Appendix
"Transductive Learning of Neural Language Models for Syntactic and Semantic Analysis"

Hiroki Ouchi[1,2] **Jun Suzuki**[2,1] **Kentaro Inui**[2,1]
[1] RIKEN Center for Advanced Intelligence Project [2] Tohoku University
`hiroki.ouchi@riken.jp`, `{jun.suzuki,inui}@ecei.tohoku.ac.jp`

A Tasks

Overview. The goal of syntactic chunking is to divide a sentence into non-overlapping segments (phrases) that consist of syntactically related words. The goal of SRL is to identify semantic arguments for each predicate. For example, consider the following sentence.

	The$_1$	man$_2$	$\underline{\text{kept}_3}$	a$_4$	cat$_5$
SynChunk	[NP	]		[NP	]
SemRole	[A0	]		[A1	]

In syntactic chunking, "The man" and "a cat" are recognized as noun phrases (`NP`). In SRL, for the predicate "kept", "The man" is the `A0` argument, and "a cat" is the `A1` argument. In the following, we give formal descriptions for these tasks.

Syntactic chunking. The inputs of the syntactic chunking task is denoted as $X = w_{1:T}$, where $w_{1:T} = (w_1, w_2, \cdots, w_T)$ is a sentence of T words. The outputs to predict is a set of labeled spans $Y = \{\langle i, j, r \rangle_k\}_{k=1}^{|Y|}$.

$$\textbf{Input: } X = w_{1:T}$$
$$\textbf{Output: } Y = \{\langle i, j, r \rangle_k\}_{k=1}^{|Y|}$$

Each labeled span $\langle i, j, r \rangle$ consists of word indices i and j in the sentence and a syntactic (constituent) label $r \in \mathcal{R}^{\text{syn}}$. For the above example, the input sentence is $\boldsymbol{w}$ = ("The", "man", "kept", "a", "cat"). The correct labeled span $\langle 1, 2, \texttt{NP} \rangle$ indicates that "The man" is a noun phrase (NP) and $\langle 4, 5, \texttt{NP} \rangle$ indicates that "a cat" is an NP.

Semantic role labeling. The inputs of the SRL task is denoted as $X = \{w_{1:T}, p\}$, where $w_{1:T} = (w_1, w_2, \cdots, w_T)$ is a sentence of T words and p is the target predicate position index. The outputs to predict is a set of labeled spans $Y = \{\langle i, j, r \rangle_k\}_{k=1}^{|Y|}$.

$$\textbf{Input: } X = \{w_{1:T}, p\}$$
$$\textbf{Output: } Y = \{\langle i, j, r \rangle_k\}_{k=1}^{|Y|}$$

Each labeled span $\langle i, j, r \rangle$ consists of word indices i and j in the sentence and a semantic role $r \in \mathcal{R}^{\text{sem}}$. For the above example, the input sentence is $\boldsymbol{w}$ = ("The", "man", "kept", "a", "cat"), and the target predicate position is $p = 3$. The correct labeled span $\langle 1, 2, \texttt{A0} \rangle$ indicates that the A0 (agent) argument is "The man", and $\langle 4, 5, \texttt{A1} \rangle$ indicates that the A1 (patient) argument is "a cat".

B Models

As a syntactic chunking model, we used a variant of The Reconciled Span Parser (Joshi et al., 2018). As an SRL model, we used a BiLSTM-Span model (Ouchi et al., 2018). These models have a common architecture in part.

The embedding layer. Both chunking and SRL models first encode the inputs as a sequence of vector representations.

$$\mathbf{x}_{1:T} \quad = \quad \text{Emb}(w_{1:T}) \ .$$

The embedding layer (Emb) outputs a sequence of vector representations $\mathbf{x}_{1:T}$ for the input sentence $w_{1:T}$. For the embedding layer of the syntactic chunking model, a language model (ELMo) f^{lm} outputs a sequence of word representations $\mathbf{x}_t^{\text{word}} \in \mathbb{R}^{d^{\text{word}}}$.

$$\text{Emb}(w_{1:T}) = f^{\text{lm}}(w_{1:T}) = \mathbf{x}_{1:T}^{\text{word}} \ .$$

