Spring 5-15-2020

Predicate Informed Syntax-Guidance for Semantic Role Labeling

Sijia Wang
Washington University in St. Louis

Washington University in St. Louis

School of Engineering and Applied Science

Department of Computer Science and Engineering

Thesis Examination Committee:
Brendan Juba
Michael Brent
Ayan Chakrabarti

Predicate Informed Syntax-Guidance for Semantic Role Labeling

by

Sijia Wang

A thesis presented to the Graduate School of Arts and Sciences
of Washington University in partial fulfillment of the
requirements for the degree of

Master of Science

May 2020
Saint Louis, Missouri

Contents

List of Tables

List of Figures

Acknowledgments

Foremost, I would like to express my deepest gratitude to my advisor Prof. Brendan Juba for the continuous support of my study and research with his immense knowledge, enthusiasm, and patience. His guidance helped me during the period of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master's study. Without his persistent help, the goal of this project would not have been realized.

I would like to extend my sincere thanks to the rest of my thesis committee: Prof. Michael Brent and Prof. Ayan Chakrabarti, for their insightful comments and suggestions of new perspectives for future research.

I'm extremely grateful to the Ph.D. candidate Zihao Deng with whom I collaborated on this project, for the illuminating discussions, the collaboration on this project, and the sleepless nights we were working together before deadlines. The completion of my thesis would not have been possible without the support of Zihao.

A special thanks goes to my friend Leiwei Yao at the Hong Kong University of Science and Technology, Zhuosheng Zhang at the Shanghai Jiao Tong University, Peng Shi at the University of Waterloo, and many graduate students and distinguished faculty within my department who have reviewed this thesis and helped support the related research.

Finally, I wish to thank my parents for their support and encouragement throughout my studies.

Sijia Wang

Washington University in Saint Louis
May 2020

Dedicated to my parents.

ABSTRACT OF THE THESIS

Predicate Informed Syntax-Guidance for Semantic Role Labeling

by

Sijia Wang

Master of Science in Computer Science

Washington University in St. Louis, May 2020

Research Advisor: Professor Brendan Juba

In this thesis, we consider neural network approaches to the semantic role labeling task in semantic parsing. Recent state-of-the-art results for semantic role labeling are achieved by combining LSTM neural networks and pre-trained features. This work offers a simple BERT-based model which shows that, contrary to the popular belief that more complexity means better performance, removing LSTM improves the state of the art for span-based semantic role labeling. This model has improved F1 scores on both the test set of CoNLL-2012, and the Brown test set of CoNLL-2005 by at least 3 percentage points.

In addition to this refinement of existing architectures, we also propose a new mechanism. There has been an active line of research focusing on incorporating syntax information into the attention mechanism for semantic parsing. However, the existing models do not make use of which sub-clause a given token belongs to or where the boundary of the sub-clause lies. In this thesis, we propose a predicate-aware attention mechanism that explicitly incorporates the portion of the parsing spanning from the predicate. The proposed Syntax-Guidance (SG) mechanism further improves the model performance. We compare the predicate informed method with three other SG

mechanisms in detailed error analysis, showing the advantage and potential research directions of the proposed method.

Chapter 1

Introduction

Given a sentence, semantic role labeling (SRL) is the task of identifying semantic roles in that sentence based on the theme of the predicate. For example, in the sentence `John told Pat to cut off the tree`, when `told` is the predicate, `John` is the first argument or performer of the action of telling, `Pat` is the receiver of this order, and the sub-clause `cut off the tree` is the order that is given. When `cut` is the predicate, `Pat` is the first argument, or performer of cutting, and `the tree` is the receiver of cutting. SRL is a fundamental task in natural language processing (NLP), and it has been shown to be useful in other challenging NLP tasks such as question answering (Shen and Lapata, 2007), and machine reading (Berant et al., 2014; Wang et al., 2015).

1.1 Current State-of-the-art Approaches to Semantic Role Labeling

Since the introduction of LSTM neural networks to SRL by Zhou and Xu (2015), He et al. (2017), and Marcheggiani et al. (2017), most recent methods for SRL employ LSTM to model the relationships between predicates and arguments. SRL is then solved by performing inference in this model. The motivation for using LSTM is that SRL can be seen as a sequential labeling task and LSTM is a powerful model for such tasks in general. However, LSTM models are usually hard to train due to their sequential computation that cannot be parallelized. Moreover, LSTM has the inherent problem of gradients vanishing over long sequences (Hochreiter et al., 2001; Pascanu

et al., 2013) and it requires a high amount of memory bandwidth Appleyard et al. (2016). If not well-trained, LSTM models can even lead to worse performance.

Pre-trained feature models have received increasing interest in NLP over the past few years. BERT (Devlin et al., 2019) is one of the most commonly used pre-trained models. BERT has led to significant improvements in multiple NLP benchmarks, hence it has been widely employed in many recent NLP models. A popular way of leveraging these pre-trained models to get better results in SRL is to use their outputs as the input features for the LSTM-based model. For example, in (Shi and Lin, 2019), an input sentence is fed into a BERT encoder, whose outputs are then used as contextual word embeddings for a one-layer BiLSTM. This approach obtained state-of-the-art performance in many SRL benchmarks. However, because they obtained their results by combining self-attention and LSTM models, it is hard to tell the extent to which these two components each contributed.

It's natural to expect that knowledge of the syntactic structure of the sentence should aid improving the solution to the SRL task. However, it's not obvious how to utilize such information in neural network models. Self-attention with syntactic guidance (SG) provides one possible mechanism for this purpose. Self-attention models the correlation between tokens in an input sequence regardless of their distance, allowing the encoder, a.k.a. attention head, of one token to attend to other tokens and draw information from them. SG self-attention restricts the attention head to attend to only the syntactically relevant tokens in a sentence, while neural nets with plain self-attention are left on their own to learn which are more important to attend to out of all the tokens in a sentence. Such mechanisms have been studied by Strubell et al. (2018) and Zhang et al. (2020). Strubell et al. (2018) trained an additional type of attention head that attends to the parent node in the dependency parsing tree when predicting the semantic role label of a token, and Zhang et al. (2020)[1] extended this mechanism so that for each token it attends to all the ancestor nodes on the path tracing back to the root in the dependency tree.

[1]This SG mechanism has only been used in Question Answering, but can be considered for SRL.

1.2 Overview of Our Contributions and Organization of This Thesis.

We introduce a new type of SG self-attention mechanism, where we train an attention head to attend to all the ancestor tokens in the syntax sub-tree spanning from the predicate. This sub-tree is essentially the parsing of the predicate's sub-clause. We call this mechanism *predicate-informed syntax guidance*. Since such attention attends to more syntactically relevant tokens, it holds more information than that proposed by Strubell et al. (2018), and since it does not attend to tokens outside the predicate sub-clause it is less noisy than that proposed by Zhang et al. (2020). We implemented a model using our new self-attention method for the SRL task, and evaluated it on the CoNLL-2005 and 2012 tasks against models using the other forms of SG self-attention. The results of these experiments show that the predicate-informed SG self-attention achieves the new state-of-the-art for these data sets.

In this thesis, we first review ingredients of the proposed model in chapter 2. We propose the baseline BERT + CRF model, which replaces LSTM with CRF compared with Shi and Lin (2019). It improves the F1 score by 3 percentage points on both CoNLL-2005 and 2012. The detailed model description and error analysis is in chapter 3. We further propose a enhanced model by adding the SG attention to the baseline BERT + CRF model, named the SG + BERT + CRF model. In chapter 4, we compare four SG attention mechanisms, and conclude that the predicate informed SG attention has the best performance. With the predicate informed SG attention, F1 increases by at least 1.5 in comparison with the baseline BERT + CRF model. All together, our model beats the previous published state-of-the-art by 4.5 percentage points in F1.

1.3 Other Task Formulations

For semantic role labeling, there are other two kinds of tasks that are widely explored, all three meant to extract the predicate-argument structure. The first one differs from our setting in that the argument span is given. The performance is intrinsically better with known span (He et al., 2017), in a sense with richer prior knowledge. Other than the fact that argument identification is an unsolved task (Shi and Lin, 2019), for SRL benchmarks such as CoNLL 2005, 2009, and 2012,

the span is not given during both training and testing. Thus, in this thesis, we only discuss the case when the argument span is unknown. In the work of (Ouchi et al., 2018), they split the task into two steps. They first select all possible argument spans and then assign labels on the span with highest label scores. Such strategy allows them to use span-level features, however, additive error comes along for the two steps.

The second one differs in that the predicate is not given. Due to the argument ambiguity in SRL, which will be discussed in Section 2.1.1, we know that the tokens will have different labels given distinct predicates. It will be complicated when the predicate is not given. On the other hand, predicate prediction is a task to identify the verb either in a clause or a sentence. Yet we intend to focus on the predicate-argument task. Thus we follow the task formulation of CoNLL-2005 (Carreras and Màrquez, 2005) and CoNLL-2012 (Pradhan et al., 2012), where the predicate is given.

Chapter 2

Task Formulation and Model Components

We propose two neural network models, one baseline model named BERT + CRF, and one with syntax-guidance, SG + BERT + CRF. As their name suggest, the model BERT + CRF consists of two major components, a pre-trained language representation model BERT and a Conditional Random Field (CRF). And SG + BERT + CRF has three major components, a Syntax Guidance layer along with BERT and CRF. In this chapter, we first formulate the task in Section 2.1 and review details of three key ingredients of both model architectures in the following sections.

