Abstract

Semantic role labeling usually models structures using sequences, trees, or graphs. Past works focused on researching novel modeling methods and neural structures and integrating more features. In this paper, we re-examined the noise in neural semantic role labeling models, a problem that has been long-ignored. By proposing a noisy channel model structure, we effectively eliminate the noise in the labeling flow and thus improve performance. Without relying on additional features, our proposed novel model significantly outperforms a strong baseline on multiple popular semantic role labeling benchmarks, which demonstrates the effectiveness and robustness of our proposed model.

1 Introduction

Semantic role labeling (SRL) extracts shallow semantic structures such as agents, goals, temporal, patient/receiver, or locative arguments for predicates. It is a popular task in natural language processing and can be useful in a variety of downstream tasks, such as information extraction (Christensen et al., 2010), machine reading comprehension (Zhang et al., 2019b), and machine translation (Liu and Gildea, 2010).

SRL’s development has paralleled that of syntax and transferred from constituency to dependency structures. As a result, SRL is typically subdivided into span (constituency) SRL and dependency SRL based on the argument formalism. In span SRL, arguments are the constituent spans of the sentence, while in dependency SRL, the head words of the constituent spans are the arguments.

A number of modeling approaches have been studied in recent work. SRL can be abstracted as the identification of predicates and arguments and the classification of their pairs, so SRL can be considered to a sequence-based labeling problem either by identifying/giving the predicate in advance (Zhou and Xu, 2015; Marcheggiani et al., 2017a; He et al., 2017, 2018b; Li et al., 2018) or, modeling SRL as a graph, the predicate is used as the root node of the tree, the arguments are treated as its child nodes, and the predicate-argument relationships are used for edge labels (Cai et al., 2018). In methods using pre-identified predicates, arguments are labeled one predicate at a time, while when modeling SRL as a graph, all predicates, arguments, and their pairs are identified and classified in one-shot (He et al., 2018a; Li et al., 2019). These modeling approaches, when coupled with large pre-trained language models, currently comprise the state-of-the-art SRL models.

While novel models are still introduced, few studies have focused on SRL’s noise issue, an important performance bottleneck in SRL that we focus on and aim to alleviate using a noisy channel model. Neural models often introduce features that are either do not help or even actively hurt target prediction during representation encoding and neural network scoring; we call the inclusion of these features the noise issue. Given an input sentence $X$, an SRL model given as a channel $P(Y|X)$ would ideally transform the input $X$ into the correct target $Y$. The model is noisy, however, making this channel a noisy channel. The noisy channel model refers to the models that can reduce the noise in the channel. Using the premise that low probability predictions (i.e. with larger uncertainty) are more likely to contain errors resulting from noise than are high probability predictions, we aim to minimize the noise of this channel and thus call our model the noisy channel model. We utilize this premise and allow the model for modeling the likelihood of making particular errors itself, instead of only relying it as loss.

Specifically, we propose a novel hierarchical network structure consisting of a traditional SRL model that provides the noisy prediction and a noise-estimating component that estimates the
amount of noisy errors caused by this prediction. In order to make the noise controllable and removable, we introduce an external noise generator to produce and model noise for the input, giving us a source on which to base noise estimations. Based on the bottom noise estimator, we build a denoising SRL model in which a two-stream self-attention mechanism is adopted to incorporate the noisy prediction and the model noise-independent word representations of the bottom model. In our model, noise is explicitly added, estimated, and eventually eliminated. Our model differs from the traditional noise channel model as we do not seek to simply restore the original input and therefore provides a new alternative model for NLP labeling tasks. Furthermore, the noise within the model is random, so performing direct modeling is extremely difficult; however, we account for this by adding artificially synthetic noise for better denoising, a critical step for ensuring performance improvement.

Our empirical evaluation is conducted on the popular multilingual dependency SRL benchmark CoNLL-2009 for multiple settings. The results show that our proposed model can effectively alleviate the noise in the baseline model and consistently improve SRL performance. Notably, our model achieves the new state-of-the-art on several datasets. Additional ablation studies demonstrate that our proposed noisy channel model can effectively remove the inherent noise in the model; and thus obtain a higher quality output.

