arXiv:1911.03875v2 [cs.CL] 2 May 2020

Rethinking Self-Attention: Towards Interpretability in Neural Parsing

Khalil Mrini[1], **Franck Dernoncourt**[2], **Trung Bui**[2], **Walter Chang**[2], **and Ndapa Nakashole**[1]

[1] University of California, San Diego, La Jolla, CA 92093
khalil@ucsd.edu, nnakashole@eng.ucsd.edu
[2] Adobe Research, San Jose, CA 95110
{franck.dernoncourt, bui, wachang}@adobe.com

Abstract

Attention mechanisms have improved the performance of NLP tasks while allowing models to remain explainable. Self-attention is currently widely used, however interpretability is difficult due to the numerous attention distributions. Recent work has shown that model representations can benefit from label-specific information, while facilitating interpretation of predictions. We introduce the Label Attention Layer: a new form of self-attention where attention heads represent labels. We test our novel layer by running constituency and dependency parsing experiments and show our new model obtains new state-of-the-art results for both tasks on both the Penn Treebank (PTB) and Chinese Treebank. Additionally, our model requires fewer layers, therefore, fewer parameters compared to existing work. Finally, we find that the Label Attention heads learn relations between syntactic categories and show pathways to analyze errors.

1 Introduction

Attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015) provide arguably explainable attention distributions that can help to interpret predictions. For example, for their machine translation predictions, Bahdanau et al. (2014) show a heat map of attention weights from source language words to target language words. Similarly, in transformer architectures (Vaswani et al., 2017), a self-attention head produces attention distributions from the input words to the same input words, as shown in the second row of the right side of Figure 1. However, self-attention mechanisms have multiple heads, making the combined outputs difficult to interpret.

Recent work in multi-label text classification (Xiao et al., 2019) and sequence labeling (Cui and Zhang, 2019) shows the efficiency and interpretability of label-specific representations. We introduce the Label Attention Layer: a modified version of self-attention, where each classification label corresponds to one or more attention heads. We project the output at the attention head level, rather than after aggregating all outputs, to preserve the source of head-specific information, thus allowing us to match labels to heads.

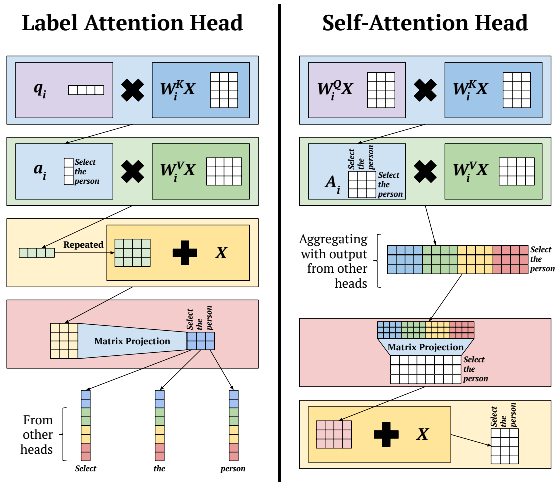


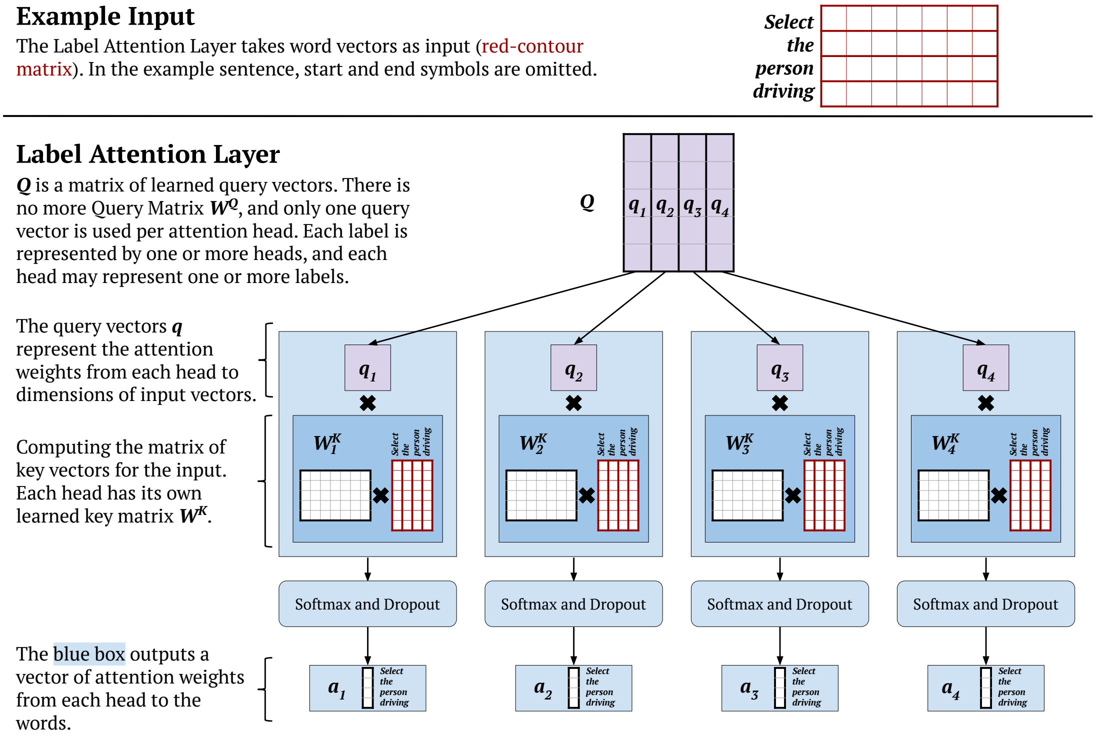
Figure 1: Comparison of the attention head architectures of our proposed Label Attention Layer and a Self-Attention Layer (Vaswani et al., 2017). The matrix **X** represents the input sentence "*Select the person*".