The embedding layer of the SRL model outputs not only word representations $\mathbf{x}_t^{\text{word}}$ but also predicate mark representations $\mathbf{x}_t^{\text{mark}} \in \mathbb{R}^{d^{\text{mark}}}$.

$$\begin{aligned}
\text{Emb}(w_{1:T}) &= \mathbf{x}_{1:T} \ , \\
\mathbf{x}_t &= [\mathbf{x}_t^{\text{word}}; \mathbf{x}_t^{\text{mark}}] \ , \\
\mathbf{x}_{1:T}^{\text{word}} &= f^{\text{lm}}(w_{1:T}) \ , \\
\mathbf{x}_{1:T}^{\text{mark}} &= f^{\text{mark}}(w_{1:T}, p) \ .
\end{aligned}$$

Each word representation is concatenated with each predicate mark representation $\mathbf{x}_t^{\text{mark}}$ in the same way as He et al. (2017).

The BiLSTM layer. The input representation $\mathbf{x}_t$ is fed to the BiLSTM layer.[1]

$$\mathbf{h}_{1:T} = \text{BiLSTM}(\mathbf{x}_{1:T}) \ .$$

The BiLSTM layer outputs a sequence of base feature vectors $\mathbf{h}_{1:T}$, each of which is a d^{hidden} dimensional vector, i.e. $\mathbf{h}_t \in \mathbb{R}^{d^{\text{hidden}}}$.

Span representation. From the base features $\mathbf{h}_{1:T}$ induced by the BiLSTMs in Equation 1, we create the span feature representations,

$$\mathbf{h}_{i,j}^{\text{span}} = [\mathbf{h}_i + \mathbf{h}_j; \mathbf{h}_i - \mathbf{h}_j] \ ,$$

where the addition and subtraction features of the i-th and j-th hidden states are concatenated and used as the feature for a span (i, j).

Labeling for syntactic chunking. For syntactic chunking, we model normalized distribution over all labels $\mathcal{R}$ for each span $(i, j) \in \mathcal{S}$. Specifically, given a span representation $\mathbf{h}_{i,j}^{\text{span}}$ as input, we first calculate the score for each span $(i, j) \in \mathcal{S}$ with a label $r \in \mathcal{R}$.

$$\text{score}_{i,j,r} = \mathbf{W}[r] \cdot \mathbf{h}_{i,j}^{\text{span}} \ , \qquad (1)$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{R}| \times 2d^{\text{hidden}}}$ has a row vector associated with each label r, and $\mathbf{W}[r]$ denotes the r-th row vector. As the result of the inner product of $\mathbf{W}[r]$ and $\mathbf{h}_{i,j}^{\text{span}}$, we obtain the score for a labeled span $\langle i, j, r \rangle$. Each score is softmaxed over all the labels $r \in \mathcal{R}$.

$$\text{P}(r \mid i, j) = \frac{\exp(\text{scre}_{i,j,r})}{\sum_{r' \in \mathcal{R}} \exp(\text{score}_{i,j,r'})} \ ,$$

To train the parameters, we minimize the negative log-likelihood,

$$\mathcal{L} = \sum_{(X,Y) \in \mathcal{D}} \ell(X, Y) \ ,$$
$$\ell(X, Y) = - \sum_{\langle i,j,r \rangle \in Y} \log \text{P}(r|i, j) \ ,$$

where function $\ell(X, Y)$ is a loss for each sample.

Labeling for SRL. For SRL, we model normalized distribution over all possible spans $\mathcal{S}$ for each label $r \in \mathcal{R}$. In the same way as the chunking model, we first calculate the score $\text{score}_{i,j,r}$ (Eq. 1). Then, each score is softmaxed over all the possible spans $(i, j) \in \mathcal{S}$.

$$\text{P}(i, j \mid r) = \frac{\exp(\text{scre}_{i,j,r})}{\sum_{(i',j') \in \mathcal{S}} \exp(\text{score}_{i',j',r})} \ ,$$

To train the parameters, we minimize the negative log-likelihood,

$$\mathcal{L} = \sum_{(X,Y) \in \mathcal{D}} \ell(X, Y) \ ,$$
$$\ell(X, Y) = - \sum_{\langle i,j,r \rangle \in Y} \log \text{P}(i, j|r) \ ,$$

where function $\ell(X, Y)$ is a loss for each sample.

