Generative Pretrained Structured Transformers: Unsupervised Syntactic Language Models at Scale

Anonymous ACL submission

Abstract

A syntactic language model (SLM) incrementally generates a sentence with its syntactic tree in a left-to-right manner. We present **G**enerative **P**retrained **S**tructured **T**ransformers (GPST), an unsupervised SLM at scale capable of being pre-trained from scratch on raw texts with high parallelism. GPST circumvents the limitations of previous SLMs such as relying on gold trees and sequential training. It consists of two components, a usual SLM supervised by a uni-directional language modeling loss, and an additional composition model, which induces syntactic parse trees and computes constituent representations, supervised by a bi-directional language modeling loss. We propose a representation surrogate to enable joint parallel training of the two models in a hard-EM fashion. We pre-train GPST on OpenWebText, a corpus with 9 billion tokens, and demonstrate the superiority of GPST over GPT-2 with a comparable size in numerous tasks covering both language understanding and language generation. Meanwhile, GPST also significantly outperforms existing unsupervised SLMs on left-to-right grammar induction, while holding a substantial acceleration on training.

1 Introduction

Pre-training a Transformer architecture (Vaswani et al., 2017) as a large language model has dominated the field of natural language processing (NLP) (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2018, 2019; Brown et al., 2020; Ouyang et al., 2022). While Transformer language models have exhibited remarkable performance over various downstream NLP tasks (Bang et al., 2023), the recursive compositions behind language are represented in an implicit and entangled form. In contrast, human language understanding exhibits explicit composition decisions, as exemplified by the garden path sentence (Dynel, 2009) "Time flies like an arrow; Fruit flies like a banana", where distinct syntactic configurations yield vastly divergent meanings[1]. In addition, human infants acquire such compositional capability without supervision (Saffran et al., 1996). These phenomena motivate us to explore an *unsupervised* approach to learning *explicit* compositions in language modeling.

A typical approach to achieve language modeling with explicit composition is to model the joint distribution of words and a syntactic tree within the framework of syntactic language models (SLMs) (Dyer et al., 2016). Though there has been a long line of research on SLMs, they are barely exploited as the backbone in state-of-the-art language modeling due to poor scalability. Recent Transformer-based SLMs (Sartran et al., 2022; Murty et al., 2023) require annotated parse trees or supervised parsers as structural supervision, leading to limited training data scales and domain adaption issues (McClosky et al., 2006). On the other hand, in unsupervised SLMs (Kim et al., 2019b; Shen et al., 2019a), non-terminal constituents are composed from their sub-constituents sequentially in a left-to-right manner, resulting in data dependencies that impede training parallelism.

In this work, we aim to pre-train an SLM *at scale* on raw texts. To this end, we propose **G**enerative **P**retrained **S**tructured **T**ransformer (GPST), an unsupervised SLM with the Transformer architecture as a backbone. A common practice in existing unsupervised SLMs is to learn structures by a uni-directional language modeling loss (LM loss). However, we empirically find such an asymmetric loss with only right-to-left feedback results in branching biases in the induced parse trees. Based on the insight, we propose two components in GPST, a composition model performing structural learning supervised by a bi-directional LM loss, and a generative model for uni-directional syntactic language modeling. Specifically, we train the GPST in a fashion similar to hard-EM (Liang et al., 2017): in E-step, the composition model runs a

[1](Fruit (flies (like a banana))) or ((Fruit flies) (like (a banana)))

pruned deep inside-outside algorithm to induce a parse tree and compute inside and outside representations of constituents simultaneously within logarithmic steps (Hu et al., 2024); while in M-step, we update all parameters of GPST by minimizing both the bi-directional (reconstructing the sentence from outside representations) and uni-directional LM loss given the induced tree. The key in the M-step lies in using the inside representations of constituents computed by the composition model as a surrogate of inputs for the generative model, which enjoys two advantages. First, the representations of all constituents pre-computed in the E-step can be simultaneously fed into the generative model, which breaks the data dependencies and facilitates training parallelism. Second, with these representations participating in generation, the uni-directional LM loss in the M-step could be back-propagated to not only the generative model but the composition model used in the E-step as well.

In experiments, we pre-train GPSTs with sizes comparable to those of GPT-2_{small} and GPT-2_{medium} on OpenWebText (Gokaslan and Cohen, 2019)(~ 9 billion tokens), and evaluate the models on various tasks including language understanding, language generation, and grammar induction. GPST demonstrates an approximately 60-fold training acceleration and over 7% absolute increase in left-to-right grammar induction in comparison with existing unsupervised SLMs. Meanwhile, GPST also shows advantages over GPT-2 across almost all language understanding/generation benchmarks. GPST provides constituent-level interfaces that are not inherent possessed by the conventional Transformer-based language models, and thus exhibits great potential to enhance interpretability (Hu et al., 2023), support multi-modality (Wan et al., 2022), and improve dense retrieval in the future. Our contributions are three-fold:

- We propose a SLM consisting of a composition model in addition to a generative model, which can be trained without gold trees via a novel approach akin to hard-EM.
- We propose a representation surrogate to enable joint parallel training of all components.
- To the best of our knowledge, GPST is the first unsupervised SLM able to be pre-trained from scratch on billions of tokens and surpass GPT-2 on various benchmarks. The experimental results demonstrate the potential of GPST as a backbone for large language models. The code will be released later.

2 Related work

Syntactic Language Models. There have been extensive studies on syntactic language modeling (Baker, 1979; Jelinek and Lafferty, 1991; Vinyals et al., 2015; Charniak et al., 2016; Dyer et al., 2016; Qian et al., 2021), in which words and constituent symbols are mixed up and generated in a left-to-right manner. Recent works (Sartran et al., 2022; Murty et al., 2023) utilize Transformers to parameterize action probability distributions, but relies on annotated parse trees or parsers trained on gold trees as structural guidance. Besides, unsupervised SLMs are also explored, by differentiable structured hidden layers (Kim et al., 2017; Shen et al., 2018, 2019a), reinforcement learning approaches (Yogatama et al., 2017), or variational approximations (Li et al., 2019; Kim et al., 2019b). Normally, these unsupervised models are trained in a sequential manner. Our model follows a similar generation paradigm, but has stark differences in model architecture and training approach.

Composition Models. A composition model transforms text encoding into a combinatorial optimization problem, learns and searches for the optimal structure, and encodes the text in a bottom-up manner along a binary tree via a composition function recursively. Maillard et al. (2019) proposes a CKY-like (Cocke, 1969; Kasami, 1966; Younger, 1967) encoder, in which high-level constituents are soft-weighted over composed representations of its sub-constituents. Drozdov et al. (2019) proposes a deep inside-outside algorithm (Baker, 1979; Lari and Young, 1990), enabling the encoder to learn underlying structures via an auto-encoding objective. Recently, a series of studies (Hu et al., 2021, 2022, 2024) have been conducted to reduce the inside algorithm complexity from cubic to linear. Our SLM is built on top of state-of-the-art composition modeling techniques, in which we achieve unsupervised learning and enhance training parallelism by taking advantage of the pruned inside-outside algorithm (Hu et al., 2024).

3 Methodology

Given a sentence $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ with x_i from a vocabulary $\mathbb{V}$ $(1 \leqslant i \leqslant n)$, our goal is to train an SLM without gold trees that can simultaneously generate $\mathbf{x}$ and its syntactic structure. We first introduce the generative architecture of GPST, and then elaborate on how to perform training and inference with the model.

3.1 Generative Model

GPST generates a sentence and its parse tree from left to right via two types of actions, GEN and COMP, along with a stack (Dyer et al., 2015) to maintain partially completed sub-trees during generation. GEN generates a word x and pushes its embedding into the stack. We denote such an action as $\text{GEN}(x)$, with $x \in \mathbb{V}$. COMP pops the top two elements off the stack, computes their composed representation, and pushes it back to the stack. A major difference in model architecture between GPST and existing unsupervised SLMs, such as URNNG (Kim et al., 2019b), is that GPST makes good use of the architecture of Transformers to parameterize the action probabilities and thus hidden states from previous actions can be directly accessed via self-attention during generation.

