Unsupervised Dependency Graph Network

Anonymous ACL submission

Abstract

Recent work has identified properties of pretrained 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.

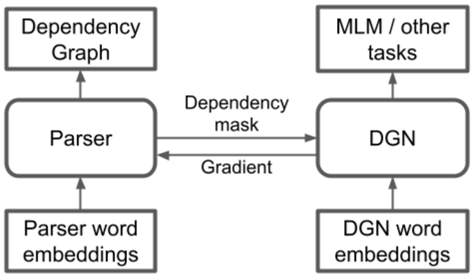


Figure 1: The architecture of Unsupervised Dependency Graph Network (UDGN). Given an input sentence, the parser can predict the dependency relation between tokens and generate a soft mask to approximate the undirected dependency graph. Conditioning on the mask, the DGN computes contextual word embeddings for the training task. Since the mask is soft, the gradient can be backpropagated from the DGN into the parser. Thus UDGN can induce grammar while training on masked language modeling or other downstream 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 pretrained 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 have used such methods for predicting structure. Koo et al. (2007) proposed using the Kirchoff 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 $\mathbf{s} = w_1 w_2 ... w_T$ 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 $w_1 w_2 ... w_T$ into a sequence of embeddings $[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]$. Then the word embeddings are fed into a stack of a bidirectional LSTM (BiLSTM):

$$\mathbf{h}_i = \mathrm{BiLSTM}(\mathbf{x}_i) \tag{1}$$

where $\mathbf{h}_i$ 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.

$$\mathbf{h}_i^{\mathrm{H}} = \mathbf{W}_{\mathrm{H}} \mathbf{h}_i + \mathbf{b}_{\mathrm{H}} \tag{2}$$

$$\mathbf{h}_i^{\mathrm{D}} = \mathbf{W}_{\mathrm{D}} \mathbf{h}_i + \mathbf{b}_{\mathrm{D}} \tag{3}$$

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

$$e_{ij} = \frac{\mathbf{h}_i^{\mathrm{D}} \mathbf{h}_j^{\mathrm{H}}}{\sqrt{D}} \tag{4}$$

$$p_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})} \tag{5}$$

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

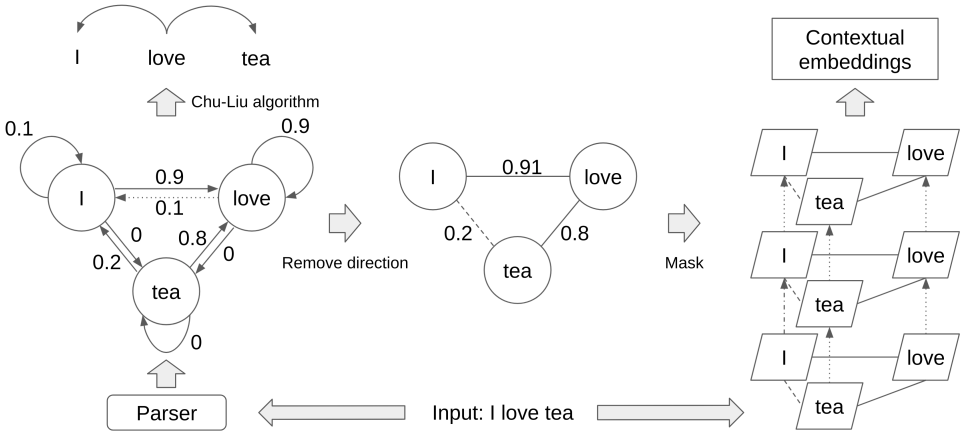


Figure 2: Details of the UDGN. Given the input sentence, the parser (left) produces a dependency head distribution for each token. These distributions form a distribution matrix p_{ij}. To do unsupervised parsing, the Chu-Liu algorithm (Chu and Liu, 1965a) generates the most likely dependency graph given p_{ij}. While training, however, we remove the direction of dependency in p_{ij} and obtain an undirected dependency mask m_{ij} (middle). m_{ij} is symmetric and with zeroes along the diagonal. The DGN (right) takes m_{ij} and the sentence as input and uses competitive gated heads to propagate information between tokens. m_{ij} controls the amount of information being propagated between nodes. If m_{ij} is small then less information will be communicated between x_i and x_j, and vice versa.