Bidirectional Encoder Representations from Transformers (BERT) is a language representation model that has been widely explored in Natural Language tasks. Upon release, BERT achieve a significant boost in a variety of applications. We will explain BERT contextual embedding in more detail in Section 2.2. To emphasise the importance of predicate to the argument identification, we use the Predicate Indicator Embedding to enhance the contribution of the predicate to the token embedding, which is in Section 2.3. Conditional Random Field (CRF) is a simple and classic probabilistic model which is able to capture sequential structure information. Thus it is widely used for sequential tasks. CRF is much less complex than LSTM so it can generalize better than LSTM and is easier to train. Moreover CRF does not suffer the vanishing gradient problem of LSTM (Zheng et al., 2015). The details are in Section 2.4

2.1 Task Formulation

We focus on the task of argument identification and classification. Given a sentence and a predicate, this task is to identify the spans of all the arguments for the given predicate and classify each

argument as the corresponding semantic role. Different predicates are parsed differently. For example, in the sentence `John told Pat to cut off the tree`, for the predicate `told`, the arguments are `John`, whose role is giver of the order; `Pat`, whose role is the receiver of the order; and `cut off the tree`, whose role is the order that is given. For the predicate `cut`, the arguments are `Pat`, whose role is the one that does the cutting, and `the tree`, whose role is to receive the cutting. This is suitable for the benchmark data sets CoNLL-2005 and CoNLL-2012, where predicates are given in the training set and test set along with the sentences.

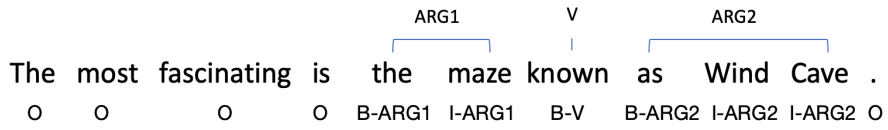


Figure 2.1: The above example illustrates the concept of argument span and the BIO labeling mechanism. Given the predicate `known`, `the maze` belongs to ARG1, `as Wind Cave` to ARG2. Within the term `the maze`, `the` is labeled with B-ARG1, which denotes the beginning of ARG1. `maze` is labeled with I-ARG1, which denotes the inside of ARG1. The rest are out of the span, labelled with O.

We treat SRL as a sequential BIO tagging problem, where the span information is encoded by the BIO prefixes in the tags. The prefix "B-" means the beginning of the span, "I-" means in the span, and "O-" means out of the span. This task is shown in Fig. 2.1. The goal is the assign each word to the correct semantic role label.

2.1.1 Argument Ambiguity in SRL

A key observation for SRL tasks is the argument ambiguity of target words given different predicates, that is, given different predicates, a word could bear distinct tags. For instance, in the following sentence in (He et al., 2017), `John told Pat to cut off the tree`. The person named `Pat` carries two distinct tags `A2` and `A0`, when given two predicates `told` and `cut off` respectively.

```
I) John told Pat to cut off the tree.
Predicate:  told(1)
A0:  John
V:   told
```

```
A2:  Pat
A1:  to cut off the tree


II) John told Pat to cut off the tree.
Predicate:  cut(4)
A0:  Pat
V:   cut off
A1:  the tree
```

To deal with the argument ambiguity problem, we will incorporate the predicate indicator in our model by converting the indicator sequence into an indicator embedding. Then we concatenate the predicate indicator embedding following the previous work of (He et al., 2017) with BERT contextual representations.

2.2 BERT Contextual Embedding

The training of a BERT model consists of two steps, pre-training and fine-tuning. During the pre-training step, the model is trained on unlabeled data over different tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and the parameters are fine-tuned using labeled data from the downstream tasks (Devlin et al., 2019). In this thesis, we use the pre-trained BERT model to generate the contextual embeddings, and then fine-tune the parameters with the labeled SRL data.

In order to add contextual information to the features, we use BERT contextual embeddings. Given a sequence of tokens as input we map each token to an embedding vector. To do that, we input the token sequence to the BERT model and it outputs a sequence of embedding vectors. We use "`[CLS]` sentence `[SEP]` predicate `[SEP]`" as the input format to the BERT model. This format was used in Shi and Lin (2019). There are two reasons for using it. First, the BERT model was pre-trained using the format with concatenated special tokens `[CLS]` and `[SEP]` so it fits the model better; second, it has the benefit of producing predicate-aware contextual embeddings for each word token. After getting the outputs of BERT, we only collect the embeddings of words in the

sentence for further computations. This is illustrated in the lower layer of the model architecture in Fig. 3.1.

The final output of BERT assigns a contextual embedding vector to each token. Note that BERT does not simply assign an embedding vector for each word in the sentence. Upon receiving the input, BERT uses the WordPiece tokenizer to split the input text into a sequence of tokens. Traditional tokenizers break the text into a sequence of words by splitting at the whitespace characters in the sentence. The WordPiece tokenizer sometimes breaks one word into multiple tokens. For example, the WordPiece tokenizer will split the sentence `here is the sentence I want embeddings for` into `here`, `is`, `the`, `sentence`, `I`, `want`, `em`, `##bed`, `##ding`, `##s`, and `for`. This is because it has a fixed sized vocabulary of tokens. So if a word does not belong to the existing vocabulary then the tokenizer will break it down into sub-tokens that are in the vocabulary.

We need to use the WordPiece tokenizer because it helps our model generalize beyond the training set. Suppose a word in the test set has not been seen in the training set. A traditional method will assign this word a randomly generated embedding vector. But this causes the vector to lose information about the new word. A model with the WordPiece tokenizer instead decomposes the word into sub-tokens in the vocabulary and uses the embedding vectors of these sub-tokens to create an embedding vector for the whole word. This is similar to recognizing a word through its roots. Since the embedding vectors of these sub-tokens have already been trained, they are informed rather random. However, since the WordPiece tokenizer can break one word into multiple sub-tokens, we need to find a way to get a single embedding vector for such a word from the multiple sub-token embeddings. If not, this creates an inconsistency between the number of tokens and labels. One can get such a single embedding by averaging the sub-token embeddings, but according to the suggestion by Devlin et al. (2019), we simply choose the embedding vector of the first sub-token as the embedding vector for the whole word.

Other than BERT, DistilBERT is a lighter version of BERT that serves the same purpose. Specifically, it reduce the size of parameters by 40%, while retaining 97% of its language understanding capabilities and is 60% faster (Sanh et al., 2019). In Section 3.2, we will show experimental performance for both BERT and DistilBERT.

2.3 Predicate Indicator Embedding

After obtaining the contextual embeddings for each word in the sentence, we now concatenate each embedding with a predicate indicator embedding.

Note that even though we have included the predicate word in the input to BERT so that we obtain a predicate-aware contextual embedding, we have not encoded the position information of the predicate. For instance, suppose we are given the sentence `I saw a saw saw a saw` and the predicate `saw` which is the second word in the sentence. The input of form `[CLS] I saw a saw saw a saw [SEP] saw [SEP]` can confuse the model because `saw` also appears in three other places in the sentence, where it has different meanings.

To include the position information of the predicate, we use a predicate indicator embedding (He et al., 2017) and concatenate it with the contextual word embeddings. Formally, we create an embedding layer of two vectors, $\mathbf{v}_0$ and $\mathbf{v}_1$. For each word token we assign it $\mathbf{v}_1$ if it is a predicate, and $\mathbf{v}_0$ if otherwise. We then concatenate the predicate indicator embeddings with the contextual word embeddings from BERT, and pass the concatenated embeddings to further computation.

2.4 CRF

The concatenation of the embedding and attention is used in a CRF layer to produce the emission score. Let $[\mathbf{A}, \mathbf{E}] \in \mathbb{R}^{n \times 2m}$ denote the concatenation. We pass it through a linear transformation layer which outputs a matrix $\mathbf{P} \in \mathbb{R}^{n \times k}$, where k is the target set size. After the linear transformation, the jth entry of the the ith vector $\mathbf{P}_i$ can be interpreted as the log probability of word i being labeled as the jth semantic role, i.e., the emission score $\mathbf{P}(w_i, y_j)$.

Suppose we are given a sequence of tokens $\mathbf{W} = \{w_1, w_2, ..., w_m\}$, and a sequence of the corresponding true labels $\mathbf{y} = \{y_1, y_2, ..., y_m\}$ from the training set. Following (Ma and Hovy, 2016) we use the score below to measure the likelihood of the labels $\mathbf{y}$ given $\mathbf{W}$:

$$score(\mathbf{y}|\mathbf{W}) = \sum_{i=1}^{m} \mathbf{P}(w_i, y_i) + \sum_{i=2}^{m} \mathbf{T}(y_{i-1}, y_i) \tag{2.1}$$

where $\mathbf{P}$ is the emission matrix, and $\mathbf{T}$ is the transition matrix in which the $\mathbf{T}(y_{i-1}, y_i)$ denotes the transition score of labeling y_i after y_{i-1}. We then model the probability of the label sequence $\mathbf{y}$ given the sequence $\mathbf{W}$ as

$$p(\mathbf{y}|\mathbf{W}) = \frac{1}{Z} \exp(score(\mathbf{y}|\mathbf{W})) \tag{2.2}$$

where $Z = \sum_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{y}|\mathbf{W})$ is the normalizing factor, which can be computed dynamically, and $\mathcal{Y}$ is the set of all possible label sequences. The negative logarithm of this probability is used as a loss for training.