Figure 1 illustrates the generative process of GPST. The generative model comprises type layers and token layers, both consisting of multi-layered Transformers. Let us denote the stack at step t as $\mathbf{S}_t$, with $\mathbf{S}_t^0$ and $\mathbf{S}_t^1$ representing the top two elements, respectively. Initially, $\mathbf{S}_0^0$ is set to the embedding of the beginning-of-sentence token (i.e., $\langle \mathbf{bos} \rangle$ in Figure 1). At each step t, $\mathbf{S}_t^0$ along with a position ID w_t is fed into the type layers, yielding a hidden state $\mathbf{h}_t$, which is then utilized to predict the next action type y_t:

- If $y_t = 0$ (COMP), we set w_{t+1} as w_t, pop off $\mathbf{S}_t^0$ and $\mathbf{S}_t^1$ from $\mathbf{S}_t$, and compose them using a composition function. The composed representation is then pushed back into the stack. In such a case, action a_t at time step t is set to COMP.
- If $y_t = 1$ (GEN), we set w_{t+1} as $w_t + 1$, feed $\mathbf{h}_t$ to the subsequent token layers, and get an output state $\mathbf{g}_{w_t}$ that is used to generate x_{w_t+1}. In such a case, we have $a_t = \text{GEN}(x_{w_t+1})$.

Suppose that $\mathbf{a}_{\mathbf{xy}}$ is the action sequence to generate a sentence $\mathbf{x}$ and its parse tree $\mathbf{y}$, then the joint distribution of $\mathbf{x}$ and $\mathbf{y}$ can be formulated as:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{a}_{\mathbf{xy}}) = \prod_t p(a_t | a_{<t}), \tag{1}$$

where $p(a_t|a_{<t})$ is computed by:

$$\begin{aligned} p(\text{COMP}|a_{<t}) &= p(y_t = 0|a_{<t}), \\ p(\text{GEN}(x_{w_t+1})|a_{<t}) &= p(x_{w_t+1}|y_t = 1, a_{<t})p(y_t = 1|a_{<t}), \\ p(x_{w_t+1}|y_t = 1, a_{<t}) &= \text{softmax}(\text{MLP}_x(\mathbf{g}_{w_t})), \\ p(y_t|a_{<t}) &= \text{softmax}(\text{MLP}_y(\mathbf{h}_t)). \end{aligned}$$

$\text{MLP}_y(\cdot)$ and $\text{MLP}_x(\cdot)$ convert inputs to a 2-dim vector and a $|\mathbb{V}|$-dim vector, respectively. By predicting action types through shallow layers and tokens through deep layers, we can keep the total computational cost close to that of vanilla Transformers.

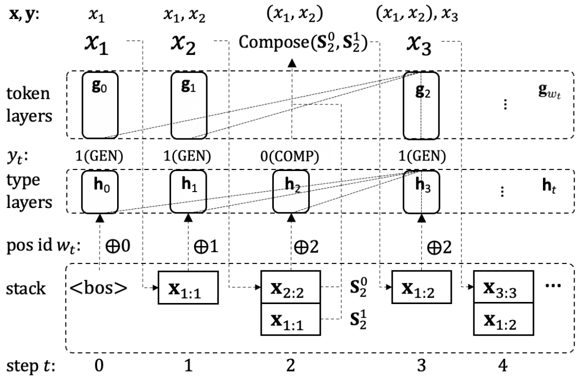


Figure 1: An illustration of the generative process of GPST. $\mathbf{x}_{i:j}$ denotes the sub tree representation spanning from i to j. As we use Transformers as the backbone, all previous hidden states are leveraged. At step t, the length of historical hidden states is t for the type layers and w_t for the token layers as illustrated with dotted lines for step 3.

3.2 Unsupervised Training

How to train an unsupervised SLM effectively and efficiently has always been a challenge. Existing methods suffer from two issues: asymmetric feedback and inability to train in parallel. The former arises from the uni-directional LM loss, and the latter stems from the inherent data dependency of each composition step on the representations of its sub-constituents from previous steps. We tackle both issues with an approach similar to hard-EM. In E-step, we employ a composition model to induce a parse tree through a pruned deep inside-outside algorithm. In M-step, we update both the composition model and the generative model by optimizing a joint objective based on the induced tree. Below we present details of the two steps and explain how they tackle the issues mentioned above.

E-step. During the E-step, the composition model searches for the best parse tree and composes representations through a deep inside-outside algorithm (Drozdov et al., 2019), as shown in Figure 2(a). In the inside pass, we compute the composed representation $\bar{\mathbf{i}}_{i,j}^k$ and the compatibility score $a_{i,j}^k$ for each span (i, j) with a split at k $(i \leq k < j)$ via function f_α and ϕ_α, respectively, which are formulated in Appendix A.2. We then compute each internal span representation $\mathbf{i}_{i,j}$ and its compatibility score $a_{i,j}$ as a weighted average over all possible $\bar{\mathbf{i}}_{i,j}^k$ and $a_{i,j}^k$:

$$\begin{aligned} \bar{a}_{i,j}^k &= \phi_\alpha(\mathbf{i}_{i,k}, \mathbf{i}_{k+1,j}) \,, \; a_{i,j}^k = \bar{a}_{i,j}^k + a_{i,k} + a_{k+1,j}, \\ \bar{\mathbf{i}}_{i,j}^k &= f_\alpha(\mathbf{i}_{i,k}, \mathbf{i}_{k+1,j}) \,, \; \hat{w}_{i,j}^k = \frac{\exp(a_{i,j}^k)}{\sum_{k'=i}^{j-1} \exp(a_{i,j}^{k'})}, \end{aligned}$$

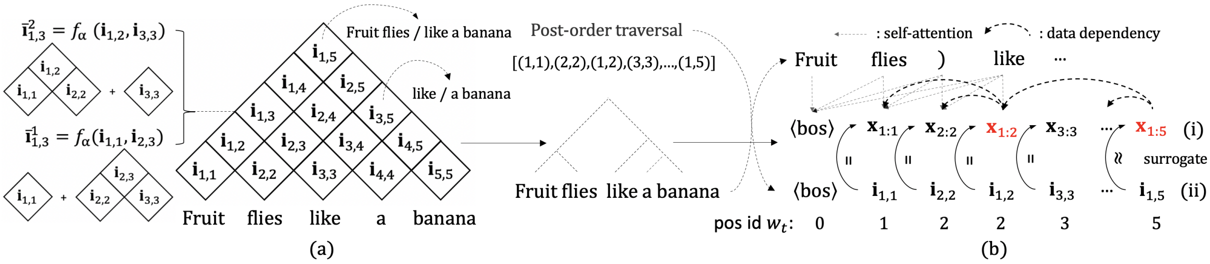


Figure 2: Illustration of the training process. ")" denotes the COMP action. (a) In the E-step, we induce a parse tree and compute constituent representations. (b)(i) Data dependencies within inputs of the generative model. (b)(ii) Illustration of the representation surrogate. $\mathbf{x}_{i,j}$ denotes the original input representation spanning over (i,j) composed from left to right.

$$\mathbf{i}_{i,j} = \sum_{k=i}^{j-1} \hat{w}_{i,j}^k \bar{\mathbf{i}}_{i,j}^k \,,\; a_{i,j} = \sum_{k=i}^{j-1} \hat{w}_{i,j}^k a_{i,j}^k.$$

Analogously, the outside pass computes each outside representation $\mathbf{o}_{i,j}$ in a top-down manner based on bi-directional information outside span (i,j). To accelerate computation, we use the pruned deep inside-outside algorithm (Hu et al., 2024) which achieves linear space complexity and approximately logarithmic parallel time complexity. The details of the algorithm and the complete outside pass are presented in Appendix A.1.

Note that for a given span (i,j), the best split point is k with the highest $a_{i,j}^k$. Thus, to derive a parse tree, we can recursively select the best split points top-down starting from the root span $(1,n)$.