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.

$$\begin{aligned} m_{ij} &= p(i \rightarrow j \quad \textbf{or} \quad j \rightarrow i) \\ &= p_{ij} + p_{ji} - p_{ij} \times p_{ji} \end{aligned} \tag{6}$$

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). 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^{l-1} is first projected into N groups of vectors, where N is the number of heads. Each group contains four different vectors, namely, query $\mathbf{q}$, key $\mathbf{k}$, value $\mathbf{v}$ and gate $\mathbf{g}$:

$$\begin{bmatrix} \mathbf{q}_{ik} \\ \mathbf{k}_{ik} \\ \mathbf{v}_{ik} \\ \mathbf{g}_{ik} \end{bmatrix} = \mathbf{W}_{\mathrm{head}_k} \mathbf{h}_i^{l-1} + \mathbf{b}_{\mathrm{head}_k} \tag{7}$$

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

$$\mathbf{c}_{ijk} = \sigma(\mathbf{v}_{jk}) \odot \mathrm{sigmoid}(\mathbf{g}_{ik}) \tag{8}$$

where σ is a non-linear activation function, and gates $\mathrm{sigmoid}(\mathbf{g})$ 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,

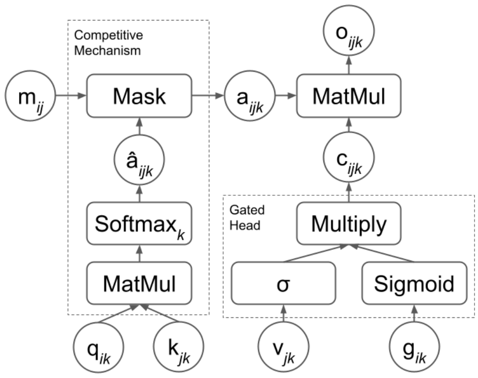


Figure 3: Competitive Gated Heads. Suppose that the information should be propagated from node j to node i, the competitive controller takes $\mathbf{q}_{i\cdot}, \mathbf{k}_{j\cdot}$ as input, output a probability distribution $\hat{a}_{ij}$ across different heads. This allows the model to select a head for the information propagation. Then the probability $\hat{a}_{ijk}$ is multiplied by dependency mask m_{ij} to get a_{ijk}. The mask m_{ij} functions as a macro gate to control the amount of information propagate between the node pair. For the k-th head, the node j send representation $\mathbf{v}_{jk}$, the node i use a gate $\mathbf{g}_{ik}$ to filter the incoming representation.

etc. In our experiment, we found that tanh function provides the best overall performance. This is probably due to two reasons: a) tanh function provides a bounded output (between -1 and 1), and b) gates and head weights are more effective while controlling a bounded value.

Competitive Controller Lamb et al. (2021) proposed the idea of using a competition method to encourage heads to specialize over training iterations to learn different functions. This idea is coherent with our intuition different heads should model different dependency relations. In UDGN, a competitive controller is designed to select a head for each pair of nodes (i, j). However discrete assignment is hard to optimize, we replace it with a soft relaxation:

$$e_{ijk} = \frac{\mathbf{q}_{ik}\mathbf{k}_{jk}}{\sqrt{D}} \tag{9}$$

$$\hat{a}_{ijk} = \mathrm{softmax}_k(e_{ijk}) \tag{10}$$

where $\hat{a}_{ijk}$ is the probability that the k-th head is assigned to propagate information from the j-th token to the i-th token. To obtain the actual head weights, we multiply the probability of edge existence with the probability of choosing a specific attention head:

$$a_{ijk} = \hat{a}_{ijk} \times m_{ij} \tag{11}$$

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:

$$\hat{a}_{ijk} = \mathrm{softmax}_k(e_{ijk} + b_k^{lr}) \tag{12}$$

$$b_k^{lr} = \begin{cases} b_k^l, & i > j \\ b_k^r, & i < j \end{cases} \tag{13}$$

where b_k^l and b_k^r 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.

$$\mathbf{o}_{ik} = \sum_j a_{ijk}\mathbf{c}_{ijk} \tag{14}$$

Then, the output $\mathbf{o}$ from different heads are concatenated, and then projected back to the hidden state space with a linear layer.

$$\mathbf{h}_i^l = \mathbf{h}_i^{l-1} + \mathbf{W}_o \begin{bmatrix} \mathbf{o}_{i1} \\ \vdots \\ \mathbf{o}_{in} \end{bmatrix} + \mathbf{b}_o \tag{15}$$

where $\mathbf{h}_i^l$ 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.

