column $x_v \in \mathbb{R}^m$ ($v \in \mathcal{V}$) encoding node features. The new node representation is computed as

$$h_v = ReLU\left(\sum_{u \in \mathcal{N}(v)} (U x_u + b)\right),$$

where $U \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$ are a weight matrix and a bias, respectively; $\mathcal{N}(v)$ are neighbors of v; $ReLU$ is the rectifier linear unit activation function.

The original GCN definition assumes that edges are undirected and unlabeled. We take inspiration from dependency GCNs (Marcheggiani and Titov, 2017) introduced for dependency syntactic structures. Our update function is defined as

$$\begin{aligned} h_v^{'} = &ReLU(LayerNorm(\\ &\sum_{u \in \mathcal{N}(v)} g_{v,u}(U_{T_c(u,v)} h_u + b_{T_f(u,v)}))), \end{aligned} \quad (1)$$

where $LayerNorm$ refers to layer normalization (Ba et al., 2016) applied after summing the messages. Expressions $T_f(u, v)$ and $T_c(u, v)$ are fine-grained and coarse-grained versions of edge labels. For example, $T_c(u, v)$ may simply return the direction of an arc (i.e. whether the message flows along the graph edge or in the opposite direction), whereas the bias can provide some additional syntactic information. The typing decides how many parameters GCN has. It is crucial to keep the number of coarse-grained types low as the model will have to estimate one $\mathbb{R}^{m \times m}$ matrix per coarse-grained type. We formally define the types in the next section. We used scalar gates $g_{u,v}$ to weight the contribution of each node in the neighborhood and potentially ignore irrelevant edges:

$$g_{u,v} = \sigma\left(\hat{u}_{T_c(u,v)} \cdot h_u + \hat{b}_{T_f(u,v)}\right), \quad (2)$$

where σ is the sigmoid activation function, whereas $\hat{u}_{T_c(u,v)} \in \mathbb{R}^m$ and $\hat{b}_{T_f(u,v)} \in \mathbb{R}$ are edge-type-specific parameters.

Now, we show how to compose GCN and LSTM layers to produce a syntactically-informed encoder.

2.3 SpanGCN

Our model is shown in Figure 2. It is composed of three modules: constituent composition, constituent GCN, and constituent decomposition. Note that there is no parameter sharing across these components.

Constituent composition The model takes as input word representations which can either be static word embeddings or contextual word vectors (Peters et al., 2018a; Liu et al., 2019b; Devlin et al., 2019). The sentence is first encoded with a BiLSTM to obtain a context-aware representation of each word. A constituency tree is composed of words ($\mathcal{V}_w$) and constituents ($\mathcal{V}_c$).[5] We add representations (initially zero vectors) for each constituent in the tree; they are shown as green blocks in Figure 2. Each constituent representation is computed using GCN updates (Equation 1), relying on the word representation corresponding to the beginning of its span and the representation corresponding to the end of its span. The coarse-grained types $T_c(u, v)$ here are binary, distinguishing messages from start tokens vs. end tokens. The fine-grained edge types $T_f(u, v)$ encode additionally the constituent label (e.g., NP or VP).

Constituent GCN The constituent composition stage is followed by a layer where constituent nodes exchange messages. This layer makes sure that information about children gets incorporated into representations of immediate parents and vice versa. GCN operates on the graph with nodes corresponding to all constituents ($\mathcal{V}_c$) in the trees. The edges connect constituents and their immediate children in the syntactic tree and do it in both directions. Again, the updates are defined as in Equation 1. As before, $T_c(u, v)$ is binary, now distinguishing parent-to-children messages from children-to-parent messages. $T_f(u, v)$ additionally

[5]We slightly abuse the notation by referring to non-terminals as constituents: part-of-speech tags (typically 'pre-terminals') are stripped off from our trees.

includes the label of the constituent sending the message. For example, consider the computation of the VP constituent in Figure 2. It receives a message from the S constituent, this is a parent-to-child message, and the 'sender' is S; these two factors determine $T_f(u, v)$ and, as a result, the parameters used in computing the corresponding message.

Constituent decomposition At this point, we 'infuse' words with information coming from constituents. The graph here is the inverse of the one used in the composition stage: the constituents pass the information to the first and the last words in their spans. As in the composition stage, $T_c(u, v)$ is binary, distinguishing messages to start and end tokens. The fine-grained edge types, as before, additionally include the constituent label. To spread syntactic information across the sentence, we use a further BiLSTM layer.

Note that residual connections indicated in grey in Figure 2, let the model bypass GCN if / where needed.

3 Semantic Role Labeling

SRL can be cast as a sequence labeling problem where given an input sentence $\mathbf{x}$ of length T, and the position of the predicate in the sentence $p \in T$, the goal is to predict a BIO sequence of semantic roles $\mathbf{y}$ (see Figure 1). We test our model on two different SRL formalisms, PropBank and FrameNet.

PropBank In PropBank conventions, a frame is specific to a predicate sense. For example, for the predicate *make*, it distinguishes 'make.01' ('create') frame from 'make.02' ('cause to be') frame. Though roles are formally frame-specific (e.g., *A0* is the 'creator' for the frame 'make.01' and the 'writer' for the frame 'write.01'), there are certain cross-frame regularities. For example, *A0* and *A1* tend to correspond to proto-agents and proto-patients, respectively.

