Abstract

Recent work has identified properties of pre-trained self-attention models that mirror those of dependency parse structures. In particular, some self-attention heads correspond well to individual dependency types. Inspired by these developments, we propose a new competitive mechanism that encourages these attention heads to model different dependency relations. We introduce a new model, the Unsupervised Dependency Graph Network (UDGN), that can induce dependency structures from raw corpora and the masked language modeling task. Experiment results show that UDGN achieves very strong unsupervised dependency parsing performance without gold POS tags and any other external information. The competitive gated heads show a strong correlation with human-annotated dependency types. Furthermore, the UDGN can also achieve competitive performance on masked language modeling and sentence textual similarity tasks.

1 Introduction

Unsupervised dependency parsing aims to learn a dependency parser from sentences that have no annotation of their correct parse trees (Han et al., 2020). Despite its difficulty, unsupervised parsing is an interesting research direction because of its capability of utilizing almost unlimited unannotated text data. The techniques developed for unsupervised dependency parsing could also be utilized for other NLP tasks, such as unsupervised discourse parsing (Nishida and Nakayama, 2020), aspect-based sentiment analysis (Dai et al., 2021) and intent discovery (Liu et al., 2021). In addition, research in unsupervised parsing inspires and verifies cognitive research of human language acquisition (Yang et al., 2020; Pate and Goldwater, 2013; Katzir, 2014; Solan et al., 2002).

Although large-scale pre-trained models have dominated most natural language processing tasks, some recent work indicates that neural network models can see accuracy gains by leveraging syntactic information rather than ignoring it (Wang et al., 2019a; Sundararaman et al., 2019; Bai et al., 2020; Kuncoro et al., 2020). These methods either include known structural information as input to the model (Sundararaman et al., 2019; Bai et al., 2020), or incorporate structural prediction tasks into the training process (Wang et al., 2019a). However, these attempts require access to large datasets with supervised parsings, which may be hard and expensive to obtain.

Recent work also identified properties of pre-trained self-attention models that mirror those of dependency parse structures (Htut et al., 2019; Hewitt and Manning, 2019; Jawahar et al., 2019). StructFormer (Shen et al., 2020) shows that a transformer-based model can induce a good dependency structure. The belief that linguistic structure may be embedded in these models is of interest to the community. Furthermore, Dai et al. (2021) shows that...
the induced trees from finetuned RoBERTa outperform parser-provided trees on aspect-based sentiment analysis tasks. This result brings interest to study task-specific structures. From this perspective, the unsupervised acquisition of dependency structure from raw data or downstream tasks appears important and feasible.

Traditionally, dependency grammars take the dependency types (a.k.a. syntactic functions) to be primitive and then derive the constellation (Debusmann, 2000). Every head-dependent dependency bears a syntactic function (Mel’cuk et al., 1988). Htut et al. (2019) shows that some attention heads in BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) track individual dependency types. In other words, these heads model different syntactic functions. Inspired by this observation and syntactic functions, we introduce competitive gated heads to model different syntactic functions and the process of selecting the right syntactic function for each edge. These heads include two key components:

- A set of gated heads that model different information propagation processes between tokens;
- A competitive controller that selects the most suitable gated head for each pair of tokens.

Building on these components, we propose a novel architecture, the Unsupervised Dependency Graph Network (UDGN). As shown in Figure 1, the UDGN is composed of two networks: a parser that computes the dependency head distribution \( p_i \) for each word \( w_i \) in the input sentence, and then convert it to a matrix of edge probability \( m_{ij} \) that approximates an undirected dependency graph; a Dependency Graph Network (DGN) that uses the edge probabilities \( \{ m_{ij} \} \) and competitive gated heads to propagate information between words to compute a contextualized embedding \( h_i \) for each word \( w_i \). While training with the masked language modeling or other objectives, the gradient can flow through the DGN to the parser network through its dependence on \( m_{ij} \). As a result, UDGN can induce a dependency grammar while solely relying on the masked language modeling objective.

In the experiment section, we first train the UDGN with masked language modeling, then evaluate it on unsupervised dependency parsing. Our experimental results show that UDGN can: 1) achieve very strong unsupervised parsing results among models that don’t have access to extra annotations (including POS tags); 2) learn attention heads that are strongly correlated to human-annotated dependency types; 3) achieve competitive performance on language modeling tasks. We also finetune the pretrained UDGN on Semantic Textual Similarity (STS) tasks. Our experiments show that UDGN outperforms a Transformer baseline trained on the same corpus.

