Semantic Role Labeling with Heterogeneous Syntactic Knowledge

Qingrong Xia[1], **Rui Wang**[2], **Zhenghua Li**[1], **Yue Zhang**[2], **Min Zhang**[1]*
[1]Institute of Artificial Intelligence, School of Computer Science and Technology,
Soochow University, China
[2]Alibaba Group
[1]`kirosummer.nlp@gmail.com`, `{zhli13, minzhang}@suda.edu.cn`
[2]`{masi.wr, shiyu.zy}@alibaba-inc.com`

Abstract

Recently, due to the interplay between syntax and semantics, incorporating syntactic knowledge into neural semantic role labeling (SRL) has achieved much attention. Most of the previous syntax-aware SRL works focus on explicitly modeling homogeneous syntactic knowledge over tree outputs. In this work, we propose to encode *heterogeneous* syntactic knowledge for SRL from both explicit and implicit representations. First, we introduce graph convolutional networks to explicitly encode multiple heterogeneous dependency parse trees. Second, we extract the implicit syntactic representations from syntactic parser trained with heterogeneous treebanks. Finally, we inject the two types of heterogeneous syntax-aware representations into the base SRL model as extra inputs. We conduct experiments on two widely-used benchmark datasets, i.e., Chinese Proposition Bank 1.0 and English CoNLL-2005 dataset. Experimental results show that incorporating heterogeneous syntactic knowledge brings significant improvements over strong baselines. We further conduct detailed analysis to gain insights on the usefulness of heterogeneous (vs. homogeneous) syntactic knowledge and the effectiveness of our proposed approaches for modeling such knowledge.

1 Introduction

Semantic role labeling (SRL) is a fundamental task in natural language processing (NLP), which aims to find the predicate argument structures (*Who* did *what* to *whom*, *when* and *where*, etc.) in a sentence (see Figure 1 as an example). Recent SRL works can mostly be divided into two categories, i.e., syntax-aware (Roth and Lapata, 2016; He et al., 2018b; Strubell et al., 2018) and syntax-agnostic (He et al., 2017; He et al., 2018a) approaches according to whether incorporating syntactic knowledge or not.

Most syntax-agnostic works employ deep BiLSTM or self-attention encoder to encode the contextual information of natural sentences, with various kinds of scorers to predict the probabilities of BIO-based semantic roles (He et al., 2017; Tan et al., 2018) or predicate-argument-role tuples (He et al., 2018a; Li et al., 2019). Motivated by the strong interplay between syntax and semantics, researchers explore various approaches to integrate syntactic knowledge into syntax-agnostic models. Roth and Lapata (2016) propose to use dependency-based embeddings in a neural SRL model for dependency-based SRL. He et al. (2018b) introduce k-order pruning algorithm to prune arguments according to dependency trees. However, previous syntax-aware works mainly employ *singleton/homogeneous* automatic dependency trees, which are generated by a syntactic parser trained on a specific syntactic treebank, like Penn Treebank (PTB) (Marcus et al., 1994).

Our work follows the syntax-aware approach and enhances SRL with heterogeneous syntactic knowledge. We define *heterogeneous* syntactic treebanks as treebanks that follow different annotation guidelines. All is well known, there exist many published dependency treebanks that follow different annotation guidelines, i.e., English PTB (Marcus et al., 1994), Universal Dependencies (UD) (Silveira et al., 2014), Penn Chinese Treebank (PCTB) (Xue et al., 2005), Chinese Dependency Treebank (CDT) (Che et

*Corresponding author.

刚刚	，	杰克	买	了	一把	吉他	。
Just now	,	Jack	**bought**		a	guitar	.
AM-TMP		A0	V		A1		

Figure 1: An example annotated with PropBank annotations, where the upper line words are Chinese.

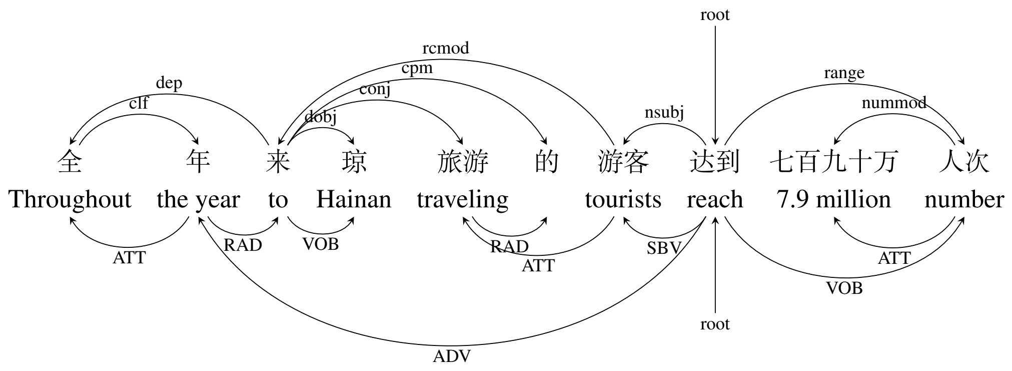

Figure 2: An example of automatic heterogeneous trees, where the top dependency tree is generated from $\texttt{Parser}_{\texttt{PCTB}}$ and the bottom dependency tree is generated from $\texttt{Parser}_{\texttt{CDT}}$.

al., 2012) and so on. These dependency treebanks contain high-quality dependency trees and provide rich syntactic knowledge. Due to different construction purposes, these treebanks have different annotation emphases and data domains. For example, Xue et al. (2005) mainly follow the annotation guideline of PTB to annotate the PCTB treebank on the news data, while Che et al. (2012) use a different annotation guideline with fewer syntactic label on the news and story data. Figure 2 shows an example of automatic heterogeneous trees, where several dependencies are different in the two trees. The word “traveling” is the conjunction modifier of “to” in the PCTB tree, while it is the attribute modifier of “tourists” in the CDT tree. We think both dependencies are grammatically reasonable. Thus, we believe that such heterogeneous syntactic treebanks provide more valid information than each *homogeneous* treebank.