FrameNet In FrameNet, every frame has its own set of role labels (frame elements in FrameNet terminology).[6] This makes the problem of predicting role labels harder. Differently from PropBank, lexically distinct predicates (lexical units or targets in FrameNet terms) may evoke the same frame. For example, *need* and *require* both can trigger frame 'Needing'.

[6]Cross-frame relations (e.g., the frame hierarchy) present in FrameNet can, in principle, be used to establish correspondences between a subset of roles.

As in previous work we compare to, we assume to have access to gold frames (Swayamdipta et al., 2018; Yang and Mitchell, 2017).

4 Semantic Role Labeling Model

For both PropBank and FrameNet, we use the same model architecture.

Word representation We represent words with pretrained word embeddings, and we keep them fixed during training. Word embeddings are concatenated with 100-dimensional embeddings of a predicate binary feature (indicating if the word is the target predicate or not). Before concatenation, the pretrained embeddings are passed through layer normalization (Ba et al., 2016) and dropout (Srivastava et al., 2014). Formally,

$$x_t = dropout(LayerNorm(w_t)) \circ predemb(t)),$$

where $predemb(t)$ is a function that returns the embedding for the presence or absence of the predicate at position t. The obtained embedding x_t is then fed to the sentence encoder.

Sentence encoder As a sentence encoder we use SpanGCN introduced in Section 2. SpanGCN is fed with word representations x_t. Its output is a sequence of hidden vectors that encode syntactic information for each candidate argument h_t. As a baseline, we use a syntax-agnostic sentence encoder that is the reimplementation of the encoder of He et al. (2017) with stacked alternating LSTMs, i.e., our model with the three GCN layers stripped off.[7]

Bilinear scorer Following Strubell et al. (2018), we used a bilinear scorer:

$$s_{pt} = (h_p^{pred})^T U (h_t^{arg}).$$

h_p^{pred} and h_t^{role} are a non-linear projection of the predicate h_p at position p in the sentence and the candidate argument h_t. The scores s_{pt} are passed through the softmax function and fed to the conditional random field (CRF) layer.

Conditional random field For the output layer, we use a first-order Markov CRF (Lafferty et al., 2001). We use the Viterbi algorithm to predict the most likely label assignment at testing time. At training time, we learn the scores for transitions

[7]To have a fair baseline, we independently tuned the number of BiLSTM layers for our model and the baseline.

	Dev		
	P	R	F1
Baseline	82.78	83.58	83.18
SpanGCN	84.48	84.26	84.37
(w/o BiLSTM)	83.31	83.35	83.33
SpanGCN (Gold)	90.50	90.65	90.58
(w/o BiLSTM)	88.96	90.02	89.49
DepGCN	83.4	83.73	83.56
(w/o BiLSTM)	83.01	83.18	83.09

Table 1: Results with predicted and gold syntax on the CoNLL-2005 development set.

between BIO labels. The entire model is trained to minimize the negative conditional log-likelihood:

$$\mathcal{L} = -\sum_{j}^{N} \log P(\mathbf{y}|\mathbf{x}, p)$$

where p is the predicate position for the training example j.

5 Experiments

5.1 Data and setting

We experiment on the CoNLL-2005 and CoNLL-2012 (OntoNotes) datasets and use the CoNLL 2005 evaluation script for evaluation. We also apply our approach to FrameNet 1.5 with the data split of Das et al. (2014) and follow the official evaluation set-up from the SemEval07 Task 19 on frame-semantic parsing (Baker et al., 2007).

We train the self-attentive constituency parser of Kitaev and Klein (2018)[8] on the training data of the CoNLL-2005 dataset (Penn Treebank) and parse the development and test sets of CoNLL-2005 dataset. We apply the same procedure for the CoNLL-2012 dataset. We perform 10-fold jackknifing to obtain syntactic predictions for the training set of CoNLL-2005 and CoNLL-2012. For FrameNet, we parse the entire corpus with the parser trained on the training set of CoNLL-2005. All hyperparameters are reported in Appendix A.

5.2 Importance of syntax and ablations

Before comparing our full model to state-of-the-art SRL systems, we show that our model genuinely benefits from incorporating syntactic information and motivate other modeling decisions (e.g., the presence of BiLSTM layers at the top).

[8]https://github.com/nikitakit/self-attentive-parser

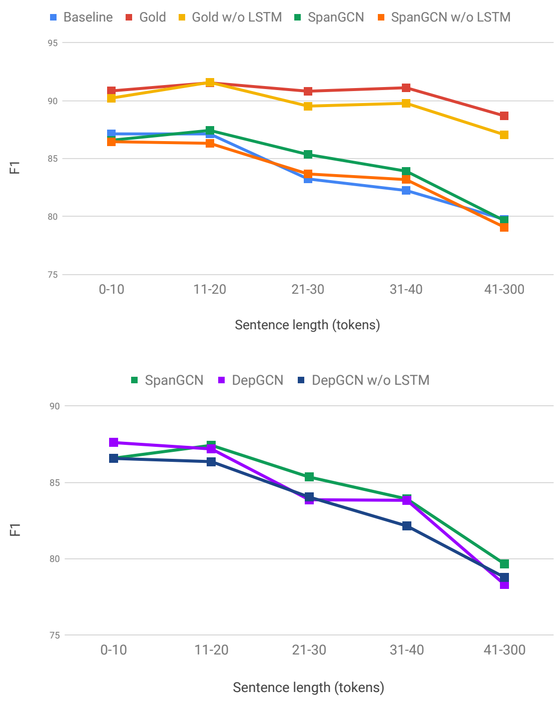

Figure 3: CoNLL-2005 F1 score as a function of sentence length.

