Semformer: Transformer Language Models with Semantic Planning

Yongjing Yin[1,2], **Junran Ding**[2], **Kai Song**[4], **Yue Zhang**[2,3]*
[1] Zhejiang University
[2] School of Engineering, Westlake University
[3] Institute of Advanced Technology, Westlake Institute for Advanced Study
[4] ByteDance
{yinyongjing,dingjunran}@westlake.edu.cn
yue.zhang@wias.org.cn

Abstract

Next-token prediction serves as the dominant component in current neural language models. During the training phase, the model employs teacher forcing, which predicts tokens based on all preceding ground truth tokens. However, this approach has been found to create shortcuts, utilizing the revealed prefix to spuriously fit future tokens, potentially compromising the accuracy of the next-token predictor. In this paper, we introduce Semformer, a novel method of training a Transformer language model that explicitly models the semantic planning of response. Specifically, we incorporate a sequence of planning tokens into the prefix, guiding the planning token representations to predict the latent semantic representations of the response, which are induced by an autoencoder. In a minimal planning task (i.e., graph path-finding), our model exhibits near-perfect performance and effectively mitigates shortcut learning, a feat that standard training methods and baseline models have been unable to accomplish. Furthermore, we pretrain Semformer from scratch with 125M parameters, demonstrating its efficacy through measures of perplexity, in-context learning, and fine-tuning on summarization tasks[1].

1 Introduction

Neural language models (LMs) (Bengio, 2008), a fundamental component of natural language processing (NLP), have witnessed significant advancements in recent years. By scaling up model sizes and pretraining on extensive text, large language models (LLMs) have successfully learned language and world knowledge, which has resulted in promising performance across various tasks and even demonstrated reasoning capabilities (Brown et al., 2020; Wei et al., 2022; OpenAI, 2023; Touvron et al., 2023; Schaeffer et al., 2023). The success of these models can be attributed to a straightforward training paradigm: next-token prediction with teacher forcing (Williams and Zipser, 1989), in which the models are trained to predict tokens using all preceding ground truth tokens as input.

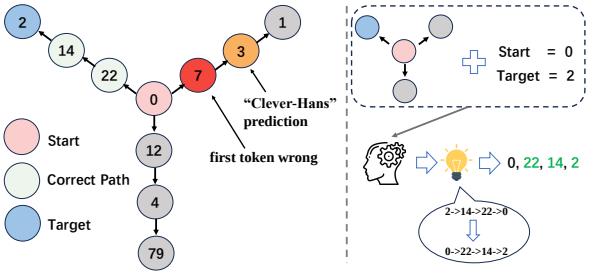


Figure 1: The Clever Hans cheat in a graph path-finding problem which is a minimal lookahead task. The task is to find the correct path based on the adjacency list, the start node, and the target node.

Recent studies, however, have raised concerns about the efficacy of the aforementioned training scheme in facilitating the learning of an accurate problem solver or planner (Malach, 2024; Wies et al., 2023; Bachmann and Nagarajan, 2024; Gloeckle et al., 2024; Pfau et al., 2024). For instance, the graph path-finding task—which necessitates lookahead and planning— demonstrates that teacher forcing can lead to a Clever Hans Cheat phenomenon characterized by shortcut learning (Bachmann and Nagarajan, 2024). Consequently, the later nodes such as 3 and 1 in Figure 1 become easier to predict, while the first node of the answer (i.e., 22) becomes more challenging to learn. This could result in a highly inaccurate next-token predictor, which would struggle to generalize to unseen problems, even without considering out-of-distribution and length generalization.

Humans, intuitively, do not rely solely on historical context to solve a problem (Du et al., 2023). Instead, they formulate an abstract plan based on the problem at hand, which subsequently guides them towards the final answers. For the problem in Figure 1, the quickest solution is to look ahead at the later nodes to identify a unique path correspond-

*Corresponding author
[1]https://github.com/ARIES-LM/Semformer.git

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 18669–18680
November 12-16, 2024

ing to the problem, and then reverse this found path to generate the correct answer. In general, a language model should internalize the process of looking ahead or "thinking about the future". The semantics of finding the response path is predicted by internal computation, with token output guided by the intended semantics.

To this end, we incorporate semantic planning into next-token prediction in a decoder-only Transformer (Vaswani et al., 2017; Radford et al., 2019), which we refer to as Semformer. Our Semformer is composed of a language model and an autoencoder that is used only during training. For the language model, we introduce a semantic planning token sequence that follows the prefix of the input. This sequence disregards the general next-token prediction loss and is utilized to predict the latent representations of the subsequent tokens. The autoencoder learns to generate the sequence of latent representations, compressing the subsequent tokens into a low-dimensional space.

