Nested Named Entity Recognition as Latent Lexicalized Constituency Parsing

Anonymous ACL submission

Abstract

Nested named entity recognition (NER) has been receiving increasing attention. Recently, Fu et al. (2020) adapt a span-based constituency parser to tackle nested NER. They treat nested entities as partially-observed constituency trees and propose the masked inside algorithm for partial marginalization. However, their method cannot leverage entity heads, which have been shown useful in entity mention detection and entity typing. In this work, we resort to more expressive structures, lexicalized constituency trees in which constituents are annotated by headwords, to model nested entities. We leverage the Eisner-Satta algorithm to perform partial marginalization and inference efficiently. In addition, we propose to use (1) a two-stage strategy (2) a head regularization loss and (3) a head-aware labeling loss in order to enhance the performance. We make a thorough ablation study to investigate the functionality of each component. Experimentally, our method achieves the state-of-the-art performance on ACE2004, ACE2005 and NNE, and competitive performance on GENIA, and meanwhile has a fast inference speed. Our code will be publicly available at: `github.com/xxx`.

1 Introduction

Named Entity Recognition (NER) is a fundamental task in information extraction, playing an essential role in many downstream tasks. Nested NER brings more flexibility than flat NER by allowing nested structures, thereby enabling more fine-grained meaning representations and broader applications (Byrne, 2007; Dai, 2018). Traditional sequence-labeling-based models have achieved remarkable performance on flat NER but fail to handle nested entities. To resolve this problem, there are many layer-based methods (Ju et al., 2018; Fisher and Vlachos, 2019; Shibuya and Hovy, 2020; Wang et al., 2020, 2021) proposed to recognize entities layer-by-layer in bottom-up or top-down manners. However, they suffer from the error propagation issue due to the cascade decoding.

Recently, Fu et al. (2020) adapt a span-based constituency parser to tackle nested NER, treating annotated entity spans as a partially-observed constituency tree and marginalizing latent spans out for training. Their parsing-based method, namely PO-TreeCRF, admits global exact inference thanks to the CYK algorithm (Cocke, 1969; Younger, 1967; Kasami, 1965), thereby eliminating the error propagation problem. However, their method does not consider entity heads, which provide important clues for entity mention detection (Lin et al., 2019; Zhang et al., 2020d) and entity typing (Katiyar and Cardie, 2018; Choi et al., 2018; Chen et al., 2021). For example, *University* and *California* are strong clues of the existence of `ORGEDU` and `STATE` entities in Fig.1. Motivated by this and inspired by head-driven phrase structures, Lin et al. (2019) propose the Anchor-Region Network (ARN), which identifies all entity heads firstly and then predicts the boundary and type of entities governed by each entity head. However, their method is heuristic and greedy, suffering from the error propagation problem as well.

Our main goal in this work is to obtain the best of two worlds: proposing a probabilistically principled method that enables exact global inference like Fu et al. (2020), meanwhile taking entity heads into accounts like Lin et al. (2019). To enable exact global inference, we also view observed entities as partially-observed trees. Since constituency trees cannot model entity heads, we resort to lexicalized trees, in which constituents are annotated with headwords. A lexicalized tree embeds a constituency tree and a dependency tree (Gaifman, 1965), and lexicalized constituency parsing can thus be viewed as joint dependency and constituency parsing (Eisner and Satta, 1999; Collins, 2003). Fig.1 illustrates an example lexicalized tree. Joint dependency and con-

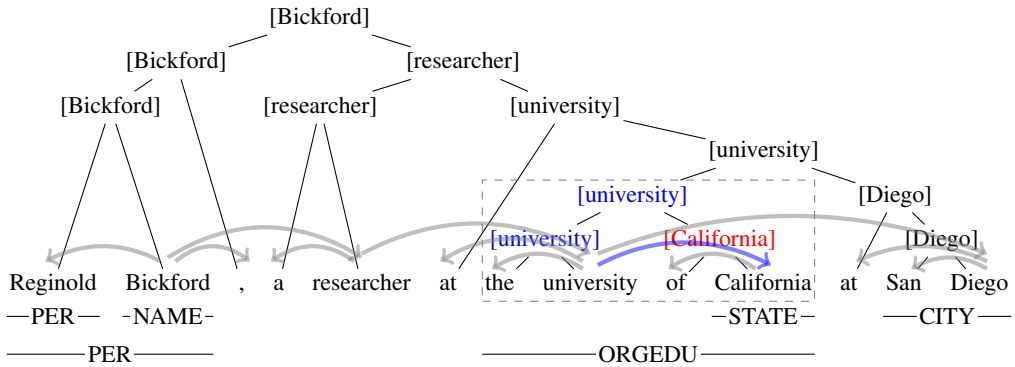


Figure 1: An example sentence with a compatible latent lexicalized constituency tree (top) and observed entities (down). All constituents are annotated by headwords with $[\cdot]$ and we omit the constituent labels. The dotted frame shows an example of inherited head (blue) and non-inherited head (red). We can draw a dependency arc from the inherited head to the non-inherited head. For example, *University* $\rightarrow$ *California*. Hence a lexicalized constituency tree embeds a constituency tree and a dependency tree.

stituency parsing has been shown to outperform standalone constituency parsing (Zhou and Zhao, 2019; Fernández-González and Gómez-Rodríguez, 2020) possibly because modeling dependencies between headwords helps predict constituents correctly. Hence, in the context of nested NER, we have reasons to believe that modeling latent lexicalized constituency trees would bring improvement in predicting entities over modeling latent constituency trees, and we verify this in experiments.

When using a lexicalized constituency tree for nested NER, only part of unlexicalized spans, i.e., entities, are observed, so we need to marginalize latent spans and dependency arcs out for training. Inspired by the masked inside algorithm of Fu et al. (2020), we propose a masked version of the Eisner-Satta algorithm (Eisner and Satta, 1999), a fast lexicalized constituency parsing algorithm, to perform partial marginalization. We also adopt the Eisner-Satta algorithm for fast inference.