We perform this analysis on the CoNLL-2005 dataset. We also experiment with gold-standard syntax, as this provides an upper bound on what SpanGCN can gain from using syntactic information.

From Table 1, we can see that SpanGCN improves over the syntax-agnostic baseline by 1.2% F1, a substantial boost from using predicted syntax. We can also observe that it is important to have the top BiLSTM layer. When we remove the BiLSTM layer, the performance drops by 1% F1. Interestingly, without this last layer, SpanGCN's performance is roughly the same as that of the baseline. This shows the importance of spreading syntactic information from constituent boundaries to the rest of the sentence.

When we provide to SpanGCN gold-standard syntax instead of the predicted one, the SRL scores improve greatly.[9] This suggests that, despite its simplicity (e.g., somewhat impoverished parameterization of constituent GCNs), SpanGCN is capable of extracting predictive features from syntactic structures.

We also measure the performance of the models above as a function of sentence length (Figure 3),

[9]The syntactic parser we use scores 92.5% F1 on the development set.

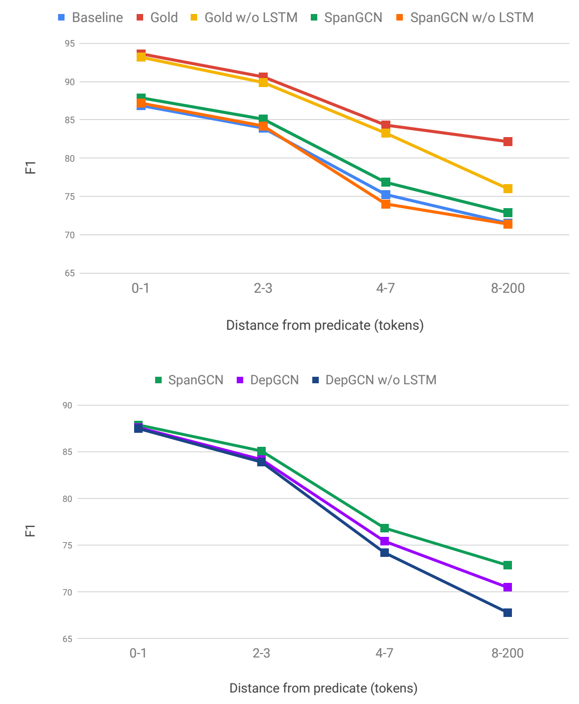

Figure 4: CoNLL-2005 F1 score as a function of the distance of a predicate from its arguments.

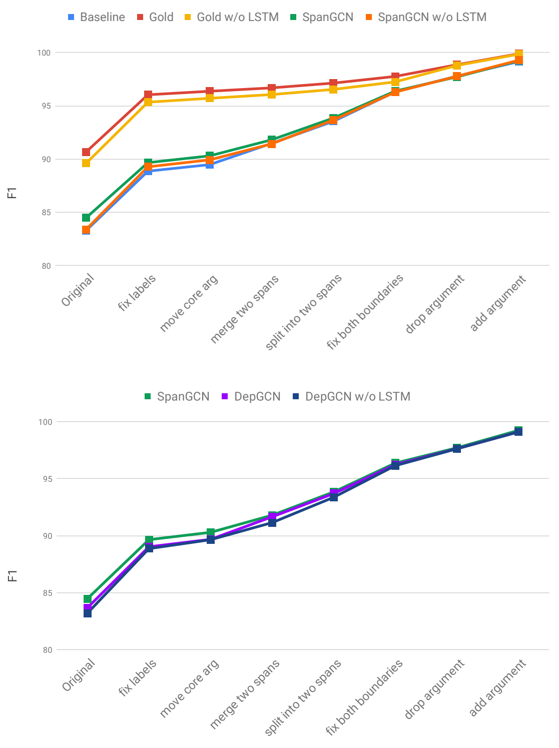

Figure 5: Performance of CoNLL-2005 models after performing corrections from He et al. (2017).

and as a function of the distance between a predicate and its arguments (Figure 4). Not surprisingly, the performance of every model degrades with the length. For the model using gold syntax, the difference between F1 scores on short sentences and long sentences is smaller (2.2% F1) than for the models using predicted syntax (6.9% F1). This is also expected as in the gold-syntax set-up, SpanGCN can rely on perfect syntactic parses even for long sentences. In contrast, in the realistic set-up syntactic features start to be unreliable. SpanGCN performs on par with the baseline for very short and very long sentences. Intuitively, for short sentences, BiLSTMs may already encode enough syntactic information, while for longer sentences, the quality of predicted syntax is not good enough to get gains over the BiLSTM baseline. When considering the performance of each model as a function of the distance between a predicate and its arguments, we observe that all models struggle with more 'remote' arguments. Evaluated in this setting, SpanGCN is slightly better than the baseline.