The outside representations of tokens can be used to define an auto-encoding loss (i.e., predicting each token from its outside representation) for the composition model, which is optimized in the M-step:

$$\mathcal{L}_{ae} = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\exp(\mathbf{o}_{i,i}^T \mathbf{e}_{\mathbf{x}_i})}{\sum_{k=1}^{|\mathbb{V}|} \exp(\mathbf{o}_{i,i}^T \mathbf{e}_k)},$$

where $\mathbf{e}_k$ is the embedding of the k-th token in the vocabulary. As the auto-encoding loss provides feedback to each token representation from both sides of the token, the asymmetric feedback issue is addressed.

M-step. With the induced tree $\mathbf{y}$, we update the parameters of the composition model and the generative model in a joint manner. Denote the sequence of node spans in post-order as $[(i_0,j_0),(i_1,j_1),...,(i_{2n-1},j_{2n-1})]$. The action sequence can be formulated as:

$$a_t = \begin{cases} \textsc{Comp} \\ \textsc{Gen}(x_{i_t}) \end{cases}, \;\text{for}\; \begin{matrix} i_t < j_t \\ i_t = j_t \end{matrix}.$$

An auto-regression loss can be defined as:

$$\mathcal{L}_{ar} = -\log p(\mathbf{x},\mathbf{y}) = -\frac{1}{2n-1}\sum_{t=0}^{2n-1} \log p(a_t|a_{<t}).$$

However, even though the action sequence is given, there are still two challenges. First, there are data dependencies within the inputs for the generative model as mentioned earlier and shown in Figure 2(b), which impedes parallel training. Second, there are no feedforwards from the composition model to the generative model, so the two models are disconnected and hence cannot be trained jointly. To address these challenges, we employ the internal span representations computed by the composition model as a surrogate for the input representations of the generative model. Note that internal span representations do not contain any information outside spans, so there is no information leakage in the uni-directional generative model. As illustrated in Figure 2(b)(i), using the same composition function f_α, representations composed from left to right are equivalent to those composed bottom-up following the same binary tree. Thus we can use the internal span representations in the induced tree traversed in post-order as an approximation[2] of the inputs for the generative model as depicted in Figure 2(b)(ii). The key lies in the generative model sharing the same composition function as used in the composition model. As the internal span representations are already computed in the E-step, they can be fed into Transformers seamlessly at once to fully leverage the parallel training ability of the architecture. Moreover, the surrogate enables the representations computed by the composition model to participate in the generative model, and thus the two models can be jointly optimized via the auto-regression loss.

The final training loss for the M-step combines the auto-regression and auto-encoding losses as:

$$\mathcal{L} = \mathcal{L}_{ae} + \mathcal{L}_{ar}.$$

[2] It is an approximation because $\mathbf{i}_{i,j}$ is soft-weighted over all $\bar{\mathbf{i}}_{i,j}^k$ with the best split-point at k, while $\mathbf{x}_{i,j}$ is supposed to be composed by $\mathbf{x}_{i,k}$ and $\mathbf{x}_{k+1,j}$ in a hard manner during inference. However, our experimental results indicate that such an approximation approach has minimal impact on actual inference.

We empirically find that the combined loss leads to left-branching bias in parse trees induced by the composition model that is not observed when training with $\mathcal{L}_{ae}$ alone. A possible reason is that left-leaning trees provide more left-side context for each step during generation and thus are reinforced in learning. To tackle the issue, we stop gradient propagation from $\mathcal{L}_{ar}$ to $a_{i,j}^k$, which means only gradients from $\mathcal{L}_{ae}$ are allowed to be back-propagated along $a_{i,j}^k$. Note that other variables like $\mathbf{i}_{i,j}$ still receive gradients from $\mathcal{L}_{ar}$.

3.3 Inference

The action space of $\textsc{Gen}(x)$ is much larger than that of $\textsc{Comp}$, leading to an imbalance between their probabilities. Stern et al. (2017) point out that during beam search decoding, hypotheses in a beam should be grouped by the length of generated tokens instead of action history, which they refer to as "word-level search". However, their approach does not guarantee that the top-k next words searched are the optimal ones. To address the issue, we propose an improved word-level search tailored for our generation paradigm. The core idea is to guarantee that all hypotheses in a beam have the same number of $\textsc{Gen}(x)$ actions. Beams satisfying the condition are marked as $sync$ and otherwise $\overline{sync}$. Below we depict the entire word-level search process through an example shown in Figure 3:

1. Starting with a $sync$ beam, e.g., A, B, C and $(A, B), D$, we estimate the probability distribution of the next action for each hypothesis within it. For each possible action, we compute the probability of the resulting new hypothesis as the product of the probabilities of the current hypothesis and the action. The new hypotheses are pooled and ranked, and the top-k are retained (e.g., $A,(B,C)$ and $(A,B),D,E$).
2. If the current beam contains hypotheses with the last step being $\textsc{Comp}$, e.g., $A,(B,C)$, we continue to explore their next actions, update their probabilities, pool them with other hypotheses in the beam, and rank the top-k, until all the top-k hypotheses have $\textsc{Gen}(x)$ as their last action.
3. End generation upon reaching the length limit or producing an end token; otherwise, go back to step 1.

This method is also applicable to top-k random sampling (Fan et al., 2018), or parsing with a given input sentence by simply setting the probabilities of all $\textsc{Gen}(x)$ actions to zeros except for the correct next token.

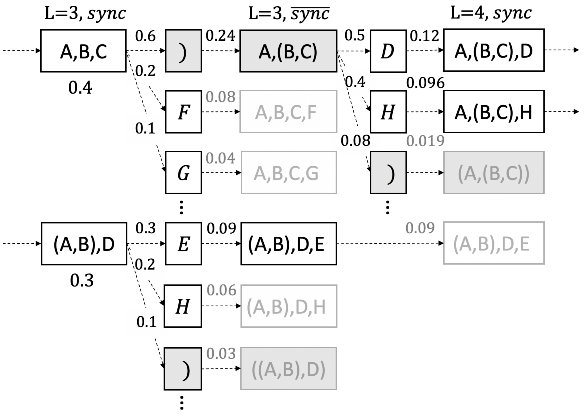


Figure 3: An illustration of beam search decoding of size 2. For simplicity, we use ")" to denote $\textsc{Comp}$ and upper case characters to denote words generated by $\textsc{Gen}(x)$. Boxes filled in gray are hypotheses with the last action being $\textsc{Comp}$. Grayed-out boxes are pruned out during beam search.

4 Experiments

To fairly compare GPST and GPT-2, we pre-train both models from scratch on the same corpus with the same setups and comparable parameter sizes. Evaluation is conducted on various language understanding/generation tasks. Besides, we also evaluate GPST on grammar induction to verify to what extent the induced parse trees are consistent with human annotation.

Pre-training Corpus. We pre-train models on WikiText-103 (Merity et al., 2017) and OpenWebText (Gokaslan and Cohen, 2019), where the two datasets contain 116 million tokens and 9 billion tokens, respectively. The context window size in pre-training is set to 1024. When a context involves more than one complete sentence, parse trees are induced for each sentence separately.