Besides the difference in parsing formalism and algorithms, our work also differs from the work of Fu et al. (2020) and Lin et al. (2019) in the following three aspects. First, inspired by Zhang et al. (2020a), we adopt a two-stage parsing strategy, i.e., we first predict an unlabeled tree and then label the predicted constituents, instead of using the one-stage parsing strategy of PO-TreeCRF. We show that two-stage parsing can improve the performance of both PO-TreeCRF and our proposed method. Second, Lin et al. (2019) observe that each entity head governs only one entity span in most cases, so they impose a hard constraint of that during learning and inference, which is potentially harmful since the constraint is not always satisfied. Instead, we add a soft KL penalty term to encourage satisfaction of the constraint, which is reminiscent of posterior regularization (Ganchev et al., 2010; Zhang et al., 2017). Third, considering that gold entity heads are not given, Lin et al. (2019) propose a "bag loss" for entity boundary detection and labeling. However, this loss is heuristic and brings an additional hyperparameter, to which the final performance is sensitive. In contrast, entity boundary detection is learned in the first stage of our method, and in the second stage, we propose a more principled labeling loss based on expectations (i.e., marginal likelihoods) of all possible entity heads within gold entity spans, which can be estimated efficiently and does not introduce new hyperparameters.

We conduct experiments on four benchmark datasets, showing that our model achieves state-of-the-art results on ACE2004, ACE2005 and NNE, and competitive results on GENIA, validating the effectiveness of our method.

2 Preliminary

2.1 One-stage and Two-stage Parsing

A labeled constituency tree can be represented as a rank-3 binary tensor T where $T_{ijk} = 1$ if there is a span from the i-th word to the j-th word with label k in the tree and $T_{ijk} = 0$ otherwise. We assume the 0-th label is preserved for $\emptyset$ (i.e., no label) without loss of generality. Similarly, an unlabeled constituency tree can be represented as a binary matrix T'. One-stage span-based constituency parsers decompose the score of a labeled constituency tree

into the scores of constituents s_{ijk}:

$$s(T) = \sum_{ijk} T_{ijk} s_{ijk}$$

They use the CYK algorithm to recover the optimal **labeled** tree. In contrast, two-stage constituency parsers score unlabeled trees and constituent labels independently. They decompose the score of an unlabeled constituency tree into the scores of spans $s_{i,j}$:

$$s(T') = \sum_{ij} T'_{ij} s_{ij}$$

They use the CYK algorithm to recover the optimal **unlabeled** tree in the first stage and then use a separate component to label spans, including the $\emptyset$ label, in the second stage. Zhang et al. (2020c) show that adopting the two-stage parsing strategy leads to a better result in constituency parsing.

2.2 PO-TreeCRF

PO-TreeCRF (Fu et al., 2020) adapts a one-stage constituency parser to tackle nested NER. It views the set of entities $\boldsymbol{y} := \{(i, j, k), \dots\}$ as observed parts of a constituency tree T where (i, j) is the unlabeled entity span and k is the entity label. We refer to other constituents as latent spans. A labeled tree T is compatible with $\boldsymbol{y}$ if $T_{ijk} = 1$ for any entity $(i, j, k) \in \boldsymbol{y}$ and $T_{ij0} = 1$ for all latent spans (i, j) (recall that the 0-th label is $\emptyset$). Define set $\tilde{\mathcal{T}}(\boldsymbol{y})$ as all compatible trees with $\boldsymbol{y}$. PO-TreeCRF maximizes the total likelihood of all compatible trees:

$$s(\boldsymbol{y}) = \log \sum_{T \in \tilde{\mathcal{T}}(\boldsymbol{y})} \exp(s(T))$$

$$\log p(\boldsymbol{y}) = s(\boldsymbol{y}) - \log Z$$

where $\log Z$ is the log-partition function. The difficulty is how to estimate $s(\boldsymbol{y})$ efficiently. Fu et al. (2020) propose the masked inside algorithm to tackle this, in which they set all incompatible span (overlapped but not nested with any of $\boldsymbol{y}$) values to negative infinity before running the inside algorithm. We refer readers to their paper for more details.

2.3 Lexicalized Parsing

Figure 1 shows an example lexicalized constituency tree. We omit all constituent labels for brevity. Each constituent is annotated by a headword. A non-leaf constituent span consists of two adjacent sub-constituents and copies the headword from one of them. We refer to the copied headword as the inherited head and the other headword as the non-inherited head. We can draw a dependency arc from the inherited head to the non-inherited head. A dependency tree can be obtained by reading off all headwords recursively, and hence in this view, a lexicalized constituency tree embeds a dependency tree and a constituency tree.

The $O(n^4)$ Eisner-Satta algorithm (Eisner and Satta, 1999) can be used to calculate the partition function or obtain the best parse if we decompose the score of a lexicalized constituency tree into scores of spans and arcs. We refer interested readers to Appendix A for details of the Eisner-Satta algorithm.

3 Model

Notations Given a length-n sentence $\boldsymbol{x} = x_0, ..., x_{n-1}$ with (gold) entity set $\boldsymbol{y} := \{(i, j, \boldsymbol{\Omega}), \dots\}$, where (i, j) is an unlabeled entity span and $\boldsymbol{\Omega}$ is the set of entity labels (there could be multiple labels for one entity). We denote $\tilde{\boldsymbol{y}}$ as the set of unlabeled entity spans, i.e., $\tilde{\boldsymbol{y}} := \{(i, j), \dots\}$.

3.1 Two-stage Strategy and Training Loss

The first stage always predicts $2n - 1$ spans[1] and most of them are not entities. Hence naively adopting the two-stage parsing strategy to nested NER suffers from the imbalanced classification problem when predicting labels in the second stage because the $\emptyset$ label would dominate all the entity labels. To bypass this problem, we modify unlabeled constituency trees by assigning 0-1 labels to unlabeled constituency trees, where 0 stands for latent spans and 1 stands for entities. It transfers the burden of identifying non-entities to the first stage, in which the binary classification problem is much more balanced and easier to tackle. The total training loss can be decomposed into:

$$L = L_{\text{tree}} + L_{\text{label}} + L_{\text{reg}}$$

where L_{tree} is a 0-1 labeled constituency tree loss, L_{label} is a head-aware labeling loss and L_{reg} is a regularization loss based on the KL divergence.