In this work, we propose two types of methods from the perspective of explicit and implicit to take advantage of heterogeneous syntactic knowledge, which we believe are highly complementary. Our baseline model follows the architecture of He et al. (2018a). Afterwards, we inject the heterogeneous syntactic knowledge into the base model using two proposed methods. For the explicit method, we try to encode the heterogeneous automatic dependency trees with the recent popular graph convolutional networks (GCN) (Kipf and Welling, 2016). For the implicit method, which is inspired by the powerful representations from pre-trained language models, like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), we introduce a method to extract implicit syntactic representations from the dependency parser trained with heterogeneous syntactic treebanks. It is well known that the main reason for the success of pre-trained language model representations is the use of large amounts of natural text. However, it is difficult to obtain and costly to annotate large amounts of syntactic data. Therefore, making full use of existing heterogeneous data is the most feasible and natural idea. Intuitively, the explicit method models the syntactic structure of a sentence, providing valuable syntactic position information, while the implicit method aims to capture the syntactic representation for each word into a vector. These two methods contain different types of syntactic knowledge, which are thus highly complementary.

To verify the effectiveness of injecting heterogeneous syntactic knowledge, we conduct experiments on the widely-used Chinese and English SRL benchmarks, i.e., Chinese Proposition Bank 1.0 and English CoNLL-2005. Our contributions are listed as follows:

- To our best knowledge, we are the first to utilize heterogeneous syntactic knowledge to help neural semantic role labeling.
- We introduce two kinds of methods that effectively encode the heterogeneous syntactic knowledge for SRL, and achieve significant improvements over the strong baselines.

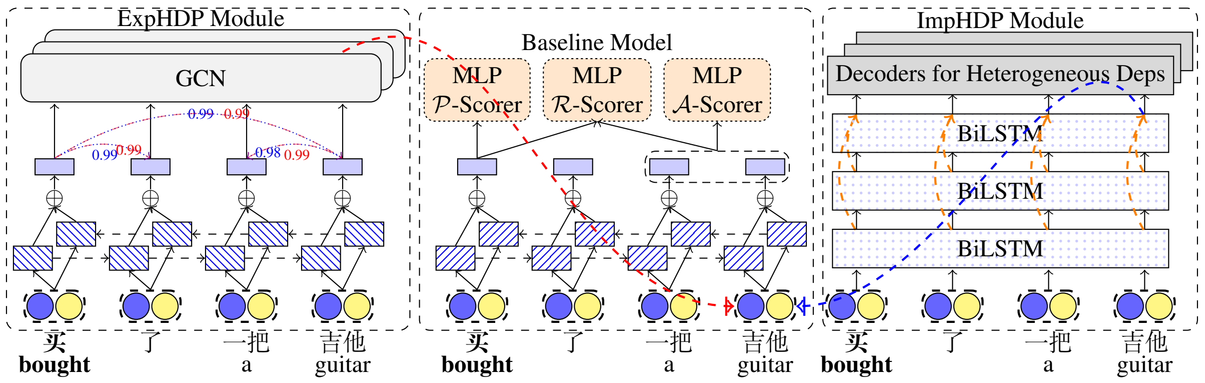

Figure 3: Overview of our model. The middle component is our basic SRL model, the left is the ExpHDP module, and the right is the ImpHDP component. Our model concatenates the ExpHDP representation and ImpHDP representation with the basic SRL input, as shown by the red and blue dashed lines. For clarity, we only show the ExpHDP and ImpHDP representation flow of the word "guitar".

- Detailed analyses clearly show that integrating heterogeneous syntactic knowledge outperforms homogeneous syntactic knowledge and also demonstrate the effectiveness of our methods for encoding heterogeneous syntactic knowledge.

2 Base Model

Following He et al. (2018a), we treat SRL as a predicate-argument-role tuple identification task in our work. Formally, given a sentence $S = w_1, w_2, ..., w_n$, we denote the candidate predicates as $\mathcal{P} = \{w_1, w_2, ..., w_n\}$, the candidate arguments as $\mathcal{A} = \{(w_i, ..., w_j)|1 \leq i \leq j \leq n\}$, and the semantic roles as $\mathcal{R}$. The goal is to predict a set of predicate-argument-role tuples $\mathcal{Y} \in \mathcal{P} \times \mathcal{A} \times \mathcal{R}$.

We basically use the framework of He et al. (2018a) as our baseline model. In general, the model consists of four modules, i.e., input layer, BiLSTMs encoder layer, predicate and argument representation layer, and MLP scorers layer. The middle component of Figure 3 shows the architecture of the baseline model. In the following, we briefly introduce the framework of the baseline model.