On the graph path-finding problem (Bachmann and Nagarajan, 2024), our Semformer achieves almost 100% accuracy scores on the settings of different levels of difficulty, showing superiority to the related baselines. Only introducing dummy tokens in the sequence (i.e., Pause Transformer) (Goyal et al., 2024) fails to learn the planning task. Moreover, our Semformer learns to solve the problem significantly faster than the baselines, merely one epoch based on the GPT2-Large (Radford et al., 2019). Further, to validate the effectiveness of this architecture on general LM pretraining, we train Transformer models with 125M parameters from scratch on OpenWebText. Semformer results in improvements on perplexity evaluation, in-context learning, and fine-tuning on abstractive summarization.

2 Related Work

Next-token Prediction for next-token prediction. Despite being the standard training objective, next-token prediction has faced several challenges. On one hand, criticisms target the error accumulation caused by *autoregressive inference* (Kääriäinen, 2006; Ross and Bagnell, 2010; Dziri et al., 2023; LeCun, 2024). On the other hand, there has been debate about whether *teacher forcing* can learn an accurate next-token predictor especially for reasoning and planning tasks. Bubeck et al. (2023) report failures on GPT4 experimental report and they speculate the failures result from the "linear thinking" in next-token prediction. Du et al. (2023) informally note that some next-tokens can be hard to learn as they require a global understanding of what will be uttered in the future. Bachmann and Nagarajan (2024) demonstrate the Clever Hans cheat and the inference error can happen at the beginning. While language models are often shown to perform worse on out-of-distribution data (McCoy et al., 2023), Bachmann and Nagarajan (2024) demonstrate that they can fail even test in the same distribution. In addition, Malach (2024) and Wies et al. (2023) argue that some complex multi-hop tasks become learnable via next-token prediction only when providing a preceding chain-of-thought supervision for each hop. Pfau et al. (2024) also find that the learning of using filler tokens necessitates specific and dense supervision. The above studies support our motivation to provide general dense supervision for language models.

Beyond the next-token prediction, various training paradigms have been proposed including non-autoregressive models (Gu et al., 2018), diffusion LM (Li et al., 2022; Zhang et al., 2023), and multiple token prediction (Qi et al., 2020; Monea et al., 2023; Gloeckle et al., 2024). Predicting multiple future tokens is originally for accelerating inference, and Gloeckle et al. (2024) recently show that it can also avoid the localness issue of next-token prediction with teacher forcing. Zhang et al. (2023) introduce a latent diffusion model to generate paragraph representations induced by a variational autoencoder (Kingma and Welling, 2014), and feed them into the language model to help paragraph generation. Rather than significantly changing the model architecture, we internalize the planning ability into the language model, achieved through the semantic representation prediction of the subsequent sequence.

Custom Tokens in Language Modeling Custom tokens can be used to increase model capacity used as additional memory (Sukhbaatar et al., 2019; Burtsev and Sapunov, 2020; Bulatov et al., 2022). For example, Bulatov et al. (2022) propose to apply custom tokens recurrently, leading to improvement on long sequence modeling and algorithmic tasks. Compressing long prompts into fixed length sequence can alleviate the heavy burden of a large key-value cache during inference (Li et al., 2023; Jung and Kim, 2023; Mu et al., 2023). Custom tokens are also used to optimize pretrained models

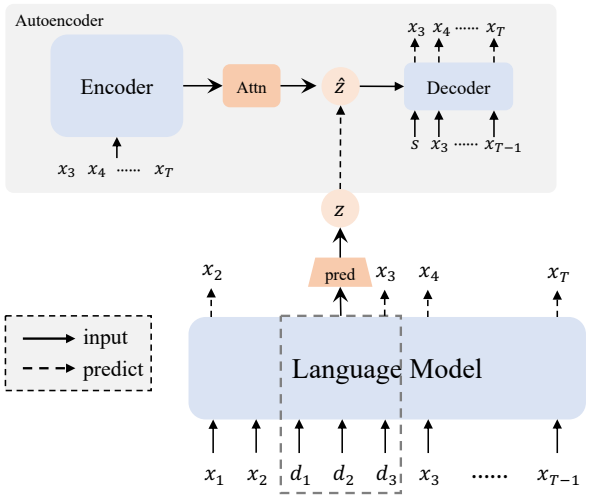


Figure 2: Illustration of our Semformer. We introduce trainable tokens in language modeling. The representations of the tokens encoded by the language model are regressed to the latent representations of the response with L_2 loss. We can share the parameters between the language model and the encoder, and utilize a small decoder to enhance training efficiency.

to accomplish specific downstream tasks, i.e., parameter efficient fine-tuning (Lester et al., 2021; Li and Liang, 2021). For vision Transformer, (Darcet et al., 2023) find that appending trainable tokens to image patches leads to smoother representation learning.