3.2 Stage I: Structure Module

Encoding and scoring We feed the sentence into the BERT encoder (Devlin et al., 2019), apply

[1] A binary (lexicalized) constituency tree consists of exactly $2n - 1$ constituents.

scalar mixing (Peters et al., 2018) to the last four layers of BERT, and apply mean-pooling to all subword embeddings to obtain word-level contextual embedding. We concatenate static word embedding, e.g., GloVe (Pennington et al., 2014), to the contextual embedding to obtain the word representation $a = a_0, .., a_{n-1}$. Then we feed a into a three-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997) network (BiLSTM):

$$\dots, (\overrightarrow{b_i}, \overleftarrow{b_i}), \cdots = \text{BiLSTM}([\dots, a_i, \dots])$$

Next, we use deep biaffine scoring functions (Dozat and Manning, 2017) to calculate span scores $s^c \in \mathbb{R}^{n\times n\times 2}$ and arc scores $s^d \in \mathbb{R}^{n\times n}$:

$$\begin{aligned}
e_i^{c,in/out} &= \text{MLP}^{c,in/out}([\overrightarrow{b_i}; \overleftarrow{b_{i+1}}]) \\
e_i^{d,in/out} &= \text{MLP}^{d,in/out}([\overrightarrow{b_i}; \overleftarrow{b_i}]) \\
s_{ij}^c &= \text{PN}([e_i^{c,in}; \mathbf{1}]^T W^c [e_j^{c,out}; \mathbf{1}]) \\
s_{ij}^d &= \text{PN}([e_i^{d,in}; \mathbf{1}]^T W^d [e_j^{d,out}; \mathbf{1}]),
\end{aligned}$$

where MLPs are multi-layer perceptrons that project embeddings into k-dimensional spaces; $W^c \in \mathbb{R}^{(k+1)\times 2\times (k+1)}, W^d \in \mathbb{R}^{(k+1)\times (k+1)}$ are trainable parameters; PN is Potential Normalization, which normalizes scores to follow unit Gaussian distributions and has been shown beneficial (Fu et al., 2020).

Scores of trees A 0-1 labeled lexicalized constituency tree l embeds an unlabeled dependency tree d and a 0-1 labeled constituency tree c. The label set is $\{0, 1\}$, where 0 denotes latent spans and 1 denotes entity spans. We use a binary rank-3 tensor $C \in R^{n\times n\times 2}$ to represent c, where $C_{ijk} = 1$ if and only if there is a span from x_i to x_j with label k in c; and a binary matrix $D \in \mathbb{R}^{n\times n}$ to represent d, where $D_{ij} = 1$ if and only if there is an arc from x_i to x_j in d. We define the score of l as :

$$\begin{aligned}
s(l) &= s(c) + s(d) \\
&= \sum_{ijk} C_{ijk} s_{ijk}^c + \sum_{ij} D_{ij} s_{ij}^d
\end{aligned}$$

Structural tree loss We marginalize all latent spans and arcs out to define the loss:

$$\begin{aligned}
s(\tilde{\boldsymbol{y}}) &= \log \sum_{\tilde{T}\in\tilde{\mathcal{T}}} \exp(s(\tilde{T})) \\
L_{\text{tree}} &= \log Z - s(\tilde{\boldsymbol{y}})
\end{aligned}$$

where $\tilde{\mathcal{T}}$ is the set of all compatible lexicalized trees whose constituents contain $\tilde{\boldsymbol{y}}$; $\log Z$ is the log-partition function that can be estimated by the Eisner-Satta algorithm. For each compatible tree $\tilde{T} \in \tilde{\mathcal{T}}$, the 0-1 labels are assigned in accordance with the entity spans in $\tilde{\boldsymbol{y}}$. We use a masked version of the Eisner-Satta algorithm (Appendix A) to estimate $s(\tilde{\boldsymbol{y}})$.

Regularization loss As previously discussed, entity heads govern only one entity in most cases. But imposing a hard constraint is sub-optimal because there are also cases violating this constraint. Hence we want to encourage the model to satisfy this constraint in a soft manner. Inspired by posterior regularization (Ganchev et al., 2010; Zhang et al., 2017), we build a constrained TreeCRF and minimize the KL divergence between constrained and original unconstrained TreeCRFs. The first problem is how to construct the constrained TreeCRF. We propose to "hack" the forward pass (i.e., inside) of the Eisner-Satta algorithm to achieve this: we decrease the arc scores by a constant value (we typically set to 0.4) whenever the parent has already governed an entity during computing the inside values, so it discourages a head having several children and thus governing several spans. We refer readers to Appendix A for more details. The second problem is how to optimize the KL divergence efficiently for exponential numbers of trees. We adopt the specific semiring designed to calculate KL divergences between structured log-linear models (Li and Eisner, 2009) from the Torch-Struct library (Rush, 2020) [2]. The calculation of KL divergence is fully differentiable and thus is amenable to gradient-based optimization methods. It has the same time complexity as the forward pass of the Eisner-Satta algorithm. We denote the value of KL divergence as L_{reg}.

3.3 Stage II: Labeling Module

To incorporate entity head information when labeling entity spans, we score the assignment of label $l \in \mathcal{L}$ to a span (i, j) with head x_k as follows:

$$\begin{aligned}
e_i^{l,in/out} &= \text{MLP}^{l,in/out}([\overrightarrow{b_i}; \overleftarrow{b_{i+1}}]) \\
e_i^{l,head} &= \text{MLP}^{l,head}([\overrightarrow{b_i}; \overleftarrow{b_i}]) \\
s_{ijkl}^{label} &= \text{TriAff}(e_i^{l,in}, e_j^{l,out}, e_k^{l,head}),
\end{aligned}$$

where Triaff is the triaffine scoring function (Zhang et al., 2020b); $\mathcal{L}$ is the set of all labels. We reuse the encoder (BiLSTM) from Stage I.

[2] https://github.com/harvardnlp/pytorch-struct/blob/master/torch_struct/semirings/semirings.py

Nested named entities could have multiple labels. For instance, 7% entity spans in the NNE dataset (Ringland et al., 2019) have multiple labels. We use a multilabel loss introduced by Su (2020). For each $(i, j, \mathbf{\Omega}) \in \boldsymbol{y}$, consider a potential head x_k with $i \leq k \leq j$, we define the loss as:

$$l(i,j,k,\mathbf{\Omega}) = \log(1 + \sum_{l \in \mathcal{L}/\mathbf{\Omega}} \exp(s^{label}_{ijkl})) + \log(1 + \sum_{l \in \mathbf{\Omega}} \exp(-s^{label}_{ijkl}))$$

Since the gold entity heads are not given, we define the **head-aware labeling loss** based on expectation over the headword for each entity span:

$$L_{\text{label}} = \sum_{(i,j,\mathbf{\Omega}) \in \boldsymbol{y}} \sum_{i \leq k \leq j} \alpha_{ijk} l(i,j,k,\mathbf{\Omega})$$

where α_{ijk} is the marginal likelihood of x_k being the headword of span (i, j) under the TreeCRF, which satisfies $\sum_{i \leq k \leq j} \alpha_{ijk} = 1$ and can be estimated efficiently via the backward pass (i.e., backpropagation (Eisner, 2016)) of the Eisner-Satta algorithm.