We also check what kind of errors these models make by using an oracle to correct one error type at the time and measuring the influence on the performance (He et al., 2017). Figure 5 (top) shows the results. We can see that all the models make the same fraction of mistakes in labeling arguments, even with gold syntax. It is also clear that using gold syntax and, to a lesser extent, predicted syntax, helps the model to figure out the exact boundaries of argument spans. These results also suggest that using gold-syntax leads to many fewer span-related errors: fixing these errors (merge two spans, spit into two spans, fix both boundaries) yields 6.1% and 1.4% improvements, when using predicted and gold syntax, respectively. The BiLSTM is even weaker here (6.8% increase in F1).

SpanGCN vs. DependencyGCN To show the benefits of using constituency syntax, we compare SpanGCN with the dependency GCN (DepGCN) of Marcheggiani and Titov (2017). We use DepGCN in our architecture in place of the 3-stage SpanGCN. We obtain dependency trees by transforming the predicted constituency trees with CoreNLP (de Marneffe and Manning, 2008). Table 1 shows that while the model with DepGCN preforms +0.38% better than the baseline, it performs worse than SpanGCN 83.56 vs. 84.36 F1. In Figure 3 (bottom), we also compare the performance of the two syntactic encoders as a function of sentence length. Interestingly, DepGCN per-

	WSJ Test		
Single	P	R	F_1
He et al. (2017)	83.1	83.0	83.1
He et al. (2018a)	84.2	83.7	83.9
Tan et al. (2018)	84.5	85.2	84.8
Ouchi et al. (2018)	84.7	82.3	83.5
Strubell et al. (2018)(LISA)†‡	84.7	84.6	84.6
SpanGCN†	85.8	85.1	85.4
Single / Context. Emb.			
He et al. (2018a)(ELMo)	-	-	87.4
Li et al. (2019)(ELMo)	87.9	87.5	87.7
Ouchi et al. (2018)(ELMo)	88.2	87.0	87.6
Wang et al. (2019)(ELMo)†	-	-	88.2
SpanGCN (ELMo)†	87.5	87.9	87.7
SpanGCN (RoBERTa)†	87.7	88.1	87.9

	Brown Test		
Single	P	R	F_1
He et al. (2017)	72.9	71.4	72.1
He et al. (2018a)	74.2	73.1	73.7
Tan et al. (2018)	73.5	74.6	74.1
Ouchi et al. (2018)	76.0	70.4	73.1
Strubell et al. (2018)(LISA)†‡	74.8	74.3	74.6
SpanGCN†	76.2	74.7	75.5
Single / Context. Emb.			
He et al. (2018a)(ELMo)	-	-	80.4
Li et al. (2019)(ELMo)	80.6	80.4	80.5
Ouchi et al. (2018)(ELMo)	79.9	77.5	78.7
Wang et al. (2019)(ELMo)†	-	-	79.3
SpanGCN(ELMo)†	79.4	79.6	79.5
SpanGCN(RoBERTa)†	80.5	80.7	80.6

Table 2: Precision, recall and F_1 on the CoNLL-2005 test sets. † indicates syntactic models and ‡ indicates multi-task learning models.

	Test		
Single	P	R	F_1
He et al. (2017)	81.7	81.6	81.7
He et al. (2018a)	-	-	82.1
Tan et al. (2018)	81.9	83.6	82.7
Ouchi et al. (2018)	84.4	81.7	83.0
Swayamdipta et al. (2018)†‡	85.1	81.2	83.8
SpanGCN†	84.5	84.3	84.4
Single / Context. Emb.			
Peters et al. (2018a)(ELMo)	-	-	84.6
He et al. (2018a)(ELMo)	-	-	85.5
Li et al. (2019)(ELMo)	85.7	86.3	86.0
Ouchi et al. (2018)(ELMo)	87.1	85.3	86.2
Wang et al. (2019)(ELMo)†	-	-	86.4
SpanGCN (ELMo)†	86.3	86.8	86.5
SpanGCN (RoBERTa)†	86.5	87.1	86.8

Table 3: Precision, recall and F_1 on the CoNLL-2012 test set. † indicates syntactic models and ‡ indicates multi-task learning models.

forms slightly better than SpanGCN on short sentences. Figure 4 (bottom) shows that SpanGCN performs on par with DepGCN when arguments are close to the predicate but better for more distant arguments. As with SpanGCN, Figure 3 and 4 (bottom) show that adding a BiLSTM on top of DepGCN helps to capture long range dependencies. In Figure 5(bottom), we show the different behaviour of DepGCN with respect to SpanGCN in terms of prediction mistakes. Unsurprisingly, the main mistake that DepGCN makes is on deciding span boundaries. Fixing span related errors (merge two spans, spit into two spans, fix both boundaries) yields an improvement of 6.6% for DepGCN vs. 6.1% of SpanGCN.

5.3 Comparing to the state of the art

We compare SpanGCN with state-of-the-art models on both CoNLL-2005 and CoNLL-2012.[10]

CoNLL-2005 In Table 2 (Single) we show results on the CoNLL-2005 dataset. We compare the model with approaches that use syntax (Strubell et al., 2018; Wang et al., 2019) and with syntax-agnostic models (He et al., 2018a, 2017; Tan et al., 2018; Ouchi et al., 2018). SpanGCN obtains the best results also outperforming the multi-task self-attention model of Strubell et al. (2018)[11] on the WSJ (in-domain) (85.43 vs. 84.64 F1) and Brown (out-of-domain) (75.45 vs. 74.55 F1) test sets. The performance on the Brown test shows that SpanGCN is robust with nosier syntax.