2 Related Work

Unsupervised dependency parsing Unsupervised dependency parsing is a long-standing task for computational linguistics. Dependency Model with Valence (DMV; Klein and Manning 2004) is the basis of several unsupervised dependency parsing methods (Daumé III, 2009; Gillenwater et al., 2010). Jiang et al. (2016) updates the method using neural networks to predict grammar rule probabilities. These methods require additional Part-of-Speech (POS) information. Spitkovsky et al. (2011) tackled the issue by performing clustering to assign tags to each word by considering its context. He et al. (2018) tackled the problem by combining DMV model with an invertible neural network to jointly model discrete syntactic structure and continuous word representations. Recently, NL-PCFG (Zhu et al., 2020) and NBL-PCFG (Yang et al., 2021) combine neural parameterization and L-PCFG to achieve good results in both unsupervised dependency and constituency parsing. StructFormer (Shen et al., 2020) proposes a joint constituency and dependency parser and use the dependency distribution to regularize the self-attention heads in the transformer model. This joint parser-language model framework can induce grammar from masked language modeling tasks.

The UDGN’s architecture is similar to StructFormer, both models include a parser and masked language model. Our model, however, has three major differences: 1) it uses competitive gated heads to improve models performance on grammar induction; 2) it uses a neural head selective parser that can produce both projective and non-projective dependency trees, whereas the distance parser in StructFormer can only produce projective trees; 3) it uses a simplified method to generate an undirected dependency mask.

Transformers, Graph Neural Networks and Dependency Graphs In many Transformer-based models, attention masks are often used to limit the
input tokens that a particular timestep can attend over. In Yang et al. (2019), for example, a mask derived from the permutation of inputs is used to induce a factorization over the tokens so that the resulting model is a valid probabilistic model. This attention mask can be viewed as an adjacency matrix over a graph whose nodes are the input tokens. From this perspective, Transformers are a form of Graph Neural Network (Scarselli et al., 2008) — specifically, a Graph Attention Network (GAT; Veličković et al. 2017), as it attends over the features of its neighbors. Several works have made this connection, and integrated dependency structures into transformers (Ahmad et al., 2020; Wang et al., 2019b; Tang et al., 2020). Results from Omote et al. (2019) and Deguchi et al. (2019) suggest that embedding these structures can improve translation models.

However, these dependency parses may not always be present to be used as input to the model. Strubell et al. (2018) trains the self-attention to attend the syntactic governor (head) of a particular token, resulting in a model that does not require dependency structure as input during inference time. We take a further step in our work and attempt to learn these structures in an unsupervised fashion from the MLM objective.

Differentiable Structured Prediction While the head selection is a good approximation of a tree structure, there are methods to obtain a relaxed adjacency matrix as the output of the parser. Previous work has used such methods for predicting structure. Koo et al. (2007) proposed using the Kirchhoff matrix tree theorem for unsupervised dependency parsing. They explain how the marginals of the edge potentials are computed, and these marginals have properties similar to a tree adjacency matrix (sum over the marginals are equal to \(N - 1\) for example, where \(N\) is the length of the sentence). Eisner (2016) describes how backpropagation can be used to compute marginals of some structured prediction algorithm. We also tried using the Kirchhoff method to normalize our dependency distributions in Appendix A.3. Corro and Titov (2018) uses similar notions but relaxes projective trees using Gumbel-softmax. Kim et al. (2017) proposed a structured form of attention and show that they are useful for certain sequence-to-sequence tasks. Mensch and Blondel (2018) gives a general theoretical treatment for these types of relaxations, while Paulus et al. (2020) gives a practical treatment of possible applications for these methods.

3 Model Architecture

As shown in Figure 2, the parser computes a dependency head distribution for each token and then converts it to a soft dependency mask \(m_{ij}\). The DGN takes \(m_{ij}\) and the sentence as input and uses a competitive mechanism to propagate information between tokens.