4 Experiment

4.1 Setup

We conduct experiments on four datasets: ACE2004 (Doddington et al., 2004), ACE2005 (Walker, Christopher et al., 2006), GENIA (Kim et al., 2003) and NNE (Ringland et al., 2019). For ACE2004, ACE2005 and GENIA, we use the same data splitting and preprocessing as in Shibuya and Hovy (2020)[3]. For NNE, we use the official preprocessing script[4] to split train/dev/test sets. We refer readers to Appendeix B.1 for implementation details and to Appendix B.2 for data statistics of each dataset. We report span-level labeled precision (P), labeled recall (R) and labeled F1 scores (F1). We select models according to the performance on development sets. All results are averaged over three runs with different random seeds.

4.2 Main Result

We show the comparison of various methods on ACE2004, ACE2005 and GENIA in Table 1. We note that there is an inconsistency in the data prepossessing. For instance, the data statistics shown in Table 1 of (Shibuya and Hovy, 2020) and Table 5 of (Shen et al., 2021) do not match. More seriously, we find Shen et al. (2021); Tan et al. (2021) use context sentences, which plays a crucial role in their performance improvement but is not standard practice in other work. In addition, they report the best result instead of the mean result. Hence we rerun the open-sourced codes of Shen et al. (2021); Tan et al. (2021) using our preprocessed data and no context sentences and we report their mean results over three different runs. We also rerun the code of PO-TreeCRF for a fair comparison.

We can see that our method outperforms PO-TreeCRF, our main baseline, by 0.30/2.42/0.64 F1 scores on the three datasets, respectively. Our method has 87.90 and 86.91 F1 scores on ACE2004 and ACE2005, achieving the state-of-the-art performances. On GENIA, our method achieves competitive performance.

We also evaluate our method on the NNE dataset, whereby there are many multilabeled entities. Table 2 shows the result: our method outperforms Pyramid by 0.27 F1 score.

5 Analysis

5.1 Ablation Studies

We conduct a thorough ablation study of our model on the ACE2005 test set. Table 3 shows the result.

Structured vs. unstructured We study the effect of structural training and structured decoding as a whole. "Unstructured" is a baseline that adopts the local span classification loss and local greedy decoding. "1-stage" is our re-implementation of PO-TreeCRF, which adopts the latent structural constituency tree loss and uses the CYK algorithm for decoding. "1-stage+LEX" adopts the latent structural lexicalized constituency tree loss and uses the Eisner-Satta algorithm for decoding. All methods use the same neural encoders. We can see that "1-stage" outperforms the unstructured baseline by 0.33 F1 score. Further, "1-stage+LEX" outperforms "1-stage" by 0.25 F1 score, verifying the effectiveness of using latent lexicalized constituency tree structures.

1-stage vs. 2-stage On the unstructured model, we adopt a 0-1 local span classification loss in the first stage of the two-stage version, and we observe that the two-stage version performs similarly the

[3] https://github.com/yahshibu/nested-ner-tacl2020-transformers

[4] https://github.com/nickyringland/nested_named_entities/tree/master/ACL2019%20Paper

Model		ACE2004			ACE2005			GENIA		
		P	R	F1	P	R	F1	P	R	F1
Comparable										
SH		-	-	-	83.30	84.69	83.99	77.46	76.65	77.05
Pyramid-Basic		86.08	86.48	86.28	83.95	85.39	84.66	78.45	78.94	79.19
W(max)		86.27	85.09	85.68	85.28	84.15	84.71	79.20	78.16	78.67
PO-TreeCRFs†		87.62	87.57	87.60	83.34	85.67	84.49	79.10	76.53	77.80
Seq2set†		87.05	86.26	86.65	83.92	84.75	84.33	78.33	76.66	77.48
Locate&Label†		87.27	86.61	86.94	86.02	85.62	85.82	76.80	79.02	77.89
BARTNER		87.27	86.41	86.84	83.16	86.38	84.74	78.57	79.3	78.93
Ours		87.39	88.40	87.90	85.97	87.87	86.91	78.39	78.50	78.44
For reference										
SH	[F]	-	-	-	83.83	84.87	84.34	77.81	76.94	77.36
Pyramid-Full	[A]	87.71	87.78	87.74	85.30	87.40	86.34	-	-	-
PO-TreeCRFs	[D]	86.7	86.5	86.6	84.5	86.4	85.4	78.2	78.2	78.2
Seq2set	[C,P,D]	88.46	86.10	87.26	87.48	86.63	87.05	82.31	78.66	80.44
Locate&Label	[C,P,D]	87.44	87.38	87.41	86.09	87.27	86.67	80.19	80.89	80.54

Table 1: Results on ACE2004, ACE2005 and GENIA. SH: Shibuya and Hovy (2020); Pyramid-Basic/Full: Wang et al. (2020)[5]; W(max/logsumexp): Wang et al. (2021)[6]; PO-TreeCRFs: Fu et al. (2020); Seq2set: Tan et al. (2021) ; Locate&Label: Shen et al. (2021); BARTNER: Yan et al. (2021). Labels in square brackets stand for the reasons of the results being incomparable to ours. F: +Flair; A: +ALBERT, C: context sentences, P: POS tags, D: different data preprocessing. † denotes that we rerun their open-sourced codes using our data.

Model	NNE		
	P	R	F1
Pyramid-Basic	93.97	94.79	94.37
Ours	94.32	94.97	94.64

Table 2: Results on NNE.

one-stage version. On the other hand, we observe improvements on structured methods: "2-stage" outperforms "1-stage" by 0.23 F1 score and "2-stage+LEX" outperforms "1-stage+LEX" by 0.18 F1 scores, validating the benefit of adopting the two-stage strategy. Moreover, "2-stage(0/1)+LEX" outperforms "2-stage+LEX" by 0.15 F1 score, suggesting the effectiveness of bypassing the imbalanced classification problem.