Hyper-parameters. Following GPT-2 (Radford et al., 2019), we use 768/1024-dimensional embeddings, a vocabulary size of 30522, 3072/4096-dimensional hidden layer representations, and 12/16 attention heads for the generative models of $\text{GPST}_{\text{small}}$ and $\text{GPST}_{\text{medium}}$, respectively. To align with Transformer layer counts in GPT-2, we configure $\text{GPST}_{\text{small}}$ with 3 type layers and 9 token layers, and $\text{GPST}_{\text{medium}}$ with 3 type layers and 21 token layers, respectively. We set the input dimension of the composition model to $256/512$, and the number of Transformer layers used in the composition function and decomposition function to 4 and 1, corresponding to the small and medium setups. The token embeddings are down-scaled before being fed into the composition model, and the constituent representations are up-scaled before

Models	corpus	SST2	COLA	MRPC(f1)	QQP(f1)	QNLI	RTE	MNLI-(m/mm)	average	#param.
GPT-2$_{\text{small}}$	wiki103	88.11	27.75	80.80	**85.37**	83.71	53.91	75.85/75.77	71.41	1.0x
GPST$_{\text{small w/o grad.stop}}$	wiki103	88.11	29.09	81.16	84.98	84.62	53.19	75.87/75.88	71.61	1.1x
GPST$_{\text{small w/o surrogate}}$	wiki103	88.07	**29.24**	80.98	85.08	84.05	52.71	76.47/76.36	71.62	1.1x
GPST$_{\text{small}}$	wiki103	**88.34**	28.41	**81.21**	85.33	**85.08**	**56.08**	**76.60/76.46**	**72.19**	1.1x
GPT-2$_{\text{small}}$	opw	90.71	40.53	83.20	86.55	85.60	58.72	79.53/79.75	75.57	1.0x
GPST$_{\text{small}}$	opw	90.94	44.51	84.72	86.70	86.91	**64.98**	79.60/80.15	77.31	1.1x
GPT-2$_{\text{medium}}$	opw	91.10	47.55	83.68	87.17	86.64	61.49	81.35/81.05	77.50	2.7x
GPST$_{\text{medium}}$	opw	**91.97**	**50.79**	**85.69**	**87.36**	**87.60**	64.86	**81.80/82.01**	**79.01**	3.0x
For Reference										
Ordered-Memory†	–	90.40	–	–/–	–/–	–	–	72.53/73.20	–	

Table 1: Evaluation results on GLUE benchmark. We mark out the best result of each group in bold. The results of Ordered-Memory† are copied from Ray Chowdhury and Caragea (2023).

being fed into GPST. All models are trained on 8 A100 GPUs with a learning rate of 5e-5/1e-4, $8 \times 32 \times 1024$ tokens per step, 5 billion and 15 billion total training tokens for WikiText-103 and OpenWebText, respectively.

4.1 Understanding Tasks

Dataset. We evaluate GPST on the GLUE benchmark (Wang et al., 2018), which collects tasks covering a broad range of natural language understanding (NLU) domains.

Evaluation Settings. We borrow and minimally modify the fine-tuning paradigm from Radford et al. (2018). Details are described in Appendix A.3. As the whole sentence is given, the composition model is utilized to induce the best tree and compose constituent representations as described in the E-step. The constituent representations in the induced tree are gathered in post-order as inputs for the generative model. We derive two additional baselines GPST$_{\text{w/o surrogate}}$ and GPST$_{\text{w/o grad.stop}}$ for ablation study. In GPST$_{\text{w/o surrogate}}$, all constituent representations of non-terminals are replaced by embeddings of a placeholder `COMP` as in Transformer Grammars (Sartran et al., 2022), and thus there is no interaction between the composition model and the generative model (i.e., they are separately optimized). In GPST$_{\text{w/o grad.stop}}$, partial gradient stopping is disabled to study the impact of left-leaning trees on downstream tasks. We run three rounds of fine-tuning with different seeds and report average results (accuracy by default) on the validation sets.

Results and Discussions. Table 1 reports the results on the GLUE benchmark. GPST significantly outperforms GPT-2 in both small and medium setups. We find that GPST$_{\text{w/o grad.stop}}$ and GPST$_{\text{w/o surrogate}}$ underperform GPST, but are still better than GPT-2. The performance drop of GPST$_{\text{w/o grad.stop}}$ indicates that poor structures compromise the performance of downstream tasks. GPST$_{\text{w/o surrogate}}$ is better than GPT-2, implying that as long as induced syntactic structures are utilized, simply replacing non-terminal representations with `COMP` embeddings is also helpful. It is, however, worse than GPST, demonstrating that computing representations of non-terminal constituents via an explicit composition function further benefits language understanding. One more interesting thing we find is that GPST consistently and most significantly outperforms baselines on the RTE task. One possible explanation is that certain relationships in RTE are predicated on negation words such as "not", which generally affects high-level semantics through compositions with other phrases. Explicit syntactic composition modeling contributes to a better representation of such cases.

4.2 Generation Tasks

4.2.1 Abstractive Summarization

Datasets. We conduct experiments on three summarization datasets: BBC extreme (XSum) (Narayan et al., 2018), CNN and DailyMail (Nallapati et al., 2016), and Gigaword (Napoles et al., 2012) to assess the performance of GPST in terms of language generation abilities. Statistics of the datasets are presented in Table 6 in Appendix A.4.

Evaluation Settings. For XSum and CNN/DailyMail, we truncate the documents and their summaries to 900 and 100 tokens respectively, and concatenate them with short prompt `Summary:`. For Gigaword, the truncating thresholds of documents and summaries are set to 400 and 120 respectively, following the settings of Rothe et al. (2020). Considering the complexity of the generation task, we primarily evaluate the models pre-trained on OpenWebText. More details are described in Appendix A.5. We apply the word-level search de-

Models	XSum				CNN/DailyMail				Gigaword			
	R-1	R-2	R-L	R-AVG	R-1	R-2	R-L	R-AVG	R-1	R-2	R-L	R-AVG
GPT-2$_{\text{small}}$	29.78	9.43	23.56	20.92	**35.54**	14.45	24.76	24.92	32.45	**14.84**	30.37	25.88
GPST$_{\text{small-w/o sync}}$	29.44	9.09	23.20	20.58	35.63	14.57	24.93	25.04	32.34	14.69	29.98	25.67
GPST$_{\text{small}}$	**29.86**	**9.51**	**23.70**	**21.02**	35.52	**14.65**	**25.01**	**25.06**	**32.53**	14.76	30.37	**25.89**
GPT-2$_{\text{medium}}$	31.91	11.11	25.28	22.76	37.18	15.23	25.59	26.00	33.13	15.27	30.85	26.42
GPST$_{\text{medium w/o sync}}$	31.66	10.91	25.16	22.58	37.07	15.45	25.69	26.07	32.83	15.06	30.59	26.16
GPST$_{\text{medium}}$	**31.96**	**11.31**	**25.58**	**22.95**	37.18	**15.69**	**26.00**	**26.29**	**33.19**	15.27	**30.91**	**26.46**

Table 2: Abstractive summarization results.

Models	Agr.	C.E.	G.P.E.	C.S.E.	Lcs.	L.D.D.	avg
WikiText-103							
GPT2$_{\text{small}}$	50.88	73.21	77.88	97.83	33.95	65.98	66.62
GPST$_{\text{small}}$	59.65	73.21	87.10	97.83	57.89	64.78	73.41
OpenWebText							
GPT$_{\text{small}}$	78.95	87.50	85.22	97.83	71.58	78.65	83.29
GPST$_{\text{small}}$	77.19	85.71	94.54	96.74	68.95	72.38	82.59
GPT$_{\text{medium}}$	64.91	94.64	86.41	98.91	73.42	79.38	82.95
GPST$_{\text{medium}}$	85.96	85.71	95.04	94.57	83.68	78.17	87.19
For Reference (Models with gold trees)							
TG	69.7	88.4	90.4	95.6	78.1	77.9	83.35
Pushdown Layers	79.0	92.0	84.2	100.0	77.8	77.5	85.08

Table 3: Syntactic generalization results. For reference, we list the results of models with gold trees from Sartran et al. (2022) and Murty et al. (2023).

scribed in §3.3 to top-k random sampling for GPST, except for models with w/o sync which only uses naive action-level beam search. ROUGE (Lin and Hovy, 2003) is employed as the evaluation metric.

4.2.2 Syntactic Generalization

Datasets. The syntactic generalization task (Hu et al., 2020) collects 34 test suites to assess syntactic generalizability of the models. The test suites are grouped into 6 circuits: Agreement (Agr.), Center Embedding (C.E), Garden-Path Effects (G.P.E), Cross Syntactic Expectation (C.S.E.), Licensing (Lcs.) and Long-Distance Dependencies (L.D.D.).