To test our proposed Label Attention Layer, we build upon the parser of Zhou and Zhao (2019) and establish a new state of the art for both constituency and dependency parsing. We also release our pre-trained parsers, as well as our code to encourage experiments with the Label Attention Layer[1].

2 Label Attention Layer

The self-attention mechanism of Vaswani et al. (2017) propagates information between the words of a sentence. Each resulting word representation contains its own attention-weighted view of the

[1] github.com/KhalilMrini/LAL-Parser



Example Input

The Label Attention Layer takes word vectors as input (red-contour matrix). In the example sentence, start and end symbols are omitted.

Label Attention Layer

Q is a matrix of learned query vectors. There is no more Query Matrix W^Q, and only one query vector is used per attention head. Each label is represented by one or more heads, and each head may represent one or more labels.

The query vectors *q* represent the attention weights from each head to dimensions of input vectors.

Computing the matrix of key vectors for the input. Each head has its own learned key matrix W^K.

The blue box outputs a vector of attention weights from each head to the words.

Figure 2: The architecture of the top of our proposed Label Attention Layer. In this figure, the example input sentence is "*Select the person driving*".

sentence. We hypothesize that a word representation can be enhanced by including each label's attention-weighted view of the sentence, on top of the information obtained from self-attention.

The Label Attention Layer (LAL) is a novel, modified form of self-attention, where only one query vector is needed per attention head. Each classification label is represented by one or more attention heads, and this allows the model to learn label-specific views of the input sentence. Figure 1 shows a high-level comparison between our Label Attention Layer and self-attention.

We explain the architecture and intuition behind our proposed *Label Attention Layer* through the example application of parsing.

Figure 2 shows one of the main differences between our Label Attention mechanism and self-attention: the absence of the Query matrix $\mathbf{W}^{\mathbf{Q}}$. Instead, we have a learned matrix $\mathbf{Q}$ of query vectors representing each head. More formally, for the attention head i and an input matrix $\mathbf{X}$ of word vectors, we compute the corresponding attention weights vector $\mathbf{a}_i$ as follows:

$$\mathbf{a}_i = \text{softmax}\left(\frac{\mathbf{q}_i * \mathbf{K}_i}{\sqrt{d}}\right) \tag{1}$$

where d is the dimension of query and key vectors, $\mathbf{K}_i$ is the matrix of key vectors. Given a learned head-specific key matrix $\mathbf{W}_i^K$, we compute $\mathbf{K}_i$ as:

$$\mathbf{K}_i = \mathbf{W}_i^K \mathbf{X} \tag{2}$$

Each attention head in our Label Attention layer has an attention *vector*, instead of an attention *matrix* as in self-attention. Consequently, we do not obtain a *matrix* of vectors, but a *single* vector that contains head-specific context information. This *context* vector corresponds to the green vector in Figure 3. We compute the context vector $\mathbf{c}_i$ of head i as follows:

$$\mathbf{c}_i = \mathbf{a}_i * \mathbf{V}_i \tag{3}$$

where $\mathbf{a}_i$ is the vector of attention weights in Equation 1, and $\mathbf{V}_i$ is the matrix of value vectors. Given a learned head-specific value matrix $\mathbf{W}_i^V$, we compute $\mathbf{V}_i$ as:

$$\mathbf{V}_i = \mathbf{W}_i^V \mathbf{X} \tag{4}$$

The context vector gets added to each individual input vector making for one residual connection per head, rather one for all heads, as shown in the yellow box in Figure 3. We project the resulting matrix of word vectors to a lower dimension before