Model	PTB	BLLIP -SM	BLLIP -MD	BLLIP -XL
Transformer	68.9	44.6	22.8	17.0
StructFormer	64.8	43.1	23.4	16.8
UDGN	59.3	40.2	24.2	19.7

Table 1: Masked Language Model perplexities on different datasets.

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 `<mask>`, except that we never replace `<unk>` token.

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 `<unk>`. The preprocessing results in a vocabulary size of 10798 (including `<unk>`, `<pad>` and `<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 `<unk>`, `<pad>` and `<mask>`), and covers more than 98% tokens in BLLIP-LG split.

The mask rate when training on both corpora is

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

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). DDA stands for Directed Dependency Accuracy. UDA stands for Undirected Dependency Accuracy. Unsupervised dependency parsing results with the knowledge of gold POS tags or other external knowledge are excluded from this table.

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.

4.2 Unsupervised Dependency Parsing

We convert the human-annotated constituency trees from the Wall Street Journal test set (Marcus et al., 1993) to dependency trees and use the Directed Dependency Accuracy (DDA) as our metric. To derive valid trees from the attention mask, we use the Chu-Liu (Chu and Liu, 1965b) (or Edmonds' Edmonds 1967) algorithm to obtain the maximum directed spanning tree. Following previous research (Shen et al., 2020), we use the model trained on the preprocessed PTB dataset (no punctuations), and test its parsing performance on section 23 of the WSJ corpus. Punctuation is ignored during the evaluation.

Table 2 shows that our model outperforms baseline models. This result suggests that, given our minimum inductive bias (a token must attach to

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

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 arrange from the highest frequency to lower frequency. PCC heat maps between all types and all heads are in Appendix A.2.

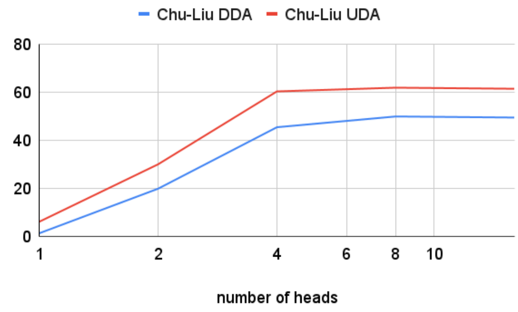


Figure 4: Relationship between the parsing performance and the number of heads in each layer. The hidden state size of heads are adjusted to maintain the same number of total parameters.

another, but the graph is not necessarily a tree), predicting missing tokens implicitly learns a good graph that correlates well with human-annotated dependency trees. This may suggest that some of the 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.

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 $A^k = \{\hat{a}^k_{ji}\}$ and type values $Y^l = \{y^l_{ij}\}$. $\hat{a}^k_{ij}$ 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^l_{ij} 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 $\{i \rightarrow j\}$:

$$\rho_{A^k,Y^l} = \frac{\mathrm{cov}(A^k, Y^l)}{\sigma_{A^k}\sigma_{Y^l}} \tag{16}$$

where $\mathrm{cov}(\cdot)$ is the covariance function, $\sigma_{\cdot}$ is the standard deviation of the respective variable. Hence, ρ_{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.

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

Model	MLM	Argmax		Chu-Liu	
	PPL	DDA	UDA	DDA	UDA
UDGN	59.3(0.5)	52.7(0.9)	58.3(0.7)	49.9(1.6)	61.8(0.9)
- Gates	69.5(1.9)	31.5(2.2)	40.7(0.3)	26.1(2.1)	48.9(0.5)
- Competition	73.6(3.1)	44.7(1.9)	54.4(1.9)	40.4(1.6)	56.6(2.1)
- relative pos bias	62.1(1.0)	51.6(1.6)	59.8(0.8)	47.4(2.6)	62.1(1.1)

Table 4: The performance of UDGN after removing different components. "- Gates" means removing the gate $\mathbf{g}$ 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.

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Transformer	76.17	61.48	73.97	74.35	53.72	64.26	80.00	69.14
UDGN (Freeze parser)	77.71	71.17	78.71	82.30	66.04	70.13	82.17	75.46
UDGN	80.51	75.02	80.54	82.16	64.73	72.49	81.94	76.77

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.

modeled by heads. These two observations may suggest that the diversity of information propagation function (multiple heads) is essential to induce a meaningful structure.