Evaluation Settings. We evaluate models on syntactic generalization test suites by comparing surprisals (Hale, 2001) without fine-tuning, as required by Hu et al. (2020). Surprisal: $S(w|C) = -\log_2 p(w|C)$ is defined as negative log conditional probabilities of a sub-sentence w given the left-side context C. In detail, when we apply word-level search with beam size b to do left-to-right parsing with a given input, we temporarily store b best hypotheses with their probability $p(\mathbf{x}_{<t}, \mathbf{y}_{<n(t)})$ at each token position t, in which $\mathbf{y}_{<n(t)}$ refers to the current latent structure before generating x_t. We marginalize $\mathbf{y}_{<n(t)}$ out of $p(\mathbf{x}_{<t}, \mathbf{y}_{<n(t)})$ by summing up all the probabilities of the b best hypotheses. Finally, we obtain the surprisal of a sub-sentence with starting position s and ending position e as $S(w|C) = -\log p(x_{<e}) + \log p(x_{<s-1})$.

To align with Murty et al. (2023) and Sartran et al. (2022), we set beam size b to 300.

4.2.3 Results and Discussions

Table 2 and 3 report the results of summarization and syntactic generalization tasks. Overall, the performance of GPST is comparable to GPT, with a slight advantage. One possible reason why the advantage of GPST on generalization tasks is not as significant as that on GLUE is the discrepancies between training and inference. During training, the constituent representations are computed via the inside algorithm, where the representations are soft-weighed over composed representations of valid sub-constituents. However, during inference, constituent representations are composed of the top two elements in the stack, which is a one-hot version of the inside algorithm. This issue could potentially be resolved using a hard inside-outside algorithm (Drozdov et al., 2020), which we may explore in our future work. Despite the discrepancies, our performance still slightly surpasses that of GPT-2, which adequately demonstrates the potential of GPST in generation tasks. One more interesting thing is that GPST$_{\text{medium}}$ even outperforms baselines with gold trees in the syntactic generalization task, and the results of all GPSTs maintain a lead on Garden-Path Effect. Note that the results have a large variance due to the relatively small size of the evaluation set, e.g., GPT$_{\text{medium}}$ even underperforms GPT$_{\text{small}}$. However, the results still imply that unsupervised syntactic LMs have reached a critical point where they can surpass approaches reliant on gold trees.

4.3 Grammar Induction

Baselines & Dataset. We select baselines that report left-to-right unsupervised parsing results: Neural variational (NV) approaches (Li et al., 2019), PRPN (Shen et al., 2018) and ON-LSTM (Shen et al., 2019b). For reference, we also select some baselines performing parsing through the inside algorithm only: URNNG (Kim et al., 2019b), C-PCFG (Kim et al., 2019a), DIORA (Drozdov et al., 2019) and Fast-R2D2 (Hu et al., 2022). We report their performance on PTB (Marcus et al., 1993).

Besides, we also report results of $\text{GPST}_{\text{w/o grad.stop}}$ and $\text{GPST}_{\text{w/o surrogate}}$ for checking the gains from partial gradient stopping and joint pre-training achieved by the representation surrogate.

Evaluation Settings. We continue to fine-tune all models on the training set of PTB for 10 epochs with batch size set to 32 after pre-training. Since GPST takes word pieces as inputs, we provide our model with word-piece boundaries as non-splittable spans to align with models with word-level inputs. We apply different inference algorithms to grammar induction. For the inside algorithm, we directly evaluate the parse tree induced by the composition model. For the left-to-right parsing, we apply improved word-level search described in §3.3 with a beam size of 20 to parse the given text, except for $\text{GPST}_{\text{w/o sync}}$ which employs action-level sync beam search for parsing. We adopt sentence-level unlabeled F_1 as the evaluation metric, with the same setup as Kim et al. (2019a) where punctuations are discarded and words are lowercased. We evaluate the checkpoints from all epochs on the validation set, pick the best one, and then report its performance on the test set.

Models	corpus	inference alg.	F1
NV(unsupervised)	WSJ	left to right	29.0
NV(+linguistic rules)	WSJ	left to right	42.0
PRPN	WSJ	left to right	37.4
ON-LSTM	WSJ	left to right	47.4
$\text{GPST}_{\text{small w/o sync}}$	wiki103	left to right	43.64
$\text{GPST}_{\text{small}}$	wiki103	left to right	**55.25**
$\text{GPST}_{\text{small w/o sync}}$	opw	left to right	43.09
$\text{GPST}_{\text{small}}$	opw	left to right	51.40
$\text{GPST}_{\text{medium w/o sync}}$	opw	left to right	43.37
$\text{GPST}_{\text{medium}}$	opw	left to right	54.71
For Reference			
$\text{GPST}_{\text{small w/o grad.stop}}$	wiki103	inside	42.46
$\text{GPST}_{\text{small w/o surrogate}}$	wiki103	inside	50.27
$\text{GPST}_{\text{small}}$	wiki103	inside	57.46
$\text{GPST}_{\text{small}}$	opw	inside	53.95
$\text{GPST}_{\text{medium}}$	opw	inside	56.27
URNNG	WSJ	inside	45.4
C-PCFG	WSJ	inside	55.2
DIORA	WSJ	inside	55.7
Fast-R2D2	wiki103	inside	57.2
Oracle	—	—	84.3

Table 4: Results on unsupervised left-to-right parsing.

Results and Discussions. There are several observations from the results shown in Table 4. First and foremost, we find that our unsupervised left-to-right parsing achieves comparable performance with the bi-directional inside algorithm, significantly surpassing previous left-to-right grammar induction baselines. Such results indicate the structures generated by GPST are meaningful and consistent with those from humans. Secondly, a larger pre-training corpus may not necessarily bring improvement. A plausible explanation is that OpenWebText, mixed with more non-natural text such as URLs, introduces additional noise, leading to a performance drop. The results indicate the importance of high-quality corpora for structural learning. Thirdly, the performance of $\text{GPST}_{\text{w/o surrogate}}$ infering with the inside algorithm drops a lot. We suppose the main reason is that disabling the representation surrogate prevents the composition model from receiving long-term feedback from tokens on the right introduced by the auto-regression loss. Lastly, the performance decline of $\text{GPST}_{\text{w/o grad.stop}}$ corroborates the impact of asymmetric loss on structural learning. We attach trees parsed by GPST in Appendix A.6 for case studies.

4.4 Training Efficiency

Finally, we conduct a fair comparison of training efficiency with other unsupervised SLMs. We keep the model sizes and memory usage comparable and the training tokens the same. We report their time consumption in Table 5, from which we can observe the huge advantage of GPST over the baselines in terms of efficiency.

		sentence length			
	#param.	128	256	512	1024
GPST	24M	1x	1x	1x	1x
URNNG	23M	130.6x	955.3x	n/a	n/a
OM	28M	2.0x	9.2x	25.4x	63.3x

Table 5: Training acceleration on the same number of tokens.

5 Conclusion

In this paper, we propose an unsupervised approach to train GPST at scale efficiently. A key insight of our work is to guide the left-to-right structural learning with symmetric supervision such as an auto-regression loss, which can receive feedback from both sides. A key technical contribution is that we propose the representation surrogate which enables joint training of all components in parallel. Besides, the composition model of GPST can be regarded as an enhancement to the conventional embedding layer, which provides context-invariant embeddings of various granularities beyond token embeddings. Our experiment results show the superiority of GPST on language understanding, generation, and left-to-right grammar induction, which demonstrate the potential of GPST as a foundational architecture for large language models.

6 Limitation

Despite GPST achieving a multiple-fold acceleration compared to previous syntactic language models, it still requires 1.5 to 5 times the training time compared to vanilla GPTs (the inference time can be comparable if well implemented). The more layers there are in the type/token layers, the lower the overall time multiplier becomes. The additional training time comes from the composition model which only accounts for one-tenth of the overall model parameters. Though the pruned inside-outside can be completed in approximately $\log n$ steps, it involves significant memory transfers, thus making it bounded by the speed of memory swapping across hard-wares. Meanwhile, our implementation is quite naive, without any operator fusion or hardware-aware implementation. Thus there should be multiple potential ways to further reduce the time consumption of the composition model in the future.

