Syntax Aware LSTM model for Semantic Role Labeling

Feng Qian[2], **Lei Sha**[1], **Baobao Chang**[1], **Lu-chen Liu**[2], **Ming Zhang**[2]
[1] Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
[2] Institute of Network Computing and Information Systems
School of Electronics Engineering and Computer Science, Peking University
{nickqian, shalei, chbb, liuluchen292, mzhang_cs}@pku.edu.cn

Abstract

In Semantic Role Labeling (SRL) task, the tree structured dependency relation is rich in syntax information, but it is not well handled by existing models. In this paper, we propose Syntax Aware Long Short Time Memory (SA-LSTM). The structure of SA-LSTM changes according to dependency structure of each sentence, so that SA-LSTM can model the whole tree structure of dependency relation in an architecture engineering way. Experiments demonstrate that on Chinese Proposition Bank (CPB) 1.0, SA-LSTM improves F_1 by 2.06% than ordinary bi-LSTM with feature engineered dependency relation information, and gives state-of-the-art F_1 of 79.92%. On English CoNLL 2005 dataset, SA-LSTM brings improvement (2.1%) to bi-LSTM model and also brings slight improvement (0.3%) when added to the state-of-the-art model.

1 Introduction

The task of Semantic Role Labeling (SRL) is to recognize arguments of a given predicate in a sentence and assign semantic role labels. Many NLP works such as machine translation (**?**Aziz et al., 2011) benefit from SRL because of the semantic structure it provides. Figure 1 shows a sentence with semantic role label.

Dependency relation is considered important for SRL task (**?**Punyakanok et al., 2008; Pradhan et al., 2005), since it can provide rich structure and syntax information for SRL. At the bottom of Figure 1 shows dependency of the sentence.

Traditional methods (**??**Ding and Chang, 2008, 2009; **?**) do classification according to manually designed features. Feature engineering requires expertise and is labor intensive. Recent works based on Recurrent Neural Network (RNN) (**??**He et al., 2017) extract features automatically, and significantly outperform traditional methods. However, because RNN methods treat language as sequential data, they fail to integrate the tree structured dependency into RNN.

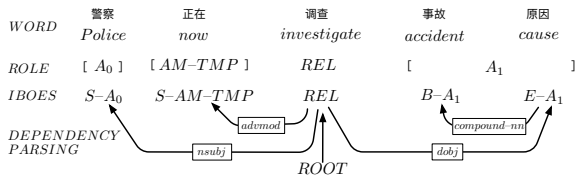

Figure 1: A sentence from Chinese Proposition Bank 1.0 (CPB 1.0) (**?**) with semantic role labels and dependency.

We propose Syntax Aware Long Short Time Memory (SA-LSTM) to directly model complex tree structure of dependency relation in an architecture engineering way. Architecture of SA-LSTM is shown in Figure 2. SA-LSTM is based on bidirectional LSTM (bi-LSTM). In order to model the whole dependency tree, we add additional directed connections between dependency related words in bi-LSTM. SA-LSTM integrates the whole dependency tree directly into the model in an architecture engineering way. Also, to take dependency relation type into account, we introduce trainable weights for different types of dependency relation. The weights can be trained to indicate importance of a dependency type.

SA-LSTM is able to directly model the whole tree structure of dependency relation in an architecture engineering way. Experiments show that SA-LSTM can model dependency relation better than traditional feature engineering way. SA-LSTM gives state of the art F_1 on CPB 1.0 and also shows improvement on English CoNLL 2005 dataset.

2 Syntax Aware LSTM

In this section, we first introduce ordinary bi-LSTM. Based on bi-LSTM, we then introduce the proposed SA-LSTM. Finally, we introduce how to do optimization for SA-LSTM.

2.1 Conventional bi-LSTM Model for SRL

In a corpus sentence, each word w_t has a feature representation x_t which is generated automatically as (**?**) did. z_t is feature embedding for w_t, calculated as followed:

$$z_t = f(W_1 x_t) \tag{1}$$

where $W_1 \in \mathbb{R}^{n_1 \times n_0}$. n_0 is the dimension of word feature representation.