The competitive controller also has an important influence on parsing performance. Its non-competitive version is the sigmoid controller used in StructFormer. If we replace it with the non-competitive controller, the DDA decreases to 44.7 which is similar to the result of StructFormer (46.2). Another interesting observation is that removing relative position bias has the least influence on parsing and language modeling. This may suggest that the dependency structure already encoded certain positional information. More ablation experiment results can be found in Appendix A.3.

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

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 81–91.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez Agirre, Rada Mihalcea, German Rigau Claramunt, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval-2016. 10th International Workshop on Semantic Evaluation; 2016 Jun 16-17; San Diego, CA. Stroudsburg (PA): ACL; 2016. p. 497-511.* ACL (Association for Computational Linguistics).

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. 2020. Gate: Graph attention transformer encoder for cross-lingual relation and event extraction. *arXiv preprint arXiv:2010.03009*.

He Bai, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. 2020. Segatron: Segment-aware transformer for language modeling and understanding. *arXiv preprint arXiv:2004.14996*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965a. On the shortest arborescence of a directed graph. *Science Sinica*.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965b. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Caio Corro and Ivan Titov. 2018. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. *arXiv preprint arXiv:1807.09875*.

Junqi Dai, Hang Yan, Tianxiang Sun, Pengfei Liu, and Xipeng Qiu. 2021. Does syntax matter? a strong baseline for aspect-based sentiment analysis with roberta. *arXiv preprint arXiv:2104.04986*.

Hal Daumé III. 2009. Unsupervised search-based structured prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 209–216.

Ralph Debusmann. 2000. An introduction to dependency grammar. *Hausarbeit fur das Hauptseminar Dependenzgrammatik SoSe*, 99:1–16.

Hiroyuki Deguchi, Akihiro Tamura, and Takashi Ninomiya. 2019. Dependency-based self-attention for transformer nmt. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 239–246.

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*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv e-prints*, pages arXiv–2104.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. *ACL 2010*, page 194.

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. 2021. Coordination among neural modules through a shared global workspace. *arXiv preprint arXiv:2103.01197*.

Wenjuan Han, Yong Jiang, Hwee Tou Ng, and Kewei Tu. 2020. A survey of unsupervised dependency parsing. *arXiv preprint arXiv:2010.01535*.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Unsupervised learning of syntactic structure with invertible neural projections. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of human language technologies: the 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 101–109.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do attention heads in bert track syntactic dependencies? *arXiv preprint arXiv:1911.12246*.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger P Levy. 2020. A systematic assessment of syntactic generalization in neural language models. *arXiv preprint arXiv:2005.03692*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Yong Jiang, Wenjuan Han, Kewei Tu, et al. 2016. Unsupervised neural dependency parsing. Association for Computational Linguistics (ACL).

Roni Katzir. 2014. A cognitively plausible model for grammar induction. *Journal of Language Modelling*, 2.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

Yoon Kim, Chris Dyer, and Alexander M Rush. 2019. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, pages 478–485.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *arXiv preprint arXiv:2005.13482*.

Alex Lamb, Di He, Anirudh Goyal, Guolin Ke, Chien-Feng Liao, Mirco Ravanelli, and Yoshua Bengio. 2021. Transformers with competitive ensembles of independent mechanisms. *arXiv preprint arXiv:2103.00336*.

Pengfei Liu, Youzhang Ning, King Keung Wu, Kun Li, and Helen Meng. 2021. Open intent discovery through unsupervised semantic clustering and dependency parsing. *arXiv preprint arXiv:2104.12114*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Lrec*, pages 216–223. Reykjavik.

Igor Aleksandrovic Mel'cuk et al. 1988. *Dependency syntax: theory and practice*. SUNY press.

Arthur Mensch and Mathieu Blondel. 2018. Differentiable dynamic programming for structured prediction and attention. In *International Conference on Machine Learning*, pages 3462–3471. PMLR.

Tomáš Mikolov et al. 2012. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 80:26.

Noriki Nishida and Hideki Nakayama. 2020. Unsupervised discourse constituency parsing using viterbi em. *Transactions of the Association for Computational Linguistics*, 8:215–230.

Yutaro Omote, Akihiro Tamura, and Takashi Ninomiya. 2019. Dependency-based relative positional encoding for transformer nmt. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 854–861.