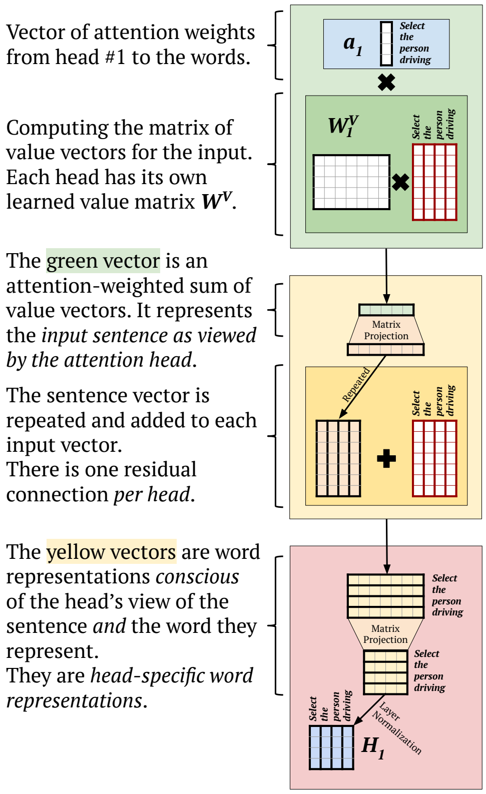

Vector of attention weights from head #1 to the words.
Computing the matrix of value vectors for the input. Each head has its own learned value matrix W^V.
The green vector is an attention-weighted sum of value vectors. It represents the *input sentence as viewed by the attention head.*
The sentence vector is repeated and added to each input vector.
There is one residual connection *per head.*
The yellow vectors are word representations *conscious* of the head's view of the sentence *and* the word they represent.
They are *head-specific word representations.*
a_1
Select the person driving
W_1^V
Select the person driving
Matrix Projection
Repeated
Select the person driving
Select the person driving
Matrix Projection
Select the person driving
Select the person driving
Layer Normalization
H_1

Figure 3: The Value vector computations in our proposed Label Attention Layer.

normalizing. We then distribute the word vectors computed by each label attention head, as shown in Figure 4.

We chose to assign as many attention heads to the Label Attention Layer as there are classification labels. As parsing labels (syntactic categories) are related, we did not apply an orthogonality loss to force the heads to learn separate information. We therefore expect an overlap when we match labels to heads. The values coming from each head are identifiable within the final word representation, as shown in the color-coded vectors in the middle of Figure 4.

The activation functions of the position-wise feed-forward layer make it difficult to follow the path of the contributions. Therefore we can remove the position-wise feed-forward layer, and compute the contributions from each label. We provide an example in Figure 6, where the contributions are computed using normalization and averaging. In this case, we are computing the contributions of each head to the span vector. The span representation for "*the person*" is computed following the method of Gaddy et al. (2018) and Kitaev and Klein (2018). However, forward and backward representations are not formed by splitting the entire word vector at the middle, but rather by splitting each head-specific word vector at the middle.

In the example in Figure 6, we show averaging as one way of computing contributions, other functions, such as softmax, can be used. Another way of interpreting predictions would be to look at the head-to-word attention distributions, which are the output vectors in the computation in Figure 2.

3 Syntactic Parsing Model

3.1 Encoder

Our parser is an encoder-decoder model. The encoder has self-attention layers (Vaswani et al., 2017), preceding the Label Attention Layer. We follow the attention partition of Kitaev and Klein (2018), who show that separating content embeddings from position embeddings improves performance.

Sentences are pre-processed following Zhou and Zhao (2019). Trees are represented using a simplified Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994). In Zhou and Zhao (2019), two kinds of span representations are proposed: the division span and the joint span. We choose the joint span representation as it is the best-performing one in their experiments. Figure 5 shows how the example sentence in Figure 2 is represented.

The token representations for our model are a concatenation of content and position embeddings. The content embeddings are a sum of word and part-of-speech embeddings.

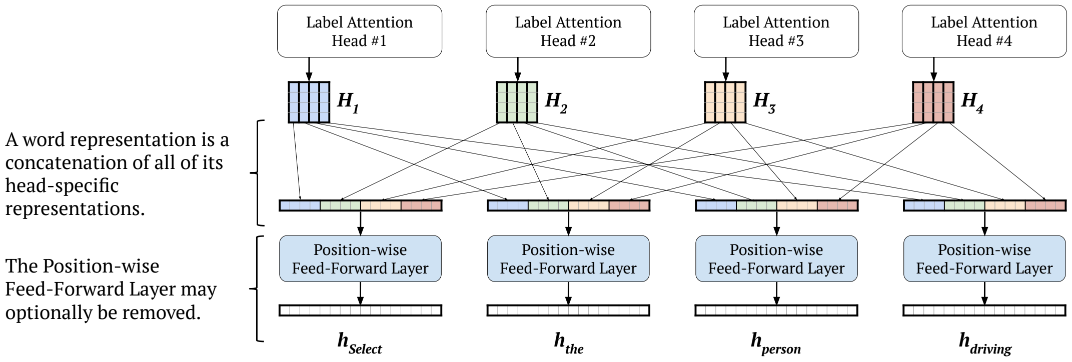
3.2 Constituency Parsing

For constituency parsing, span representations follow the definition of Gaddy et al. (2018) and Kitaev and Klein (2018). For a span starting at the i-th word and ending at the j-th word, the corresponding span vector s_{ij} is computed as:

$$\mathbf{s_{ij}} = \left[\overrightarrow{\mathbf{h_j}} - \overrightarrow{\mathbf{h_{i-1}}} ; \overleftarrow{\mathbf{h_{j+1}}} - \overleftarrow{\mathbf{h_i}}\right] \tag{5}$$

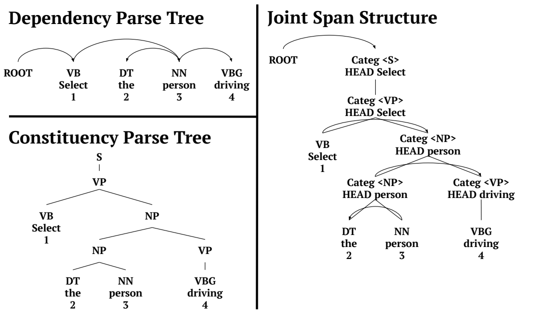
where $\overleftarrow{\mathbf{h_i}}$ and $\overrightarrow{\mathbf{h_i}}$ are respectively the backward and forward representation of the i-th word obtained by splitting its representation in half. An example of a span representation is shown in the middle of Figure 6.