In a corpus sentence, each word w_t has six internal vectors, $\widetilde{C}$, g_i, g_f, g_o, C_t, and h_t, shown in Equation 2:

$$\begin{aligned} \widetilde{C} &= f(W_c z_t + U_c h_{t-1} + b_c) \\ g_j &= \sigma(W_j z_t + U_j h_{t-1} + b_j) \;\; j \in \{i, f, o\} \\ C_t &= g_i \odot \widetilde{C} + g_f \odot C_{t-1} \\ h_t &= g_o \odot f(C_t) \end{aligned} \tag{2}$$

where $\widetilde{C}$ is the candidate value of the current cell state. g are gates used to control the flow of information. C_t is the current cell state. h_t is hidden state of w_t. W_x and U_x are matrixes used in linear transformation:

$$\begin{aligned} W_x, x \in \{c, i, f, o\} \in \mathbb{R}^{n_h \times n_1} \\ U_x, x \in \{c, i, f, o\} \in \mathbb{R}^{n_h \times n_h} \end{aligned} \tag{3}$$

As convention, f stands for $\tanh$ and σ stands for $sigmoid$. $\odot$ means the element-wise multiplication.

In order to make use of bidirectional information, the forward $\overrightarrow{h_t}^T$ and backward $\overleftarrow{h_t}^T$ are concatenated together, as shown in Equation 4:

$$a_t = [\overrightarrow{h_t}^T, \overleftarrow{h_t}^T] \tag{4}$$

Finally, o_t is the result vector with each dimension corresponding to the score of each semantic role tag, and are calculated as shown in Equation 5:

$$o_t = W_3 f(W_2 a_t) \tag{5}$$

where $W_2 \in \mathbb{R}^{n_3 \times n_2}$, n_2 is $2 \times h_t$, $W_3 \in \mathbb{R}^{n_4 \times n_3}$ and n_4 is the number of tags in IOBES tagging schema.

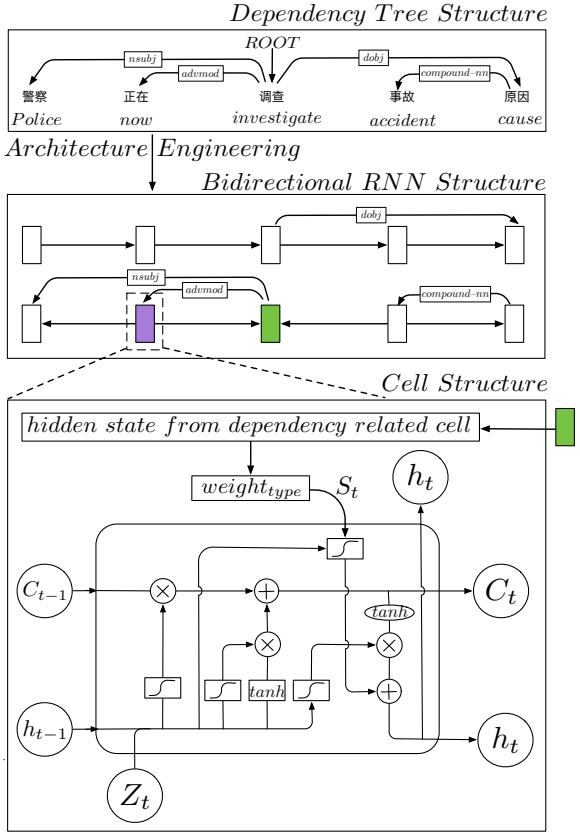

Figure 2: Structure of Syntax Aware LSTM. The purple square is the current cell that is calculating. The green square is a dependency related cell.

2.2 Syntax Aware LSTM Model for SRL

This section introduces the proposed SA-LSTM model. Figure 2 shows the structure of SA-LSTM. SA-LSTM is based on bidirectional LSTM. By architecture engineering, SA-LSTM can model the whole tree structure of dependency relation.