CoNLL-2012 In Table 3 (Single) we report results on the CoNLL-2012 dataset. SpanGCN obtains 84.4 F1, outperforming all previous models evaluated on this data.

Experiments using contextualized embeddings We also test SpanGCN using contextualized word embeddings. We use ELMo (Peters et al., 2018a) to train the syntactic parser of Kitaev and Klein (2018), and provide ELMo and RoBERTa (Liu et al., 2019b) embeddings as input to our model.

[10]We only consider single, non-ensemble models.

[11]We compare with the LISA model where no ELMo information (Peters et al., 2018a) is used, neither in the syntactic parser nor the SRL components.

Model	P	R	F_1
Yang and Mitchell (2017) (SEQ)	63.4	66.4	64.9
Yang and Mitchell (2017) (ALL)	70.2	60.2	65.5
Swayamdipta et al. (2018)†‡	69.2	69.0	69.1
SpanGCN†	69.8	68.8	69.3

Table 4: Results on FrameNet 1.5 test set using gold frames. † indicates syntactic models and ‡ indicates multi-task learning models.

In Table 2 (Single / Context. Emb.) we show results of models that employ contextualized embeddings on the CoNLL-2005 test set. Both SpanGCN models with contextualized embeddings perform better than the models with GloVe embeddings in both test sets. SpanGCN(RoBERTa) is outperformed by the syntax-aware model of Wang et al. (2019) on the WSJ test but obtains results on par with the state of the art (Li et al., 2019) on the Brown test set.

When we train the syntax-agnostic baseline of Section 4 with RoBERTa embeddings, we obtain 87.0 F1 on the WSJ test set and 79.7 on the Brown test set, 0.9% F1 worse than SpanGCN on both test sets. This suggests that although contextualized word embeddings contain information about syntax (Tenney et al., 2019; Peters et al., 2018b; Hewitt and Manning, 2019), explicitly encoding high-quality syntax is still useful. SpanGCN(ELMo) has comparable results to SpanGCN(RoBERTa) when tested on the WSJ test set, but has a 1.1% difference when tested on the Brown test set. This difference is not surprising; BERT-like embeddings have been shown to perform better than ELMo embeddings in various probing tasks (Liu et al., 2019a). We believe that on top of this, the sheer volume of data used to train RoBERTa (160GB of text) is beneficial in the out-of-domain setting.

We report results with contextualized embeddings on CoNLL-2012 in Table 3 (Single / Context. Emb.). SpanGCN(RoBERTa) obtains the best results. It is interesting to notice, though, that results of the syntax-aware model of Wang et al. (2019) are overall (on both CoNLL 2005 - 2012) similar to SpanGCN(RoBERTa). Also in this setting, SpanGCN(ELMo) obtains similar (although inferior) results to SpanGCN(RoBERTa) 86.5 vs. 86.8 F1. Compared with the best ELMo-based model (Wang et al., 2019), SpanGCN(ELMo) obtains similar (0.1% lower) results.

FrameNet On FrameNet data, we compare SpanGCN with the sequential and sequential-span ensemble models of Yang and Mitchell (2017), and with the multi-task learning model of Swayamdipta et al. (2018). Swayamdipta et al. (2018) use a multi-task learning objective where the syntactic scaffolding model and the SRL model share the same sentence encoder and are trained together on disjoint data. Like our method, this approach injects syntactic information (though dependency rather than constituent syntax) into word representations used by the SRL model. We show results obtained on the FrameNet test set in Table 4. SpanGCN obtains 69.3% F1 score. It performs better than the syntax-agnostic baseline (2.9% F1) and better than the syntax-agnostic ensemble model (ALL) of Yang and Mitchell (2017) (3.8% F1). SpanGCN also slightly outperforms (0.2% F1) the multi-task model of Swayamdipta et al. (2018).

6 Related Work

Among earlier approaches to incorporating syntax into neural networks, Socher et al. (2013); Tai et al. (2015) proposed recursive neural networks that encode constituency trees by recursively creating representations of constituents. There are two important differences between these approaches and ours. First, in our model, the syntactic information in the constituents flows back to word representations. This may be achieved with their inside-outside versions (Le and Zuidema, 2014; Teng and Zhang, 2017). Second, these previous models do a global pass over the tree, whereas GCNs consider only small fragments of the graph. This may make GCNs more robust when using noisy, predicted syntactic structures.

In SRL, dependency syntax has gained a lot of attention. Similarly to this work, Marcheggiani and Titov (2017) encoded dependency structures using GCNs. Strubell et al. (2018) used a multi-task objective to force the self-attention model to predict syntactic edges. Roth and Lapata (2016) encoded dependency paths between predicates and arguments using an LSTM. Li et al. (2018) analysed different ways of encoding syntactic dependencies for dependency-based SRL, while He et al. (2018b) and He et al. (2019) proposed an argument pruning technique which calculates promising candidate arguments. Recently, Wang et al. (2019) used syntax linearizaton approaches of Gómez-Rodríguez and Vilares (2018) and employed this information as a word-level feature in a SRL model. Swayamdipta et al. (2018); Cai and Lapata (2019) used multi-task learning to produce syntactically-informed word representation, with a sentence encoder shared between SRL and an auxiliary syntax-related task.