The score vector for the span is obtained by applying a one-layer feed-forward layer:


A word representation is a concatenation of all of its head-specific representations.

The Position-wise Feed-Forward Layer may optionally be removed.

Figure 4: Redistribution of the head-specific word representations to form word vectors by concatenation. We use different colors for each label attention head. The colors show where the head outputs go in the word representations. We do not use colors for the vectors resulting from the position-wise feed-forward layer, as the head-specific information moved.


Dependency Parse Tree

Joint Span Structure

Constituency Parse Tree

Figure 5: Parsing representations of the example sentence in Figure 2.

$$\mathbf{S}(i,j) = \mathbf{W_2}\text{ReLU}(\text{LN}(\mathbf{W_1 s_{ij}} + \mathbf{b_1})) + \mathbf{b_2} \quad (6)$$

where LN is Layer Normalization, and $\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{b_1}$ and $\mathbf{b_2}$ are learned parameters. For the l-th syntactic category, the corresponding score $s(i,j,l)$ is then the l-th value in the $\mathbf{S}(i,j)$ vector.

Consequently, the score of a constituency parse tree T is the sum of all of the scores of its spans and their syntactic categories:

$$s(T) = \sum_{(i,j,l)\in T} s(i,j,l) \quad (7)$$

We then use a CKY-style algorithm (Stern et al., 2017; Gaddy et al., 2018) to find the highest scoring tree $\hat{T}$. The model is trained to find the correct parse tree T^*, such that for all trees T, the following margin constraint is satisfied:

$$s(T^*) \geq s(T) + \Delta(T, T^*) \quad (8)$$

where Δ is the Hamming loss on labeled spans. The corresponding loss function is the hinge loss:

$$L_c = \max\left(0, \max_T[s(T) + \Delta(T, T^*)] - s(T^*)\right) \quad (9)$$

3.3 Dependency Parsing

We use the biaffine attention mechanism (Dozat and Manning, 2016) to compute a probability distribution for the dependency head of each word. The child-parent score α_{ij} for the j-th word to be the head of the i-th word is:

$$\alpha_{ij} = \mathbf{h_i^{(d)}}^T \mathbf{W} \mathbf{h_j^{(h)}} + \mathbf{U}^T \mathbf{h_i^{(d)}} + \mathbf{V}^T \mathbf{h_j^{(h)}} + b \quad (10)$$

where $\mathbf{h_i^{(d)}}$ is the dependent representation of the i-th word obtained by putting its representation $\mathbf{h_i}$ through a one-layer perceptron. Likewise, $\mathbf{h_j^{(h)}}$ is the head representation of the j-th word obtained by putting its representation $\mathbf{h_j}$ through a separate one-layer perceptron. The matrices $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{V}$ are learned parameters.

The model trains on dependency parsing by minimizing the negative likelihood of the correct dependency tree. The loss function is cross-entropy:

$$L_d = -\log\left(P\left(h_i|d_i\right) P\left(l_i|d_i, h_i\right)\right) \quad (11)$$

where h_i is the correct head for dependent d_i, $P\left(h_i|d_i\right)$ is the probability that h_i is the head of d_i, and $P\left(l_i|d_i, h_i\right)$ is the probability of the correct dependency label l_i for the child-parent pair (d_i, h_i).