When the training is complete, a Viterbi decoder is used to predict the labels for sentences in the test set. It finds the label sequence $\hat{\mathbf{y}}$ that maximizes the joint score $score(\mathbf{y}|\mathbf{W})$. Training the transition matrix $\mathbf{T}$ is computationally efficient since all we need to learn are k^2 parameters where k is the size of role label set.

Chapter 3

The BERT + CRF Model

In this chapter, we first introduce the baseline BERT + CRF model in Section 3.1. The baseline model follows (Shi and Lin, 2019), with a replacement of LSTM with CRF. The experiment results are in Section 3.2. The error analysis mainly focuses on two types of confusion, A0/A1 confusion and A2 confusion, which will be discussed in Section 3.3

3.1 Model Description

The training process of the baseline model BERT + CRF is as follows.

Given a sentence $\boldsymbol{W} = \{w_1, w_2, ..., w_m\}$

1. Tokenize $\boldsymbol{W}$, some of the tokens might break down into several sub-tokens.
2. Obtain token embeddings through BERT. Pick embedding for the first sub-token as the embedding for the token, $\boldsymbol{E}' = \{\boldsymbol{e}'_1, \boldsymbol{e}'_2, ..., \boldsymbol{e}'_m\}$, where $\boldsymbol{e}'_j \in \mathbb{R}^{d_{BERT}}$ and d_{BERT} is the embedding dimension.
3. Concatenate the contextual embedding $\boldsymbol{e}'_j$ with the predicate embedding $\boldsymbol{p}_j \in \mathbb{R}^{d_p}$. Then

$$\boldsymbol{e}_j = \boldsymbol{e}'_j \oplus \boldsymbol{p}_j, \tag{3.1}$$

   $\boldsymbol{e}_j \in \mathbb{R}^{d_{BERT}+d_p}$. Put $\boldsymbol{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_m\}$.

4. Pass $\mathbf{E}$ to a linear layer to make the dimension fit the token size.

$$\boldsymbol{L} = C\boldsymbol{E} \tag{3.2}$$

5. Compute loss based on (2.1) and (2.2) and propagate backward. Repeat three epochs.

6. Decoding. Use the Viterbi algorithm to get predictions.

We pick the embeddings after passing to BERT instead of at Step 1 because we believe that the subsequent tokens contain information that can be encoded by BERT. Figure 3.1 shows the baseline model architecture.

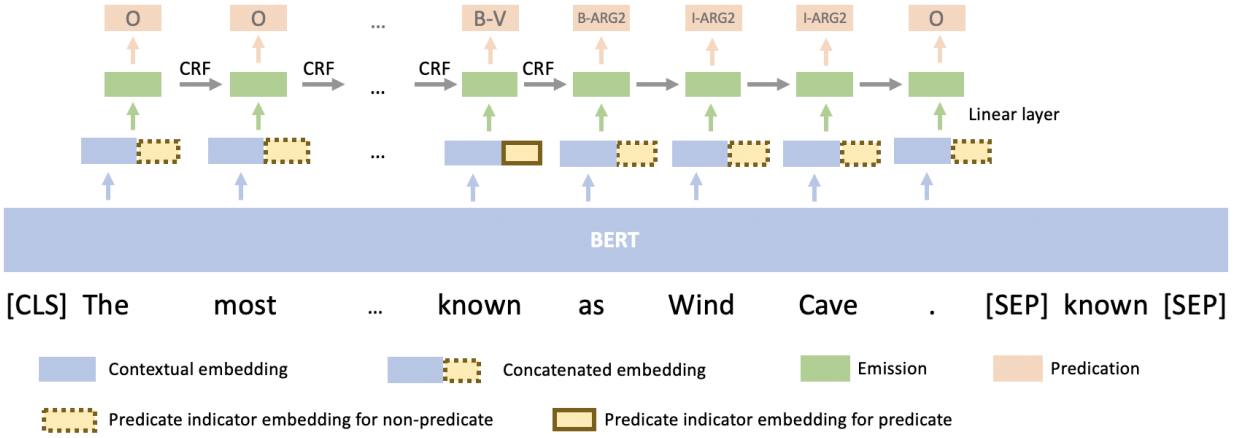


Figure 3.1: In this model, we first obtain BERT contextual embeddings for the word tokens in the input sentence. Then we concatenate these embeddings with predicate indicator embeddings. Finally we pass these to a CRF.

3.2 Performance Evaluation

The trainable parameters that we introduce in this model are predicate indicator embeddings, the attention head vectors, the linear layer matrix, and the transition matrix in a Conditional Random Field (CRF). During training, we fine-tune the parameters of BERT.

For BERT + CRF, we experimented using both BERT and DistilBert. DistilBert (Sanh et al., 2019) is a smaller version of BERT, which retains most of the performance of the original version. For

Model	**F1**(WSJ)	**F1**(Brown)
LISA (Strubell et al., 2018)	86.0	78.3
Joint predication + ELMo (He et al., 2018)	87.4	80.4
ELMo SPAN (ensemble) (Ouchi et al., 2018)	88.5	79.6
BERT + LSTM (Shi and Lin, 2019)	87.2	82.0
BERT-CRF	**91.2**	**85.4**
DistilBERT-CRF	89.1	84.8

Table 3.1: Experimental results on CoNLL-2005 in terms of F1.

Model	**F1**
LISA (Strubell et al., 2018)	83.4
Joint predication + ELMo (He et al., 2018)	85.5
ELMo SPAN (ensemble) (Ouchi et al., 2018)	87.0
BERT + LSTM (Shi and Lin, 2019)	86.5
BERT-CRF	**90.2**
DistilBERT-CRF	89.7

Table 3.2: Experimental results on CoNLL-2012 test set in terms of F1

both BERT and DistilBert we use the base uncased version. For BERT-CRF, the batch size is 25 and for DistilBERT-CRF, 100. Our learning rate is 0.0001. We use 4 GPUs for training. For the CoNLL-2005 data set, the running times are 3 hours and 8 hours, using DistilBERT-CRF and BERT-CRF, respectively. For CoNLL-2012, the running times are 10 hours and 17 hours with DistilBERT-CRF and BERT-CRF, respectively.

The results of our BERT/DistilBERT-CRF model for span-based SRL is shown in Tables 3.1 and 3.2. BERT-CRF has an improvement of 3.4 and 3.2 percentage points respectively on the CoNLL-2005 and CoNLL-2012 data sets over the previous state-of-the-art. By the Chernoff bound, the 95% statistical significance thresholds for the three data sets are 1.33, 4.75 and 0.77. Thus BERT + CRF has significantly better performance in terms of F1 on CoNLL-2005 WSJ data set and CoNLL-2012 test set. The detailed calculation is in Appendix.

Since the only change we made to the model of (Shi and Lin, 2019) was to remove the LSTM and we obtained a significant boost in the performance, we see that LSTM was indeed harmful for this approach to span-based SRL. We suspect that this may be due to the fact that LSTM is much

harder to optimize than a CRF model. While it is certainly possible that an unconventional use of LSTM can achieve better performance, we can't rule out all possible training methods. What we argue here is simply that the standard approach is not effective. Rather, it's actually harmful.

3.3 Error Analysis

With the experimental results reported in Section 3.2, we wonder where potential scope for further improvement lies. We will mainly focus on the errors due to the lack of syntactic information in the baseline BERT + CRF model, which could be addressed with SG, and the comparison with previous methods in the performance of a particular error type.

Though the BERT + CRF model outperforms the previous state-of-the-art by a considerable amount, we note the following weakness of the model, illustrated by the mislabeled example in Figure 3.2. The green labels above the given sentence are true labels, and red ones are predictions by our model. Note that the last word `there` is mislabeled. Its true label, B-ARGM-LOC, indicates `there` should be a location argument of the verb `is`. It seems obvious to us that `a little out of context` and `there` belong to separate spans. We suspect that this error may be due to a lack of knowledge of this syntactical information. Therefore, if we can incorporate such syntactic knowledge via parsing, then we can potentially avoid such errors.

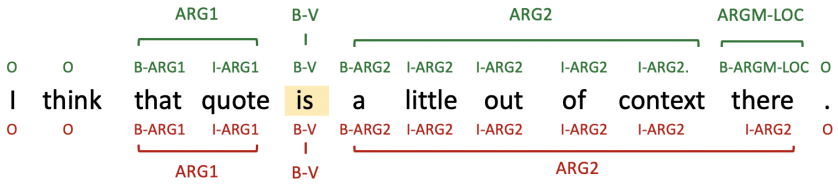


Figure 3.2: A mislabeled example. The green labels are true labels, and the red ones are predictions. The predicate is the token `is`.

According to (Ouchi et al., 2018) and (He et al., 2017), there are two major types of labeling confusion. The first type is between A0 and A1 due to ergative verbs. For instance in the sentence

`Worse, Congress has started to jump on the Skinner bandwagon`

When `started` is the predicate, `Congress` should be labeled as A1, meaning `thing starting`; when `jump` is the predicate, `Congress` should be labeled as A0, meaning `causer of jumping`. Since such differences are subtle, it's hard for an SRL model to tell them apart.

The second type of confusion is between A2 and DIR or LOC. According to (He et al., 2017) these confusions can arise due to the use of A2 in many verb frames to represent semantic relations such as direction or location. We computed the labeling confusion matrix of our model, following (He et al., 2017) and (Ouchi et al., 2018), in Table 3.4. We observed that even though these two kinds of confusion still appear to be the most prominent, which is similar to previous works, as seen in Table 3.3, their rate has decreased significantly (decreased by $> 30\%$ for A0 and A1 confusion and by $> 10\%$ for A2 confusion).