2 The Method

2.1 Overview

We present our noisy channel model for SRL in this section. First, our full model is split into bottom and top components. The lower component is a variant of a regular SRL model. We choose a simple and intuitive BiLSTM+MLP sequence labeling model as our basic model in the bottom component. We use this for noisy label prediction and noise estimation. The top component is designed to denoise the base model’s noisy predicted probabilities and result in a more accurate prediction. Specifically, for this top component, we adopt a two-stream self-attention denoiser. On the one hand, it encodes the word-level representation with a word-based self-attention; and on the other hand it denoises the label representation using a word-label cross-attention based on a probability-soft embedding of the noisy label prediction. It then combines the two streams to make the final prediction. The overall architecture of the noisy channel model is shown in Figure 1.

2.2 Base Model

First, we explain the base SRL model of our bottom component. Formally, given an input sentence \( X = \{x_1, x_2, \ldots, x_n\} \), the SRL model predicts a semantic triple of the predicate and argument and the relationship between them: i.e. \( Y = \{(p, a, r)\}, p \in X, a \in X, r \in \mathcal{R} \), where \( \mathcal{R} \) is the vocabulary of semantic relationships. Although the target prediction is based on triples, in the sequence-based modeling method, the triple is transformed into several label sequences using a task decomposition, and then sequence labeling is performed separately. SRL is typically decomposed into four subtasks: predicate identification, predicate disambiguation, argument recognition, and argument classification. If the predicate is prespecified, the problem is changed to then only entails identifying and classifying its arguments.

Following He et al. (2018b)’s practice, we adopted a similar model structure and made some necessary changes to meet our overall needs. To vectorize sentence input \( X \), we employed word embeddings \( e_{\text{word}} \) and a character CNN encoding network \( e_{\text{char}} \), which not only takes into account word information but also better handles the out-of-vocabulary (OOV) problem. Other features like Parts-of-Speech (POS, \( e_{\text{pos}} \)) and lemmas (\( e_{\text{lem}} \)) are also integrated into the embeddings. Since the labeling of arguments is related to the predicate, predicate awareness is crucial to the implementation of the sequence labeling. Therefore, we use additional predicate indicator embedding \( e_{\text{ind}} \) is used to indicate which predicate is currently being processed. A word is then represented by concatenating its embeddings:

\[
e_{i}^{w} = [e_{i}^{\text{word}}; e_{i}^{\text{char}}; e_{i}^{\text{pos}}; e_{i}^{\text{lem}}; e_{i}^{\text{ind}}],
\]

where \([::] \) denotes a concatenation operation. Recently, pre-trained language models like ELMo (Peters et al., 2018), BERT have further improved the performance of many NLP tasks, our method can also further enhance its embeddings by concatenating language model features \( e_{\text{plm}} \).

SRL is a context-related task, while the vector representation \( e_{i} \) of word \( w_{i} \) is context-independent. To further contextualize the representation, we encode the word representation \( h_{i} \in H \)
using a Bidirectional Long Short-Term Memory (LSTM) encoder (Hochreiter and Schmidhuber, 1997):

\[ H = \text{BiLSTM}(e_1^w, ..., e_n^w). \]

The BiLSTM encoder was chosen to facilitate a more fair comparison with LSTM-based SRL works. Encoders such as CNN or Transformer can obviously also be adopted for contextualizing representations.

We can employ Multi-layer Perceptron (MLP) layers to project the contextualized representation into the predicted probability distribution of each position:

\[ P(y_i|X, \theta) = \text{Softmax}(\text{MLP}(h_i)), \]

where \( \theta \) is the parameters of base model.

### 2.3 Noise Estimation

Since the inherent noise of the model will have a negative impact on the model’s prediction, further denoising is beneficial to performance improvement. We define the inherent noise of the base model as \( \zeta \). Since there is no direct way to model the real inherent noise as it may be unstructured and changing, we artificially synthesize a number of different noises and apply them to the same example so that the model can learn to capture and remove this noise.

Following (Gui et al., 2020), we use sampling based on Monte Carlo Dropout (Gal and Ghahramani, 2016a) to create rational noise, which Gal and Ghahramani refer to as uncertainty. We sample the dropout distribution \( M \) times for a single example. Assuming that the noise generated by sampling \( M \) times is \( N = \{\eta_1, \eta_2, ..., \eta_M\} \), then the predicted probability of the instance with the \( k \)-th sampling noise can be written as:

\[ P(y_i|X, \theta, \zeta + \eta_k) = \text{Softmax}(\text{MLP}(h_i(\eta_k))), \]

where \( h_i(\eta_k) \) represents the contextual representation of \( w_i \) with noise \( \eta_k \).