Input layer. The input of the i-th word in S is composed of fixed word embedding and fine-tuned char representation. Formally, $\mathbf{x}_i = \mathbf{emb}_{w_i}^{word} \oplus \mathbf{rep}_{w_i}^{char}$, where $\oplus$ is the concatenate operation. The char representation $\mathbf{rep}_{w_i}^{char}$ is generated by CNNs on the characters of the i-th word.

BiLSTMs encoder layer. The baseline model employs three layer BiLSTMs as the encoder layer, which is enhanced by the highway connections. We denote the i-th output of BiLSTMs as $\mathbf{h}_i$.

Predicate and argument representation layer. The model directly treats the output hidden states from the top BiLSTM layer as the representations of candidate predicates. The representation of the k-th word as the candidate predicate is denoted as $\mathbf{r}_k^p = \mathbf{h}_k$. The representation of candidate argument is composed of four parts: 1) the BiLSTM output of the beginning word in the argument, 2) the BiLSTM output of the end word in the argument, 3) an embedding indicating the length of the argument, and 4) a softmax weighted summation of the BiLSTM hidden outputs in the range of candidate argument, where the softmax weights are computed by attention mechanism over words in the argument span. Formally, for a candidate argument from i-th word to j-th word, its representation is defined as $\mathbf{r}_{i,j}^a = \mathbf{h}_i \oplus \mathbf{h}_j \oplus \mathbf{emb}_{j-i+1}^{len} \oplus \mathbf{e}_{i,j}$, where $\mathbf{e}_{i,j}$ is the softmax weighted summation. We use p and a as the abbreviation of predicate and argument.

MLP scorers layer. Three MLP scorers are employed to compute the scores of the candidate predicates, arguments, and the semantic roles between the predicted predicates and arguments, respectively.

$$\begin{aligned} s_p(p) &= \mathbf{W}_p^\top \mathrm{MLP}(\mathbf{r}_p) \\ s_a(a) &= \mathbf{W}_a^\top \mathrm{MLP}(\mathbf{r}_a) \\ s_r(p,a) &= \mathbf{W}_r^\top \mathrm{MLP}(\mathbf{r}_p \oplus \mathbf{r}_a) \end{aligned} \tag{1}$$

The objective is to find the highest-scoring semantic structure, which is computed as:

$$\begin{aligned} P_\theta(Y|S) &= \prod_{p\in\mathcal{P},a\in\mathrm{A},r\in\mathrm{R}} P_\theta(Y_{p,a,r}|S) \\ &= \prod_{p\in\mathcal{P},a\in\mathcal{A},r\in\mathcal{R}} \frac{s(p,a,r)}{\sum_{\hat{r}\in\mathcal{R}} s(p,a,\hat{r})} \end{aligned} \tag{2}$$

where θ represents the model parameters, and $s(p,a,r) = s_p(p) + s_a(a) + s_r(p,a)$ is the score of the candidate predicate-argument-role tuple.

3 Syntactic Knowledge

In this section, we introduce the proposed methods for extracting syntactic knowledge from heterogeneous dependency treebanks. First, we introduce the method for encoding singleton dependency trees and then detailedly describe the variations to extract heterogeneous syntactic knowledge.

3.1 Syntactic Representation

We employ two different methods to fully encode the homogeneous syntactic trees, i.e., GCN that encodes the syntactic structures and implicit representations that encode the syntactic features.

Explicit Method. Graph convolutional networks (GCN) are neural networks that work on graph structures, which have been explored in many NLP tasks (Guo et al., 2019; Zhang et al., 2020). Formally, we denote an undirected graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges, respectively. The GCN computation flow of node $v \in \mathcal{V}$ at l-th layer is defined as

$$\mathbf{h}_v^l = \rho\left(\sum_{u\in\mathcal{N}(v)} \mathbf{W}^l\mathbf{h}_u^{l-1} + \mathbf{b}^l\right), \tag{3}$$

where $\mathbf{W}^l \in \mathrm{R}^{m\times m}$ is the weight matrix, $\mathbf{b}^l \in \mathrm{R}^m$ is the bias term, $\mathcal{N}(v)$ is the set of all one-hop neighbour nodes of v, and ρ is an activation function. Especially, $\mathbf{h}_u^0 \in \mathrm{R}^m$ is the initial input representation, and m is the representation dimension. In our work, we employ a 1-layer BiLSTM encoder over the input layer[1], and treat the BiLSTM outputs as the input of the GCN module, as depicted in the left component in Figure 3. We enhance the basic GCN module with the dense connections (Huang et al., 2017; Guo et al., 2019). The key idea is that the node v of the l-th layer takes input from the concatenation of $\mathbf{h}_u^{l-1}$ and all the representations from previous layers. Formally, the input representation in the l-th layer is defined as $\mathbf{r}_u^l$:

$$\mathbf{r}_u^l = \mathbf{h}_u^0 \oplus \mathbf{h}_u^1 \oplus ... \oplus \mathbf{h}_u^{l-1}. \tag{4}$$

Then, the GCN computation at l-th layer would be modified as:

$$\mathbf{h}_v^l = \rho\left(\sum_{u\in\mathcal{N}(v)} \mathbf{W}^l\mathbf{r}_u^l + \mathbf{b}^l\right), \tag{5}$$

where the weight matrix $\mathbf{W}^l$ increases its column dimension by d_{hidden} per layer, i.e., $\mathbf{W}^l \in \mathrm{R}^{d_{hidden}\times d^l}$ $(d^l = d + d_{hidden} \times (l-1))$.