3.1 Head Selective Parser

We use a simplified version of the Dependency Neural Selection parser (DENSE; Zhang et al. 2016) that only predicts unlabelled dependency relations. The parser takes the sentence \(s = w_1w_2...w_T\) as input, and, for each token \(w_i\), it produces a distribution \(p_i\) over all tokens in the sentence, resulting in a \(T \times T\) weight matrix.

The parser first maps the sequence of tokens \(w_1w_2...w_T\) into a sequence of embeddings \([x_1, x_2, ..., x_T]\). Then the word embeddings are fed into a stack of a bidirectional LSTM (BiLSTM):

\[
h_i = \text{BiLSTM}(x_i) \quad (1)
\]

where \(h_i\) is the output of the BiLSTM at \(i\)-th timestep. Linear transforms are applied to the output of the BiLSTM to extract head and dependent information.

\[
h_i^H = W_H h_i + b_H \quad (2)
\]

\[
h_i^D = W_D h_i + b_D \quad (3)
\]

To map the head and dependents, we use bilinear attention:

\[
e_{ij} = \frac{h_i^D h_j^H}{\sqrt{D}} \quad (4)
\]

\[
p_{ij} = \frac{\exp(e_{ij})}{\sum_{k} \exp(e_{ik})} \quad (5)
\]

where \(p_{ij}\) is the probability that \(w_i\) depends on \(w_j\), \(D\) is the dimension of hidden states. During the inference for parsing, the Chu-Liu/Edmonds’ algorithm (Chu and Liu, 1965b) is used to extract the most likely directed dependency graph from the matrix \(p_{ij}\).

3.2 Dependency Mask

Given the dependency probabilities, StructFormer (Shen et al., 2020) uses a weighted sum of matrix \(p\) and \(p^\top\) to produce a mask for self-attention layers in the transformer. We found that simply using
the adjacency matrix of the undirected dependency graph provides better parsing results and perplexities. However, simply using the sum of the matrix and its transpose to create a symmetric weight matrix does not ensure that the attention mask has values < 1. When \( p_{ij} = 1 \) and \( p_{ji} = 1 \), for instance, the mask violates the constraints of a dependency mask. Thus, we treat \( p_{ij} \) and \( p_{ji} \) as parameters for independent Bernoulli variables, and we compute the probability that either \( w_i \) depends on \( w_j \) or \( w_j \) depends on \( w_i \).

However, DGN learns these functions from training tasks, which in our experiments is the masked language model objective. Since these objectives tend to be statistical in nature, these functions may not be correlated with ground truth labels given by human experts.

Inside each layer, the input vector \( h_{i-1} \) is first projected into \( N \) groups of vectors, where \( N \) is the number of heads. Each group contains four different vectors, namely, query \( q \), key \( k \), value \( v \), and gate \( g \):

\[
\begin{bmatrix}
q_{ik} \\
k_{ik} \\
v_{ik} \\
g_{ik}
\end{bmatrix} = W_{\text{head}_k} h_{i-1} + b_{\text{head}_k} \tag{7}
\]

**Gated Head** To model the information propagation from node \( j \) to node \( i \), we proposed a gated head:

\[
c_{ijk} = \sigma(v_{jk}) \odot \text{sigmoid}(g_{ik}) \tag{8}
\]

where \( \sigma \) is a non-linear activation function, and gates \( \text{sigmoid}(g) \) allows the \( i \)-th token to filter the extracted information. We also found that the gate effectively improves the model’s ability to induce latent dependency structures that are coherent to human-annotated trees. The activation function can be chosen from a wide variety of functions, including the identity function, tanh, ReLU, and ELU.

---

**3.3 Dependency Graph Network**

To better induce and model the dependency relations, we propose a new Dependency Graph Network (DGN). One DGN layer includes several gated heads and a competitive controller. A gated head can process and propagate information from one node to another. Different heads can learn to process and propagate different types of information. The competitive controller is designed to select the correct head to propagate information between a specific pair of nodes.

We take inspiration from the linguistic theory that dependencies are associated with different syntactic functions. These functions can appear as labels, e.g. ATTR (attribute), COMP-P (complement of preposition), and COMP-TO (complement of to).
where $a_{ijk}$ is the weight from the node $j$ to the node $i$ for $k$-th attention head.