John K Pate and Sharon Goldwater. 2013. Unsupervised dependency parsing with acoustic cues. *Transactions of the Association for Computational Linguistics*, 1:63–74.

Max B Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. 2020. Gradient estimation with stochastic softmax tricks. *arXiv preprint arXiv:2006.08063*.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2020. Structformer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. *arXiv preprint arXiv:2012.00857*.

Zach Solan, Eytan Ruppin, David Horn, and Shimon Edelman. 2002. Automatic acquisition and efficient representation of syntactic structures. *Advances in Neural Information Processing Systems*, 15:107–114.

Valentin I Spitkovsky, Hiyan Alshawi, Angel Chang, and Dan Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1281–1290.

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*.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. 2019. Syntax-infused transformer and bert models for machine translation and natural language understanding. *arXiv preprint arXiv:1911.06156*.

Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. 2020. Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6578–6588.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1324–1334.

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.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019a. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019b. Tree transformer: Integrating tree structures into self-attention. 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 1060–1070.

Jinbiao Yang, Stefan L Frank, and Antal van den Bosch. 2020. Less is better: A cognitively inspired unsupervised model for language segmentation. In *Proceedings of the Workshop on the Cognitive Aspects of the Lexicon*, pages 33–45.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021. Neural bi-lexicalized pcfg induction. *arXiv preprint arXiv:2105.15021*.

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*.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2016. Dependency parsing as head selection. *arXiv preprint arXiv:1606.01280*.

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 Appendix

A.1 Hyperparameters

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

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

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

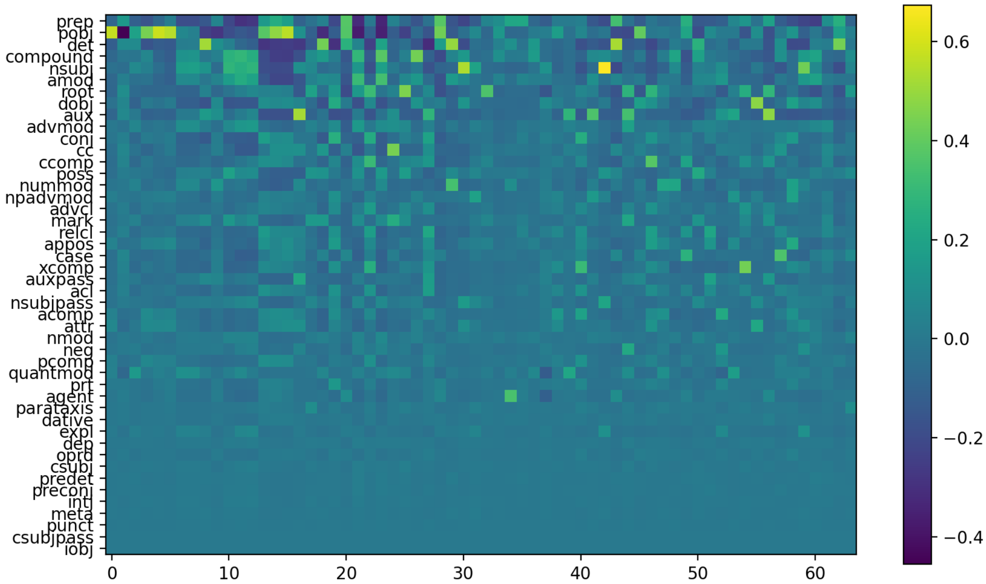
Table 7: The pearson correlation coefficients between most frequent dependency types (*the child to parent direction*) and their most correlated head. Types are arrange from the highest frequency to lower frequency.