According to the idea of boosting (Schapire, 2003; Wang et al., 2008), we combine these predicted probabilities with various synthetic noises,

\[ P(y_i|X, \theta) = \frac{1}{M} \sum_{k=1}^{M} P(y_i|X, \theta, \zeta + \eta_k). \]

In terms of implementation, we repeat the input batch \( M \) times to allow parallelization on the GPU; and then use the standard dropout on the sentence length dimension. Notably, we also enable this dropout for synthesizing noise in the inference phase.

The synthesized noise is thus integrated into the predicted distribution. Since these probability scores are computed by a probabilistically-weighted average of various noises, synthesized noise that better resembles the true noise will be emphasized in this averaging operation, and the parts of true noise that are inconsistent with the synthesized noise will be reduced by average operation. Reducing the artificial noise (as seen in the

---

**Figure 1:** The overall architecture of our noisy channel model for semantic role labeling.
next step), should then lead to a reduction in real noise as well because of the isomorphism.

Based on predicted probabilities, we obtain the noisy label predictions \( \hat{y}_i \) and calculate entropy as their noise estimation:

\[
\hat{y}_i = \arg\max(P(y_i|X, \theta)),
\]

\[
\tau_i = -\sum_{r \in R} P(y_i = r|X, \theta)\log P(y_i = r|X, \theta).
\]

Entropy \( \tau_i \) is a good noise estimation since when \( \tau_i \) is larger, the predicted label \( \hat{y}_i \) has a greater probability of being wrong, which means that the label in position \( i \) needs to be further processed by the denoiser.

### 2.4 Denoiser

The denoiser eliminates noise from the base model’s label prediction. We leverage a two-stream self-attention structure to be able to focus on both the original word sequence and the noisy label sequence. Two-stream self-attention was first proposed in (Yang et al., 2019) to use two sets of hidden representations and model the two-stream interactions. In our work, the original word embedding (which lacks the inherent noise of the model) interacts with the soft embedding of the noisy label using the two-stream attention mechanism, which helps to remove the noise in the noisy label prediction from the base model.

For the two-stream self-attention structure, we implement a multi-head self-attention (Vaswani et al., 2017) with relative position encoding following (Yang et al., 2019) as the basis. The calculation of two-stream attention between word sequence and noise label sequence can then be expressed as follows:

\[
o^w2w = \text{LayerNorm}(e^w + \text{RelMHAttn}(e^wW_Q, e^wW_K, e^wW_V)),
\]

\[
h^w = \text{FeedForward}(o^w2w),
\]

\[
o^w2l = \text{LayerNorm}(e^w + \text{RelMHAttn}(e^wW_Q, e^lW_K, e^lW_V)),
\]

\[
h^l = \text{FeedForward}(o^w2l),
\]

where \text{RelMHAttn} denotes relative multi-head attention, \text{LayerNorm} denotes layer normalization, and \text{FeedForward} denotes a feed forward layer.

There are two reasons for using relative multi-head attention rather than ordinary multi-head attention. On the one hand, no additional position encoding features are introduced. New features like those could corrupt the original features and have a negative impact on the denoising effect. On the other hand, the relative distance between labels is a valuable feature that can be beneficial for denoising.

To make the gradient for the noisy label embedding differentiable in the training phase, we did not use the embedding of predicted label from argmax operation directly; but rather adopted a soft embedding technique, which can be expressed as:

\[
e_i^l = \sum_{r \in R} P(y_i = r|X, \theta)\text{Emb}(\text{label})(r),
\]

in which \( \text{Emb}(\text{label}) \) represents the embedding space for semantic role labels. The basic idea is to weight sum all label embeddings using the predicted probabilities of each label as weights.

After two-stream encoding and denosing, we concatenate the output features in the two streams and use the MLP layer to project the features to the label probability space:

\[
P(y_i|X, \theta, \phi) = \text{Softmax}(\text{MLP}([h^w_i; h^l_i])),
\]

where \( \phi \) denotes the parameters of the denoiser.