**Relative Position Bias** Transformer models use positional encoding to represent the absolute position for each token. In DGN, we only model whether the token is before or after the current token. The motivating intuition is the association of different heads with different directions. In equation 10, we can introduce a relative position bias:

$$
\hat{a}_{ijk} = \text{softmax}_k(e_{ijk} + b_k^r) \quad (12)
$$

$$
b_k^r = \begin{cases} 
     b_k^l, & i > j \\
     b_k^s, & i < j 
\end{cases} \quad (13)
$$

where $b_k^l$ and $b_k^s$ are trainable parameters. The relative position bias allows the attention head $k$ to prioritize forward or backward directions. A mere forward and backward differentiation may seem weak compared to other parameterizations of positional encoding (Vaswani et al., 2017; Shaw et al., 2018), but in conjunction with the dependency constraints, this method is a more effective way to model the relative position in a tree structure. As shown in Table 8, the relative position bias achieves stronger masked language modeling and parsing performance than positional encoding.

At the end, a matrix multiplication is used to aggregate information from different positions:

$$
o_{lk} = \sum_{j} a_{ijk} c_{ijk} \quad (14)
$$

Then, the output $o$ from different heads are concatenated, and then projected back to the hidden state space with a linear layer:

$$
h_l^i = h_{l-1}^i + W_o \begin{bmatrix} o_{i1} \\ \vdots \\ o_{itn} \end{bmatrix} + b_o \quad (15)
$$

where $h_l^i$ is the output of the $l$-th gated self attention layers. The shared hidden state space can be seen as the shared global workspace (Goyal et al., 2021) for different independent mechanisms (heads).

### 4 Experiments

#### 4.1 Masked Language Modeling

Masked Language Modeling (MLM) is a macroscopic evaluation of the model’s ability to deal with various semantic and linguistic phenomena (e.g. co-occurrence, syntactic structure, verb-subject agreement, etc.). The performance of MLM is evaluated by measuring perplexity on masked words.
We trained and evaluated our model on 2 different datasets: the Penn TreeBank (PTB) and BLLIP. In our MLM experiments, each token has an independent chance to be replaced by a mask token \texttt{<mask>}, except that we never replace \texttt{<unk>} tokens.

**PTB** The Penn Treebank (Marcus et al., 1993) is a standard dataset for language modeling (Mikolov et al., 2012) and unsupervised constituency parsing (Shen et al., 2018; Kim et al., 2019). It contains 1M words (2499 stories) from Wall Street Journal. Following the setting proposed in Shen et al. (2020), we preprocess the Penn Treebank dataset by removing all punctuations, lower case all letters, and replaces low frequency tokens (< 5) with \texttt{<unk>}. The preprocessing results in a vocabulary size of 10798 (including \texttt{<unk>}, \texttt{<pad>} and \texttt{<mask>}).

**BLLIP** The Brown Laboratory for Linguistic Information Processing dataset is a large Penn Treebank-style parsed corpus of approximately 24 million sentences from Wall Street Journal. We train and evaluate UDGN on four splits of BLLIP: BLLIP-XS (40k sentences, 1M tokens), BLLIP-SM (200K sentences, 5M tokens), BLLIP-MD (600K sentences, 14M tokens), and BLLIP-LG (2M sentences, 42M tokens). Following the same setting proposed in Hu et al. (2020) for sentence selection, resulting in each BLLIP split being a superset of smaller splits. All models are then tested on a shared held-out test set (20k sentences, 500k tokens). To make the mask language modeling and parsing results comparable, we use a shared vocabulary for all splits. Just like the PTB dataset, we preprocess the BLLIP dataset by removing all punctuations and lower case all letters. The shared vocabulary is obtained by counting word frequencies on BLLIP-LG dataset and select the words that appear more than 27 times. The resulting vocabulary size is 30232 (including \texttt{<unk>}, \texttt{<pad>} and \texttt{<mask>}), and covers more than 98% tokens in BLLIP-LG split.

The mask rate when training on both corpora is 30%. In Section A.4, we further explore the relationship between mask rate and parsing results. Other hyperparameters are tuned separately for each model and dataset and are further described in Section A.1. The masked language model results are shown in Table 1. UDGN outperforms the baselines on smaller datasets (PTB, BLLIP-SM), but underperforms against baselines trained on large datasets (BLLIP-MD, BLLIP-LG). However, in Section 4.5, we find that the UDGN pretrained on BLLIP-LG dataset can achieve stronger performance when finetuned on a downstream task. This may suggest that our model learns more generic contextual embeddings.