A.3 More Ablation Experiments

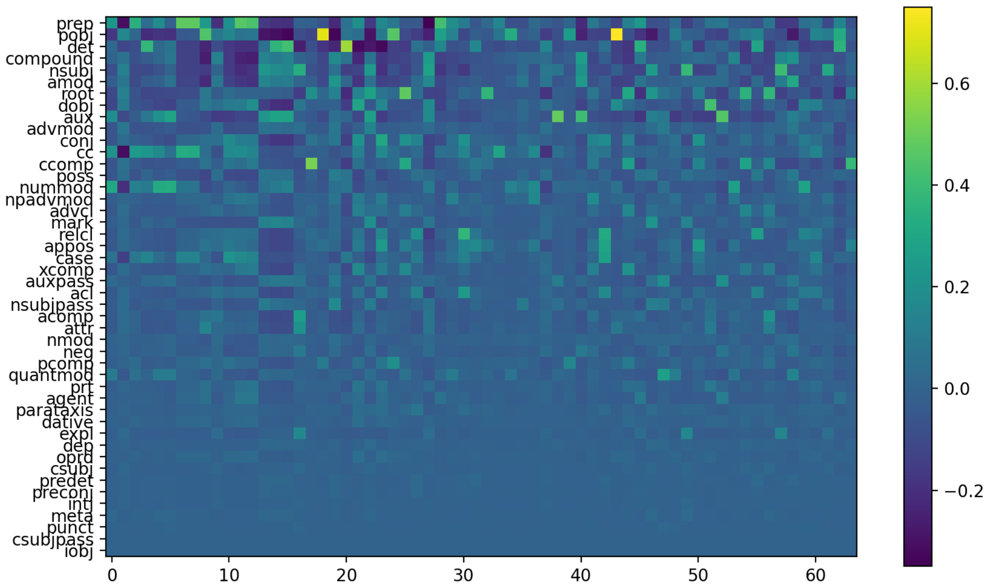
In this section, we evaluate UDGN's performance after removing the nonlinear function in gated heads, replacing relative positional bias with a standard positional encoding, and using Kirchhoff matrix tree theorem (Koo et al., 2007) to normalize the dependency probabilities. It's interesting to notice 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.

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

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).



(a) PCC heat map for heads and child to parent dependency relations.



(b) PCC heat map for heads and parent to child dependency relations.

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.

Dataset	#tokens	MLM PPL	Argmax DDA	Argmax UDA	Chu-Liu DDA	Chu-Liu UDA
BLLIP-XS	1M	133.7(3.1)	51.4(2.0)	57.6(1.6)	47.9(2.7)	61.2(1.6)
BLLIP-SM	5M	40.2(0.8)	53.7(2.5)	60.7(0.6)	50.9(5.3)	65.1(1.6)
BLLIP-MD	14M	24.2(0.5)	50.5(6.1)	59.8(2.9)	47.7(8.1)	63.0(4.2)
BLLIP-LG	42M	19.7(0.3)	45.6(2.9)	61.7(1.8)	41.6(4.2)	62.5(1.6)

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.

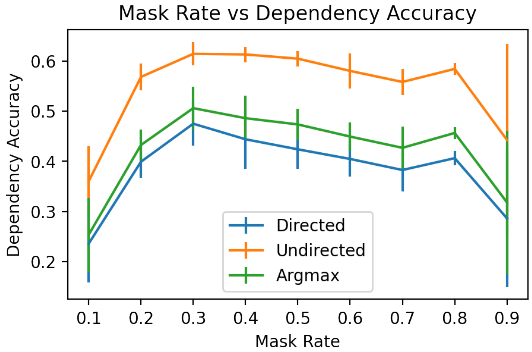


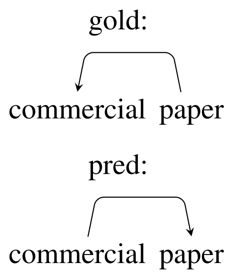
Figure 6: Relationship between the parsing performance and the mask rate for MLM.