References

James K. Baker. 1979. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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.

John Cocke. 1969. *Programming Languages and Their Compilers: Preliminary Notes*. New York University, USA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Andrew Drozdov, Subendhu Rongali, Yi-Pei Chen, Tim O'Gorman, Mohit Iyyer, and Andrew McCallum. 2020. Unsupervised parsing with S-DIORA: single tree encoding for deep inside-outside recursive autoencoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4832–4845. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. 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 1129–1141, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

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.

Marta Dynel. 2009. Humorous garden-paths: A pragmatic-cognitive study. Cambridge Scholars.

Angela Fan, Mike Lewis, and Yann N. Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 889–898. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Second meeting of the north american chapter of the association for computational linguistics*.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Xiang Hu, Xinyu Kong, and Kewei Tu. 2023. A multi-grained self-interpretable symbolic-neural model for single/multi-labeled text classification. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Xiang Hu, Haitao Mi, Liang Li, and Gerard de Melo. 2022. Fast-R2D2: A pretrained recursive neural network based on pruned CKY for grammar induction and text representation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2809–2821, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiang Hu, Haitao Mi, Zujie Wen, Yafang Wang, Yi Su, Jing Zheng, and Gerard de Melo. 2021. R2D2: recursive transformer based on differentiable tree for interpretable hierarchical language modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4897–4908. Association for Computational Linguistics.

Xiang Hu, Qingyang Zhu, Kewei Tu, and Wei Wu. 2024. Augmenting transformers with recursively composed multi-grained representations. In *The Twelfth International Conference on Learning Representations*.

Frederick Jelinek and John D. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–353.

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

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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

Yoon Kim, Alexander M Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pages 1105–1117.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 4(1):35–56.

Bowen Li, Jianpeng Cheng, Yang Liu, and Frank Keller. 2019. Dependency grammar induction with a neural variational transition-based parser. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6658–6665. AAAI Press.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*, pages 150–157.

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*.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2019. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *Nat. Lang. Eng.*, 25(4):433–449.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 4-9, 2006, New York, New York, USA*. The Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. 2023. Pushdown layers: Encoding recursive structure in transformer language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3233–3247.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016.

Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montréal, Canada. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Peng Qian, Tahira Naseem, Roger Levy, and Ramón Fernandez Astudillo. 2021. Structural guidance for transformer language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3735–3745.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jishnu Ray Chowdhury and Cornelia Caragea. 2023. Beam tree recursive cells. In *Proceedings of the 40th International Conference on Machine Learning*.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Jenny R. Saffran, Richard N. Aslin, and Elissa L. Newport. 1996. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928.

Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. 2022. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. *Transactions of the Association for Computational Linguistics*, 10:1423–1439.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron C. Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yikang Shen, Shawn Tan, Arian Hosseini, Zhouhan Lin, Alessandro Sordoni, and Aaron C Courville. 2019a. Ordered memory. *Advances in Neural Information Processing Systems*, 32.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2019b. Ordered neurons: Integrating tree structures into recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1695–1700, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

Bo Wan, Wenjuan Han, Zilong Zheng, and Tinne Tuytelaars. 2022. Unsupervised vision-language grammar induction with shared structure modeling. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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

A Appendix

A.1 Pruned inside-outside algorithm

Fast-R2D2 (Hu et al., 2022) introduces a pruned variant of the inside algorithm that reduces its complexity from $O(n^3)$ to $O(n)$ in both space and time. Building on this, ReCAT (Hu et al., 2024) extends the pruning method to the inside-outside algorithm, and further enables it to complete in approximate $\log n$ steps, whose key idea is to prune out unnecessary cells in the chart-table and encode cells in different rows simultaneously. The main idea of the pruning process is to decide which two spans should be merged at each step during the inside pass and prune out cells that would break the non-splittable span. An unsupervised top-down parser is applied to determine the merge order of spans. Given a sentence $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, the top-down parser assigns each split point a score v_i s.t. $1 \leq i \leq n-1$, and recursively split the sentence into two in the descending order of the scores shown in Figure 4 (a). Hence, the reverse order of the split points could be used to decide which cells to merge. Specifically, the pruned inside-outside algorithm works as follows:

0. Prepare merge batches according to the height of merge points in the induced tree, with the lowest merge points in the first batch, as illustrated in Figure 4(b).
1. Merge each pair of adjacent cells into one according to the current merge batch. For example, in Figure 5(b), at merge point 1, we merge x_1 and x_2 into $x_{1:2}$; at merge point 5, we merge x_5 and x_6 into $x_{5:6}$.
2. Remove all conflicting cells that would break the now non-splittable span from Step 1, e.g., the dark cells in Figure 5(c), and reorganize the chart table much like in the Tetris game as in (d).
3. Encode the cells that just descend to height m and record their valid splits in $\mathcal{K}$, e.g., the cells highlighted with stripes in Figure 5(d) with valid splits $\{2, 3\}$ for span $(1, 4)$ and $\{3, 4\}$ span $(3, 6)$. Go back to Step 1 until no blank cells are left.

Therefore, the entire inside process can be completed within steps equal to the height of the tree. Using the valid splits $\mathcal{K}$ recorded for each cell during the pruning process, we now have the new inside state transition equation as:

$$\bar{a}^k_{i,j} = \phi_\alpha(\mathbf{i}_{i,k}, \mathbf{i}_{k+1,j})\,, a^k_{i,j} = \bar{a}^k_{i,j} + a_{i,k} + a_{k+1,j}$$

$$\bar{\mathbf{i}}^k_{i,j} = f_\alpha(\mathbf{i}_{i,k}, \mathbf{i}_{k+1,j})\,, \hat{w}^k_{i,j} = \frac{\exp(a^k_{i,j})}{\sum_{k' \in \mathcal{K}_{i,j}} \exp(a^{k'}_{i,j})}$$

$$\mathbf{i}_{i,j} = \sum_{k \in \mathcal{K}_{i,j}} \hat{w}^k_{i,j} \bar{\mathbf{i}}^k_{i,j}\,, a_{i,j} = \sum_{k \in \mathcal{K}_{i,j}} \hat{w}^k_{i,j} a^k_{i,j}$$

where $\mathcal{K}_{i,j}$ is the valid splits set for span (i, j). According to $\mathcal{K}$, we can obtain a mapping from a span to its immediate sub-spans. By reversing such mapping, we get a mapping from a span to its valid immediate parent spans denoted as $\mathcal{P}$, which records the non-overlapping endpoint k in the parent span (i, k) or (k, j) for a given span (i, j).