C Training Details for Language Models

We used Embeddings from Language Models (ELMo) (Peters et al., 2018), 1024-dimensional vectors ($d^{\text{word}} = 1024$)[2]. We also used the same hyperparameters and training method as Peters et al. (2018).

Training and fine-tuning of ELMo. First, we trained ELMo on the 1B billion word benchmark (Chelba et al., 2013). It took about a week on four GPUs. Second, we fine-tuned ELMo on each test set. Because each test set is much smaller than the 1B billion word benchmark corpus, it takes less than an hour. During training of syntactic and semantic models, we freezed the fine-tuned ELMo (not updated it).

Cross-domain settings. In cross-domain settings, we first built a fine-tuned ELMo and then built each syntactic and semantic model. Consider the case where `NW` $\rightarrow$ `BC`, i.e., the source domain

[1]We used the stacked BiLSTMs in an interleaving fashion (Zhou and Xu, 2015; He et al., 2017). The details can be found in their papers.

[2]https://allennlp.org/elmo

is the newswire NW and the target domain is the broadcast conversation BC. We first trained ELMo on the 1B billion word benchmark corpus and fine-tuned it on the BC test set. We then trained syntactic and semantic models that use the fine-tuned ELMo on the NW training set. We selected hyperparameters by using the NW development set. Finally, we evaluated the trained model on the BC test set. In the same way, we conducted training and evaluation for each cross-domain setting.

Standard benchmarks. In standard benchmark settings using the CoNLL-2000/2005/2012 datasets, each model used the ELMo fine-tuned on each test set. Consider the case where the benchmark setting on the CoNLL-2005 dataset. We first trained ELMo on the 1B billion word benchmark corpus and fine-tuned it on the CoNLL-2005 test set. We then trained an SRL model that uses the fine-tuned ELMo on the CoNLL-2005 training set. We selected hyperparameters by using the CoNLL-2005 development set. Finally, we evaluated the trained model on the CoNLL-2005 test set. In the same way, we conducted training and evaluation on the other datasets.

D Training Details for Syntactic and Semantic Models

Name	Value
Word representation d^{word}	1024-dimensional ELMo
Mark representation d^{mark}	50-dimensional vector
BiLSTM layers	4
BiLSTM hidden units d^{hidden}	300 dimensions
Mini-batch Size	32
Optimization	Adam
Learning Rate	0.001
L2 Regularization λ	0.0001
Dropout Ratio for BiLSTMs	0.1
Dropout Ratio for ELMo	0.5

Table 1: Hyperparameters used in the experiments.

Table 1 lists the hyperparameters used in the experiments.

Network setup. As predicate mark representations $\mathbf{x}^{\text{mark}}$, we used randomly initialized 50-dimensional vectors ($d^{\text{mark}} = 50$). During training, we updated them. For the BiLSTM layer, we used 4 stacked BiLSTMs (2 forward and 2 backward LSTMs) with 300-dimensional hidden units. Following He et al. (2017), we initialized all the parameter matrices in BiLSTMs with random orthonormal matrices (Saxe et al., 2013). Other parameters were initialized following Glorot and Bengio (2010), and bias parameters were initialized with zero vectors.

Regularization. We applied dropout (Srivastava et al., 2014) to the input vectors of each LSTM with dropout ratio of 0.1 and the ELMo embeddings with dropout ratio of 0.5.

Training. To optimize the parameters, we used Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was initialized to 0.001. After training 50 epochs, we halved the learning rate every 25 epochs. Parameter updates were performed in mini-batches of 32. The number of training epochs was set to 100. We saved the parameters that achieve the best F1 score on the development set and evaluated them on the test set. Training the models takes 24 - 48 hours on a single GPU.

References

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256.

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

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of ACL*, pages 1190–1199.

D.P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

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

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

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

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.