Effect of structural training and decoding We study the importance of structural training and decoding in a decoupled way here. "-parsing" denotes the case that we use the latent lexicalized constituency tree loss for training, while we do not use the Eisner-Satta algorithm for parsing and instead predict spans locally whenever their label score of 1 is greater than that of 0. We can see that it causes a performance drop of 0.49 F1 score, indicating the importance of structural decoding, i.e., parsing. It is also worth noting that "-parsing" outperforms the unstructured baseline by 0.42 F1 score, showing the benefit of structural training even without structural decoding.

Effect of head regularization We can see that using the regularization loss brings an improvement of 0.24 F1 score (86.32->86.56). In the case study (Section 5.2), we observe that some common errors are avoided because of this regularization.

Effect of head-aware labeling loss We can see that using the head-aware labeling loss brings an improvement of 0.30 F1 score (86.32 -> 86.62). When combined with the head regularization, we achieve further improvements because of more accurate head estimation (Appendix B.3).

5.2 Case Study

Table 4 shows example predictions of our models. In the first pair, "2-stage" predict reasonable structures (visualized in B.5), but fail to label entities, whereas "2-stage (0-1)" predicts further correct labels. The second pair shows that, by constrain-

[5]They did not report Pyramid-Full with BERT only. However, with BERT+ALBERT, Pyramid-Full only outperforms Pyramid-Basic with a small margin (< 0.1).

[6]The *max* and *logsumexp* versions are the best models for BERT only and BERT+Flair respectively.

Model	P	R	F1
Unstructured(1-stage)	83.76	87.17	85.43
Unstructured(2-stage)	84.23	86.62	85.41
1-stage	84.08	87.52	85.76
1-stage + LEX	84.26	87.83	86.01
2-stage	84.68	87.33	85.99
2-stage + LEX	84.60	87.80	86.17
2-stage (0-1) + LEX	84.83	87.87	86.32
- parsing	84.26	87.40	85.83
+ head regularization	85.84	87.30	86.56
+ head-aware labeling	85.50	87.77	86.62
+ both (our final model)	**85.97**	**87.87**	**86.91**

Table 3: Ablation studies on the ACE2005 test set. LEX represents lexicalized structures.

ing head sharing and head-aware entity labeling, "+both" successfully detect bus as a headword, then produce correct entity boundaries and labels. Besides, "+both" can be seen to handle both fine-grained and coarse-grained entities in the last two predictions: *this bus near the airport* is predicted into two entities but *all sites and people in Iraq* remains one multilabeled entity.

Table 5 gives the most common headwords of each type predicted by our model on ACE2005. We find that the most frequently predicted headwords are gold headwords[7], except for some common function words, e.g., *in* and *of*. It proves the ability of our model in recognizing headwords.

5.3 Speed Comparison

One concern regarding our method is that since the Eisner-Satta algorithm has a $O(n^4)$ theoretical time complexity, it would be too slow to use for NER practitioners. Fortunately, the Eisner-Satta algorithm is amenable to highly-parallelized implementation so that $O(n^3)$ out of $O(n^4)$ can be computed in parallel (Zhang et al., 2020b; Rush, 2020), which greatly accelerates parsing. Empirically, we observe linear running time on GPUs in most cases. We show the comparison of (both training and decoding) running time in Table 6. We measure the time on a machine with Intel Xeon Gold 6278C CPU and NVIDIA V100 GPU.

We can see that compared with PO-TreeCRF, which also uses a highly-parallelized implementation of the $O(n^3)$ CYK algorithm, our method is around 20% slower in training and decoding, which is acceptable. Notably, both PO-TreeCRF and our method are much faster than Seq2Set (Tan et al., 2021) and Locate&Label(Shen et al., 2021).

[7] ACE2005 is additionally annotated with headwords. We only use them for evaluation.

6 Related Work

Nested NER Nested NER has been receiving increasing attentions and there are many methods proposed to tackle it. We roughly categorize the methods into the following groups: (1) Span-based methods: Luan et al. (2019); Yu et al. (2020); Li et al. (2021) directly assign scores to each potential entity span. (2) Layered methods: Ju et al. (2018); Fisher and Vlachos (2019) dynamically merge subspans to larger spans and Shibuya and Hovy (2020); Wang et al. (2021) use linear-chain CRFs and recursively find second-best paths for predicting nested entities. (3) Hypergraph-based methods: Lu and Roth (2015); Katiyar and Cardie (2018) propose different hypergraph structures to model nested entities but suffer from the spurious structure issue, and Wang and Lu (2018) solve this issue later. (4) Object-detection-based methods: Shen et al. (2021) adapt classical two-stage object detectors to tackle nested NER and Tan et al. (2021) borrow the idea from DETR (Carion et al., 2020). (5) Parsing-based methods (Finkel and Manning, 2009; Wang et al., 2018; Fu et al., 2020). (6) Sequence-to-sequence methods (Yan et al., 2021).

Our method belongs to parsing-based methods. Finkel and Manning (2009) use a non-neural TreeCRF parser. Wang et al. (2018) adapt a shift-reduce transition-based parser. Fu et al. (2020) use a span-based neural TreeCRF parser. All of them cast nested NER to constituency parsing, while we cast nested NER to lexicalized constituency parsing and our method is thus able to model entity heads.

Structured models using partial trees Full gold parse trees are expensive to obtain, so there are many methods proposed to marginalize over latent parts of partial trees, performing either approximate marginalization via loopy belief propagation or other approximate algorithms (Naradowsky et al., 2012; Durrett and Klein, 2014) or exact marginalization via dynamic programming algorithms (Li et al., 2016; Zhang et al., 2020b; Fu et al., 2020; Zhang et al., 2021). Naradowsky et al. (2012); Durrett and Klein (2014) construct factor graph representations of syntactically-coupled NLP tasks whose structures can be viewed as latent dependency or constituency trees, such as

Model	Prediction
2-stage	[I]$^{\text{PER}}$ have never heard of [a pig like [this]$^{\text{WEA}}$]$^{\text{WEA}}$ before !
2-stage (0-1)‡	[I]$^{\text{PER}}$ have never heard of a pig like this before !
2-stage (0-1)	[Police]$^{\text{PER}}$ surrounded [this bus near [the airport]$^{\text{FAC}}$]$^{\text{VEH,FAC}}$ with [guns]$^{\text{WEA}}$ drawn .
+ both‡	[Police]$^{\text{PER}}$ surrounded [this bus]$^{\text{VEH}}$ near [the airport]$^{\text{FAC}}$ with [guns]$^{\text{WEA}}$ drawn .
+ both‡	[Blix]$^{\text{PER}}$ stressed that [council]$^{\text{ORG}}$ resolutions call for [[U.N.]$^{\text{ORG}}$ inspectors]$^{\text{PER}}$ to have access to [all sites and people in [Iraq]$^{\text{GPE}}$]$^{\text{FAC,PER}}$.