Pred. \ True	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	55	11	13	4	0	0	0	0	0
A1	78	-	46	0	0	22	11	10	25	14
A2	11	23	-	48	15	56	33	41	25	0
A3	3	2	2	-	4	0	0	0	25	14
ADV	0	0	0	4	-	0	15	29	25	36
DIR	0	0	5	4	0	-	11	2	0	0
LOC	5	9	12	0	4	0	-	10	0	14
MNR	3	0	12	26	33	0	0	-	0	21
PNC	0	3	5	4	0	11	4	2	-	0
TMP	0	8	5	0	41	11	26	6	0	-

Table 3.3: Confusion matrix from Joint predication + ELMo (He et al., 2017)

Our introduction of syntactic information was motivated by a desire to further reduce the second type of confusion, which has decreased less. The error sentence,

`I think that quote is a little out of context there`

exemplifies such confusion.

When `is` is the predicate, `there` can either indicate the location where the `quote` is `a little out of context` or modify `context`. In the first scenario, `there` should be labeled LOC. But in the second, `there` wrongly attaches to `context`, so it's labeled inside the span of label A2. Such confusion is related to prepositional phrase (PPs) attachment error, a well-known linguistic ambiguity (Kummerfeld et al., 2012). We notice that this can be fixed by inserting the correct syntactic parsing: if a reliable parser can correctly link `there` to the predicate `is` instead of `context`, as done by our syntax-guided attention, for example, then it can help the SRL model clear such confusion.

Pred. \ True	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	16	4	13	0	0	0	1	0	2
A1	43	-	34	18	26	26	24	13	57	31
A2	14	28	-	49	13	39	22	18	10	3
A3	0	3	4	-	6	15	0	22	15	0
ADV	21	7	2	0	-	0	14	26	13	33
DIR	0	0	3	0	0	-	6	0	0	0
LOC	4	15	14	2	5	6	-	8	2	16
MNR	3	8	13	11	27	0	17	-	0	12
PNC	0	5	18	0	1	0	8	7	-	0
TMP	10	14	3	3	16	13	5	3	0	-

Table 3.4: Confusion matrix for BERT + CRF

Chapter 4

The SG + BERT + CRF Models

We now improve the model by incorporating information about syntactic dependencies. For each sentence, the syntax dependency tree is generated by another pre-trained neural model from (Zhou and Zhao, 2019). We do not use the golden standard parsing trees from data sets such as Penn Tree Bank because even if the golden standard parsing is available for each sentence in the CoNLL-2005 and 2012 data sets, they might not be available in practice. So it's more realistic to use a parsing tree that is predicted at test time. The key ingredient is the SG attention head, so the comparisons in the experimental results and the error analysis will mainly focus on the methods for SG attention.

This chapter consists of four parts. We first introduce the SG model in Section 4.1, followed by the SG + BERT + CRF model description in Section 4.2. The experiment results for the four mechanisms proposed in Section 4.1 are listed in Section 4.3. We discuss their performance in Section 4.4.

4.1 The Syntax Guided (SG) Model

Syntax Guided (SG) attention is a mechanism where each token attends only to syntactically related tokens, in contrast to general self attention." We propose four types of SG attention. We will show how to generate their dependency matrices in Section 4.1.1 and how to calculate the corresponding attention in Section 4.1.2.

4.1.1 SG Dependency Matrix

To obtain the syntax-guided attention, we need a matrix to indicate which words to attend to. A standard approach to encode syntactic dependencies is through a parsing tree. An example is shown in Figure 4.1. In this thesis, we investigate four mechanisms using the syntactic dependency information: *full-tree informed*, *parent informed*, *predicate informed*, and *full-subclause predicate informed*.

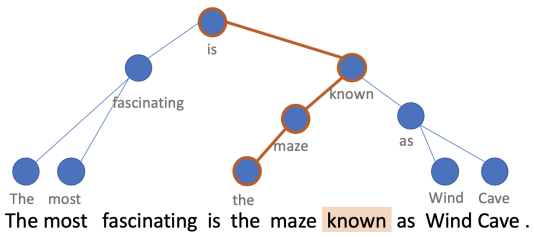
Given a sequence of words $\mathbf{W} = \{w_1, w_2, ..., w_n\}$, we generate a dependency array $\mathbf{H} = [h_1, h_2, ..., h_n]$ where h_i indicates the head (parent) word of w_i. For instance, in Figure 4.1, `fascinating` is the head of `The` and `most`, and likewise `is` is the head of `fascinating`. Thus we have $h_1 = h_2 = 3$ and $h_3 = 4$.

Given the dependency array, we first construct trees as shown in Figures 4.2a-4.2b. Figures 4.3a-4.3b illustrate the corresponding dependency matrices $\mathbf{I}$, where $\mathbf{I}_{ij} = 1$ denotes that w_j is a word that w_i should attend to. Note that in contrast to (Zhang et al., 2020), we set the diagonal entries $\mathbf{I}_{ii}$ to be 0. Since we will concatenate a word's own embedding vector with its syntax-guided attention vector, it would be redundant for them to attend to themselves here.

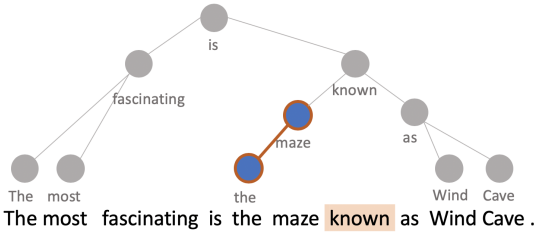
Here is the description of the dependency matrix for each one of them:

- The Full-tree informed dependency matrix:
  For each word w_i we set $\mathbf{I}_{ij} = 1$ for all its ancestors w_j from its adjacent parent to the most remote ancestor, which is the root of the dependency tree.
- Parent informed dependency matrix:
  We only set $\mathbf{I}_{ij} = 1$ when w_j is the adjacent/direct parent of w_i.
- Predicate informed dependency matrix:
  For each word w_i we set $\mathbf{I}_{ij} = 1$ for all its ancestors w_j from its adjacent parent up to the predicate.
- Full-subclause predicate informed dependency matrix:
  We set $\mathbf{I}_{ij} = 1$ if w_i and w_j are in the same subclause rooted with the predicate and $i \neq j$

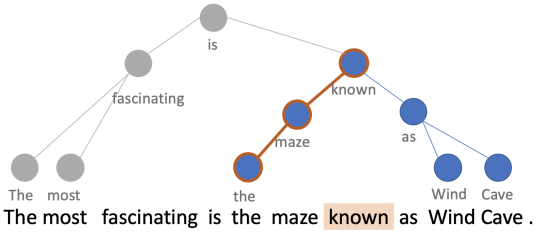
Figure 4.1: Parsing tree of the example sentence



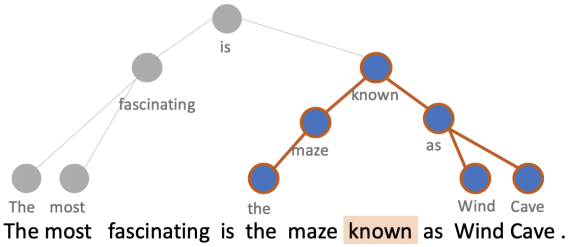
(a) Dependency tree of the full-tree informed SG. Each node attends to all its ancestors all the way to the root node.



(b) An illustration of Strubell's attention mechanism, which only attends to the adjacent parent



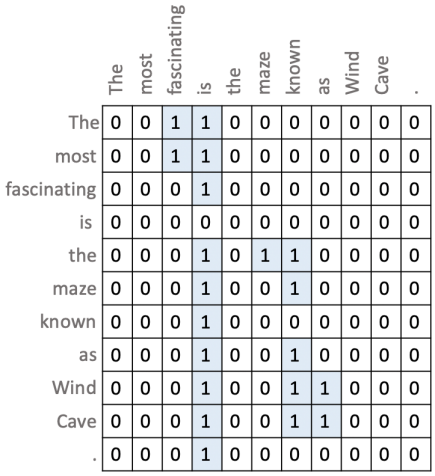
(c) Dependency tree of predicate informed SG with respect to the predicate `known` (subtree in blue). Each node attends to all its ancestors all the way to the predicate node.



(d) Dependency tree of full-subclause predicate informed SG with respect to the predicate `known` (subtree in blue). Every token pays attention to every other token in the subclause except itself.

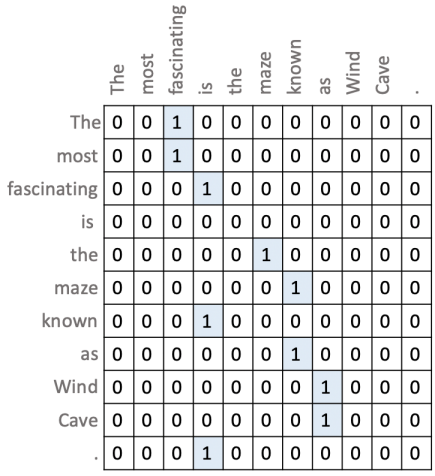
Figure 4.2: A visualization of the tokens attended to under the four mechanisms.