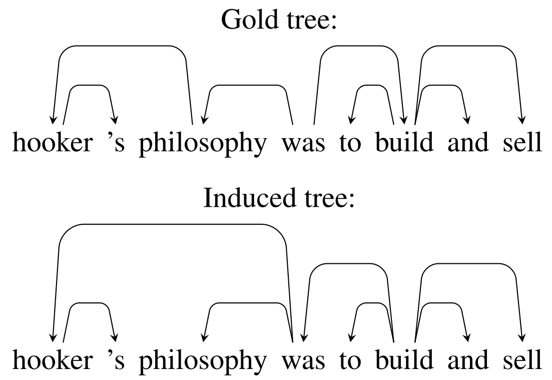
A.4 Mask rate

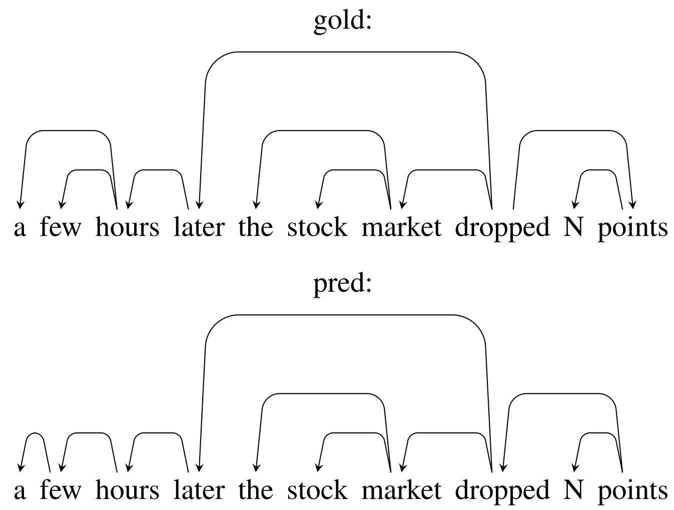
One of the more surprising findings in our experiments with this architecture was the relationship between the word mask rate in the MLM task and how much the resulting parse trees corresponded to the ground-truth parse trees. We trained 5 models for different word masking rates from 0.1 to 0.9, in 0.1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

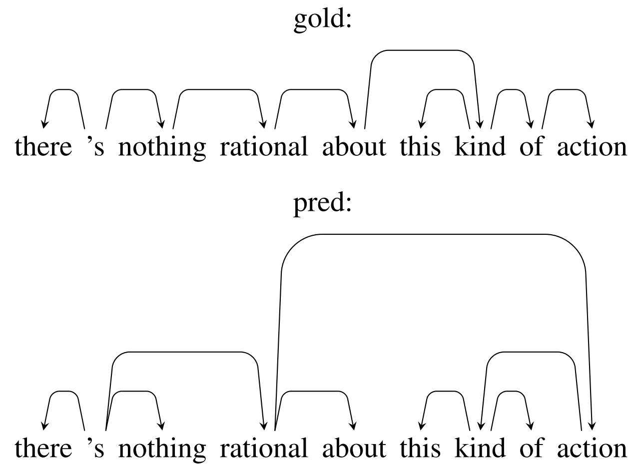
Firstly, we observe that the acceptable range of masking rate for achieving a decent UDA score was fairly large: the optimal was at about 0.3, but values of 0.2 up to 0.8 worked to induce tree structures that resulted in fairly good undirected trees. Secondly, as we move away from the optimum of 0.3-0.4, 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

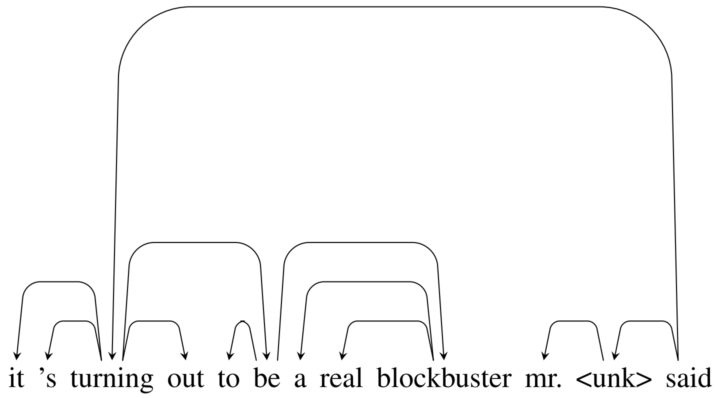



Gold tree:
hooker 's philosophy was to build and sell
Induced tree:
hooker 's philosophy was to build and sell

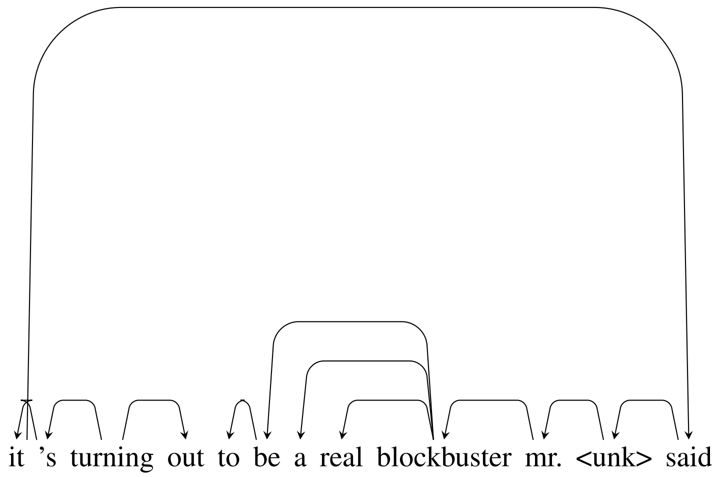

gold:
a few hours later the stock market dropped N points
pred:
a few hours later the stock market dropped N points


gold:
there 's nothing rational about this kind of action
pred:
there 's nothing rational about this kind of action