### 2.5 Training and Inference

Since our model makes two label predictions during the training process, the total training loss naturally consists of two parts:

\[
\mathcal{L}(\theta) = -\sum_{i=1}^{n} P(y_i = \hat{y}_i|X, \theta)\log P(y_i = \hat{y}_i|X, \theta),
\]

\[
\mathcal{L}(\theta, \phi) = -\sum_{i=1}^{n} P(y_i = \hat{y}_i|X, \theta, \phi)\log P(y_i = \hat{y}_i|X, \theta, \phi),
\]

\[
\mathcal{L} = \mathcal{L}(\theta) + \mathcal{L}(\theta, \phi).
\]

To optimize the model, we use cross-entropy to calculate the loss. The loss of the base model is denoted by \( \mathcal{L}(\theta) \) and is used to make the base model predict the correct label as much as possible. \( \mathcal{L}(\theta, \phi) \) is the loss of denoising during training and not only trains the denoiser; but also optimizes the whole model jointly.

In the inference stage, we do not directly take the final prediction of the denoiser as the output of the model. According to Ockham’s razor, “entities should not be multiplied without necessity;” we therefore only use the output of the denoiser for some labels that are affected by noise and keep the rest. In terms of implementation, we set a threshold \( \rho \) for noise estimation \( \tau \) and combine the two...
predictions thus:

\[
\hat{y}_i^* = \begin{cases} 
\arg\max(P(y_i|X, \theta)), & \tau_i < \rho, \\
\arg\max(P(y_i|X, \theta, \phi)), & \tau_i \geq \rho.
\end{cases}
\]

### 3 Experiments and Analysis

#### 3.1 Setup

We conducted experiments on the CoNLL-2009 shared task's multilingual dataset, which includes Catalan, Chinese, Czech, English, German, Japanese, and Spanish. In the experiments, we used two settings: *predicate-given* and *end-to-end*. In the *predicate-given* setting, we use the official dataset’s pre-specified predicate but predict the predicate sense, argument, and semantic roles. In the *end-to-end* setting, all of the predicate and argument must be predicted since they are all unknown. Additionally, in keeping with (He et al., 2018b), we use POS and lemma features in the model. These are the predicted POS tags and lemma as given by the CoNLL-2009 shared task for each language. To keep the model concise, we did not leverage syntactic tree information, which makes our model syntax-agnostic.

Our model uses pre-trained fastText (Grave et al., 2018) embeddings as a word embedding initialization. Other POS embeddings, lemma embeddings, and label embeddings are initialized randomly. In the case of using a pre-trained language model, the ELMo-original-5.5B model is used for ELMo feature extraction, while for BERT (Devlin et al., 2019), BERT-large-cased is used for English and BERT-base-chinese for Chinese. To keep the results comparable to (Li et al., 2020a), BERT for other languages in multilingual benchmarks adopted is the same as them. All models are trained for up to 400 epochs, early stopping patient is set to 20, and the batch size is 64. We use the categorial cross-entropy as the objective and the Adam optimizer (Kingma and Ba, 2015). For other model hyper-parameters, please see Appendix A.1.

#### 3.2 Analysis

**Predicate-given Results** In the CoNLL-2009 multilingual benchmark, the English and Chinese datasets are used in the majority of SRL works. To compare with these works, we list the results from recent works and our models in Table 1. When comparing the baseline to our proposed full model, in the case when not using any pre-trained language models, our full model obtained 1.0+ Sem-F1 improvement on both the English in-domain (ID) and out-of-domain (OOD) tests, as well as the Chinese test set, demonstrating the effectiveness of our proposed method.

As previously mentioned, when we compared the SRL performance of previous works, we found that the baseline results of different modeling methods differed. There is generally a trend in the performance of the models: Graph > Tree > Sequence. This trend may be caused by the more complex modeling methods (i.e., graph-based methods) taking into account more features. Conversely, however, decoding speed follows its own trend: Sequence > Tree > Graph in terms of decoding speed (for details see our speed analysis in Appendix A.3). Without using pre-trained language models or additional features such as syntax parse trees, our method first achieves state-of-the-art among sequence-based modeling approaches. Furthermore, our sequence-based model outperformed the best results of the tree-based modeling approach (Cai et al., 2018) and achieved results comparable to those of the state-of-the-art graph-based modeling work (Fei et al., 2021). This shows that our proposed method is both fast and effective.