Figure 4.1-4.3 show three steps to generate the dependency matrices for the four mechanisms, Full-tree informed, Parent informed, Predicate informed and Full-subclause predicate informed. Figure 4.1 shows the parsing tree obtained from a standard parser. Figures 4.2a-4.2d visualize the part of that tree that should be attended to. In this example, the token `is` is the root and `known` is the given predicate. For illustration, the attended tokens for labeling `the` are marked with orange circles. Figures 4.3a-4.3d are the corresponding dependency matrices.



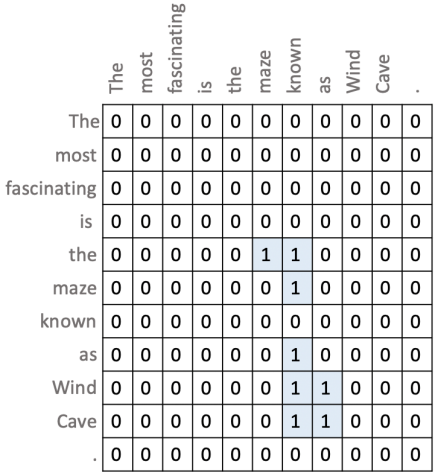
	The	most	fascinating	is	the	maze	known	as	Wind	Cave	.
The	0	0	1	1	0	0	0	0	0	0	0
most	0	0	1	1	0	0	0	0	0	0	0
fascinating	0	0	0	1	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	1	0	1	1	0	0	0	0
maze	0	0	0	1	0	0	1	0	0	0	0
known	0	0	0	1	0	0	0	0	0	0	0
as	0	0	0	1	0	0	1	0	0	0	0
Wind	0	0	0	1	0	0	1	1	0	0	0
Cave	0	0	0	1	0	0	1	1	0	0	0
.	0	0	0	1	0	0	0	0	0	0	0

(a) Full-tree-informed dependency matrix



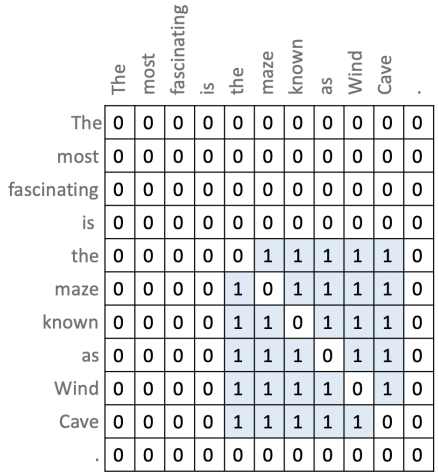
	The	most	fascinating	is	the	maze	known	as	Wind	Cave	.
The	0	0	1	0	0	0	0	0	0	0	0
most	0	0	1	0	0	0	0	0	0	0	0
fascinating	0	0	0	1	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1	0	0	0	0	0
maze	0	0	0	0	0	0	1	0	0	0	0
known	0	0	0	1	0	0	0	0	0	0	0
as	0	0	0	0	0	0	1	0	0	0	0
Wind	0	0	0	0	0	0	0	1	0	0	0
Cave	0	0	0	0	0	0	0	1	0	0	0
.	0	0	0	1	0	0	0	0	0	0	0

(b) Parent-informed dependency matrix



	The	most	fascinating	is	the	maze	known	as	Wind	Cave	.
The	0	0	0	0	0	0	0	0	0	0	0
most	0	0	0	0	0	0	0	0	0	0	0
fascinating	0	0	0	0	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1	1	0	0	0	0
maze	0	0	0	0	0	0	1	0	0	0	0
known	0	0	0	0	0	0	0	0	0	0	0
as	0	0	0	0	0	0	1	0	0	0	0
Wind	0	0	0	0	0	0	1	1	0	0	0
Cave	0	0	0	0	0	0	1	1	0	0	0
.	0	0	0	0	0	0	0	0	0	0	0

(c) Predicate-informed dependency matrix



	The	most	fascinating	is	the	maze	known	as	Wind	Cave	.
The	0	0	0	0	0	0	0	0	0	0	0
most	0	0	0	0	0	0	0	0	0	0	0
fascinating	0	0	0	0	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1	1	1	1	1	0
maze	0	0	0	0	1	0	1	1	1	1	0
known	0	0	0	0	1	1	0	1	1	1	0
as	0	0	0	0	1	1	1	0	1	1	0
Wind	0	0	0	0	1	1	1	1	0	1	0
Cave	0	0	0	0	1	1	1	1	1	0	0
.	0	0	0	0	0	0	0	0	0	0	0

(d) Full-subclause predicate-informed dependency matrix

Figure 4.3: Dependency matrices of the four mechanisms.

4.1.2 SG Attention

We compute the attention vectors using a $n \times n$ SG dependency matrix $\mathbf{I}$. Let $\mathbf{E} \in \mathbb{R}^{n \times m}$ denote the BERT features concatenated with predicate indicator vectors, where m is the concatenated embedding dimension. For each token w_i, we compute the attention scores $\mathbf{s}_i$

$$\mathbf{s}_i = \frac{\mathbf{E}\mathbf{E}_i^\top}{\sqrt{m}}. \tag{4.1}$$

We then take a softmax to get the attention weights

$$\mathbf{p}_i = \mathrm{softmax}(\mathbf{s}_i). \tag{4.2}$$

The SG attention vector $\mathbf{A}_i$ of token w_i is the weighted sum of embeddings that we should attend to:

$$\mathbf{A}_i = \sum_{j=1}^{n} \mathbf{p}_{ij}\mathbf{I}_{ij}\mathbf{E}_j. \tag{4.3}$$

Each SG attention vector $\mathbf{A}_i$ is then concatenated with the embedding $\mathbf{E}_i$, which will be the input to the next layer.

4.2 Model Description

The training process of the SG + BERT + CRF model is as follows.

Given a sentence $\boldsymbol{W} = \{w_1, w_2, ..., w_m\}$

1. Tokenize $\boldsymbol{W}$, some of the tokens might break down into several sub-tokens. Get the dependency head using the pre-trained model (Zhou and Zhao, 2019) and then generate dependency matrices $\boldsymbol{M}$ accordingly.

2. Obtain token embeddings through BERT. Pick embedding for the first sub-token as the embedding for the token, $\boldsymbol{E}' = \{\boldsymbol{e}'_1, \boldsymbol{e}'_2, ..., \boldsymbol{e}'_m\}$, where $\boldsymbol{e}'_j \in \mathbb{R}^{d_{BERT}}$ and d_{BERT} is the embedding dimension.

3. Concatenate the contextual embedding $\boldsymbol{e}'_j$ with the predicate embedding $\boldsymbol{p}_j \in \mathbb{R}^{d_p}$. Then

$$\boldsymbol{e}_j = \boldsymbol{e}'_j \oplus \boldsymbol{p}_j, \tag{4.4}$$

   where $\boldsymbol{e}_j \in \mathbb{R}^{d_{BERT}+d_p}$. Put $\boldsymbol{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_m\}$.

4. Calculate the SG attention using (4.3)

5. Pass $\mathbf{E}$ to a linear layer to make the dimension fit the token size.

$$\boldsymbol{L} = C'(\boldsymbol{A} \oplus \boldsymbol{E}) \tag{4.5}$$

6. Compute loss based on (2.1) and (2.2) and propagate backwards. Repeat three epochs.

7. Decoding. Use the Viterbi algorithm to get predictions.

An illustration of the architecture is in Figure 4.4.

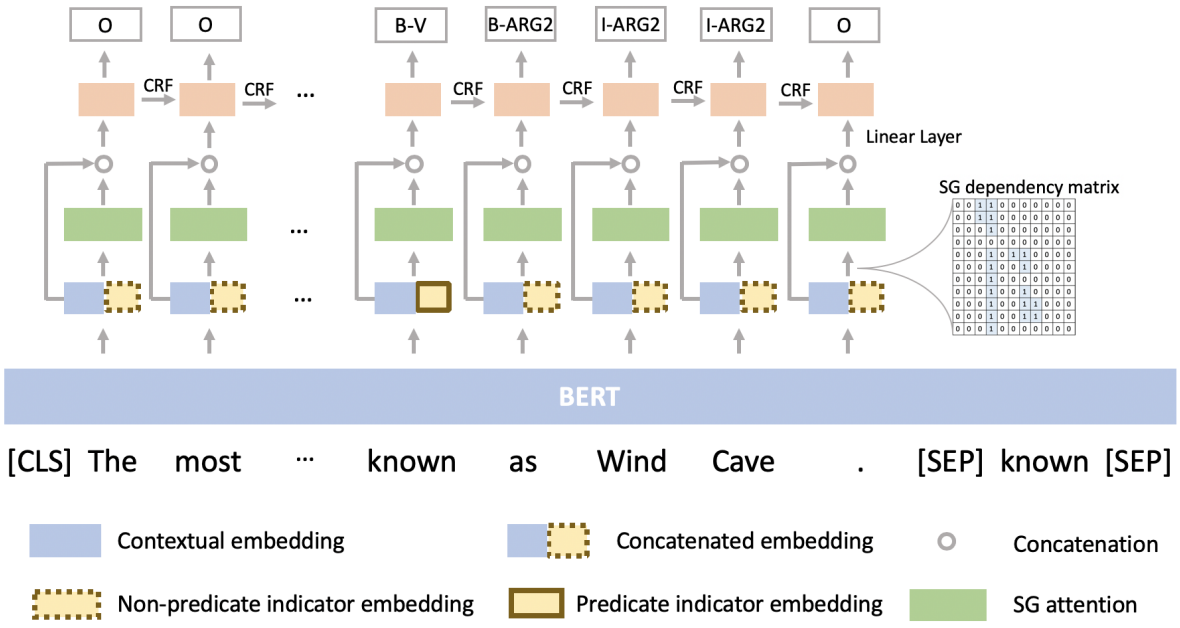


Figure 4.4: An illustration of the BERT + CRF + SG architecture. In this model, we first concatenate BERT contextual embeddings with predicate indicator embeddings. The SG attention is computed with a specific dependency matrix. Then we concatenate SG attention with the embeddings. A linear layer follows to ensure the input dimension of the next layer matches the target label size. CRF is used to encode the sequential transition information.