In earlier work, Naradowsky et al. (2012) used graphical models to encode syntactic structures while Moschitti et al. (2008) applied tree kernels for encoding constituency trees. Many methods cast the problem of SRL as a span classification problem. FitzGerald et al. (2015) used hand-crafted features to represent spans, while He et al. (2018a) and Ouchi et al. (2018) adopted a BiLSTM feature extractor. In principle, SpanGCN can be used as a syntactic feature extractor within this class of models.

7 Conclusions

In this paper, we introduced SpanGCN, a novel neural architecture for encoding constituency syntax at the word level. We applied SpanGCN to SRL, on PropBank and FrameNet. We observed substantial improvements from using constituent syntax on both datasets, and also in the realistic out-of-domain setting. By comparing to dependency GCN, we observed that for SRL constituent structures yield more informative features that the dependency ones. Given that GCNs over dependency and constituency structure have access to very different information, it would be interesting to see in future work if combining two types of representations can lead to further improvements. While we experimented only with constituency syntax, SpanGCN may be able to encode any kind of span structure, for example, co-reference graphs, and can be used to produce linguistically-informed encoders for other NLP tasks rather than only SRL.

Acknowledgments

We thank Luheng He for her helpful suggestions. The project was supported by the European Research Council (ERC StG BroadSem 678254), and the Dutch National Science Foundation (NWO VIDI 639.022.518). We thank NVIDIA for donating the GPUs used for this research.

References

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

Collin F. Baker, Michael Ellsworth, and Katrin Erk. 2007. Semeval-2007 task 19: Frame semantic structure extraction. In *Proceedings of SemEval@ACL*, pages 99–104.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of COLING-ACL*, pages 86–90.

Rui Cai and Mirella Lapata. 2019. Syntax-aware semantic role labeling without parsing. *Trans. Assoc. Comput. Linguistics*, 7:343–356.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 152–164.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 93–98.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*, pages 33–43.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of EMNLP*, pages 960–970.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*, pages 1019–1027.

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

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of ICML*, pages 1263–1272.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of EMNLP*, pages 1314–1324.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of ACL*, pages 364–369.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of ACL*, pages 473–483.

Shexia He, Zuchao Li, and Hai Zhao. 2019. Syntax-aware multilingual semantic role labeling. In *Proceedings of EMNLP-IJCNLP*, pages 5349–5358.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of ACL*, pages 2061–2071.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of NAACL-HLT*, pages 4129–4138.

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

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

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of ACL*, pages 2675–2685.

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 ICML*, pages 282–289.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of EMNLP*, pages 729–739.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of EMNLP*, pages 2401–2411.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *Proceedings of AAAI*, pages 6730–6737.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of NAACL-HLT*, pages 1073–1094.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of CoNLL*, pages 411–420.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*, pages 1506–1515.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation@COLING 2008*, pages 1–8.

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

Jason Naradowsky, Sebastian Riedel, and David A. Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proceedings of EMNLP*, pages 810–820.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of EMNLP*, pages 1630–1642.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of EMNLP*, pages 1499–1509.

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*, pages 217–220.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of EMNLP-CoNLL*, pages 1–40.

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*, pages 1192–1202.

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*, pages 1631–1642.

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.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of NIPS*, pages 2377–2385.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of EMNLP*, pages 5027–5038.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*, pages 159–177.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of EMNLP*, pages 3772–3782.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL-IJCNLP*, pages 1556–1566.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of AAAI*, pages 4929–4936.

Zhiyang Teng and Yue Zhang. 2017. Head-lexicalized bidirectional tree lstms. *TACL*, 5:163–177.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *Proceedings of ICLR*.

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 *Proceedings of NIPS*, pages 6000–6010.

Yufei Wang, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. 2019. How to best use syntax in semantic role labelling. In *Proceedings of ACL*, pages 5338–5343.

Bishan Yang and Tom M. Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of EMNLP*, pages 1247–1256.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL*, pages 1127–1137.

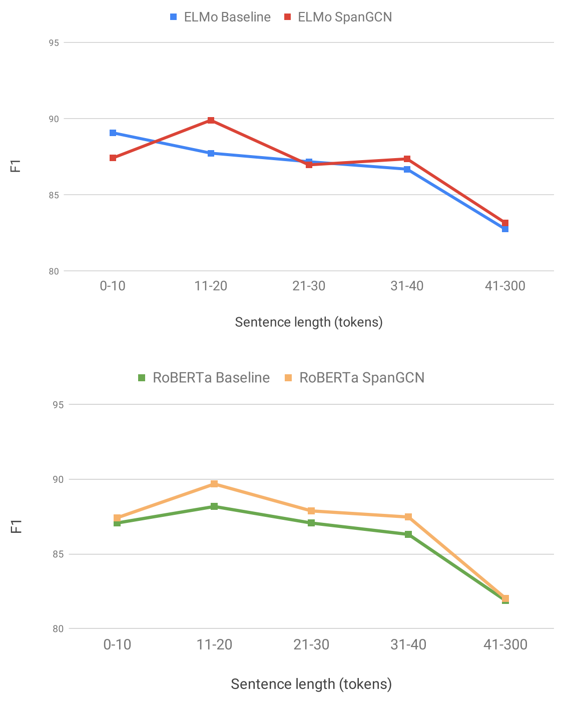

Figure 6: CoNLL-2005 F1 score as a function of sentence length.