Whereas previous works that typically integrated more features had disparities in their improvements on ID and OOD results, our approach interestingly boasts similar improvements both the ID and OOD settings. That our approach provides an even improvement across both settings suggests that it does reduce the noise inherent in the model, as this is a problem that affects both settings. This contrasts the development of new features, which are typically biased towards one of the two settings and thus give disparate performance improvements.

The results in the *predicate-given* setting on CoNLL-09 multilingual are shown in Table 2. The study of multilingual SRL has resurfaced in recent years, especially after the introduction of the multilingual pre-trained language model - BERT. The improvements of our full models over the baselines on the multilingual test sets are consistent. Although the performance of our baseline (sequence-based) lags behind that of tree-based (He et al., 2019) and graph-based models (Li et al., 2020), our full model achieves comparable results to these models in most cases and further obtains state-of-the-art results in Catalan, Czech, and Japanese languages with the help of BERT. Furthermore, our
approach is neither limited to the sequence modeling model nor the SRL task. The noise channel model is a task-independent method of alleviating a model’s inherent noise, and our approach model-and task-independent. Thus, our method also can be transplanted to a tree or graph-based baseline. We leave this to future work since in this paper, we are primarily interested in enhancing performance by alleviating the model’s inherent noise in this paper.

End-to-end Results The end-to-end setting necessitates fewer external preset conditions and therefore better resembles realistic applications. Table 3 shows the results of our approach in this more challenging setting. The outcome of predicate identification has a significant impact on the overall F1, but this has usually been overlooked in previous work. Inconsistent choices in predicate identifier often render findings significantly incomparable, so in this paper, we advocate for reporting the predicate’s F1 score as well so that we can ensure the overall Sem-F1 increase is due to better role labeling rather than better predicate recognition. The results show not only is our model’s improvement is stable in this challenge setting, but when compared to the previous results, our results are comparable or even superior, despite using a simpler modeling method and simpler neural structures.

Where did Denoising Work? For the base model, the longer the sequence is, the more likely it is affected by noise, so we hypothesize that denoising should bring greater effect to longer sequences. To verify our hypothesis, we compared the performance of the baseline and the full model on different length sentences, as shown in Figure 2(a).

According to the figure, the baseline model’s performance is obviously better when handling short sentences, which is common in sequence-based modeling. This suggests that long sentences may be more influenced by noise. The figure also shows that our full model improves on this baseline and its improvement grows when handling longer sentences, which is consistent with our hypothesis.

Denoising, Refining, or Smoothing? Generally speaking, if we do not use synthesized noise, our approach can be thought of as refining; additionally, if we do not use dropout but instead sum noise features from distribution sampling, it can be thought of as model smoothing. We thus performed two experiments to contrast our method with these similar
with several novel techniques. To illustrate the importance of introducing these novel techniques, we performed an ablation study as shown in the bottom of Table 4. Removing the relative position encoding, two-stream attention mechanism (using two separate Transformers instead), and soft label embedding (using the embedding obtained on the argmax prediction) led to performance reductions of varying degrees. Among them, the removal of the two-stream attention mechanism affected model performance the most, which shows that the interaction between the word representations (without model noise) and the label representations (with model noise) is critical for denoising.

Furthermore, we explore the sampling size $M$ in order to determine the optimal sampling size for improving performance. Figure 2(b) depicts

### 3.3 Ablation Study

In our full model, we augmented the base model with several novel techniques. To illustrate the importance of introducing these novel techniques, we performed an ablation study as shown in Table 4. Removing the relative position encoding, two-stream attention mechanism (using two separate Transformers instead), and soft label embedding (using the embedding obtained on the argmax prediction) led to performance reductions of varying degrees. Among them, the removal of the two-stream attention mechanism affected model performance the most, which shows that the interaction between the word representations (without model noise) and the label representations (with model noise) is critical for denoising.