Table 4: Two sentences with predicted entity decorated. Blue entities are correct and red entities are wrong. The underlined words are the entity heads. Models annotated with ‡ predict all entities correctly.

Type	Most Frequent Headwords
PER	you, I, he, they, i, his, of, their, we, who
LOC	world, of, area, there, coast, where, beach, desert, Southeast, that
ORG	we, they, Starbucks, its, court, company, military, of, their, companies
GPE	U.S., Indonesia, Baghdad, city, state, Russian, we, country, Iraqi, where
FAC	airport, house, jail, in, prison, street, of, it, hospital, home
VEH	of, car, in, aircraft, that, bus, plane, lincoln, deck, its
WEA	gun, weapons, arms, guns, firearms, missile, bullet, knife, rifles, Kalashnikov

Table 5: The most common (top 10) headwords of each entity type predicted by our method on the ACE2005 test set. Red words are not headwords in the gold annotation.

Model	Train	Sents/sec
PO-TreeCRF	2m1s	205
2-stage	2m15s	184
2-stage + LEX	2m23s	173
Seq2set	3m24s	122
Locate&Label	4m23s	94

Table 6: Speed comparison for training one epoch on ACE2005.

NER, semantic role labeling (SRL), and relation extraction. Li et al. (2016); Zhang et al. (2020b) perform partial marginalization to train (second-order) TreeCRF parsers for partially-annotated dependency parsing. Zhang et al. (2021) view arcs in SRL as partially-observed dependency trees; Fu et al. (2020) view entities in nested NER as partially-observed constituency trees; and we view entities in nested NER as partially-observed lexicalized constituency trees in this work.

Lexicalized parsing Probabilistic context-free grammars (PCFGs) have been widely used in syntactic parsing. Lexicalized PCFGs (L-PCFGs) leverage headword information to disambiguate parsing and are thus more expressive. Eisner and Satta (1999) propose an efficient $O(n^4)$ algorithm for lexicalized parsing. Collins (2003) conduct a thorough study of lexicalized parsing. Recently, neurally parameterized L-PCFGs have been used in unsupervised joint dependency and constituency parsing (Zhu et al., 2020; Yang et al., 2021). Our work removes the grammar components and adapts the dynamic programming algorithm of lexicalized parsing (Eisner and Satta, 1999) in the spirit of span-based constituency parsing (Stern et al., 2017).

7 Conclusion

We have presented a parsing-based method for nested NER, viewing entities as partially-observed lexicalized constituency trees, motivated by the close relationship between entity heads and entity recognition. Benefiting from structural modeling, our model does not suffer from error propagation and heuristic head choosing and is easy for regularizing predictions. Furthermore, our highly-parallelized implementation enables fast training and inference on GPUs. Experiments on four benchmark datasets validate the effectiveness and efficiency of our proposed method.

References

Kate Byrne. 2007. Nested named entity recognition in historical archive text. In *International Conference on Semantic Computing (ICSC 2007)*, pages 589–596.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer.

Pei Chen, Haibo Ding, Jun Araki, and Ruihong Huang. 2021. Explicitly capturing relations between entity mentions via graph neural networks for domain-specific named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 735–742, Online. Association for Computational Linguistics.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical NLP. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174, Berlin, Germany. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

J. Cocke. 1969. Programming languages and their compilers: Preliminary notes.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Xiang Dai. 2018. Recognizing complex entity mentions: A review and future directions. In *Proceedings of ACL 2018, Student Research Workshop*, pages 37–44, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the association for computational linguistics*, 2:477–490.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX. Association for Computational Linguistics.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464, College Park, Maryland, USA. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Multitask pointer network for multi-representational parsing. *CoRR*, abs/2009.09730.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 141–150, Singapore. Association for Computational Linguistics.

Joseph Fisher and Andreas Vlachos. 2019. Merge and label: A novel neural network architecture for nested NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5840–5850, Florence, Italy. Association for Computational Linguistics.

Yao Fu, Chuanqi Tan, Mosha Chen, Songfang Huang, and Fei Huang. 2020. Nested named entity recognition with partially-observed treecrfs.

Haim Gaifman. 1965. Dependency systems and phrase-structure systems. *Inf. Control.*, 8(3):304–337.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning Research*, 11(67):2001–2049.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana. Association for Computational Linguistics.

Tadao Kasami. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus–a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl 1):i180–i182.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Fei Li, ZhiChao Lin, Meishan Zhang, and Donghong Ji. 2021. A span-based model for joint overlapped and discontinuous named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4814–4828, Online. Association for Computational Linguistics.

Zhenghua Li, Min Zhang, Yue Zhang, Zhanyi Liu, Wenliang Chen, Hua Wu, and Haifeng Wang. 2016. Active learning for dependency parsing with partial annotation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 344–354, Berlin, Germany. Association for Computational Linguistics.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51, Singapore. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192, Florence, Italy. Association for Computational Linguistics.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal. Association for Computational Linguistics.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Jason Naradowsky, Sebastian Riedel, and David A Smith. 2012. Improving nlp through marginalization of hidden syntactic structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 810–820.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Nicky Ringland, Xiang Dai, Ben Hachey, Sarvnaz Karimi, Cecile Paris, and James R. Curran. 2019. NNE: A dataset for nested named entity recognition in English newswire. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5176–5181, Florence, Italy. Association for Computational Linguistics.

Alexander Rush. 2020. Torch-struct: Deep structured prediction library. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342, Online. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021. Locate and

label: A two-stage identifier for nested named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2782–2794, Online. Association for Computational Linguistics.

Takashi Shibuya and Eduard Hovy. 2020. Nested named entity recognition via second-best sequence learning and decoding. *Transactions of the Association for Computational Linguistics*, 8:605–620.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Jianlin Su. 2020. Extend "softmax+cross entropy" to multi-label classification problem.

Zeqi Tan, Yongliang Shen, Shuai Zhang, Weiming Lu, and Yueting Zhuang. 2021. A sequence-to-set network for nested named entity recognition. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI-21*.