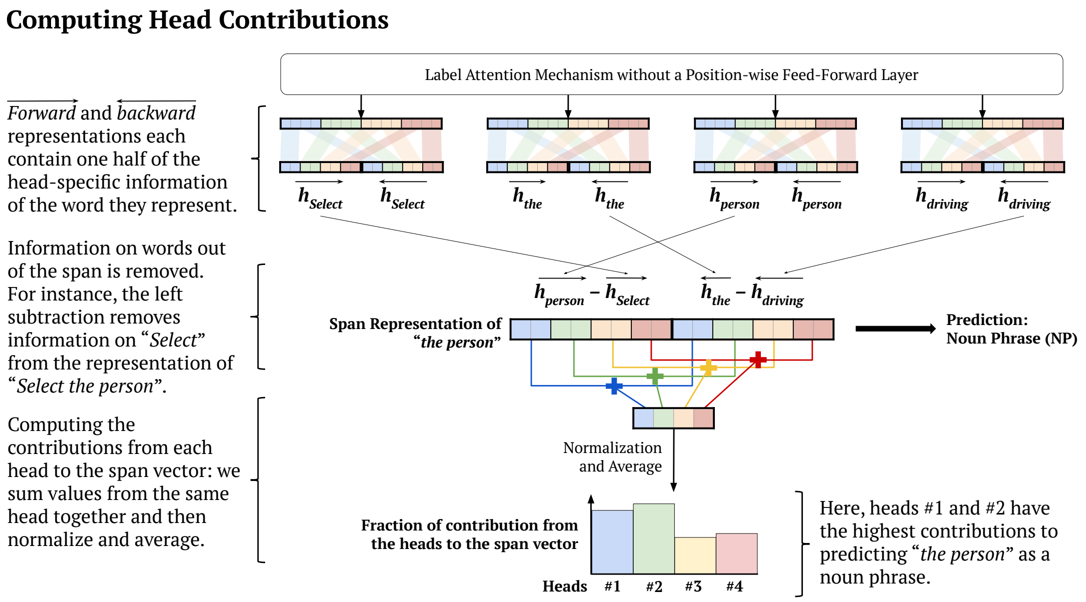

Computing Head Contributions
Label Attention Mechanism without a Position-wise Feed-Forward Layer
Forward and backward representations each contain one half of the head-specific information of the word they represent.
$\overrightarrow{h}_{Select}$ $\overleftarrow{h}_{Select}$ $\overrightarrow{h}_{the}$ $\overleftarrow{h}_{the}$ $\overrightarrow{h}_{person}$ $\overleftarrow{h}_{person}$ $\overrightarrow{h}_{driving}$ $\overleftarrow{h}_{driving}$
Information on words out of the span is removed. For instance, the left subtraction removes information on "Select" from the representation of "Select the person".
$\overrightarrow{h}_{person} - \overrightarrow{h}_{Select}$
$\overleftarrow{h}_{the} - \overleftarrow{h}_{driving}$
Span Representation of "the person"
Prediction: Noun Phrase (NP)
Computing the contributions from each head to the span vector: we sum values from the same head together and then normalize and average.
Normalization and Average
Fraction of contribution from the heads to the span vector
Heads #1 #2 #3 #4
Here, heads #1 and #2 have the highest contributions to predicting "the person" as a noun phrase.

Figure 6: If we remove the position-wise feed-forward layer, we can compute the contributions from each label attention head to the span representation, and thus interpret head contributions. This illustrative example follows the label color scheme in Figure 4.

3.4 Decoder

The model jointly trains on constituency and dependency parsing by minimizing the sum of the constituency and dependency losses:

$$L = L_c + L_d \tag{12}$$

The decoder is a CKY-style (Kasami, 1966; Younger, 1967; Cocke, 1969; Stern et al., 2017) algorithm, modified by Zhou and Zhao (2019) to include dependency scores.

4 Experiments

We evaluate our model on the English Penn Treebank (PTB) (Marcus et al., 1993) and on the Chinese Treebank (CTB) (Xue et al., 2005). We use the Stanford tagger (Toutanova et al., 2003) to predict part-of-speech tags and follow standard data splits.

Following standard practice, we use the EVALB algorithm (Sekine and Collins, 1997) for constituency parsing, and report results without punctuation for dependency parsing.

4.1 Setup

In our English-language experiments, the Label Attention Layer has 112 heads: one per syntactic category. The Chinese Treebank is a smaller dataset, and therefore we use 64 heads in Chinese-language experiments. For both languages, the query, key and value vectors, as well as the output vectors of each label attention head, have 128 dimensions, as determined through short parameter-tuning experiments. For the dependency and span scores, we use the same hyperparameters as Zhou and Zhao (2019). We use the large cased pre-trained XLNet (Yang et al., 2019) as our embedding model for our English-language experiments, and a base pre-trained BERT (Devlin et al., 2018) for Chinese. Each English experiment is performed on a single 32GB GPU, while each Chinese experiment is performed on a single 12GB GPU.

We try English-language parsers with 2, 3, 4, 6, 8, 12 and 16 self-attention layers. Our parsers with 3 and 4 self-attention layers are tied in terms of F1 score, and sum of UAS and LAS scores. The results of our fine-tuning experiments are in the appendix. We decide to use 3 self-attention layers for all the following experiments, for lower computational complexity.

4.2 Ablation Study

As shown in Figure 6, we can compute the contributions from label attention heads only if there is no position-wise feed-forward layer. Residual dropout in self-attention applies to the aggregated outputs from all heads. In label attention, residual dropout applies separately to the output of each head, and

PFL	RD	Prec.	Recall	F1	UAS	LAS
Yes	Yes	96.47	96.20	96.34	97.33	**96.29**
No	Yes	96.51	96.15	96.33	97.25	96.11
Yes	No	**96.53**	**96.24**	**96.38**	**97.42**	96.26
No	No	96.29	96.05	96.17	97.23	96.11

Table 1: Results on the PTB test set of the ablation study on the Position-wise Feed-forward Layer (**PFL**) and Residual Dropout (**RD**) of the Label Attention Layer.

QV	Conc.	Prec.	Recall	F1	UAS	LAS
Yes	Yes	**96.53**	**96.24**	**96.38**	**97.42**	**96.26**
No	Yes	96.43	96.03	96.23	97.25	96.12
Yes	No	96.30	96.10	96.20	97.23	96.15
No	No	96.30	96.06	96.18	97.26	96.17

Table 2: Results on the PTB test set of the ablation study on the Query Vectors (**QV**) and Concatenation (**Conc.**) parts of the Label Attention Layer.

therefore can cancel out parts of the label contributions. We investigate the impact of removing these two components from the LAL.

We show the results on the PTB dataset of our ablation study on Residual Dropout and Position-wise Feed-forward Layer in Table 1. We use the same residual dropout probability as Zhou and Zhao (2019). When removing the position-wise feed-forward layer and keeping residual dropout, we observe only a slight decrease in overall performance, as shown in the second row. There is therefore no significant loss in performance in exchange for the interpretability of the attention heads.