4.3 Performance Evaluation

We use the base uncased version of BERT. The batch size is 48. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0001 and weight decay rate 0.0001. The parameter matrix used in the linear layer is of shape $\mathbb{R}^{2d_{BERT} \times k}$, where k is the size of the target label set. ($k = 105$ for CoNLL-2005 and $k = 129$ for CoNLL-2012) The parser we used to generate the dependency array is by (Zhou and Zhao, 2019). The running times for training CoNLL-2005 and CoNLL-2012 are 7 hours and 19 hours respectively using 8 GPUs (GeForce GTX TITAN Black).

Model	**CoNLL-2005**		**CoNLL-2012**
	Test WSJ	Test Brown	Test
LISA(Strubell et al., 2018)	86.0	78.3	83.4
Joint predication + ELMo(He et al., 2018)	87.4	80.4	85.5
ELMo SPAN (ensemble) (Ouchi et al., 2018)	88.5	79.6	87.0
BERT + LSTM (Shi and Lin, 2019)	88.8	82.0	86.5
SpanGCN (Marcheggiani and Titov, 2019)	87.2	78.4	85.9
BERT + CRF	91.2	85.4	90.2
Full-attention + BERT + CRF	89.55	86.47	90.34
Full-tree-informed SG + BERT + CRF	91.93	88.06	90.90
Parent-informed SG + BERT + CRF	90.93	87.85	90.57
Predicate-informed SG + BERT + CRF	**92.62**	**88.47**	**91.67**
Full-subclause predicate-informed SG+BERT+CRF	90.34	87.03	90.19

Table 4.1: A comparison of performance on the CoNLL-2005 WSJ test set and Brown test set, and the CoNLL-2012 test set in terms of F1

The results of the model with the four SG mechanisms for span-based SRL are shown in Table 4.1. The baseline model for this work is the BERT + CRF moel that we proposed in Section 3.1. For comparison, we also included a model with a "vanilla" attention head, i.e., without any syntax guidance. It attends to every token in the sentence. We refer to it as full-attention in Table 4.1. The model without syntax guidance performs worse than the models with SG self-attention, which indicates the efficacy of syntax guidance.

Among all four SG mechanisms, we observed a consistently improved performance using the Predicate-informed mechanism over the others. With the Predicate-informed mechanism, we

achieved an improvement of 3.1 and 1.5 percentage points in F1 respectively on the CoNLL-2005 Brown test set and CoNLL-2012 test set against the baseline model. Moreover, we gained 3.8 percentage points in F1 on the CoNLL-2005 WSJ test set against the previous state-of-the-art models. The 95% significance threshold for the three test sets are 1.16, 4.28 and 0.67. Thus it is significantly better then BERT + CRF on CoNLL-2012. And we are 93% confident that it performs better on CoNLL-2005 WSJ test set. The detailed calculation is in Appendix. Since the only change we made to the model of was to concatenate features modified by SG attention and we obtained a significant boost in the performance, we conclude that syntactic information was indeed beneficial for this approach to span-based SRL. We discuss possible reasons for the advantage of the Predicate-informed model over the other forms of SG self-attention in the next section.

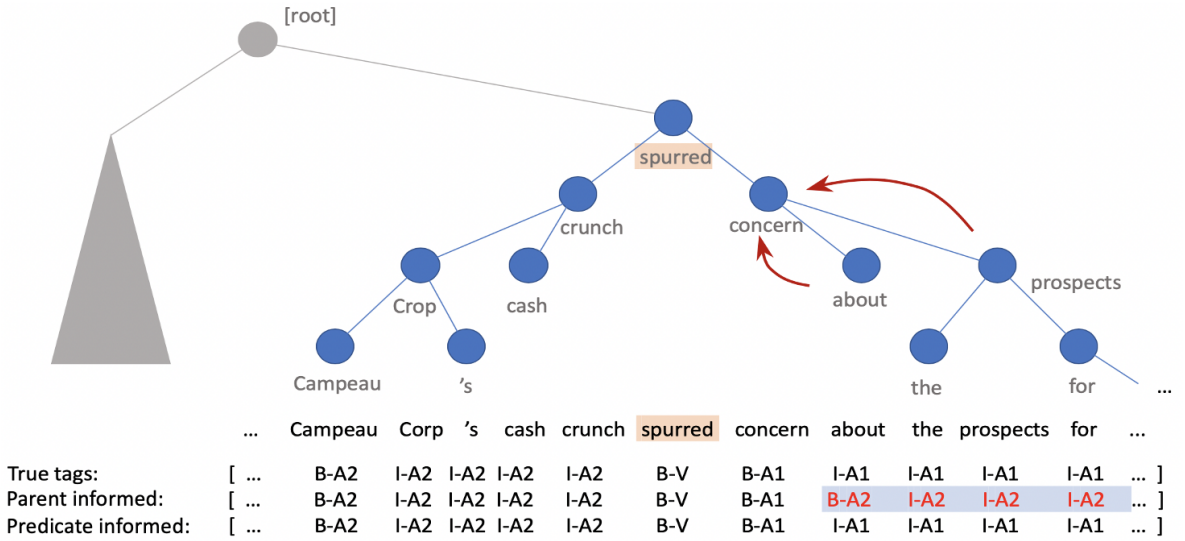
4.4 Error Analysis

In this section we analyze the reason behind the improvement of the SG model over the other three forms of SG attention.

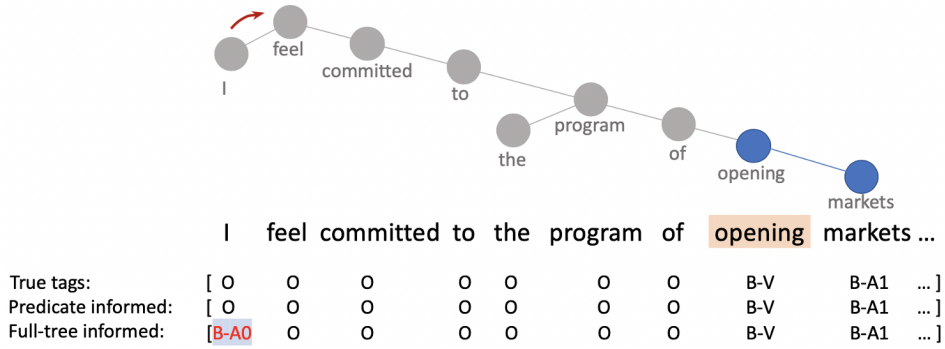
For the parent-informed SG from (Strubell et al., 2018), even though the shallow syntactical dependencies of this model can reconstruct the whole dependency tree, hence fully incorporating the syntactic information, this can cause the model to be misguided. We observe that sometimes a word can be mislabelled by parent-aware SG due to the fact that the predicate is not its direct parent. For example, in the sentence

`For weeks, the market had been nervous about takeovers, after Campeau Corp's cash crunch spurred concern about the prospects for future highly leveraged takeovers`

when the given predicate is `spurred`, the entire clause `concern about the prospects for future highly leveraged takeovers` should be labeled as "ARG1" since it is the object of the predicate verb `spurred`. However, in the dependency parsing tree, which is illustrated in Fig. 4.5b, the parent-informed model mislabelled `prospects` as "ARG2", which is a modifier type class. This is because the parent of `prospects` is `concern`. Since the parent-informed model only pays attention to the parent, it mistakes `prospects` as a modifier to `concern`, rather than part of an object of the predicate `spurred`.



(a) Parent-informed attention mistakes `concern` and `about the prospects` as separate semantic parts since they only attend to the parent `concern` thus missing the predicate `spurred`



(b) Full-tree attention mistakes `feel` as the predicate, when it is `opening`, labeling `I` as a subject argument.

Figure 4.5: Example of errors caused by attention to improper portions of the parsing tree.

Therefore, we argue that it's important to enforce attention to tokens all the way to the predicate at the SG layer. To confirm this intuition, we separate each model's label prediction for the individual tokens of CoNLL-2005 WSJ into two groups. In one group, the token has the predicate as their parent, which makes parent-informed SG pay attention to the predicate. In the other group, the parent of the token is not the predicate, which causes the attention of parent-informed SG to miss the predicate. The results are shown in Table 4.2. We observe that the performance of the two methods is essentially identical when the parent is the predicate, and there is a significant difference when the parent is not the predicate. So the improvement of the predicate-informed SG is

	Total count	Parent informed	Predicate informed	Diff.
Predicate = Parent	19228	92.5	92.6	+0.1
Predicate ≠ Parent	129419	90.7	92.6	**+1.9**

Table 4.2: A comparison of accuracy (%) between Parent-informed SG and Predicate-informed SG on CoNLL 2005 WSJ. (Statistical significance thresholds are 0.98 and 0.38.)