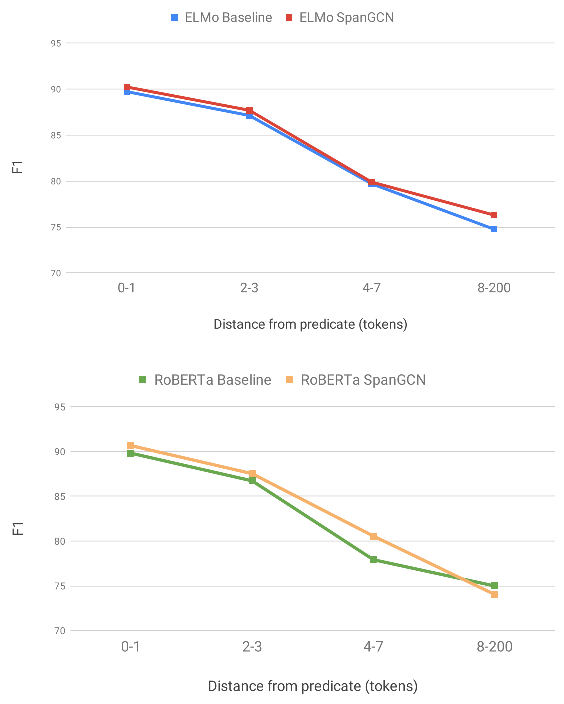

Figure 7: CoNLL-2005 F1 score as a function of the distance of a predicate from its arguments.

A Implementation details

We used 100-dimensional GloVe embeddings (Pennington et al., 2014) for all our experiments unless otherwise specified. We tuned the hyperparameters on the CoNLL-2005 development set. The LSTMs hidden state dimensions were set to 300 for CoNLL experiments and to 200 for FrameNet ones. In our model, we used a four-layer BiLSTM below GCN layers and a two-layer BiLSTM on top. We used an eight-layer BiLSTM in our syntax-agnostic baseline; the number of layers was independently tuned on the CoNLL-2005 development set. For RoBERTa (Liu et al., 2019b) experiments, we used the last layer of the 12-layers (roberta-base) pretrained transformer (Vaswani et al., 2017) without fine tuning it. In the case words got split into multiple subwords by the RoBERTa tokenizer, we took the vector of the first subword unit as the representation of the word as in Devlin et al. (2019).

For ELMo experiments, we learned the mixing coefficients of ELMo, and we concatenated the weighted sum of the ELMo layers with a GloVe 100-dimensional vector. We used the original 5.5B ELMo model [12].

For FrameNet experiments, we constrained the CRF layer to accept only BIO tags compatible with the selected frame.

We used Adam (Kingma and Ba, 2015) as an optimizer with an initial learning rate of 0.001; we halved the learning rate if we did not see an improvement on the development set for two epochs. We trained the model for a maximum of 100 epochs. We clipped the norm of the gradient to 1.

All models were implemented with PyTorch.[13] We used some modules from AllenNLP[14] and the reimplementation of the FrameNet evaluation scripts by Swayamdipta et al. (2018).[15]

B Analysis on Syntax Plus Contextualized Embeddings

We perform an analysis on the use of syntax on top of contextualized representations ELMo and RoBERTa. We perform this analysis on the CoNLL-2005 development set and we measure the impact of contextualized syntax-agnostic vs contextualized syntactic model in function of: sentence length (Figure 6), of the distance of arguments from the predicate (Figure 7), and in function of the type of mistakes they make (Figure 8). In Figures 6,

[12] `https://allennlp.org/elmo`

[13] `https://pytorch.org`

[14] `https://github.com/allenai/allennlp`

[15] `https://github.com/swabhs/scaffolding`

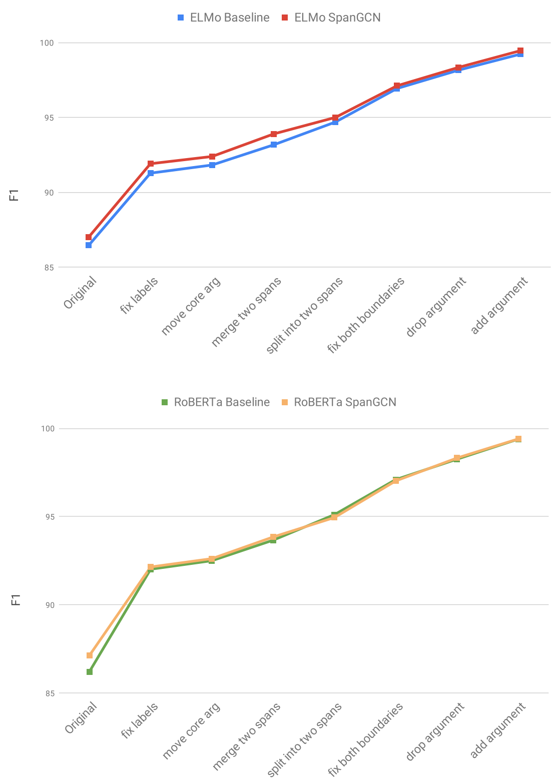

Figure 8: Performance of CoNLL-2005 models after performing corrections from He et al. (2017).