Implicit Method. Recently popular pre-trained language model embeddings (such as ELMo and BERT) have received much attention. These language models are trained on large amounts of natural text and can produce powerful implicit representations, whose effectiveness is shown in many NLP tasks. Inspired by these pre-trained language model representations and previous works on syntactic representations (Yu et al., 2018; Xia et al., 2019a), we make a trial to train a syntactic parser and extract similar implicit syntactic representations for SRL. We choose the state-of-the-art BiAffine parser (Dozat and Manning, 2017) as our basic dependency parser module. Concisely, BiAffine parser consists of an input layer, BiLSTMs encoder layer, and BiAffine scorers layer, as shown by the right component of Figure 3. We extract the hidden outputs from the 3-layer BiLSTMs encoder of the dependency parser module and make a softmax weighted summation on the outputs as the implicit syntactic representations.

[1]The ExpHDP module shares the same input representations with the basic input of the baseline model.

3.2 Heterogeous Dependency Parsing (HDP)

ExpHDP. In order to encode the automatic heterogeneous syntactic trees, we extend the basic GCN module into the heterogeneous scenarios with two techniques. First, we propose to use the prior probability of two neighbouring nodes as the weight, namely probability-based GCN (P-GCN). Specifically, the probability between node v and u in an automatic tree is the softmax score of node u as v's parent node. Our preliminary experiment shows that this modification would yield a slight improvement of +0.2 F1 score on the CPB1.0 test data. Second, we propose to make a summation of the prior probabilities between each node pair in heterogeneous syntactic trees of a sentence, which we call explicit heterogeneous dependency parsing method (ExpHDP). Intuitively, this approach can combine the different syntactic structures together and enhance the general node pairs, such as verb-subject, verb-object, etc, which would make the graph denser. Finally, the ExpHDP computation of node v in the l-th layer becomes:

$$\mathbf{h}_v^l = \rho\Bigg(\sum_{u\in\mathcal{N}(v)}\Big(\sum_{\mathcal{P}\in\mathcal{HT}s}\mathcal{P}_{uv}\Big)\mathbf{W}^l\mathbf{r}_u^l + \mathbf{b}^l\Bigg), \tag{6}$$

where $\mathcal{HT}s$ is the set of automatic heterogeneous syntactic trees, and $\mathcal{P}_{uv}$ represent the prior probability between node u and v. We treat the outputs of ExpHDP as the explicit representations and concatenate with the input representations of the SRL module to enhance the basic SRL model, as demonstrated by the left two components in Figure 3.

ImpHDP. We can not directly train a dependency parser on heterogeneous syntactic treebanks because of the different annotation guidelines. To solve this problem, we adapt the vanilla BiAffine parser into heterogeneous dependency parser by adding more BiAffine scorer modules according to the number of heterogeneous treebanks, as shown by the rightmost part of Figure 3. Thus, the shared input and BiLSTMs layer can learn more knowledge by training with heterogeneous dependency treebanks. Afterwards, we extract the hidden outputs from the 3-layer BiLSTMs encoder of the dependency parser module, and make a softmax weighted summation on the outputs as the implicit syntactic representations, as depicted by orange dashed lines in the ImpHDP module in Figure 3, which we call ImpHDP (Implicit Heterogeneous Dependency Parsing) method. Formally, the implicit syntactic representation of the i-th word is formulated as $\mathbf{h}_i^{isr} = \sum_{j=1}^{N}\alpha_j\mathbf{h}_{j,i}^{dep}$, where N is the number of BiLSTMs encoder of the dependency parser, α is the softmax weight, and $\mathbf{h}_{j,i}^{dep}$ is the i-th output hidden states of the j-th BiLSTM layer. Considering the relatively small data size of syntactic data, the representation ability of ImpHDP may be not that strong. So we make multi-task learning between SRL and dependency parsing, the work flow is shown by the right two parts of Figure 3. Please note that the losses of SRL and dependency parsing are not accumulated, so our model back-propagates and updates the gradients once a batch of SRL data or dependency data completes the forward process.

3.3 Hybrid HDP

Our model combines the two representations together, according to our intuition that explicit and implicit syntactic representations are highly complementary, which is denoted as "HybridHDP" (Hybrid Heterogeneous Dependency Parsing) in later sections. In detail, we concatenate the two heterogeneous syntactic representations with the SRL input, formulated as $\mathbf{x}_i = \mathbf{emb}_{w_i}^{word} \oplus \mathbf{rep}_{w_i}^{char} \oplus \mathbf{h}_v^l \oplus \mathbf{h}_i^{isr}$.

4 Experiments and Analysis

4.1 Experimental Setup

We conduct experiments on the commonly used Chinese Proposition Bank 1.0 (CPB1.0) (Xue, 2008) and English CoNLL-2005 (Carreras and Màrquez, 2005) benchmarks. We implement our methods and baseline model with Pytorch, and our code, configurations, and models are released in `https://github.com/KiroSummer/HDP-SRL`.

Heterogeneous Dependency Treebanks. We employ PCTB7 and CDT as the heterogeneous dependency treebanks for Chinese, PTB and UD[2] dependency treebanks for English. We employ BiAffine

[2]We use the combination of EWT, GUM, LinES, and ParTUT of UD English corpus in our experiments.