### Table 1: Masked Language Model perplexities on different datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>PTB</th>
<th>BLLIP-SM</th>
<th>BLLIP-MD</th>
<th>BLLIP-LG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>68.9</td>
<td>44.6</td>
<td>22.8</td>
<td>17.0</td>
</tr>
<tr>
<td>StructFormer</td>
<td>64.8</td>
<td>43.1</td>
<td>23.4</td>
<td>16.8</td>
</tr>
<tr>
<td>UDGN</td>
<td>59.3</td>
<td>40.2</td>
<td>24.2</td>
<td>19.7</td>
</tr>
</tbody>
</table>

### Table 2: Dependency Parsing Results on WSJ test set without gold POS tags. Daggered entries (†) takes the argmax of head distribution without a tree constraint. DMV-based baseline results are from He et al. (2018).

<table>
<thead>
<tr>
<th>Methods</th>
<th>DDA</th>
<th>UDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMV (Klein and Manning, 2004)</td>
<td>35.8</td>
<td></td>
</tr>
<tr>
<td>E-DMV (Headenden III et al., 2009)</td>
<td>38.2</td>
<td></td>
</tr>
<tr>
<td>UR-A E-DMV (Tu and Honavar, 2012)</td>
<td>46.1</td>
<td></td>
</tr>
<tr>
<td>Neural E-DMV (Jiang et al., 2016)</td>
<td>42.7</td>
<td></td>
</tr>
<tr>
<td>Gaussian DMV (He et al., 2018)</td>
<td>43.1</td>
<td></td>
</tr>
<tr>
<td>INP (He et al., 2018)</td>
<td>47.9</td>
<td></td>
</tr>
<tr>
<td>NL-PCFGs (Zhu et al., 2020)</td>
<td>40.5</td>
<td>55.9</td>
</tr>
<tr>
<td>NBL-PCFGs (Yang et al., 2021)</td>
<td>39.1</td>
<td>56.1</td>
</tr>
<tr>
<td>StructFormer (Shen et al., 2020)</td>
<td>46.2</td>
<td>61.6</td>
</tr>
<tr>
<td>UDGN</td>
<td><strong>49.9</strong></td>
<td><strong>61.8</strong></td>
</tr>
</tbody>
</table>

Table 2 shows that our model outperforms baseline models. This result suggests that, given our minimum inductive bias (a token must attach to
We can then compute Pearson Correlation Coefficients (PCC) for every pair of ground truth edges \(\{i \rightarrow j\}\) where \(i\) is assigned to edge \(i \rightarrow j\). Details about this value can be found in the literature. The PCC heat maps between all types and all heads are in Appendix A.2.

4.3 Correlation Between Heads and Dependency Types

In this section, we test the correlation between heads and dependency types. We consider each dependency edge \(i \rightarrow j\) (\(i\) depends on \(j\)) in the ground truth structure as a data point. Given all the edges, we can obtain three sets of quantities: head probabilities \(A^k = \{\hat{a}_{ij}^k\}\) and type values \(Y^l = \{y_{ij}^l\}\). \(\hat{a}_{ij}^k\) is a real value between 0 and 1, represents the probability that heads \(k\) is used to model the information propagation from the child \(i\) to the parent \(j\). Details about this value can be found at Equation 12. \(y_{ij}^l\) is a binary value, represents whether the label \(l\) is assigned to edge \(i \rightarrow j\). We can then compute Pearson Correlation Coefficient (PCC) for every pair of \(A^k\) and \(Y^l\) across all ground truth edges \(\{i \rightarrow j\}\):

\[
\rho_{A^k,Y^l} = \frac{\text{cov}(A^k,Y^l)}{\sigma_{A^k}\sigma_{Y^l}}
\]

where \(\text{cov}(\cdot)\) is the covariance function, \(\sigma\) is the standard deviation of the respective variable. Hence, \(\rho_{A^k,Y^l}\) measures the correlation between head \(k\) and dependency type \(l\). \(\rho_{A^k,Y^l} > 0\) means that the model tends to use head \(k\) for propagating information from child to parent for dependency edges of the type \(l\). Here, we only consider the information propagation from child to parent even though information can propagate in both directions in masked language models. In Appendix A.2, we also computed the PCC for the parent to child direction.