Thus, for the outside pass, we have:

$$\bar{\mathbf{o}}^k_{i,j} = \begin{cases} f_\beta(\mathbf{o}_{i,k}, \mathbf{i}_{j+1,k}) \\ f_\beta(\mathbf{o}_{k,j}, \mathbf{i}_{k,i-1}) \end{cases}, \bar{b}^k_{i,j} = \begin{cases} \phi_\beta(\mathbf{o}_{i,k}, \mathbf{i}_{j+1,k}) \\ \phi_\beta(\mathbf{o}_{k,j}, \mathbf{i}_{k,i-1}) \end{cases},$$

$$b^k_{i,j} = \begin{cases} a_{j+1,k} + \bar{b}^k_{i,j} + b_{i,k}, \\ a_{k,i-1} + \bar{b}^k_{i,j} + b_{k,j} \end{cases}, \text{ for } \begin{matrix} k > j \\ k < i \end{matrix},$$

$$\check{w}^k_{i,j} = \frac{\exp(b^k_{i,j})}{\sum_{k' \in \mathcal{P}_{i,j}} \exp(b^{k'}_{i,j})},$$

$$\mathbf{o}_{i,j} = \sum_{k' \in \mathcal{P}_{i,j}} \check{w}^k_{i,j} \bar{\mathbf{o}}^k_{i,j}\,, b_{i,j} = \sum_{k' \in \mathcal{P}_{i,j}} \check{w}^k_{i,j} b^k_{i,j}.$$

We optimize the top-down parser jointly at the M-step with GPST. Given the parse tree $\mathbf{y}$ induced at the E-step, we maximize $p(\mathbf{y}|\mathbf{x}; \Theta)$ for the top-down parser, whose parameters are denoted as Θ. As shown in Figure 4(a), at t step, the span corresponding to a given split a_t is determined, which is denoted as (i^t, j^t). Thus we can minimize the negative log-likelihood of the parser as follows:

$$p(a_t|\mathbf{x}, \Theta) = \frac{\exp(v_{a_t})}{\sum_{k=i^t}^{j^t-1} \exp(v_k)},$$

$$\mathcal{L}_p = -\log p(\mathbf{y}|\mathbf{x}; \Theta) = -\sum_{t=1}^{n-1} \log p(a_t|\mathbf{x}; \Theta).$$

We notice that the steps to finish the pruned inside-outside algorithm depend on the highest tree in a batch, thus any extremely skewed tree may result in a significant increase in time consumption. A straightforward approach to reduce the maximum height of parse trees in a batch is to introduce a height penalty. During the inside pass, the weighted tree height of span (i, j) could be computed as:

$$\bar{h}^k_{i,j} = \max(h_{i,k}, h_{k+1,j}) + 1, h_{i,j} = \sum_{k \in \mathcal{K}_{i,j}} \hat{w}^k_{i,j} \bar{h}^k_{i,j}$$

To minimize the impact of height penalties on grammar induction, we set a threshold H_{thrs} which

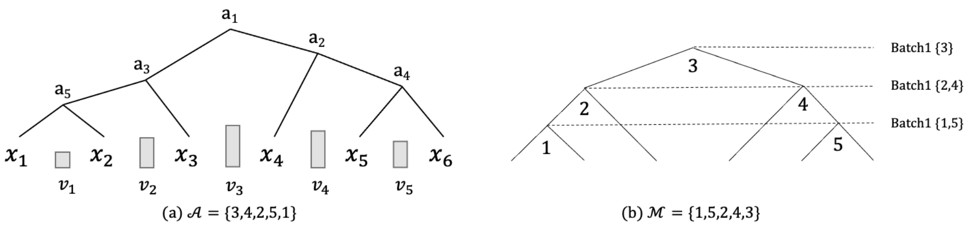


Figure 4: Fast encoding follows the order given by a top-down parser, with the merging order $\mathcal{M}$ being the reverse order of the split point sequence $\mathcal{A}$. x_i denotes the i_{th} token in a sentence of length 6. Numbers in $\mathcal{A}$ and $\mathcal{M}$ denote the indices of the split/merge point between tokens. v_j denotes the split score of j_{th} split point, predicted by the top-down parser.

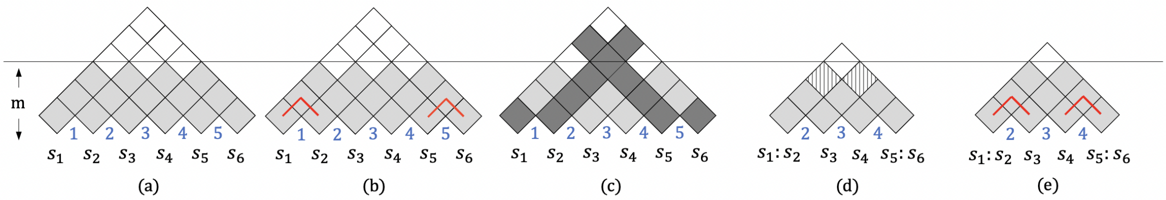


Figure 5: The initial step of encoding in $O(\log n)$ steps. The numbers in blue correspond to the indices of the split points introduced in Figure 4.

is 15 by default. Only trees that exceed this threshold will be affected.

$$\mathcal{L}_h = \frac{1}{n}\max(h_{1,n} - H_{thrs}, 0)$$

Thus the final auto-encoding objective is:

$$\mathcal{L}^*_{ae} = \mathcal{L}_{ae} + \mathcal{L}_h + \mathcal{L}_p$$

A.2 Composition function and score function

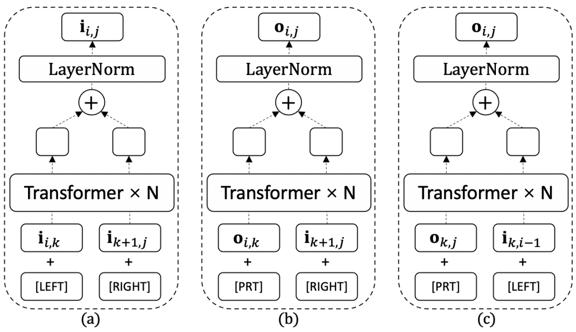


Figure 6: Model illustrations for the composition and decomposition functions.

We borrow the idea from Hu et al. (2021) to use Transformers as the backbone of the composition function f_α. As shown in Figure 6(a), composition function f_α takes left/right constituent representations $\mathbf{i}_{i,k}$/$\mathbf{i}_{k+1,j}$ along with their role embeddings `[LEFT]`/`[RIGHT]` into N-layered Transformers as inputs, passes the summation of their corresponding outputs through a layer normalization layer to get the composed representation. Decomposition function f_β works analogously as shown in Figure 6(b) and (c), with `[PRT]` as the role embedding for parents.

We define the score function ϕ_α as:

$$\phi_\alpha(\mathbf{l}, \mathbf{r}) = \mathrm{MLP}^l_\alpha(\mathbf{l})^T\,\mathrm{MLP}^r_\alpha(\mathbf{r})/\sqrt{d} \qquad (2)$$

where $\mathbf{l}$ and $\mathbf{r}$ are representations for left/right constituents. MLP^l_α and MLP^r_α are used to capture syntactic features from the left and right inputs, which convert inputs to d-dimensional vectors. Analogously, ϕ_β is defined as:

$$\phi_\beta(\mathbf{p}, \mathbf{l}) = \mathrm{MLP}^p_\beta(\mathbf{p})^T\,\mathrm{MLP}^l_\beta(\mathbf{l})/\sqrt{d}$$
$$\phi_\beta(\mathbf{p}, \mathbf{r}) = \mathrm{MLP}^p_\beta(\mathbf{p})^T\,\mathrm{MLP}^r_\beta(\mathbf{r})/\sqrt{d}$$

where $\mathbf{p}$ is the outside representation of a parent. MLP^l_β, MLP^r_β, and MLP^p_β are used to capture features from left/right children and parents respectively.

A.3 Glue fine-tuning

In detail, we append a `CLS` token after the input sequence and then feed the hidden states of the `CLS` tokens to a linear layer as the logits for classification. An additional cross-entropy loss along with the pre-training objective is used during fine-tuning.

A.4 Summarization dataset statistics

BBC extreme (XSum) comprises 204k document-summary pairs for single-sentence summarization of long documents. CNN and DailyMail (CNN/-DailyMail) contains 287k training pairs, each consisting of a document annotated with highlights. Gigaword focuses on sentence summarization with 3.8M sentence-summary training pairs conversely. We organize the statistics in Table 6.