7, and 8, baselines consist of a syntax agnostic 8 layer BiLSTMs on top of the frozen contextualized representation.

Figure 6 shows that for both ELMo and RoBERTa, SpanGCN is beneficial. For ELMo though SpanGCN is not helpful for short sentences (up to length 10), while for RoBERTa, the syntax is beneficial across all sentence lengths.

Figure 7 shows that syntax is beneficial for both contextualized representations. An interesting difference is that for ELMo, syntax is more helpful for arguments very far from the predicate. In contrast, for RoBERTa, syntax is helpful on arguments 4-7 tokens away from the predicate, but hurts performance on arguments farther away from the predicate.

Finally, in Figure 8, we see rather different behaviour between the two representations. For ELMo, the errors that the syntax agnostic model makes are the ones related to span boundaries. For RoBERTa, the syntax-agnostic model makes errors regarding labels, but it is as good as the syntactic model at predicting span boundaries.

C Additional Results

Additional development results for CoNLL-2005 (Table 5) and CoNLL-2012 (Table 6) datasets.

	Dev			WSJ Test			Brown Test		
Single	$\boldsymbol{P}$	$\boldsymbol{R}$	$\boldsymbol{F_1}$	$\boldsymbol{P}$	$\boldsymbol{R}$	$\boldsymbol{F_1}$	$\boldsymbol{P}$	$\boldsymbol{R}$	$\boldsymbol{F_1}$
He et al. (2017)	81.6	81.6	81.6	83.1	83.0	83.1	72.9	71.4	72.1
He et al. (2018a)	-	-	-	84.2	83.7	83.9	74.2	73.1	73.7
Tan et al. (2018)	82.6	83.6	83.1	84.5	85.2	84.8	73.5	74.6	74.1
Ouchi et al. (2018)	83.6	81.4	82.5	84.7	82.3	83.5	76.0	70.4	73.1
Strubell et al. (2018)†‡	83.6	83.74	83.67	84.72	84.57	84.64	74.77	74.32	74.55
DepGCN†	83.4	83.73	83.56	85.07	84.7	84.88	75.5	74.46	74.98
SpanGCN†	84.48	84.26	84.37	85.8	85.05	85.43	76.17	74.74	75.45
Single / Contextualized Embeddings									
He et al. (2018a)(ELMo)	-	-	83.9	-	-	87.4	-	-	80.4
Li et al. (2019)(ELMo)	-	-	-	87.9	87.5	87.7	80.6	80.4	80.5
Ouchi et al. (2018)(ELMo)	87.4	86.3	86.9	88.2	87.0	87.6	79.9	77.5	78.7
Wang et al. (2019)(ELMo)†	-	-	-	-	-	88.2	-	-	79.3
Baseline(ELMo)†	86.07	86.84	86.46	86.81	87.13	86.97	78.43	77.81	78.12
Baseline(RoBERTa)†	85.95	86.3	86.13	86.85	87.19	87.02	79.99	79.33	79.66
SpanGCN(ELMo)†	86.46	87.38	86.92	87.47	87.85	87.66	79.38	79.56	79.47
SpanGCN(RoBERTa)†	86.77	87.56	87.17	87.72	88.05	87.89	80.45	80.71	80.58

Table 5: Precision, recall and $\boldsymbol{F_1}$ on the CoNLL-2005 development and test sets. † indicates syntactic models and ‡ indicates multi-task learning models.

	Dev			Test		
Single	$\boldsymbol{P}$	$\boldsymbol{R}$	$\boldsymbol{F_1}$	$\boldsymbol{P}$	$\boldsymbol{R}$	$\boldsymbol{F_1}$
He et al. (2017)	81.8	81.4	81.5	81.7	81.6	81.7
Tan et al. (2018)	82.2	83.6	82.9	81.9	83.6	82.7
Ouchi et al. (2018)	84.3	81.5	82.9	84.4	81.7	83.0
Swayamdipta et al. (2018)†‡	-	-	-	85.1	81.2	83.8
SpanGCN†	84.45	84.16	84.31	84.47	84.26	84.37
Single / Contextualized Embeddings						
Peters et al. (2018a)(ELMo)	-	-	-	-	-	84.6
Li et al. (2019)(ELMo)	-	-		85.7	86.3	86.0
Ouchi et al. (2018)(ELMo)	87.2	85.5	86.3	87.1	85.3	86.2
Wang et al. (2019)(ELMo)†	-	-	-	-	-	86.4
Baseline(ELMo)†	84.55	83.7	84.13	84.55	83.56	84.06
Baseline(RoBERTa)†	84.6	84.69	84.64	84.71	84.85	84.78
SpanGCN(ELMo)†	86.26	86.74	86.5	86.25	86.83	86.54
SpanGCN(RoBERTa)†	86.69	87.22	86.95	86.48	87.09	86.78

Table 6: Precision, recall and $\boldsymbol{F_1}$ on the CoNLL-2012 development and test sets. † indicates syntactic models and ‡ indicates multi-task learning models.