Models	Dev			Test		
	P	R	F1	P	R	F1
Predefined predicates.						
Sha et al. (2016)	-	-	-	-	-	77.69
Xia et al. (2017)	-	-	-	-	-	79.67
Xia et al. (2019a)	-	-	83.39	-	-	83.91
Xia et al. (2019a) (w/ BERT)	-	-	-	-	-	87.54
Baseline	81.62	82.36	81.99	81.94	80.59	81.26
HybridHDP	84.92	85.59	85.25	84.86	84.45	84.65
Baseline (w/ RoBERTa)	87.17	86.65	86.91	87.48	87.11	87.29
HybridHDP (w/ RoBERTa)	86.93	87.02	**86.97**	87.93	87.57	**87.75**
End-to-end setting.						
Xia et al. (2019a)	-	-	82.39	-	-	81.73
Xia et al. (2019a) (w/ BERT)	-	-	85.92	-	-	85.57
Baseline	80.35	80.89	80.62	79.82	78.22	79.01
HybridHDP	83.4	84.23	83.81	83.67	82.89	83.28
Baseline (w/ RoBERTa)	85.47	86.31	85.89	86.30	86.14	86.22
HybridHDP (w/ RoBERTa)	86.73	86.41	**86.57**	87.04	86.23	**86.63**

Table 1: Experimental results and comparison with previous works on CPB1.0 in the predefined predicates and end-to-end settings, respectively.

parser (Dozat and Manning, 2017) to train dependency parsers to generate automatic dependency trees for ExpHDP, which achieve 89.12% and 89.72% UAS on the development data of the Chinese PCTB7 and CDT datasets, 95.73% and 90.14% UAS on the development data of the English PTB and UD datasets, respectively. Besides, we use 5-way jackknifing to obtain automatic dependency trees of the training data of CPB1.0 and CoNLL-2005 from parsers trained with PCTB7 and PTB, respectively.

RoBERTa Representations. We employ pre-trained language model embeddings from RoBERTa (Liu et al., 2019) to boost the performance of both Chinese[3] and English[4] SRL. In detail, we average the fixed subword-level RoBERTa representations as the word-level representations from the last four encoder layers. Then, we employ a softmax weighted operation to sum the four word-level real vectors as the final RoBERTa representations for each word in a sentence. Please note that we only use the RoBERTa representations in the SRL module.

Hyperparameters and Training Criterion. We employ word2vec (Mikolov et al., 2013) to train the Chinese word embeddings with the Chinese Gigaword corpus. The English word embeddings are 300-dimension GloVe vectors (Pennington et al., 2014). We choose Adam (Kingma and Ba, 2015) optimizer with 0.001 as the initial learning rate and decays 0.1% for every 100 steps. Gradients bigger than 1.0 are clipped. All the models are trained for at most 180k steps, and early stop when no further improvement over 30 epochs. We pick the best model according to the performance of the development data to evaluate the test data.

Evaluation. We adopt the official evaluation scripts from CoNLL-2005[5] to evaluate our system outputs. Significant tests are conducted using Dan Bikel's randomized parsing evaluation comparer.

4.2 Results and Analyses on CPB1.0

Results and comparison with previous works on CPB1.0 are shown in Table 1. First, our model, i.e., "HybridHDP" brings absolute improvements of +3.26 and +3.39 F1 scores on the development and test data over our baseline model, and outperforms Xia et al. (2019a) by 1.86 and 0.74 F1 scores, respectively.

[3]https://github.com/brightmart/roberta_zh
[4]https://github.com/pytorch/fairseq/tree/master/examples/roberta
[5]http://www.cs.upc.edu/~srlconll/st05/st05.html

Models	P	R	F1
Baseline	81.62	82.36	81.99
+ ExpHDP (PCTB)	84.59	82.99	83.78
+ ExpHDP (CDT)	83.71	83.15	83.43
+ ExpHDP (Both)	84.44	84.05	84.24

Table 2: Results regarding heterogeneous automatic syntactic trees of ExpHDP.

Models	P	R	F1
Baseline	81.62	82.36	81.99
+ ImpHDP (PCTB)	83.65	83.78	83.71
+ ImpHDP (CDT)	83.73	84.21	83.97
+ ImpHDP (Both)	84.00	84.39	84.19

Table 3: Results regarding heterogeneous syntactic treebanks of ImpHDP.

Models	Development			Test		
	P	**R**	**F1**	**P**	**R**	**F1**
HybridHDP	84.92	85.59	**85.25**	84.86	84.45	**84.65**
HybridHDP - ImHDP	84.44	84.05	84.24	84.19	82.64	83.41
HybridHDP - ExpHDP	84.00	84.39	84.19	84.35	83.90	84.12
Baseline	81.62	82.36	81.99	81.94	80.59	81.26

Table 4: Ablation results of ExpHDP and ImpHDP on CPB1.0 development and test sets.

Second, our approach still achieves significant ($p < 0.03$) improvements by +0.46 F1 score on the test data when our basic SRL module is enhanced with RoBERTa representations, which achieves the new state-of-the-art result of 87.75 F1 score on CPB1.0 test data. We also report the results in the end-to-end setting, i.e., using predicted predicates, in Table 1. Our model achieves the new best result of 86.63 F1 score on CPB1.0 test data.

Ablation study on heterogeneous treebanks. To clearly show the performance gains from singleton syntactic knowledge and heterogeneous syntactic knowledge, we report the results of the two methods when only using single syntactic trees in Table 2 and Table 3. For the ExpHDP method, we can see that using single automatic syntactic trees from $\texttt{Parser}_{\texttt{PCBT}}$ and $\texttt{Parser}_{\texttt{CDT}}$ can both achieve substantial improvements. Combining the two automatic dependency trees with the proposed ExpHDP reaches a higher performance, which demonstrates the effectiveness of our proposed ExpHDP to explicitly encode heterogeneous syntactic knowledge. The results of Table 3 also verify that encoding implicit syntactic knowledge from heterogeneous syntactic treebanks is better than only using singleton treebank.