Furthermore, we explore the sampling size $M$ in order to determine the optimal sampling size for improving performance. Figure 2(b) depicts

### Table 2: Semantic F1-score on the CoNLL-2009 in-domain multilingual test sets with the predicate-given setting.

<table>
<thead>
<tr>
<th>Model</th>
<th>Catalan</th>
<th>Chinese</th>
<th>Czech</th>
<th>English</th>
<th>German</th>
<th>Japanese</th>
<th>Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li et al. (2020a)</td>
<td>86.90</td>
<td>88.69</td>
<td>91.93</td>
<td>91.77</td>
<td>85.54</td>
<td>85.90</td>
<td>86.96</td>
</tr>
<tr>
<td>Baseline</td>
<td>81.42</td>
<td>83.20</td>
<td>88.60</td>
<td>87.76</td>
<td>81.34</td>
<td>80.52</td>
<td>85.65</td>
</tr>
<tr>
<td>Full Model</td>
<td>82.58</td>
<td>84.56</td>
<td>89.74</td>
<td>89.83</td>
<td>79.30</td>
<td>82.53</td>
<td>81.47</td>
</tr>
<tr>
<td>Full Model (w/ BERT)</td>
<td>87.05</td>
<td>87.96</td>
<td>92.24</td>
<td>92.03</td>
<td>83.55</td>
<td>86.04</td>
<td>85.65</td>
</tr>
</tbody>
</table>

† The predicate disambiguators in (Li et al., 2020a) in w/ BERT and w/o BERT setting use the same sequence labeling model (w/ BERT) which improves the overall Sem-F$_1$, while two separate disambiguators are used in our work, so the w/o BERT results are not entirely comparable.

### Table 3: Semantic F1-score on the CoNLL-2009 multilingual test sets with the end-to-end setting.

<table>
<thead>
<tr>
<th>Model</th>
<th>Catalan</th>
<th>Chinese</th>
<th>Czech</th>
<th>English</th>
<th>German</th>
<th>Japanese</th>
<th>Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li et al. (2020a)</td>
<td>84.07</td>
<td>82.01</td>
<td>89.45</td>
<td>86.16</td>
<td>60.48</td>
<td>74.20</td>
<td>83.11</td>
</tr>
<tr>
<td>Li et al. (2020a) (w/ BERT)</td>
<td>85.82</td>
<td>85.68</td>
<td>91.22</td>
<td>88.70</td>
<td>67.15</td>
<td>78.88</td>
<td>86.00</td>
</tr>
<tr>
<td>Baseline</td>
<td>90.82</td>
<td>90.47</td>
<td>90.05</td>
<td>88.18</td>
<td>91.62</td>
<td>85.06</td>
<td>85.83</td>
</tr>
<tr>
<td>Full Model</td>
<td>90.82</td>
<td>90.47</td>
<td>90.22</td>
<td>89.50</td>
<td>91.62</td>
<td>85.98</td>
<td>85.83</td>
</tr>
<tr>
<td>Full Model (w/ BERT)</td>
<td>95.05</td>
<td>85.96</td>
<td>93.11</td>
<td>85.22</td>
<td>91.45</td>
<td>86.63</td>
<td>84.83</td>
</tr>
</tbody>
</table>

† The predicate recognizer in (Li et al., 2020a) in w/ BERT and w/o BERT setting use the same sequence labeling model (w/ BERT) which improves the overall Sem-F$_1$, while two separate recognizers are used in our work, so the w/o BERT results are not entirely comparable.