Table 3 shows the PCC between the most frequent dependency types and their most correlated heads. We can observe that all three models have heads that are positively correlated to human-annotated dependency types. This result is coherent with the observation of Htut et al. (2019). Meanwhile, the UDGN achieves a significantly better correlation than the StructFormer and the Transformer. This confirms our intuition that competitive gated heads can better induce dependency types.

### Table 3: The Pearson correlation coefficients between most frequent dependency types and their most correlated head. All results are average across four random seeds, standard derivation are in parentheses. Types are arranged in masked language models. In Appendix A.2, though information can propagate in both directions, this may suggest that some dependency relations proposed by linguists correspond with efficient ways of propagating information through the sentence. Parsing examples of our model can be found in Appendix A.5.

<table>
<thead>
<tr>
<th>Models</th>
<th>prep</th>
<th>pobj</th>
<th>det</th>
<th>compound</th>
<th>nsubj</th>
<th>amod</th>
<th>dobj</th>
<th>aux</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDGN</td>
<td>0.65(0.12)</td>
<td>0.60(0.11)</td>
<td>0.68(0.15)</td>
<td>0.42(0.04)</td>
<td>0.50(0.06)</td>
<td>0.39(0.07)</td>
<td>0.39(0.07)</td>
<td>0.62(0.10)</td>
</tr>
<tr>
<td>StructFormer</td>
<td>0.39(0.05)</td>
<td>0.38(0.07)</td>
<td>0.57(0.03)</td>
<td>0.33(0.01)</td>
<td>0.25(0.06)</td>
<td>0.26(0.01)</td>
<td>0.22(0.05)</td>
<td>0.23(0.04)</td>
</tr>
<tr>
<td>Transformer</td>
<td>0.43(0.00)</td>
<td>0.46(0.03)</td>
<td>0.46(0.12)</td>
<td>0.30(0.01)</td>
<td>0.39(0.07)</td>
<td>0.26(0.02)</td>
<td>0.28(0.01)</td>
<td>0.30(0.10)</td>
</tr>
</tbody>
</table>

4.4 Ablation Experiments

Figure 4 shows the relation between the number of heads in each UDGN layer and the model’s unsupervised parsing performance. Table 8 shows the model’s performance when individual components are removed. We can observe that the number of heads has the most significant influence on unsupervised parsing performance. While this is only one head, the model fails to learn any meaningful structure. Then the parsing performance increase as the number of heads increase. And we observe marginal improvement after the number of heads reaching 8. The second most significant parsing performance decrease is caused by removing the gating mechanism. This change forces each head to always extract the same information from a given key node \(h_j\), regardless of the query node \(h_i\). This has a similar effect as the previous change, reducing the diversity of different functions that can be
Table 4: The performance of UDGN after removing different components. "- Gates" means removing the gate g in gated heads. "- Competition" means using a non-competitive sigmoid function to replace the softmax in the competitive controller. "- relative pos bias" means removing the relative positional bias. "Chu-Liu" means that we use the Chu-Liu algorithm to extract the maximum directed spanning tree. "Argmax" means that we take the word at the maximum p value as the dependency head. This could result in non-tree structures, but we believe that this metric gives a better indication of how often the parser predicts the right head of each word.

Table 5: Sentence embedding performance on STS tasks. All models are pretrained on BLLIP-LG, and finetuned on STS. Freeze parser means that the parameters for the parser are not updated during finetuning.

4.5 Fine-tuning
In this experiment, the goal was to determine if a better representation of semantics can be encoded if the model was constrained for structure. We pretrain a UDGN model on the BLLIP-XL dataset, and then finetune it on the STS-B (Cer et al., 2017) dataset. For a controlled experiment, we compare the results we attain with the previously mentioned Transformer model. We then evaluate the resulting classifier on the STS 2012-2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), the SICK-Relatedness (Marelli et al., 2014) dataset, and STS-B (Cer et al., 2017). We then report the Spearman correlation score for each dataset (the ‘all’ setting in Gao et al. 2021).