Walker, Christopher, Strassel, Stephanie, Medero, Julie, and Maeda, Kazuaki. 2006. ACE 2005 Multilingual Training Corpus. Type: dataset.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 204–214, Brussels, Belgium. Association for Computational Linguistics.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1011–1017, Brussels, Belgium. Association for Computational Linguistics.

Jue Wang, Lidan Shou, Ke Chen, and Gang Chen. 2020. Pyramid: A layered model for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5918–5928, Online. Association for Computational Linguistics.

Yiran Wang, Hiroyuki Shindo, Yuji Matsumoto, and Taro Watanabe. 2021. Nested named entity recognition via explicitly excluding the influence of the best path. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3547–3557, Online. Association for Computational Linguistics.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5808–5822, Online. Association for Computational Linguistics.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021. Neural bi-lexicalized PCFG induction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2688–2699, Online. Association for Computational Linguistics.

D. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Inf. Control.*, 10:189–208.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2020a. Fast interleaved bidirectional sequence generation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 503–515, Online. Association for Computational Linguistics.

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1514–1523, Vancouver, Canada. Association for Computational Linguistics.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020b. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

Yu Zhang, Qingrong Xia, Shilin Zhou, Yong Jiang, Zhenghua Li, Guohong Fu, and Min Zhang. 2021. Semantic role labeling as dependency parsing: Exploring latent tree structures inside arguments. *ArXiv*, abs/2110.06865.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020c. Fast and accurate neural crf constituency parsing. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

Zhisong Zhang, Xiang Kong, Zhengzhong Liu, Xuezhe Ma, and Eduard Hovy. 2020d. A two-step approach for implicit event argument detection. In *Proceedings of the 58th Annual Meeting of the Association*

for Computational Linguistics, pages 7479–7485, Online. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

Hao Zhu, Yonatan Bisk, and Graham Neubig. 2020. The return of lexical dependencies: Neural lexicalized PCFGs. *Transactions of the Association for Computational Linguistics*, 8:647–661.

A Details of the Eisner-Satta algorithm

Table 7 describes the Eisner-Satta algorithm in the parsing-as-deduction framework. Each deductive rule of the Eisner-Satta algorithm has only one word participating in the computation in addition, e.g., p and h, resulting in one-order higher than the CYK algorithm.

The masked version of the Eisner-Satta algorithm masks scores similar to PO-TreeCRF except for different label sets in our model "2-stage" and "2-stage (0/1)". For the construction of constrained trees, we introduce a minor penalty (0.4 in our paper) on type I items' scores if the item represents a gold entity. We show the pseudocode of the standard Eisner-Satta algorithm, the masked version of the Esiner-Satta algorithm and the construction of constrained trees all in Algorithm 1.

B Experiments

B.1 Implementation Details

We use BERT (bert-large-cased) and GloVe (6B-100d) to obtain word representations for ACE2004, ACE2005, and NNE. For GENIA, we use BioBERT (biobert-large-cased-v1.1) (Lee et al., 2019) and BioWordvec (Chiu et al., 2016) instead to match its domain. The hidden size of BiLSTM is set to 400. We use an Adam optimizer (Kingma and Ba, 2015) and a linear learning rate scheduler. We warm up training for 2 epochs and decay learning rates to 0 linearly for the rest of the epochs. The peak learning rates are $5e-5$ for BERT/BioBERT and $5e-3$ for the other parts of the neural networks.

B.2 Data statistics

Table 9 shows the statistics of ACE2004, ACE2005, GENIA and NNE. We report the number of multi-labeled entities and single-word entities in addition.

Items:

I $[i, j, h, -]$: span $[i, j]$ is headed by word w_h and its parent is not determined. $i \leq h \leq j$.

II $[i, j, -, p]$: span $[i, j]$ is headed by arbitrary word w_h. The common parent is w_p. $p < i$ or $k < p$.

Axiom items: $[i, i, i, -], 1 \leq i \leq n$
Goal items: $[1, n, r, -], 1 \leq r \leq n$
Deductive rules:

I $\dfrac{[i, k, h, -]}{[i, k, -, p]}$ attach left/right

II $\dfrac{[i, j, -, p] \quad [j+1, k, p, -]}{[i, k, p, -]}$ complete left

III $\dfrac{[i, j, p, -] \quad [j+1, k, -, p]}{[i, k, p, -]}$ complete right

Table 7: The Eisner-Satta algorithm described in the parsing-as-deduction framework.

	PER	LOC	ORG	GPE	FAC	VEH	WEA
ρ	0.57	0.02	0.18	0.14	0.05	0.03	0.02
PER	0.92	0.00	0.06	0.03	0.01	0.03	0.00
LOC	0.00	0.74	0.00	0.02	0.01	0.01	0.00
ORG	0.02	0.00	0.83	0.02	0.03	0.02	0.00
GPE	0.00	0.07	0.03	0.87	0.04	0.00	0.00
FAC	0.00	0.06	0.01	0.00	0.77	0.04	0.00
VEH	0.00	0.00	0.00	0.00	0.01	0.73	0.00
WEA	0.00	0.00	0.00	0.00	0.01	0.00	0.90
$\emptyset$	0.06	0.13	0.08	0.06	0.12	0.18	0.10

Table 8: Error distribution on the ACE2005 test set normalized along with columns. ρ is the gold label distribution. Each row is a gold label and each column is a predicted label. $\emptyset$ denotes entities not recognized by our model.

B.3 Studies on Headwords

We conduct more experiments to analyze the behavior of head regularization. Table 10 shows the results of models trained with different penalty constants of the head regularization. $c = 0$ means no constraint applied, and larger c means harder constraint. We observe that too hard constraints (e.g., $c = 1$) are less effective than proper constraints (e.g., $c = 0.4$). We choose $c = 0.4$ as the penalty constant for experiments in the main body. Table 11 shows the results if we apply head regularization only when decoding. We observe that the overall performance changes marginally, although the number of shared heads is significantly reduced,

	ACE2004			ACE2005			GENIA			NNE		
	train	dev	test	train	dev	test	train	dev	test	train	dev	test
# sentences	6198	742	809	7285	968	1058	15022	1669	1855	43457	1989	3762
- nested	2718	294	388	2797	352	339	3222	328	448	28606	1292	2489
# entities	22195	2514	3034	24827	3234	3041	47006	4461	5596	248136	10463	21196
- nested	10157	1092	1417	9946	1191	1179	8382	818	1212	206618	8487	17670
- single-word	11527	1363	1553	13988	1852	1706	12933	1009	1392	166183	7291	14397
- multi-type	3	1	1	9	3	2	21	5	5	16769	792	1583

Table 9: Statistics of ACE2004, ACE2005, GENIA and NNE. An entity is considered nested if contains any entity or is contained by any entity. A sentence is considered nested if contains any nested entity.

possibly because the head accuracy is still low and the labeling module is trained to pay less attention to the headwords as they are noisy. Finally, we analyze the number of shared heads and the head accuracy for models trained with head regularization and head-aware entity labeling. Table 12 shows few shared heads and high head accuracy, consistent with the high overall performance. Besides, we observe that adding the head-aware entity labeling does not reduce the shared headwords much, showing the limitation of models to learn such prior knowledge.