	XSum	CNN/DailyMail	Gigaword
Training Set	204k	287k	3.8M
Test Set	11.3k	11.5k	1.95k

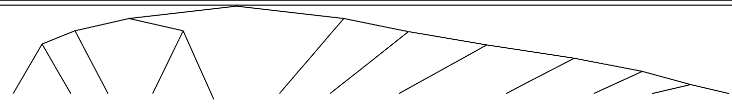
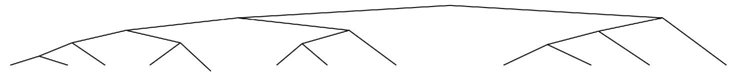
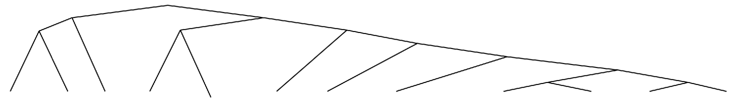
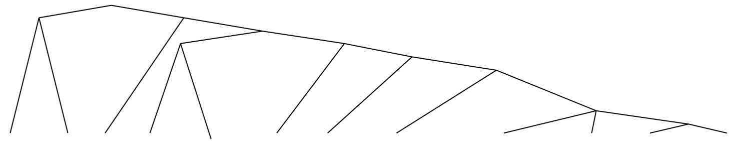
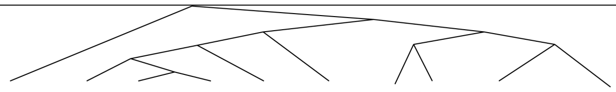
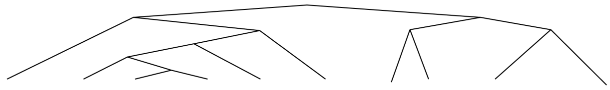
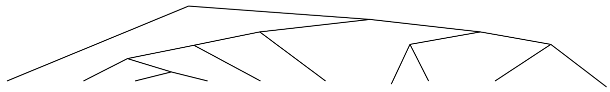
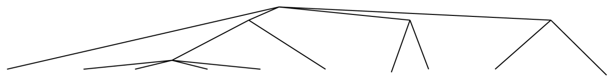
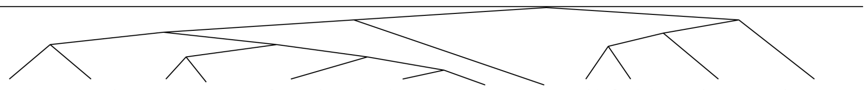
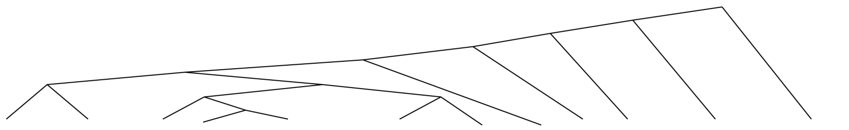
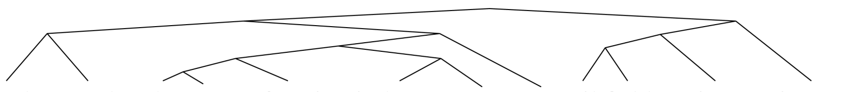
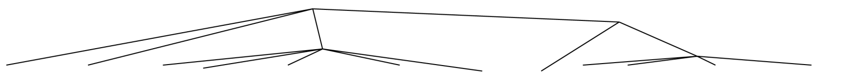
Table 6: Detailed statistics for summarization datasets.

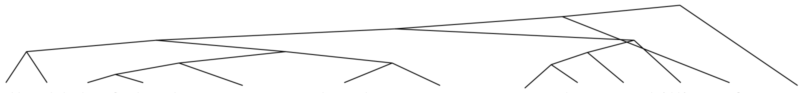
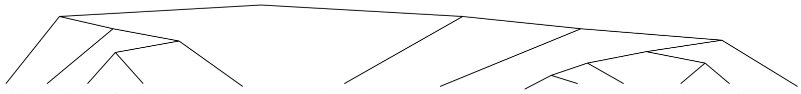
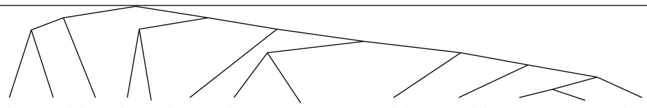
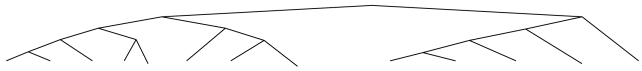
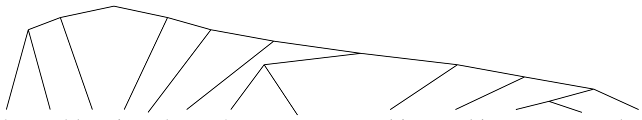
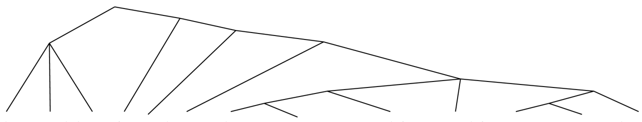
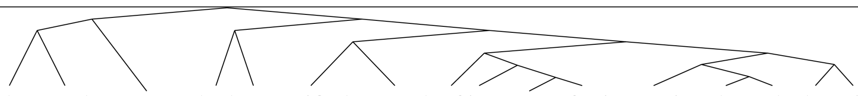
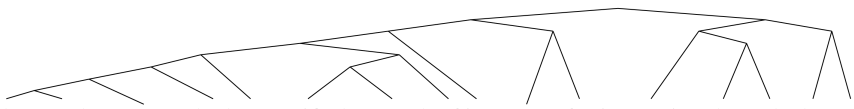
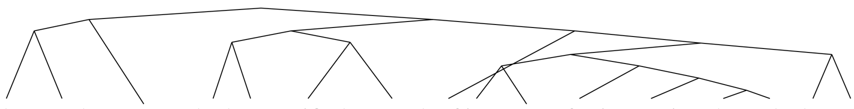
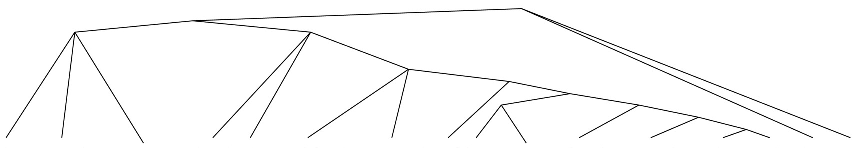
A.5 Summarization fine-tuning

We fine-tune for 15 epochs with a batch size of 16 on XSum and CNN/DailyMail datasets. For Gigaword, we fine-tune for 10 epochs with a batch size of 64. Top-k random sampling with $k = 2$ is used as the basic inference method as suggested in GPT-2 (Radford et al., 2019).

A.6 Case studies

Please refer to the following pages.

System	Tree
Left to right	 Skipper 's said the merger will help finance remodeling and future growth
Left to right w/o sync	 Skipper 's said the merger will help finance remodeling and future growth
Inside	 Skipper 's said the merger will help finance remodeling and future growth
Gold	 Skipper 's said the merger will help finance remodeling and future growth
Left to right	 Yesterday the stock market 's influence at first created nervousness
Left to right w/o sync	 Yesterday the stock market 's influence at first created nervousness
Inside	 Yesterday the stock market 's influence at first created nervousness
Gold	 Yesterday the stock market 's influence at first created nervousness
Left to right	 And yesterday the top performing industry group was oil field equipment issues
Left to right w/o sync	 And yesterday the top performing industry group was oil field equipment issues
Inside	 And yesterday the top performing industry group was oil field equipment issues
Gold	 And yesterday the top performing industry group was oil field equipment issues

System	Tree
Left to right	 All told the federal government already guarantees more than 900 billion of mortgages
Left to right w/o sync	 All told the federal government already guarantees more than 900 billion of mortgages
Inside	 All told the federal government already guarantees more than 900 billion of mortgages
Gold	 All told the federal government already guarantees more than 900 billion of mortgages
Left to right	 The real key is to have the economy working and interest rates down
Left to right w/o sync	 The real key is to have the economy working and interest rates down
Inside	 The real key is to have the economy working and interest rates down
Gold	 The real key is to have the economy working and interest rates down
Left to right	 The Dutch company had n't notified Burmah of its reason for increasing the stake he said
Left to right w/o sync	 The Dutch company had n't notified Burmah of its reason for increasing the stake he said
Inside	 The Dutch company had n't notified Burmah of its reason for increasing the stake he said
Gold	 The Dutch company had n't notified Burmah of its reason for increasing the stake he said