We observe an increase in performance when removing residual dropout only. This suggests that all label contributions are important for performance, and that we were likely over-regularizing.

Finally, removing both position-wise feed-forward layer and residual dropout brings about a noticeable decrease in performance. We continue our experiments without residual dropout.

4.3 Comparison with Self-Attention

The two main architecture novelties of our proposed Label Attention Layer are the learned Query Vectors that represent labels and replace the Query Matrix in self-attention, and the Concatenation of the outputs of each attention head that replaces the Matrix Projection in self-attention.

In this subsection, we evaluate whether our proposed architecture novelties bring about performance improvements. To this end, we establish an ablation study to compare Label Attention with Self-Attention. We propose three additional model architectures based on our best parser: all models have 3 self-attention layers and a modified Label Attention Layer with 112 attention heads. The three modified Label Attention Layers are as follows:

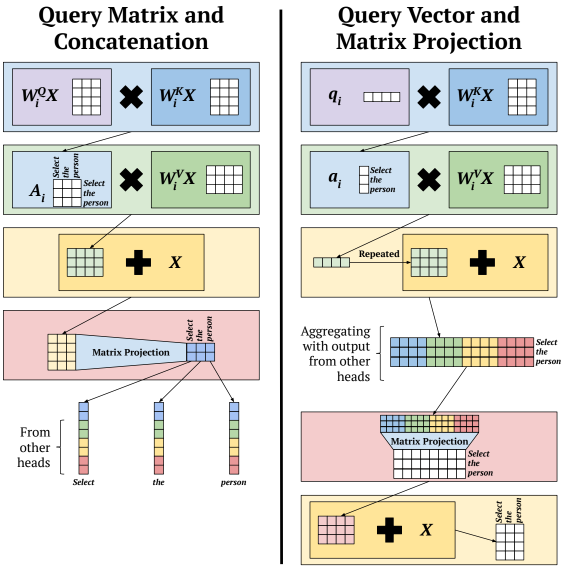


Figure 7: The two hybrid parser architectures for the ablation study on the Label Attention Layer's Query Vectors and Concatenation.

(1) Ablation of Query Vectors: the first model (left of Figure 7) has a Query Matrix like self-attention, and concatenates attention head outputs like Label Attention.

(2) Ablation of Concatenation: the second model (right of Figure 7) has a Query Vector like Label Attention, and applies matrix projection to all head outputs like self-attention.

(3) Ablation of Query Vectors and Concatenation: the third model (right of Figure 1) has a 112-head self-attention layer.

The results of our experiments are in Table 2. The second row shows that, even though query matrices employ more parameters and computation than query vectors, replacing query vectors by query matrices decreases performance. There is a similar decrease in performance when removing concatenation as well, as shown in the last row. This suggests that our Label Attention Layer learns meaningful representations in its query vectors, and that label-to-word attention distributions are more helpful to performance than query matrices and word-to-word attention distributions.

In self-attention, the output vector is a matrix projection of the concatenation of head outputs. In Label Attention, the head outputs do not interact

Model	English			Chinese		
	LR	**LP**	**F1**	**LR**	**LP**	**F1**
Shen et al. (2018)	92.0	91.7	91.8	86.6	86.4	86.5
Fried and Klein (2018)	-	-	92.2	-	-	87.0
Teng and Zhang (2018)	92.2	92.5	92.4	86.6	88.0	87.3
Vaswani et al. (2017)	-	-	92.7	-	-	-
Dyer et al. (2016)	-	-	93.3	-	-	84.6
Kuncoro et al. (2017)	-	-	93.6	-	-	-
Charniak et al. (2016)	-	-	93.8	-	-	-
Liu and Zhang (2017b)	91.3	92.1	91.7	85.9	85.2	85.5
Liu and Zhang (2017a)	-	-	94.2	-	-	86.1
Suzuki et al. (2018)	-	-	94.32	-	-	-
Takase et al. (2018)	-	-	94.47	-	-	-
Fried et al. (2017)	-	-	94.66	-	-	-
Kitaev and Klein (2018)	94.85	95.40	95.13	-	-	-
Kitaev et al. (2018)	95.51	96.03	95.77	91.55	91.96	91.75
Zhou and Zhao (2019) (BERT)	95.70	95.98	95.84	**92.03**	92.33	92.18
Zhou and Zhao (2019) (XLNet)	96.21	96.46	96.33	-	-	-
Our work	**96.24**	**96.53**	**96.38**	91.85	**93.45**	**92.64**

Table 3: Constituency Parsing on PTB & CTB test sets.

through matrix projection, but are concatenated. The third and fourth rows of Table 2 show that there is a significant decrease in performance when replacing concatenation with the matrix projection. This decrease suggests that the model benefits from having one residual connection per attention head, rather than one for all attention heads, and from separating head-specific information in word representations. In particular, the last row shows that replacing our LAL with a self-attention layer with an equal number of attention heads decreases performance: the difference between the performance of the first row and the last row is due to the Label Attention Layer's architecture novelties.

4.4 English and Chinese Results

Our best-performing English-language parser does not have residual dropout, but has a position-wise feed-forward layer. We train Chinese-language parsers using the same configuration. The Chinese Treebank has two data splits for the training, development and testing sets: one for Constituency (Liu and Zhang, 2017b) and one for Dependency parsing (Zhang and Clark, 2008).