### Table 4: Ablation study on CoNLL-09 English test set.

<table>
<thead>
<tr>
<th>Method</th>
<th>ID P R F$_1$</th>
<th>OOD P R F$_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>89.06 88.54 88.80 78.57 77.10 77.83</td>
<td>10.36 77.88 10.36 77.88 10.36 77.88</td>
</tr>
<tr>
<td>Boosting Only</td>
<td>88.97 88.85 88.90 78.52 77.24 77.87</td>
<td>10.36 77.88 10.36 77.88 10.36 77.88</td>
</tr>
<tr>
<td>Full Model</td>
<td>90.66 89.01 89.83 80.35 77.46 78.88</td>
<td>10.36 77.88 10.36 77.88 10.36 77.88</td>
</tr>
<tr>
<td>w/o Noise Sampling</td>
<td>89.34 88.92 89.12 78.92 76.95 77.92</td>
<td>10.36 77.88 10.36 77.88 10.36 77.88</td>
</tr>
<tr>
<td>w/o Gaussian Noise</td>
<td>90.53 88.72 89.61 80.44 76.87 78.61</td>
<td>10.36 77.88 10.36 77.88 10.36 77.88</td>
</tr>
<tr>
<td>w/o Relative Pos</td>
<td>90.27 88.92 89.58 80.16 77.30 78.70</td>
<td>10.36 77.88 10.36 77.88 10.36 77.88</td>
</tr>
<tr>
<td>w/o Two-stream Self-att</td>
<td>89.51 88.79 89.14 79.98 76.55 78.22</td>
<td>10.36 77.88 10.36 77.88 10.36 77.88</td>
</tr>
<tr>
<td>w/o Soft Label Emb</td>
<td>90.16 88.66 89.40 80.08 77.13 78.57</td>
<td>10.36 77.88 10.36 77.88 10.36 77.88</td>
</tr>
</tbody>
</table>
the SRL performance curve with various sampling sizes $M$. The optimal sampling size, as shown in the figure, is 8. When it is less than 8, the performance improves as $M$ becomes larger. After $M$ reaches 8, the performance is essentially stable, which demonstrates that 8 is sufficient for denoising, and no additional gain will be available by increasing $M$ further. Please refer to Appendix A.2 for other ablation studies.

4 Related Work

Semantic Role Labeling SRL has been a heated research realm since the introduction of neural networks. Early neural network-based methods (Wang et al., 2015) simply modeled semantic role labeling as a word classification task and employed recurrent networks for annotation. Leveraging syntax is a common way of boosting performance for SRL. While using syntactic treebanks, Graph Convolutional Networks (GCN) can be applied for SRL for syntax-aware labeling (Marcheggiani and Titov, 2017b), though syntax-agnostic models were also argued efficient by (Marcheggiani et al., 2017b). Still, there remains a strong connection between the studies of syntax and SRL (He et al., 2018c; Marcheggiani and Titov, 2020; Shi et al., 2020).

Refinement on output from SRL models has been increasingly popular for research. Iterative refinement on SRL has been shown to outperform base models (Lyu et al., 2019b). Higher order scorers have also been used as a source of more accurate arc scores in semantic graph (Li et al., 2020b). Our model is similar to those refining models in process, but rather than just refining, our model also specifically focuses on removing noise.

Noise Processing Noise, in NLP tasks, represented in the model as uncertainty when processing complex information or structures. Denoising can be leveraged to produce better results, as high certainty implicates faults for refinement. The noise channel model, which applies Bayesian approximation constraints to eliminate noise in generated outputs, has become a popular method in NLG tasks, including NMT (Wang et al., 2019; Zhou et al., 2020b) and summary generation (Xu et al., 2020). In domain of linguistic parsing, noise refers to labels predicted with high uncertainty. (Zhang et al., 2019a) applied an adaptive uncertainty-aware decoder for semantic parsing. Dependency parsing can also benefit from adaptive strategy based on uncertainty detection, as demonstrated in (van der Goot and van Noord, 2018). Uncertainty mechanism has also been applied in suspense prediction (Wilmot and Keller, 2020), spoken language assessment (Malinin et al., 2017), and document class prevalence inference (Keith and O’Connor, 2018).

Specifically speaking, mainstream noise processing can be categorized into two topics: evaluation and elimination. Noise elimination generally refers to he works introduced above that discuss about result refinement, while noise evaluation refers to modeling the uncertainty of a model. In this topic, Gai and Ghahramani (2016b) have suggested that the softmax function may not be a solid indication of model uncertainty, which suggests that this topic needs more research. Also on this topic, (He et al., 2020) rectified confidence scores using their MSD model to better evaluate result uncertainty, and Ethayarajh (2020) measured the bias in classification models using Bernstein-bounded unfairness.

5 Conclusion

In this paper, we propose a noisy channel model for the SRL model’s inherent noise problem. In our proposed model, synthesized noise is combined and then averaged to best emphasize similar authentic noise while weaken the different inherent noise, and this noise is then reduced by a denoiser based on two-stream attention to obtain the final output. We demonstrated the effectiveness of our approach by evaluating our models on the CoNLL-09 multilingual benchmark, and we also investigated the differences between our method and refining and smoothing techniques. Apart from being successful in SRL, our approach is also notable because it broadly applicable to other NLP tasks.
References


Hao Fei, Fei Li, Bobo Li, and Donghong Ji. 2021. Encoder-decoder based unified semantic role labeling with label-aware syntax.