Ablation study on syntactic representations. Table 4 gives the results of models with ablation on the two key components of our method, which clearly shows the contribution of each module in our model. The model without ImpHDP, which is “baseline + ExpHDP”, drops from 85.25 F1 to 84.24 F1 (-1.01 F1) and 84.65 F1 to 83.41 F1 (-1.24 F1) on the development and test data, respectively. And the model without ExpHDP, i.e., “baseline + ImpHDP”, drops from 85.25 F1 to 84.19 F1 (-1.06 F1) and 84.65 F1 to 84.12 F1 (-0.53 F1) on the development and test data, respectively. From the results, we can conclude that 1) both the two methods can effectively encode syntactic knowledge and 2) *simultaneously integrating the explicit heterogeneous syntactic knowledge and implicit heterogeneous syntactic knowledge performs better than using any single syntactic information*, which proves our intuition that the explicit and implicit methods are highly complementary.

Error breakdrown. In order to understand where syntactic knowledge helps in SRL, we follow the work of He et al. (2017) and Table 4 shows the results after fixing various kinds of prediction errors on the model outputs incrementally. In simple terms, the smoother the curves in the table, the fewer errors the model makes at each mistake item. “Orig.” represents the F1 scores of the original model outputs. From Figure 4, we can see that our two syntax-aware components both effectively outperform our baseline model, especially on the span mistakes, as shown by “Merge Spans” and “Fix Span Boundary” errors, demonstrating that *syntactic knowledge can effectively help the determination of argument boundaries*. To better understand the improvements on the span prediction performance, we report the Binary, Proportional, and Exact F1 scores on the spans, where Binary treats an argument as correct if it

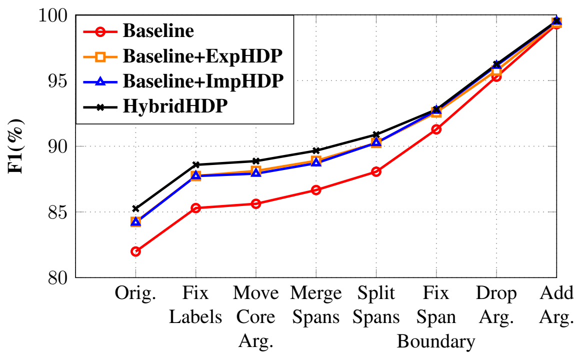

Figure 4: Results of our methods after performing oracle transformations.

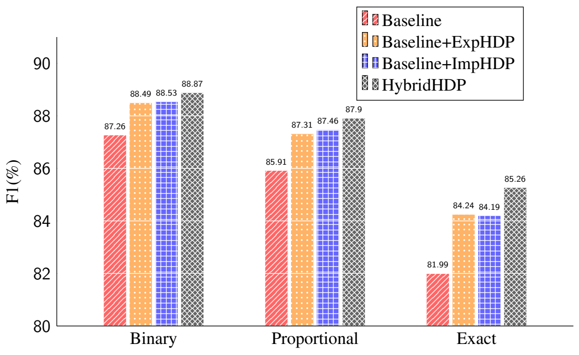

Figure 5: Results of our methods on span performance.

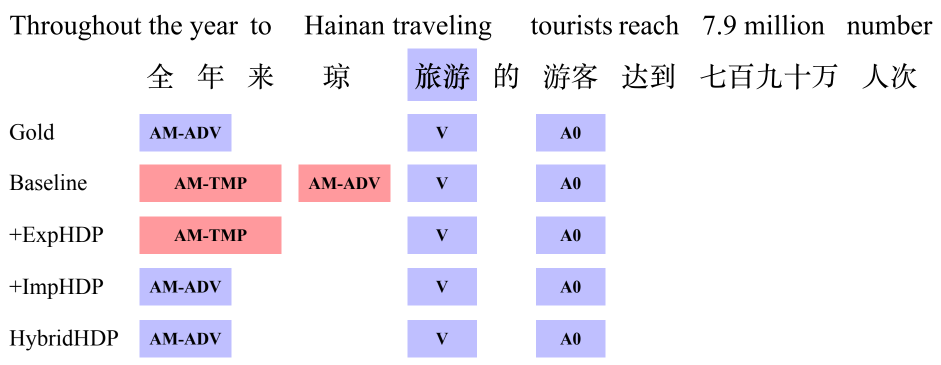

Figure 6: A case study of using heterogeneous syntactic knowledge with the ExpHDP and ImpHDP methods, where the blue block means the gold predicate or argument and red block means the wrongly predicted argument.

overlaps with a gold-standard argument and the Proportional measures the overlapped region between a predicted argument and a gold-standard argument. Table 5 shows the results and we can see that introducing syntactic knowledge can consistently help the determination on the spans of SRL arguments.