S_t is the key component of SA-LSTM. It stands for information from other dependency related words, and is calculated as shown in Equation 6:

$$S_t = f(\sum_{i=0}^{t-1} \alpha \times h_i) \tag{6}$$

$$\alpha = \begin{cases} 1 & \text{If there exists dependency} \\ & \text{relation from } w_i \text{ to } w_t \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

S_t is the weighted sum of all hidden state vectors h_i which come from previous words w_i . Note that, $\alpha \in \{0, 1\}$ indicates whether there is a dependency relation pointed from w_i to w_t.

We add a gate g_s to constrain information from

S_t, as shown in Equation 8:

$$g_s = \sigma(W_s z_t + U_s h_{t-1} + b_s) \quad (8)$$

To protect the original word information from being diluted (**?**) by S_t, we add S_t to hidden layer vector h_t instead of adding to cell state C_t. So h_t in SA-LSTM cell is calculated as:

$$h_t = g_o \odot f(C_t) + g_s \odot S_t \quad (9)$$

For example, in Figure 2, there is a dependency relation "advmod" from green square to purple square. By Equation 7, only the hidden state of green square is selected, then transformed by g_s in Equation 8, finally added to hidden layer of the purple cell.

SA-LSTM changes structure by adding different connections according to dependency relation. In this way, SA-LSTM integrates the whole tree structure of dependency.

However, by using α in Equation 7, we do not take dependency type into account, so we further improve the way α is calculated from Equation 7 to Equation 10. Each $type_m$ of dependency relation is assigned a trainable weight α_m. In this way, SA-LSTM can model differences between types of dependency relation.

$$\alpha = \begin{cases} \alpha_m & \text{exists } type_m \text{ dependency} \\ & \text{relation from } w_i \text{ to } w_t \\ 0 & \text{Otherwise} \end{cases} \quad (10)$$

2.3 Optimization

This section introduces optimization methods for SA-LSTM. We use maximum likelihood criterion to train SA-LSTM. We choose stochastic gradient descent algorithm to optimize parameters.

Given a training pair $T = (x, y)$ where T is the current training pair, x denotes current training sentence, and y is the corresponding correct answer path. $y_t = k$ means that the t-th word has the k-th semantic role label.

The score of o_t is calculated as:

$$s(x, y, \theta) = \sum_{t=1}^{N_i} o_{ty_t} \quad (11)$$

where N_i is the word number of the current sentence and θ stands for all parameters. So the log likelihood of a single sentence is:

$$\begin{aligned} \log p(y|x, \theta) &= \log \frac{exp(s(x, y, \theta))}{\sum_{y'} exp(s(x, y', \theta))} \\ &= s(x, y, \theta) - log \sum\nolimits_{y'} exp(s(x, y', \theta)) \end{aligned} \quad (12)$$

Method	F_1 **%**
Xue(2008)	71.90
Sun et al.(2009)	74.12
Yand and Zong(2014)	75.31
Wang et al.(Bi-LSTM)(2015)	77.09
Sha et al.(2016)	77.69
Path LSTM, Roth et al. (2016)[3]	**79.01**
BiLSTM+feature engineering dependency	77.75
SA-LSTM(**Random Initialized**)	**79.81**
SA-LSTM(**Pre-trained Embedding**)	**79.92**

Table 1: Results comparison on CPB 1.0

Method	F_1 **%**
Bi-LSTM(2 layers)	74.52
Bi-LSTM + SA-LSTM(2 layers)	**76.63**
He(2017)(Single Model, state of the art)	81.62
He(Single Model, 8 layers) + SA-LSTM	**81.90**

Table 2: Results on English CoNLL 2005

where y' ranges from all valid paths of answers. We use Viterbi algorithm to calculate the global normalization. Besides, we collected those impossible transitions from corpus beforehand. When calculating global normalization, we prevent calculating impossible paths which contains impossible transitions.

3 Experiment

3.1 Experiment setting

In order to compare with previous Chinese SRL works, we choose to do experiment on CPB 1.0. We also follow the same data setting as previous Chinese SRL work (**??**) did. Pre-trained[1] word embeddings are tested on SA-LSTM and shows improvement.

For English SRL, we test on CoNLL 2005 dataset.