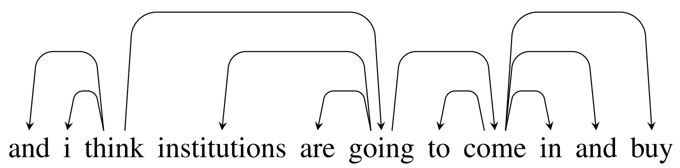
gold:



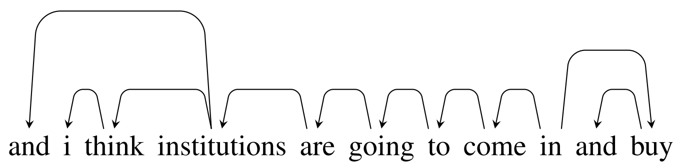
pred:

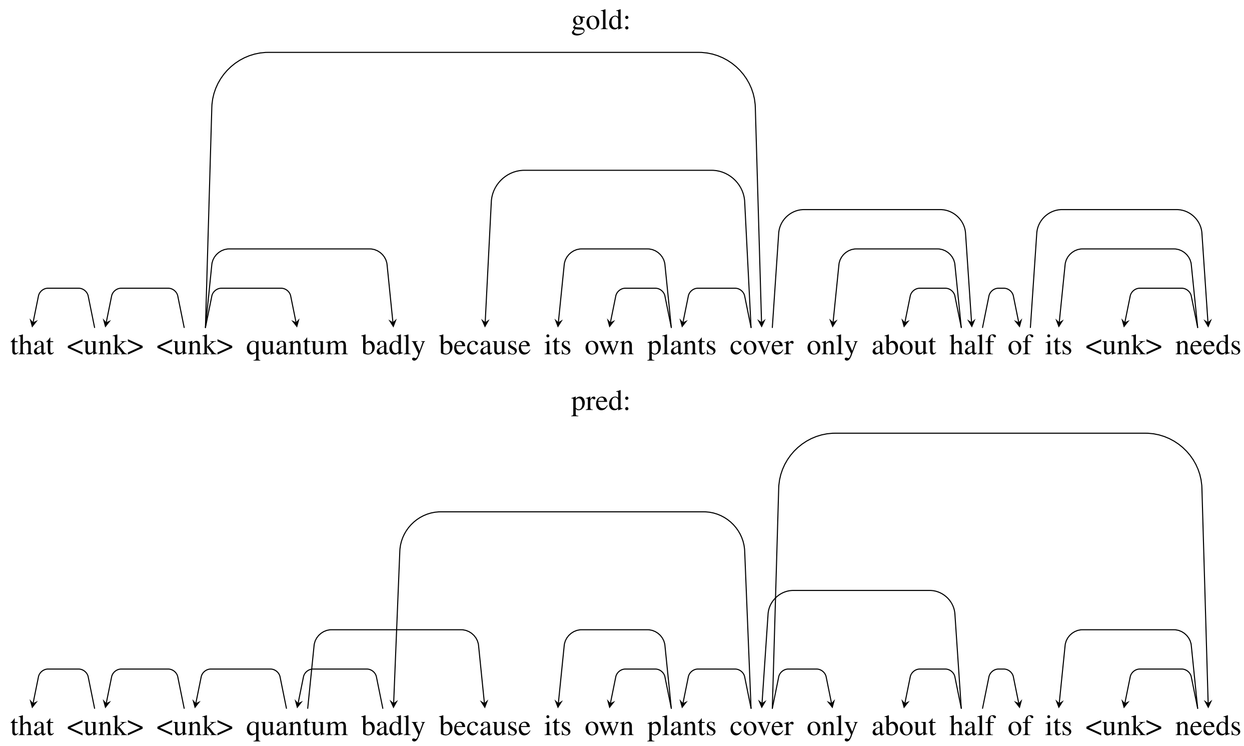


gold:



pred:




gold:
that <unk> <unk> quantum badly because its own plants cover only about half of its <unk> needs
pred:
that <unk> <unk> quantum badly because its own plants cover only about half of its <unk> needs