Case Study. To better understand the usefulness of the heterogeneous syntactic knowledge, we give a case study of using the ExpHDP and ImpHDP method in Figure 6. We observe that the baseline model can not correctly predict the "AM-ADV" argument and wrongly treat "Hainan" as an "AM-ADV" argument. With the help of ExpHDP, our model successfully excludes the wrongly predicted "AM-ADV" of "Hainan". We think this is because our model learnt the information that "Hainan" and "traveling" are not syntactic-relevant in the trees, especially in the CDT tree, as shown in Figure 2. However, only using the explicit syntactic knowledge can not fix the error of "AM-TMP". Finally, adding the implicit syntactic representation successfully predicts the correct predicate-argument structures.

4.3 Results and Analyses on English SRL

Table 5 shows our results on English CoNLL-2005 development and test data, where WSJ is the in-domain data and Brown is the out-of-domain data. Our implemented baseline model achieves slightly higher performance than the model (He et al., 2018a) we follow. The proposed methods can further improve our baseline model by +0.76 ($p <$ 1e-4) and +1.88 ($p <$ 1e-4) F1 scores on WSJ and Brown test data, respectively. With the help of RoBERTa representations, our full model achieves 88.59 F1 score on the test WSJ data, slightly outperforming Ouchi et al. (2018) by +0.2 F1. In the end-to-end setting of the CoNLL-2005 dataset, our model achieves 86.95 and 80.53 F1 scores in the WSJ and Brown test data, respectively.

Ablation study on heterogeneous treebanks. Even though integrating syntactic knowledge into SRL brings significant improvements to the CPB1.0 dataset, we also find that it is not obvious on the CoNLL-2005 dataset. To know why the improvements on CoNLL-2005 are smaller than on the CPB1.0 dataset,

Models	Dev			WSJ			Brown		
	P	R	F1	P	R	F1	P	R	F1
Predefined predicates.									
He et al. (2018a)	-	-	-	-	-	83.9	-	-	73.7
He et al. (2018a) (w/ ELMo)	-	-	-	-	-	87.4	-	-	80.4
Ouchi et al. (2018) (w/ ELMo)*	88.0	86.9	87.4	89.2	87.9	88.5	81.0	78.4	79.6
Li et al. (2020) (w/ RoBERTa)	87.24	87.26	87.25	88.05	88.00	88.03	80.04	79.56	79.80
Baseline	82.28	82.76	82.52	84.21	84.39	84.30	74.37	73.59	73.98
HybridHDP	83.65	84.06	83.85	85.12	85.0	85.06	76.3	75.42	75.86
Baseline (w/ RoBERTa)	86.87	87.89	**87.38**	88.11	88.64	88.37	82.49	83.51	83.00
HybridHDP (w/ RoBERTa)	86.99	87.41	87.20	88.43	88.75	**88.59**	83.05	83.28	**83.16**
End-to-end setting.									
He et al. (2018a)	-	-	81.6	81.2	83.9	82.5	69.7	71.9	70.8
He et al. (2018a) (w/ ELMo)	-	-	85.3	84.8	87.2	86.0	73.9	78.4	76.1
Baseline	81.53	82.15	81.84	81.81	84.12	82.95	70.35	73.04	71.67
HybridHDP	82.95	82.31	82.63	83.05	84.49	84.49	73.47	74.92	74.19
Baseline(w/ RoBERTa)	85.81	86.76	86.28	85.43	88.46	86.92	78.9	82.77	**80.79**
HybridHDP (w/ RoBERTa)	85.93	86.71	**86.32**	85.66	88.28	**86.95**	78.78	82.36	80.53

Table 5: Experimental results and comparison with previous works on CoNLL-2005 development and test data in the predefined predicates and end-to-end setting, respectively. $\star$ represents model ensemble.

Models	**P**	**R**	**F1**
Baseline	82.28	82.76	82.52
+ ExpHDP (PTB)	83.41	83.39	83.40
+ ExpHDP (UD)	83.18	82.83	83.00

Models	**P**	**R**	**F1**
Baseline	82.28	82.76	82.52
+ ImpHDP (PTB)	83.66	83.64	83.65
+ ImpHDP (UD)	82.99	82.55	82.77

Table 6: Results regarding syntactic treebanks of ExpHDP and ImpHDP on the development data of CoNLL-2005, respectively.

Number	Baseline	HybridHDP	Δ
10,000	79.71	81.66	+1.95
20,000	81.88	83.27	+1.39
30,000	83.16	84.49	+1.33
39,832	84.30	85.06	+0.76

Table 7: Performance gains of syntactic knowledge regarding the different number of SRL training data in the CoNLL-2005 test WSJ data.

we report the results of our methods on the development data when only using singleton dependency trees in Table 6. We can clearly see that compared with the corresponding improvements on the CPB1.0 dataset, the performance gains on the CoNLL-2005 is relatively small, especially when using the UD dependency treebanks. We think this is due to the intention of the construction of UD treebank, which is designed for cross-lingual studies and aims to capture similarities among different languages, thus may be relatively weak in morphosyntax.

Limits of syntactic knowledge. Another possible reason for the relatively lower improvements is the larger number of training samples of the CoNLL-2005 dataset, which can strengthen the basic SRL model and thus weaken the effect of syntactic information. Table 7 shows the results of models with the different number of SRL training samples on the CoNLL-2005 test WSJ data. This indicates that syntactic knowledge works better on those tasks that contain relatively fewer training samples. We also find that the syntactic knowledge nearly brings no performance gains on CoNLL-2005 when the model uses RoBERTa representations. We think it is understandable because the RoBERTa model is trained using very large scale text data and advanced training techniques. So, with the pre-trained language models become more and more powerful, is syntactic knowledge useless anymore for other tasks? It is apparently no because integrating syntactic knowledge into pre-trained language models has attracted some attention (Wang et al., 2020). And unitizing heterogeneous syntactic knowledge would be a direct and natural idea, which we leave for future work.