Hai Zhao, Wenliang Chen, Jun’ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features...


A Appendix

A.1 Hyper-parameters

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Embed</td>
<td>100</td>
</tr>
<tr>
<td>Char</td>
<td>100</td>
</tr>
<tr>
<td>POS</td>
<td>64</td>
</tr>
<tr>
<td>Lemma</td>
<td>100</td>
</tr>
<tr>
<td>Label</td>
<td>400</td>
</tr>
<tr>
<td>Predicate Indicator</td>
<td>32</td>
</tr>
<tr>
<td>ELMo(^\d)</td>
<td>300</td>
</tr>
<tr>
<td>BERT(^\d)</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Encoder Size</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BiLSTMs</td>
<td>256 $\times$ 2</td>
</tr>
<tr>
<td>BiLSTMs Layers</td>
<td>3</td>
</tr>
<tr>
<td>BiLSTMs Out MLP</td>
<td>128</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Two Stream Attention</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer Hidden</td>
<td>512</td>
</tr>
<tr>
<td>Transformer FFN</td>
<td>1024</td>
</tr>
<tr>
<td>Transformer Heads</td>
<td>8</td>
</tr>
<tr>
<td>Transformer Layers</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dropout Probability</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BiLSTM Input</td>
<td>0.33</td>
</tr>
<tr>
<td>BiLSTM Output</td>
<td>0.33</td>
</tr>
<tr>
<td>BiLSTMs</td>
<td>[0.33, 0.33]</td>
</tr>
<tr>
<td>Transformers</td>
<td>0.05</td>
</tr>
<tr>
<td>Noise</td>
<td>0.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimizer</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>0.001</td>
</tr>
<tr>
<td>Adam $\mu$</td>
<td>0.9</td>
</tr>
<tr>
<td>Adam $\nu$</td>
<td>0.999</td>
</tr>
<tr>
<td>Batch Size</td>
<td>64</td>
</tr>
<tr>
<td>Decay Rate</td>
<td>0.999995</td>
</tr>
<tr>
<td>Warmup Steps</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 5: Model hyper-parameters. \(^\d\) denotes optional.

A.2 Noisy Channel Model on BERT Baseline

The main experimental results show that BERT can significantly boost the performance of SRL. To demonstrate that our approach will continue to work on the strong BERT baseline, we present the w/ BERT baseline results in Table 6. The results show that, while BERT is a great help to the baseline performance, the use of our Noisy Channel Model can further play a useful role.

<table>
<thead>
<tr>
<th>Method</th>
<th>P</th>
<th>R</th>
<th>F(_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>89.06</td>
<td>88.54</td>
<td>88.80</td>
</tr>
<tr>
<td>Baseline (w/ BERT)</td>
<td>91.97</td>
<td>91.23</td>
<td>91.59</td>
</tr>
<tr>
<td>Full Model</td>
<td>90.66</td>
<td>89.01</td>
<td>89.83</td>
</tr>
<tr>
<td>Full Model (w/ BERT)</td>
<td>92.11</td>
<td>91.95</td>
<td>92.03</td>
</tr>
</tbody>
</table>

Table 6: Performance comparison between baseline and noisy channel model with BERT enhancement.

A.3 Inference Speed Analysis

To analyze the inference speed of different modeling approaches, we measured the total inference time on the CoNLL-2009 English in-domain test set with the scale of model parameters similar (i.e., same hidden size, model layers). 5 runs are performed and then reported the average speed for better stability. The comparison results are shown in Table 7. From the comparison, the inference speed order is Sequence $>$ Tree $>$ Graph, and our full model only slightly decreases the speed due to a good parallel design compared to the baseline.

<table>
<thead>
<tr>
<th>Method</th>
<th>Modeling</th>
<th>Speed (sent./s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Cai et al., 2018)</td>
<td>T</td>
<td>199.5</td>
</tr>
<tr>
<td>(Li et al., 2020a)</td>
<td>G</td>
<td>165.9</td>
</tr>
<tr>
<td>Baseline</td>
<td>S</td>
<td>245.1</td>
</tr>
<tr>
<td>Full Model</td>
<td>S</td>
<td>240.6</td>
</tr>
</tbody>
</table>

Table 7: Inference speed for different modeling approaches.