We use Stanford Parser (Chen and Manning, 2014) to get dependency relation. The training set of Chinese parser overlaps a part of CPB 1.0 test set, so we retrained the parser. Dimension of hyper parameters are tuned according to development set. $n_1 = 200$, $n_h = 100$, $n_2 = 200$, $n_3 = 100$, $learning\ rate = 0.001$.

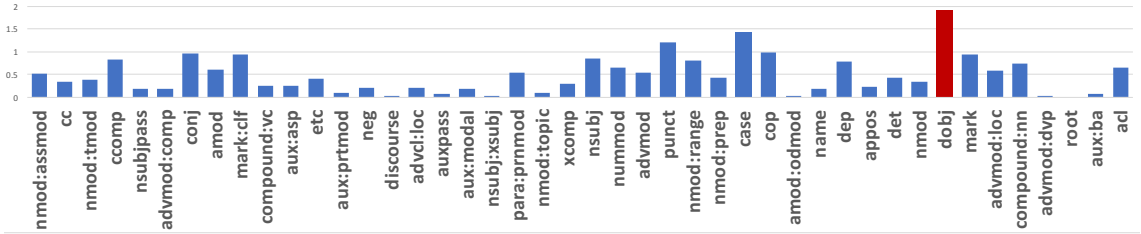

Figure 3: Visualization of trained weight α_m. X axis is Universal Dependency type, Y axis is the weight.

3.2 Syntax Aware LSTM Performance

To prove that SA-LSTM models dependency relation better than simple feature engineering method, we design an experiment in which dependency relation is added to bi-LSTM in a traditional feature engineering way.

Given a word w_t, F_t is the average of all dependency related x_i of previous words w_i, as shown in Equation 13:

$$F_t = \frac{1}{T} \sum_{i=0}^{t-1} \alpha \times x_i \tag{13}$$

where T is the number of dependency related words and α is a $0, 1$ variable calculated as in Equation 7.

Then F_t is concatenated to x_t to form a new feature representation. Then these representations are fed into bi-LSTM.

As shown in Table 1, on CPB 1.0, SA-LSTM reaches $79.81\%F_1$ score with random initialization and $79.92\%F_1$ score with pre-trained word embedding. Both of them are the best F_1 score ever published on CPB 1.0 dataset.

In contrast to the "bi-LSTM+feature engineering dependency" model, it is clear that architecture method of SA-LSTM gains more improvement(77.09% to 79.81%) than simple feature engineering method(77.09% to 77.75%). Path-LSTM (Roth and Lapata, 2016) embeds dependency path between predicate and argument for each word using LSTM, then does classification according to such path embedding and some other features. SA-LSTM ($79.81\%F_1$) outperforms Path-LSTM ($79.01\%F_1$) on CPB 1.0.

Both "bi-LSTM + feature engineering dependency" and Path-LSTM only model dependency parsing information for each single word, which can not model the whole dependency tree structure. However, by building the dependency relation directly into the structure of SA-LSTM and changing the way information flows, SA-LSTM is able to model the whole tree structure of dependency relation.

We also test our SA-LSTM on English CoNLL 2005 dataset. Replacing conventional bi-LSTM by SA-LSTM brings $1.7\%F_1$ improvement. Replacing bi-LSTM layers of the state of the art model (He et al., 2017) by SA-LSTM [1] brings $0.3\%F_1$ improvement.

3.3 Visualization of Trained Weights

According to Equation 10, influence from a single type of dependency relation will be multiplied with type weight α_m. When α_m is 0, the influence from this type of dependency relation will be ignored totally. When the weight is bigger, the type of dependency relation will have more influence on the whole system.

As shown in Figure 3, dependency relation type $dobj$ receives the highest weight after training, as shown by the red bar. According to grammar knowledge, $dobj$ should be an informative relation for SRL task, and SA-LSTM gives $dobj$ the most influence automatically. This further demonstrate that the result of SA-LSTM is highly in accordance with grammar knowledge.