	Total count	Full-tree informed	Predicate informed	Diff.
Predicate = Root	46121	92.2	92.3	+0.1
Predicate ≠ Root	102526	91.8	92.7	**+0.9**

Table 4.3: A comparison of accuracy (%) between Full-tree-informed SG and Predicate-informed SG on CoNLL 2005 WSJ. (Statistical significance thresholds are 0.63 and 0.42.)

significant precisely when the parent-informed model misses the predicate. This corroborates our intuition.

On the other hand, the Full-tree-informed SG from (Zhang et al., 2020) encounters ambiguity when it is applied to the SRL task because it uses the same parsing for each sentence even if different predicates are given. Such syntactic guidance is likely to add ambiguity since the different predicates induce different semantic parsings. For instance, in the sentence

`I feel committed to the program of opening markets`

if `opening` is given as the predicate then we should label `opening` as "B-V" and `markets` as "B-A1", while the rest of tokens are "O". However, the parent-aware SG wrongly labels `I` as "B-A0", which is the subject class. This is due to the fact that in the parsing `feel` is the root predicate, and since Full-tree-informed SG pays attention to the root when labeling `I`, it misinterprets `I` as the subject of `feel`. This shows that paying attention to tokens outside the predicate sub-clause can be distracting. If the root is not the predicate, then paying attention to all tokens up to the root

node can reduce the accuracy. Such problems do not appear in the predicate-informed SG because it does not pay attention to any token outside the predicate's sub-clause. So intuitively it's more reasonable to restrict the attention only to the sub-clause of the predicate.

We have performed another experiment to confirm this intuition. In this experiment we divided the tokens of CoNLL-2005 WSJ into two groups, where in the first group the root of the dependency tree where the token resides is different from the predicate, and in the second group they are the same. The results are shown in Table 4.3. We observe that there is no significant difference in performance between the methods when the root is the predicate, whereas there is a significant difference of 0.9% when the root is not the predicate. Therefore our predicate-informed SG achieves a significant improvement precisely when the root is not the predicate, corroborating our intuition.

4.4.1 Labeling Confusions

We now examine the errors made by the full model, SG + BERT + CRF. The SG + BERT + CRF models preserve the decrease in `A0-A1` confusion obtained by BERT + CRF over the prior works A clear advantage of the Predicate-informed SG mechanism is a decrease in PPs confusion. Tables 4.4 - 4.7 show the confusion matrices of the four SG mechanisms. We can see that Predicate-informed SG outperforms other mechanisms in either A0/A1 confusion or A2 confusion as discussed in Section 4.4.

True / Pred.	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	13	8	3	0	0	0	0	0	2
A1	53	-	51	9	18	9	21	8	63	29
A2	30	30	-	50	8	62	30	24	10	5
A3	0	4	2	-	5	9	0	1	7	0
ADV	10	11	11	18	-	0	12	41	18	57
DIR	0	0	3	1	0	-	3	0	0	0
LOC	0	23	0	2	9	1	-	9	0	2
MNR	0	5	15	6	36	3	14	-	0	2
PNC	0	2	4	0	1	5	8	10	-	0
TMP	4	8	2	4	20	9	7	1	0	-

Table 4.4: Confusion matrix for Full-tree-informed SG BERT + CRF

True / Pred.	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	15	10	6	3	0	1	1	0	0
A1	61	-	48	32	47	43	32	18	60	26
A2	15	21	-	31	12	38	19	20	17	3
A3	1	2	3	-	5	11	1	11	3	0
ADV	13	20	7	3	-	0	4	33	18	63
DIR	0	2	0	0	0	-	0	0	0	0
LOC	6	14	11	1	7	2	-	10	0	0
MNR	0	7	11	8	7	1	16	-	0	1
PNC	0	7	5	13	1	0	10	1	-	2
TMP	2	8	1	1	12	1	12	1	0	-

Table 4.5: Confusion matrix for Parent-informed SG + BERT + CRF

True / Pred.	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	15	9	10	3	0	0	0	0	2
A1	67	-	40	23	27	44	43	8	52	42
A2	6	23	-	11	5	22	14	17	15	7
A3	0	3	2	-	0	10	0	7	5	0
ADV	13	9	5	1	-	0	1	44	19	29
DIR	0	0	5	3	0	-	4	0	0	0
LOC	3	29	16	7	9	22	-	13	1	8
MNR	0	1	8	18	21	0	6	-	0	6
PNC	0	3	4	6	1	0	10	1	-	3
TMP	8	13	5	16	29	0	18	6	3	-

Table 4.6: Confusion matrix for Predicate-informed SG + BERT + CRF

True / Pred.	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	0	26	7	8	4	0	0	1	0	0
A1	59	0	32	13	41	20	33	13	60	29
A2	12	20	0	25	7	22	14	21	15	3
A3	0	0	1	0	0	11	0	4	7	0
ADV	6	5	12	13	0	3	5	38	14	61
DIR	0	0	1	1	0	0	3	0	0	0
LOC	13	8	11	2	8	24	0	11	1	1
MNR	3	13	12	15	22	3	20	0	0	2
PNC	0	9	13	12	2	4	6	3	0	0
TMP	5	15	6	6	13	9	16	5	0	0

Table 4.7: Confusion matrix for Full-subclause predicate-informed SG + BERT + CRF

4.4.2 BIO Violations

In order for a sequence of labels to be a possible sequence under the BIO labeling mechanism, it must obey the following BIO constraints: the sequence of labels within a single semantic role span must be of the form {B-X, I-X*, O*}, where * denotes the Kleene star operator, i.e., any number of repetitions. Following (He et al., 2017), we counted the number of instances that violate these BIO constraints

The CoNLL 2005 development set has 3248 sentences and 94763 tokens, and and the CoNLL 2012 development set has 35297 sentences and 934744 tokens. The number of BIO violations of (He et al., 2017) were 0.07 per token for the CoNLL 2005 development set. We had 0.03 violations per token for CoNLL 2005 and less than 0.01 per token for the CoNLL 2012 development set. We believe that the improvement is mainly due to the CRF layer. The transition matrix in CRF provides the ability to penalize the forbidden label transitions. Table 4.8 and Table 4.9 further compare BIO violations among the four proposed SG mechanisms. The results show that Full-tree-informed SG obtains the fewest BIO violations on the CoNLL 2005 development set, while Predicate-informed SG incurs the fewest BIO violations on CoNLL 2012 development set.

As a matter of fact, CRF is capable of eliminating BIO violations by assigning negative infinite values in the transition matrix. Suppose the transition from label s to label t is not allowed. Then we assign $T(s,t) = \eta$ for every such pairs, where η is the penalty for making such transitions. The experimental result shows that setting $\eta = -\infty$ indeed eliminates BIO violation, however, such a modification turns out to harm the overall performance. Specifically, on CoNLL 2005 development set, the violation is eliminated yet the F1 decreases to 82.28. We suspect that such a hard constraint rules out a large proportion of labellings, so that the learning becomes tough. If we relax the penalty η by increasing the constant scalars, and viewing them as prior knowledge, and then enable training on them, we might potentially obtain better performance.

Model	BIO violation	
	violated / mislabeled sentences	violated / mislabeled tokens
Full-tree-informed	**231** / 1269	**309** / 8011
Parent informed	289 / 1305	330 / 8435
Predicate-informed	278 / **1242**	346 / **7124**
Full subclause predicate-informed	240 / 1348	369 / 9055

Table 4.8: BIO violation on CoNLL-2005

Model	BIO violation	
	violated / mislabeled sentences	violated / mislabeled tokens
Full-tree-informed	763 / 12751	811 / 88740
Parent-informed	773 / 13045	812 / 92691
Predicate-informed	**505 / 12574**	**512 / 82192**
Full subclause predicate-informed	987 / 13744	1200 / 98621

Table 4.9: BIO violation on CoNLL-2012

4.4.3 Mutual Exclusion Violations of Unique Core Roles

We also investigate the violations of mutual exclusion of Unique Core Roles (UCR) committed by the proposed architecture. Mutual exclusion of Unique Core Roles refers to the constraint that UCRs must not occur more than once, such as `A0` and `A1`. Specifically, for any given predicate, there should be at most one `A0` (Agent) or `A1` (Patient). Table 4.10 and Table 4.11 report the number of mutual exclusion violations of four of the semantic role labels $A0, A1, A2$ and $A3$.

Model	Mutual Exclusion violation	
	violated / mislabeled sentences	violated / mislabeled tokens
Full-tree-informed	75 / 1269	548 / 8011
Parent-informed	112 / 1305	750 / 8435
Predicate-informed	**69 / 1242**	**350 / 7124**
Full subclause predicate-informed	121 / 1348	847 / 9055

Table 4.10: Violations of mutual exclusion of core roles on CoNLL-2005

Model	Mutual Exclusion violations	
	violated / mislabeled sentences	violated / mislabeled tokens
Full-tree-informed	750 / 12751	6253 / 88740
Parent-informed	**556** / 13045	**4965** / 92691
Predicate-informed	832 / **12574**	5934 / **82192**
Full-subclause predicate-informed	857 / 13744	6509 / 98621

Table 4.11: Violation of mutual exclusion of core roles on CoNLL-2012

Semantic Loss

We observe that the number of mutual exclusion violations can be decreased by introducing a semantic loss penalty, as proposed by Xu et al. (2018), for these constraints. We can assign semantic loss for every unique core role in each sentence

$$L_s(p) \propto -\log \sum_{i \in [m]} p_i \prod_{i \neq j} (1 - p_j) + \prod_j (1 - p_j) \tag{4.6}$$

where m is the number of tokens in the sentence, and p is the core role of interest. The first term $p_i \prod_{i \neq j}(1 - p_j)$ corresponds to the case when exactly one token is labeled as i and the second term $\prod_j (1 - p_j)$ the case when none of the tokens is labeled as i. Therefore whenever the mutual

exclusion constraint is not violated, semantic loss becomes 0. Otherwise, the semantic loss is positive.