B.4 Error Distribution

We report the error distribution in Table 8. Compared with PO-TreeCRF, we reduce the error rates off all extremely imbalanced classes (VEH, FAC, LOC and WEA).

B.5 Predicted Parse Tree

Here we draw the parse trees in 5.2. Fig. 2a shows a tree produced by "2-stage", which is reasonable. But the label module of "2-stage" fail to label spans correctly due to the label imbalance problem. "2-stage (0-1)" predict the same tree but correct labels. Fig. 2b shows a tree predicted by "2-stage (0-1)". The model fail to detect headwords, e.g., *bus* and *airport*. In contrast, Fig. 2c shows a tree predicted by "2-stage (0-1) + both", in which shared heads are much fewer and correct headwords are found.

Algorithm 1: The Eisner-Satta Algorithm

input: $s_c \in \mathbb{R}^{n \times n \times B}$ for span scores, where B is #sent in a batch
input: $s_d \in \mathbb{R}^{n \times n \times B}$ for arc scores
input: $enable\_soft\_constraint$ for whether enable the soft exclusive head constraint
input: $mask \in \mathbb{R}^{n \times n}$ for incompatible spans. (optional)
define: $H \in \mathbb{R}^{n \times n \times n \times B}$ for type I span in Table 7
define: $P \in \mathbb{R}^{n \times n \times n \times B}$ for type II span in Table 7
initialize: $H_{:,:,:} = -\infty, P_{:,:,:} = -\infty$
if *mask is given* **then**
  for all i, j, $s_c[i, j] = -\infty$ if $mask[i, j]$ is true.
end
for $i = 0$ **to** $n - 1$ **do**
  $H[i, i, i] = s_c[i, i]$
  for $j = 0$ **to** $n - 1$ **do**
    $P[i, i, j] = s_d[i, j] + H[i, i, i]$
  end
  if $enable\_soft\_constraint$ **then**
    $H[i, i, i] -= c$ `// c is a small positive constant (0.4 in our paper). // Equivalent to minus c for arcs headed by i.`
  end
end
for $w = 1$ **to** $n - 1$ **do**
  for $i = 0$ **to** $n - w - 1$ **do**
    $j = i + w$
    for $h = i$ **to** j **do**
      $H[i, j, h] = s_c[i, j] + \log \sum_{r \in [i,j]} [\exp(P[i, r, h] + H[r + 1, j, h]) + \exp(H[i, r, h] + P[r + 1, j, h])]$ `// complete left/right`
    end
    for $p = 0$ **to** $n - 1$ **do**
      $P[i, j, p] = \log \sum_{h \in [i,j]} \exp(H[i, j, h] + s_d[h, p])$ `// attach left/right`
    end
    if $enable\_soft\_constraint$ **then**
      for $h = i$ **to** j **do**
        $H[i, j, h] -= c$
      end
    end
  end
end
return $H[0, n - 1, 0] \equiv \log Z$

c	0	0.1	0.2	0.3	0.4	0.5	0.6
F1	86.32	86.45	86.54	86.53	86.56	86.49	86.41

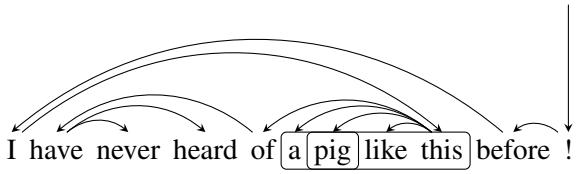
Table 10: The impact of different constants used to construct constrained trees for training on ACE2005. A higher value means harder constraints.

c	-2	0	0.2	0.4	0.6	1
F1	86.38	86.44	86.46	86.46	86.43	86.41
#shared	347	234	30	10	7	6
Head acc.	43.19	48.45	57.27	57.94	58.33	58.08

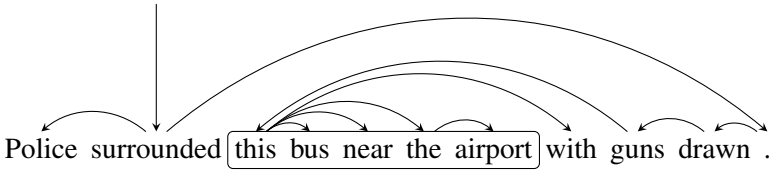
Table 11: Results of different constants when decoding. #shared denotes the number of entities having shared headwords. Models are trained without the head regularization. Head accuracy do not count single word spans. Results are of one run.

	0	0.4	0 + HA	0.4 + HA
#shared	234	73	216	10
Head acc.	48.45	59.42	73.58	81.00

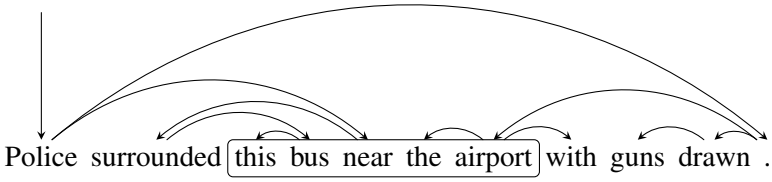
Table 12: Number of shared heads and head accuracy on the ACE2005 test set. HA means head-aware entity labeling. The head accuracy do not count single word spans. Results are of one run.



(a) A tree predicted by "2-stage". It produce reasonable structures, but the labeling module can not label them well.



(b) A tree predicted by "2stage (0-1)". It fails to detect "bus" and "airport" as headwords.



(c) A tree predicted by "2-stage (0-1) + both". It detect "bus" and "airport" as headwords correctly. The span *this bus near the airport* do not exist on the tree.

Figure 2: Predicted dependency trees. We highlight interesting spans.