Finally, we compare our results with the state of the art in constituency and dependency parsing in both English and Chinese. We show our Constituency Parsing results in Table 3, and our Dependency Parsing results in Table 4. Our LAL parser establishes new state-of-the-art results in both languages, improving significantly in dependency parsing.

4.5 Interpreting Head Contributions

We follow the method in Figure 6 to identify which attention heads contribute to predictions. We collect the span vectors from the Penn Treebank test set, and we use our LAL parser with no position-wise feed-forward layer for predictions.

Model	English		Chinese	
	UAS	**LAS**	**UAS**	**LAS**
Kuncoro et al. (2016)	94.26	92.06	88.87	87.30
Li et al. (2018)	94.11	92.08	88.78	86.23
Ma and Hovy (2017)	94.88	92.98	89.05	87.74
Dozat and Manning (2016)	95.74	94.08	89.30	88.23
Choe and Charniak (2016)	95.9	94.1	-	-
Ma et al. (2018)	95.87	94.19	90.59	**89.29**
Ji et al. (2019)	95.97	94.31	-	-
Fernández-González and Gómez-Rodríguez (2019)	96.04	94.43	-	-
Kuncoro et al. (2017)	95.8	94.6	-	-
Clark et al. (2018)	96.61	95.02	-	-
Wang et al. (2018)	96.35	95.25	-	-
Zhou and Zhao (2019) (BERT)	97.00	95.43	91.21	89.15
Zhou and Zhao (2019) (XLNet)	97.20	95.72	-	-
Our work	**97.42**	**96.26**	**94.56**	89.28

Table 4: Dependency Parsing on PTB & CTB test sets.

Figure 8 displays the bar charts for the three most common syntactic categories: Noun Phrases (NP), Verb Phrases (VP) and Sentences (S). We notice several heads explain each predicted category.

We collect statistics about the top-contributing heads for each predicted category. Out of the NP spans, 44.9% get their top contribution from head 35, 13.0% from head 47, and 7.3% from head 0. The top-contributing heads for VP spans are heads 31 (61.1%), 111 (13.2%), and 71 (7.5%). As for S spans, the top-contributing heads are 52 (48.6%), 31 (22.8%), 35 (6.9%), and 111 (5.2%). We see that S spans share top-contributing heads with VP spans (heads 31 and 111), and NP spans (head 35). The similarities reflect the relations between the syntactic categories. In this case, our Label Attention Layer learned the rule S $\rightarrow$ NP VP.

Moreover, the top-contributing heads for PP spans are 35 (29.6%), 31 (26.7%), 111 (10.3%), and 47 (9.4%): they are equally split between NP spans (heads 35 and 47) and VP spans (heads 31 and 111). Here, the LAL has learned that both verb and noun phrases can contain preposition phrases.

We see that head 52 is unique to S spans. Actually, 64.7% of spans with head 52 as the highest contribution are S spans. Therefore our model has learned to represent the label S using head 52.

All of the aforementioned heads are represented in Figure 8. We see that heads that have low contributions for NP spans, peak in contribution for VP spans (heads 31, 71 and 111), and vice-versa (heads 0, 35 and 47). Moreover, NP spans do not share any top-contributing head with VP spans. This shows that our parser has also learned the differences be-

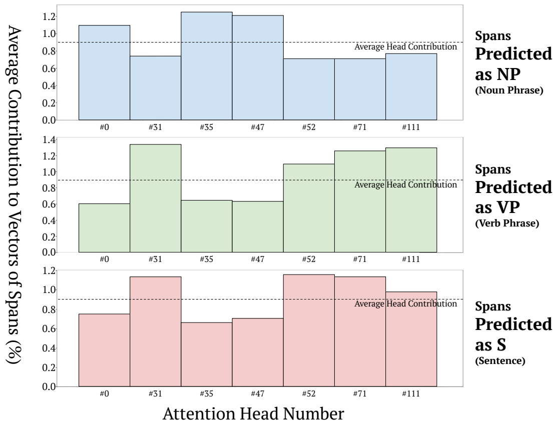


Figure 8: Average contribution of select attention heads to span vectors with different predicted syntactic category labels.

tween dissimilar syntactic categories.

4.6 Error Analysis

Finally, we analyze prediction errors from the PTB test set. One example is the span "*Fed Ready to Inject Big Funds*", predicted as NP but labelled as S. We trace back the attention weights for each word, and find that, out of the 9 top-contributing heads, only 2 focus their attention on the root verb of the sentence (*Inject*), while 4 focus on a noun (*Funds*), resulting in a noun phrase prediction. We notice similar patterns in other wrongly predicted spans, suggesting that forcing the attention distribution to focus on a relevant word might correct these errors.

We analyze wrongly predicted spans by their true category. Out of the 53 spans labelled as NP but not predicted as such, we still see the top-contributing head for 36 of them is either head 35 or 47, both top-contributing heads of spans predicted as NP. Likewise, for the 193 spans labelled as S but not predicted as such, the top-contributing head of 141 of them is one of the four top-contributing heads for spans predicted as S. This suggests that a stronger prediction link to the label attention heads may increase the performance.