We find that the UDGN model performs better overall compared to the transformer model. While these are not state-of-the-art results on these tasks, the purpose of our comparison was to examine the benefit of the UDGN model over the Transformer architecture. It’s also interesting to notice that if parameters in the parser are frozen during the finetuning, the model will get worse performance. This result suggests that fine-tuning on STS forces pretrained language models to learn more task-oriented trees. Dai et al. (2021) observed similar results with finetuned RoBERTa on Aspect-Based Sentiment Analysis tasks.

5 Conclusion
In this paper, we proposed the Unsupervised Dependency Graph Network (UDGN), a novel architecture to induce and accommodate dependency graphs in a transformer-like framework. The model is inspired by linguistic theories. Experiment results show that UDGN achieves state-of-the-art dependency grammar induction performance. The competitive gated heads show a strong correlation to human-annotated dependency types. We hope these interesting observations will build new connections between classic linguistic theories and modern neural network models. Another interesting future research direction is exploring how the newly proposed components can help large-scale pretrained languages models.
References


A Appendix

A.1 Hyperparameters

<table>
<thead>
<tr>
<th>Model</th>
<th>Hidden Size</th>
<th>head/Head Size</th>
<th>Dropout</th>
<th>DropAtt</th>
<th>lr</th>
<th>#tags</th>
<th>Feedforward Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDGN (PTB)</td>
<td>512</td>
<td>128</td>
<td>0.2</td>
<td>0.1</td>
<td>0.001</td>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>UDGN (BLLIP-XS,SM)</td>
<td>512</td>
<td>128</td>
<td>0.2</td>
<td>0.1</td>
<td>0.001</td>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>UDGN (BLLIP-MD,LG)</td>
<td>512</td>
<td>128</td>
<td>0.2</td>
<td>0.1</td>
<td>0.001</td>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>Transformer</td>
<td>512</td>
<td>64</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0003</td>
<td>–</td>
<td>2048</td>
</tr>
<tr>
<td>StructFormer</td>
<td>512</td>
<td>64</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0003</td>
<td>–</td>
<td>2048</td>
</tr>
</tbody>
</table>

Table 6: Hyperparameters used in Masked Language Modeling experiments. All model has 8 layers and 8 heads or attention heads. For UDGN, we apply dropout in front of all linear layers; dropatt randomly drops heads; the parser is a 3-layer biLSTM model, which has 6 tag embeddings, 1 of them is a zero vector, 5 of them are trainable. For transformer and structformer, the dropout is applied to the output of each sublayers; dropatt randomly drops attention weights; the size of their feedforward sublayers is 2048.

A.2 Correlation between Heads and Dependency Types

<table>
<thead>
<tr>
<th>Models</th>
<th>prep</th>
<th>pobj</th>
<th>det</th>
<th>compound</th>
<th>nsubj</th>
<th>amod</th>
<th>dobj</th>
<th>aux</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDGN</td>
<td>0.45(0.15)</td>
<td>0.84(0.05)</td>
<td>0.59(0.08)</td>
<td>0.38(0.03)</td>
<td>0.47(0.08)</td>
<td>0.43(0.08)</td>
<td>0.32(0.04)</td>
<td>0.45(0.08)</td>
</tr>
<tr>
<td>StructFormer</td>
<td>0.28(0.04)</td>
<td>0.43(0.13)</td>
<td>0.38(0.06)</td>
<td>0.34(0.02)</td>
<td>0.30(0.03)</td>
<td>0.27(0.01)</td>
<td>0.19(0.02)</td>
<td>0.22(0.02)</td>
</tr>
<tr>
<td>Transformer</td>
<td>0.44(0.03)</td>
<td>0.31(0.05)</td>
<td>0.37(0.03)</td>
<td>0.32(0.00)</td>
<td>0.16(0.01)</td>
<td>0.28(0.01)</td>
<td>0.20(0.01)</td>
<td>0.26(0.03)</td>
</tr>
</tbody>
</table>

Table 7: The pearson correlation coefficients between most frequent dependency types (the child to parent direction) and their most correlated head. Types are arrange from the highest frequency to lower frequency.

A.3 More Ablation Experiments

In this section, we evaluate UDGN’s performance after removing the nonlinear function in gated heads, replacing relative positional bias with a standard positional encoding, and using Kirchhoff matrix tree theorem (Koo et al., 2007) to normalize the dependency probabilities. It’s interesting to notice that, although Kirchhoff method can produce a valid marginal distribution for dependency probabilities, adding the normalization can’t improve the unsupervised parsing performance. We believe it’s due to the extra optimization complexity introduced by the matrix inversion in Kirchhoff method. Another observation is that relative position bias helps the model to achieve better perplexity and parsing performance in comparison with positional encoding. This may suggest that the combination of dependency graphs and relative positions is more informative than absolute positions.