Suppose L_{NLL} is the negative log likelihood, then the augmented loss becomes:

$$L' = L_{NLL} + \lambda L_s(p) \tag{4.7}$$

where λ controls the importance of semantic loss.

Figure 4.6 and Table 4.12 show the experimental result of applying the semantic loss regularizer and the hard constraint in the transition matrix as discussed in Section 4.4.2. It confirms the claim that semantic loss can decrease the Unique Core Roles violations, and has a better effect as λ increases. F1 improves 2.67 percentage points compared to the case when we don't apply semantic loss.

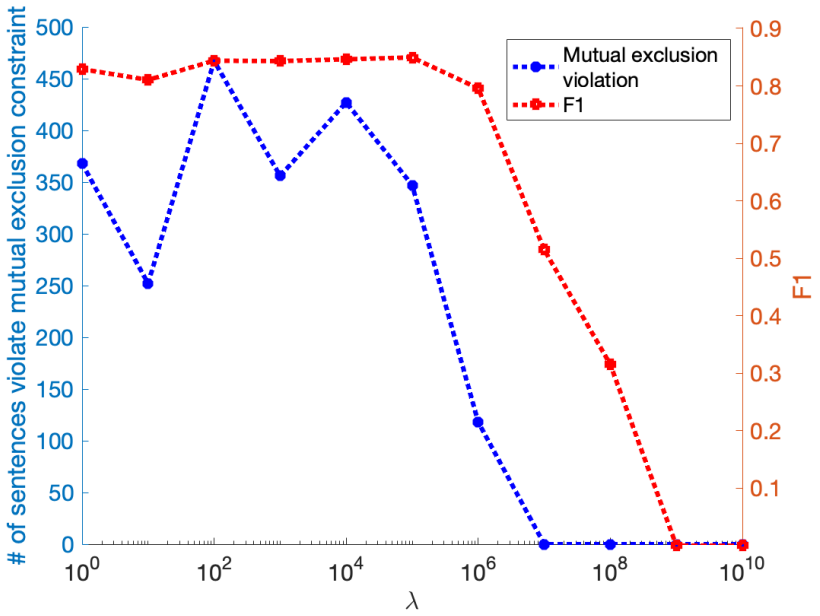


Figure 4.6: The experimental result of adding the semantic loss and forbidden illegal transition through transition matrix in CRF.

λ	Mutual Exclusion violation		**F1**
	violated / mislabeled sentences	violated / mislabeled tokens	
0	435 / 2480	3914 / 16789	82.28
1.00E+00	368 / 2400	2511 / 16191	82.91
1.00E+01	252 / 2428	1993 / 17961	81.05
1.00E+02	467 / 2419	3005 / 14795	84.39
1.00E+03	356 / 2354	1903 / 14858	84.32
1.00E+04	427 / 2358	3427 / 14567	84.63
1.00E+05	347 / 2376	2168 / 14260	**84.95**
1.00E+06	118 / 2718	1241 / 19354	79.58
1.00E+07	0 / 3242	0 / 45966	51.49
1.00E+08	0 / 3248	0 / 64886	31.53
1.00E+09	0 / 3248	0 / 94742	0.02
1.00E+10	0 / 3248	0 / 94763	0.00

Table 4.12: Experimental result with different λ on CoNLL-2005 development set. The illegal transitions are forbidden through transition matrix in CRF.

Chapter 5

Conclusion

We first examined the role of LSTM in the state-of-the-art BERT-LSTM model by replacing it with a CRF, and examining the performance on the CoNLL-2005 and CoNLL-2012 data sets. We found that by removing LSTM the model can achieve much better results, hence LSTM is harming the performance of this approach to span-based SRL. We also observed that the lack of syntactic information might be responsible for some errors made by our model, thus we conclude that incorporating syntactic parsing information might help improve the performance of this baseline model.

We then proposed a syntax-guided (SG) neural architecture SG + BERT + CRF, which achieves the new state-of-the-art for CoNLL-2005 and 2012. We have compared its performance with the SG attention models that have been previously studied, and analyzed the source of the improvement. However, we notice that the attention mechanism with highest accuracy only traces back to the predicate when predicting the semantic role of a token, and does not take into consideration the tokens of other semantic roles of this predicate. We suspect that sometimes these are relevant. As a future direction, we would like to develop a new attention mechanism that includes such tokens, which could potentially further improve the accuracy. We also emphasize the potential in the transition matrix of CRF and semantic loss regularizer to exclude two types of errors, specifically, the BIO violation and the violation of unique core roles.

Appendix A

Statistical Significance in F1

The multiplicative Chernoff bound has the following form (Kearns and Vazirani, 1994):

$$\mathbb{P}\Big[S - \mu m > \gamma \mu m\Big] \leq e^{-m\mu\gamma^2/3} \tag{A.1}$$

where m is the number of variables. pm is the expectation of S. γ controls the significance threshold. Based on the discussion, the calculation is as follows. Suppose S denote the number of tokens that are not correctly labeled, then we have $1 - F1 \approx S \times (k/2)$, where k denote label size (since positive rate equals $1/$(lable size)). $m = nk$ where n is the number of tokens, and k is the size of labels.

Then

$$\mathbb{P}\left[\frac{k}{2}E_{acc} - \frac{k}{2}\mu > \frac{k}{2}\gamma\mu\right] \leq e^{-m\mu\gamma^2/3}. \tag{A.2}$$

Let $t = \frac{k}{2}\gamma\mu$, then

$$\mathbb{P}\Big[E_{F1} - E_{F1,SOTA} > t\Big] \leq e^{-\frac{4nt^2}{3k\mu}}. \tag{A.3}$$

Table A.1 and Table A.2 show the values for each variable on different data sets for the BERT + CRF model and the SG + BERT + CRF model. Table A.1 shows that the BERT + CRF model performs significantly better on the CoNLL-2005 WSJ and the CoNLL-2012 test set. Though it obtains 3.4 percentage points gain on the CoNLL-2005 Brown set, due to the size of the corpus being small, its gain does not pass the significance test.

Table A.2 shows that the SG + BERT + CRF model performs significantly better than the BERT + CRF on CoNLL-2012 test set. We are 93.83% confident that it is also better than the BERT + CRF on the CoNLL-2005 WSJ test set. On the CoNLL-2005 Brown test set, even though it obtains 3.07 percentage points gain, it still doesn't pass the significance test.

Var	Notation	**CoNLL-2005**		**CoNLL-2012**
		WSJ	Brown	Test
n	# of tokens	148647	18814	639622
k	label size	105	105	129
μ	SOTA 1-F1	0.112	0.180	0.130
t^*	95% Sign.	+0.0133	+0.0475	+0.0077
Δ	F1 Gain	**+0.027**	**+0.034**	**+0.032**
$1-\delta'$	Conf.*	**100.00%**	78.44%	**100.00%**

Table A.1: Some statistics and the significance threshold with respect to the BERT + CRF model. t^* means the 95% statistical significance threshold. $1-\delta'$ is the confidence level that the proposed model is better.

Var	Notation	**CoNLL-2005**		**CoNLL-2012**
		WSJ	Brown	Test
n	# of tokens	148647	18814	639622
k	label size	105	105	129
μ	1-F1 (BERT+CRF)	0.112	0.180	0.130
t^*	95% Sign.	+0.0116	+0.0428	+0.0067
Δ	F1 Gain	**+0.0112**	**+0.0307**	**+0.0147**
$1-\delta'$	Conf.*	**93.83%**	78.61%	**100.00%**

Table A.2: Some statistics and the significance threshold with respect to the SG + BERT + CRF model. t^* means the 95% statistical significance threshold. $1-\delta'$ is the confidence level that the proposed model is better.

References

Jeremy Appleyard, Tomas Kocisky, and Phil Blunsom. Optimizing performance of recurrent neural networks on gpus. *arXiv preprint arXiv:1604.01946*, 2016.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, 2014.

Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W05-0620.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT*, 2019.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, 2017.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2058. URL https://www.aclweb.org/anthology/P18-2058.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994. ISBN 0262111934.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D12-1096.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1101. URL https://www.aclweb.org/anthology/P16-1101.

Diego Marcheggiani and Ivan Titov. Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Association for Computational Linguistics*, 2019.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1041. URL https://www.aclweb.org/anthology/K17-1041.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1630–1642, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1191. URL https://www.aclweb.org/anthology/D18-1191.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning (CoNLL 2012)*, Jeju, Korea, 2012.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 12–21, 2007.

Peng Shi and Jimmy Lin. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*, 2019.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. Deep semantic role labeling with self-attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 700–706, 2015.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *The 35th International Conference on Machine Learning Conference*, 2018.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, and Hai Zhao. Sg-net: Syntax-guided machine reading comprehension. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.

Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, 2015.

Junru Zhou and Hai Zhao. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, July 2019. Association for Computational Linguistics.