5 Related Work

Since their introduction in Machine Translation, attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015) have been extended to other tasks, such as text classification (Yang et al., 2016), natural language inference (Chen et al., 2016) and language modeling (Salton et al., 2017).

Self-attention and transformer architectures (Vaswani et al., 2017) are now the state of the art in language understanding (Devlin et al., 2018; Yang et al., 2019), and semantic role labeling (Strubell et al., 2018), extractive summarization (Liu, 2019), semantic role labeling (Strubell et al., 2018) and machine translation for low-resource languages (Rikters, 2018; Rikters et al., 2018).

While attention mechanisms can provide explanations for model predictions, Serrano and Smith (2019) challenge that assumption and find that attention weights only noisily predict overall importance with regard to the model. Jain and Wallace (2019) find that attention distributions rarely correlate with feature importance weights. However, Wiegreffe and Pinter (2019) show through alternative tests that prior work does not discredit the usefulness of attention for interpretability.

Xiao et al. (2019) introduce the Label-Specific Attention Network (LSAN) for multi-label document classification. They use label descriptions to compute attention scores for words, and follow the self-attention of Lin et al. (2017). Cui and Zhang (2019) introduce a Label Attention Inference Layer for sequence labeling, which uses the self-attention of Vaswani et al. (2017). In this case, the key and value vectors are learned label embeddings, and the query vectors are hidden vectors obtained from a Bi-LSTM encoder. Our work is unrelated to these two papers, as they were published towards the end of our project.

6 Conclusions

In this paper, we introduce a revised form of self-attention: the Label Attention Layer. In our proposed architecture, attention heads represent labels. We have only one learned vector as query, rather than a matrix, thereby diminishing the number of parameters per attention head. We incorporate our Label Attention Layer into the HPSG parser (Zhou and Zhao, 2019) and obtain new state-of-the-art results on the Penn Treebank and Chinese Treebank. In English, our results show 96.38 F1 for constituency parsing, and 97.42 UAS and 96.26 LAS for dependency parsing. In Chinese, our model achieves 92.64 F1, 94.56 UAS and 89.28 LAS. Finally, our Label Attention heads learn the relations between the syntactic categories, as we show by computing contributions from each attention head to span vectors. The heads also help to analyze prediction errors, and show suggestions to correct them.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Eugene Charniak et al. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925.

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

Leyang Cui and Yue Zhang. 2019. Hierarchically-refined label attention network for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4106–4119.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. 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 710–716.

Daniel Fried and Dan Klein. 2018. Policy gradient as a proxy for dynamic oracles in constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 469–476, Melbourne, Australia. Association for Computational Linguistics.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–166.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. Whats going on in neural constituency parsers? an analysis. 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 999–1010.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. 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 3543–3556.

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2018. Multilingual constituency parsing with self-attention and pre-training. *arXiv preprint arXiv:1812.11760*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1744–1753, Austin, Texas. Association for Computational Linguistics.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Jiangming Liu and Yue Zhang. 2017a. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Jiangming Liu and Yue Zhang. 2017b. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics*, 5:45–58.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective MST parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 59–69, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Matīss Rikters. 2018. Impact of corpora quality on neural machine translation. *arXiv preprint arXiv:1810.08392*.

Matīss Rikters, Mārcis Pinnis, and Rihards Krišlauks. 2018. Training and adapting multilingual nmt for less-resourced and morphologically rich languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Giancarlo Salton, Robert Ross, and John Kelleher. 2017. Attentive language models. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 441–450.

Satoshi Sekine and Michael Collins. 1997. Evalb bracket scoring program. *URL: http://www. cs. nyu. edu/cs/projects/proteus/evalb*.

Sofia Serrano and Noah A Smith. 2019. Is attention interpretable? *arXiv preprint arXiv:1906.03731*.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia. Association for Computational Linguistics.

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)*, volume 1, pages 818–827.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*.

Jun Suzuki, Sho Takase, Hidetaka Kamigaito, Makoto Morishita, and Masaaki Nagata. 2018. An empirical study of building a strong baseline for constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 612–618.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4599–4609.

Zhiyang Teng and Yue Zhang. 2018. Two local models for neural constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 119–132, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for computational Linguistics.

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

Wenhui Wang, Baobao Chang, and Mairgup Mansur. 2018. Improved dependency parsing using implicit word connections learned from unlabeled data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2857–2863, Brussels, Belgium. Association for Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20.

Lin Xiao, Xin Huang, Boli Chen, and Liping Jing. 2019. Label-specific document representation for multi-label text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 466–475.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on penn treebank. *arXiv preprint arXiv:1907.02684*.

A Additional Experiment Results

Self-Attention Layers	**Precision**	**Recall**	**F1**	**UAS**	**LAS**
2	96.23	96.03	96.13	97.16	96.09
3	96.47	**96.20**	**96.34**	97.33	**96.29**
4	**96.52**	96.15	**96.34**	**97.39**	96.23
6	96.48	96.09	96.29	97.30	96.16
8	96.43	96.09	96.26	97.33	96.15
12	96.27	96.06	96.16	97.24	96.14
16	96.38	96.02	96.20	97.32	96.11

Table 5: Performance on the Penn Treebank test set of our LAL parser according to the number of self-attention layers. All parsers here include the Position-wise Feed-forward Layer and Residual Dropout.