<table>
<thead>
<tr>
<th>Model</th>
<th>MLM PPL</th>
<th>Argmax DDA</th>
<th>UDA DDA</th>
<th>Chu-Liu DDA</th>
<th>UDA DDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDGN</td>
<td>60.4(0.8)</td>
<td>52.5(0.7)</td>
<td>58.8(0.9)</td>
<td>50.2(1.5)</td>
<td>61.2(0.4)</td>
</tr>
<tr>
<td>- Nonlinear</td>
<td>61.2(1.0)</td>
<td>49.5(1.1)</td>
<td>56.8(1.4)</td>
<td>45.6(2.0)</td>
<td>60.8(1.4)</td>
</tr>
<tr>
<td>- relative pos bias + pos encoding</td>
<td>65.2(3.4)</td>
<td>47.1(7.3)</td>
<td>55.4(4.1)</td>
<td>44.8(7.2)</td>
<td>58.2(5.2)</td>
</tr>
<tr>
<td>+ Kirchhoff</td>
<td>59.7(0.5)</td>
<td>50.2(2.2)</td>
<td>58.4(1.2)</td>
<td>46.5(2.1)</td>
<td>60.7(1.2)</td>
</tr>
</tbody>
</table>

Table 8: The performance of UDGN after removing different components. “- Nonlinear” means remove the tanh activation function in gated heads. “- relative pos bias + pos enc” means using a trainable positional encoding to replace the relative position bias. “+ Kirchhoff” means using Kirchhoff matrix tree theorem (Koo et al., 2007) to compute the marginal probabilities of each edge, and these marginals have properties similar to a tree adjacency matrix (sum over the marginals are equal to N-1 for example, where N is the length of the sentence).
Figure 5: Pearson Correlation Coefficients heat maps. Dependency types are arranged from highest frequency to lowest. We can observe that high frequent types have more strongly correlated heads. Strongly correlated heads also evenly distributed across layers.
Table 9: The performance of UDGN after trained on different BLLIP splits. Since all BLLIP splits share the same vocabulary and test set, results are comparable. While DDA have a high variance, UDA remain stable across different corpus sizes. This may due to the reason that DGN only use an undirected dependency mask, the choice of dependency direction could be arbitrary. This result may suggest that syntax can be acquired with a relatively small amount of data. It is possible then, that where extra data helps is in terms of semantic knowledge, like common sense.

A.4 Mask rate

One of the more surprising findings in our experiments with this architecture was the relationship between the word mask rate in the MLM task and how much the resulting parse trees corresponded to the ground-truth parse trees. We trained 5 models for different word masking rates from 0.1 to 0.9, in 0.1 increments, and computed the argmax, DDA, and undirected DDA (UDA) scores for each of these models. Figure 6 shows the plot for these results.

Firstly, we observe that the acceptable range of masking rate for achieving a decent UDA score was fairly large: the optimal was at about 0.3, but values of 0.2 up to 0.8 worked to induce tree structures that resulted in fairly good undirected trees. Secondly, as we move away from the optimum of 0.3-0.4, the variance of our results increases, with the highest variance when we mask at a rate of 0.9. Finally, our model supplies the attention mask as a symmetric matrix—the directionality of the mask is decimated when we perform Equation 6. Consequently, we find that the variance of the DDA is higher than UDA as the connectivity of the nodes in the tree is more important than the direction of the connection in our architecture.

A.5 Dependency Graph Examples

gold:

commercial paper

pred:

commercial paper
Gold tree:

hooker’s philosophy was to build and sell

Induced tree:

hooker’s philosophy was to build and sell

Gold:

a few hours later the stock market dropped N points

Pred:

a few hours later the stock market dropped N points

Gold:

there’s nothing rational about this kind of action

Pred:

there’s nothing rational about this kind of action
it’s turning out to be a real blockbuster. <unk> said

and I think institutions are going to come in and buy
that <unk> <unk> quantum badly because its own plants cover only about half of its <unk> needs