4 Related works

Semantic role labeling (SRL) was first defined by (Gildea and Jurafsky, 2002). Early works (Gildea and Jurafsky, 2002; **?**) on SRL got promising result without large annotated SRL corpus. **?** built the Chinese Proposition Bank to standardize Chinese SRL research.

Traditional works such as (**??**Ding and Chang, 2009; **?**; Chen et al., 2006; **?**) use feature engineering methods. Their methods can take dependency

[1]Trained by word2vec on Chinese Gigaword Corpus

[2]All experiment code and related files are available on request

[3]We test the model on CPB 1.0

[1]We add syntax-aware connections to every bi-LSTM layer in the 8-layer model of (He et al., 2017)

relation into account in feature engineering way, such as syntactic path feature. It is obvious that feature engineering method can not fully capture the tree structure of dependency relation.

More recent SRL works often use neural network based methods. Collobert and Weston (2008) proposed a Convolutional Neural Network (CNN) method for SRL. **?** proposed bidirectional RNN-LSTM method for English SRL, and **?** proposed a bi-RNN-LSTM method for Chinese SRL on which SA-LSTM is based. He et al. (2017) further extends the work of **?**. NN based methods extract features automatically and significantly outperforms traditional methods. However, most NN based methods can not utilize dependency relation which is considered important for semantic related NLP tasks (**?**Punyakanok et al., 2008; Pradhan et al., 2005).

The work of Roth and Lapata (2016) and Sha et al. (2016) have the same motivation as SA-LSTM, but in different ways. Sha et al. (2016) uses dependency relation as feature to do argument relations classification. Roth and Lapata (2016) embeds dependency path into feature representation for each word using LSTM. In contrast, SA-LSTM utilizes dependency relation in an architecture engineering way, by integrating the whole dependency tree structure directly into SA-LSTM structure.

5 Conclusion

We propose Syntax Aware LSTM model for Semantic Role Labeling (SRL). SA-LSTM is able to directly model the whole tree structure of dependency relation in an architecture engineering way. Experiments show that SA-LSTM can model dependency relation better than traditional feature engineering way. SA-LSTM gives state of the art F_1 on CPB 1.0 and also shows improvement on English CoNLL 2005 dataset.

Acknowledgments

Thanks Natali for proofreading the paper and giving so many valuable suggestions.

This paper is partially supported by the National Natural Science Foundation of China (NSFC Grant Nos. 61472006 and 91646202) as well as the National Basic Research Program (973 Program No. 2014CB340405).

References

Wilker Aziz, Miguel Rios, and Lucia Specia. 2011. Shallow semantic trees for smt. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 316–322. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of chinese chunking. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 97–104. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Weiwei Ding and Baobao Chang. 2008. Improving chinese semantic role classification with hierarchical feature selection strategy. In *Proceedings of the conference on empirical methods in natural language processing*, pages 324–333. Association for Computational Linguistics.

Weiwei Ding and Baobao Chang. 2009. Word based chinese semantic role labeling with semantic chunking. *International Journal of Computer Processing Of Languages*, 22(02n03):133–154.

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

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

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

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. *arXiv preprint arXiv:1605.07515*.

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

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of chinese. In *Proceedings of NAACL 2004*, pages 249–256.

Weiwei Sun. 2010. Improving chinese semantic role labeling with rich syntactic features. In *Proceedings of the ACL 2010 conference short papers*, pages 168–172. Association for Computational Linguistics.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese semantic role labeling with shallow parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1475–1483. Association for Computational Linguistics.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *EMNLP*, pages 1626–1631.

Huijia Wu, Jiajun Zhang, and Chengqing Zong. 2016. An empirical exploration of skip connections for sequential tagging. *arXiv preprint arXiv:1610.03167*.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 902–911. Association for Computational Linguistics.

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

Nianwen Xue and Martha Palmer. 2003. Annotating the propositions in the penn chinese treebank. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 47–54. Association for Computational Linguistics.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *IJCAI*, volume 5, pages 1160–1165. Citeseer.

Haitong Yang, Chengqing Zong, et al. 2014. Multi-predicate semantic role labeling. In *EMNLP*, pages 363–373.

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