5 Related Work

Recently, SRL has achieved significant improvements because of the development of deep learning. Previous works can mostly be divided into two kinds of methods, syntax-agnostic methods, which focus on the SRL problem itself, and syntax-aware methods, which explore various ways to integrate syntactic knowledge into the SRL models. Zhou and Xu (2015) propose to use deep BiLSTMs for English span-based SRL. He et al. (2017) further employ several deep learning advanced practices into the stacked BiLSTMs. With the rise of Transformer in machine translation, Tan et al. (2018) employ deep self-attention encoder for SRL, achieving strong performance. Marcheggiani et al. (2017) propose a simple and fast model with rich input representations. Cai et al. (2018) present a full end-to-end model which composed of BiLSTM encoder and BiAffine scorer. He et al. (2018a) first treat SRL as a predicate-argument-role tuple identification task. Following the trend, Li et al. (2019) extend this framework for both span-based and dependency-based SRL, with constraining the argument length to be 1 for dependency-based SRL.

Syntactic knowledge has been explored in various ways to promote the performance of SRL models. Roth and Lapata (2016) propose to integrate the dependency path embeddings into the basic SRL model for dependency-based SRL. Strubell et al. (2018) propose a multi-task learning framework based on the self-attention encoder, which treats dependency parsing as an auxiliary task. He et al. (2019) propose an argument pruning method based on dependency tree positions for multilingual SRL. Recently, Xia et al. (2019a) propose a similar framework to extract syntactic representation for SRL, but they only focus on Chinese SRL. Xia et al. (2019b) compares four explicit methods to encode automatic dependency trees for SRL. Zhang et al. (2019) present different methods to encode dependency trees and compare various incorporation ways into a self-attention based SRL model. These previous works mainly focus on encoding single-sourced dependency treebank, which can only provide limited syntactic knowledge. Our work focus on exploiting heterogeneous dependency benchmarks and the results verify our intuition that heterogeneous syntactic knowledge can provide more valid information.

6 Conclusion

We propose to encode heterogeneous syntactic knowledge with explicit and implicit methods to help SRL. For the explicit aspect, we propose ExpHDP to encode the heterogeneous automatic dependency trees, which can provide more information compared with singleton automatic dependency trees. For the implicit aspect, we extract implicit syntactic representations from the dependency parser trained with heterogeneous dependency treebanks. Experimental results and detailed analyses demonstrate that our proposed methods effectively capture heterogeneous syntactic knowledge, and thus achieve more improvements compared with using singleton dependency trees. We also discuss the limitations of syntactic knowledge, for which we will explore ways to integrate heterogeneous syntactic knowledge into pre-trained language models in the future.

Acknowledgments

We thank our anonymous reviewers for their helpful comments. This work was supported by the National Key Research and Development Program of China (2017YFB1002104), National Natural Science Foundation of China (Grant No. 61876116), a Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, and was partially supported by the joint research project of Alibaba and Soochow University.

References

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of COLING*, pages 2753–2765.

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

Wanxiang Che, Zhenghua Li, and Ting Liu. 2012. Chinese dependency treebank 1.0 ldc2012t05. *Web Download. Philadelphia: Linguistic Data Consortium.*

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.

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

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

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.

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.

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.

Shexia He, Zuchao Li, and Hai Zhao. 2019. Syntax-aware multilingual semantic role labeling. In *Proceedings of EMNLP-IJCNLP*, pages 5353–5362.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*, pages 4700–4708.

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

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

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.

Tao Li, Parth Anand Jawale, Martha Palmer, and Vivek Srikumar. 2020. Structured tuning for semantic role labeling. In *Proceedings of ACL*, pages 8402–8412.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv: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.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on HLT*, pages 114–119. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

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

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. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of ACL*, pages 1192–1202.

Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui, and Tingsong Jiang. 2016. Capturing argument relationship for chinese semantic role labeling. In *Proceedings of EMNLP*, pages 2011–2016.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

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.

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.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Qiaolin Xia, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. A progressive learning approach to chinese srl using heterogeneous data. In *Proceedings of ACL*, pages 2069–2077.

Qingrong Xia, Zhenghua Li, and Min Zhang. 2019a. A syntax-aware multi-task learning framework for chinese semantic role labeling. In *Proceedings of EMNLP-IJCNLP*, pages 5385–5395.

Qingrong Xia, Zhenghua Li, Min Zhang, Zhang Meishan, Guohong Fu, Rui Wang, and Luo Si. 2019b. Syntax-aware neural semantic role labeling. In *Proceedings of AAAI*, pages 7305–7313.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational linguistics*, 34(2):225–255.

Nan Yu, Meishan Zhang, and Guohong Fu. 2018. Transition-based neural rst parsing with implicit syntax features. In *Proceedings of COLING*, pages 559–570.

Yue Zhang, Rui Wang, and Luo Si. 2019. Syntax-enhanced self-attention-based semantic role labeling. In *Proceedings of EMNLP-IJCNLP*, pages 616–626.

Bo Zhang, Yue Zhang, Rui Wang, Zhenghua Li, and Min Zhang. 2020. Syntax-aware opinion role labeling with dependency graph convolutional networks. In *Proceedings of ACL*, pages 3249–3258.

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