Incorporating trainable tokens has been demonstrated as an effective way to enhance the Transformer's reasoning and planning capabilities. Herel and Mikolov (2024) find that such a method leads to small perplexity gains on reasoning tasks, and Goyal et al. (2024) investigate its effectiveness on the setting of pretraining on C4 with the evaluation on math and question answering. Wang et al. (2024) propose the addition of new tokens preceding each CoT step. Zelikman et al. (2024) generate rationales post every token to elucidate future thinking using REINFORCE learning. Our work is in line with the above studies in introducing additional tokens. The differences lie in that our purpose is to alleviate the shortcut learning induced by teacher forcing, and the tokens are used to generate the semantic plan representations of the subsequent tokens. More importantly, we use a simple and general representation prediction method to guide the function learning of the custom tokens.

3 Method

3.1 Next-token Prediction

Given an observed text sequence of length T, $x = \{x_1, ..., x_T\}$, neural language models (Bengio, 2008) (NLM) are trained to predict every token conditioned on the previous tokens defined by the chain rule of probability, i.e., teacher forcing:

$$\log p_\theta(x) = \sum_{t=1}^{T} \log p_\theta(x_t | x_{<t}), \quad (1)$$

where θ is the model parameter. During inference, the model autoregressively generates the response token-by-token by sampling or searching strategies, given the prefix and all previously generated tokens. We use decoder-only Transformer as the language model. However, our method can also be applied to other architectures such as Mamba (Gu and Dao, 2024).

3.2 Semformer

In addition to next-token prediction, we introduce the prediction in the representation space. The overall framework of our Semformer is illustrated in Figure 2. Specifically, we use an autoencoder to learn latent representations of the target sequence, which guides the representation learning of the language model.

During training, we segment each input sequence x into the prefix $x_{1:n}$ and target $x_{n+1:T}$ where n is the segmentation position between the prefix and the response. For general LM pretraining, the position is selected randomly for each sequence block. Then, we append k trainable planning tokens $d = \{d_1, d_2, ..., d_k\}$ to the prefix. The input of the language model can be rewritten to $x' = \{x_{1:n}; d; x_{n+1:T}\}$. We feed x' into the language model and the planning tokens are not used for the loss calculation for predicting the next token. Formally, the training loss is defined by:

$$\mathcal{L}_{\text{LM}} = \sum_{\substack{t=1 \\ x'_t \notin d}}^{T+k} \log p_\theta(x'_t | x'_{<t}). \quad (2)$$

Latent Semantic Planning We provide the plan tokens with generic supervision information, enabling them to serve as the function to compute a future plan before response generation. The supervision is to predict the latent semantic planning representations of the response, and we introduce an autoencoder with a bottleneck layer to this end.

The encoder of the autoencoder takes the response $x'_{n+1:T}$ as the input and encode them into contextualized representations H_r, which are then compressed into a sequence of latent vectors $Z = \{z_1, z_2, ..., z_k\}$ using a cross-attention layer:

$$H_r = \text{Encoder}(x'_{n+1:T}), \quad (3)$$

$$Z = \text{CrossAttend}(Q, H_r, H_r), \quad (4)$$

where Q is the trainable query input of the cross-attention layer. We use a linear transformation to project Z into a low-dimension representation space. The number of latent vectors is the same as the number of the planning tokens. Using cross-attention provides us with more flexible options for the encoder, such as sharing parameters with the language model or using an off-the-shelf pretrained encoder.

We treat Z as additional memory for the decoder. Before being fed into the decoder, each latent vector is projected into the same dimension of hidden states of the decoder with a distinct linear transformation. Then, the latent vectors are attended by other tokens via self-attention. Such an infusion mechanism of latent vectors is convenient to apply pretrained language models without any modification, and has been shown superior to regarding Z as extra input token embeddings (Li et al., 2020). The objective is the standard reconstruction loss:

$$\mathcal{L}_{\text{AE}} = \log p_{\theta_{\text{AE}}}(x_{n+1:T}|Z), \quad (5)$$

where θ_{AE} is the parameter set of the autoencoder.

To alleviate the training burden, we can adopt the following strategies: sharing the parameters between the encoder and the language model, using an off-the-shelf encoder, stopping the gradient flow into the encoder in the autoencoding branch, and using a compact decoder.

Latent Representation Prediction Given the contextualized representations H of the input x' encoded by the language model, we use a predictor head to output the predicted latent representations. The loss is defined as the L_2 distance between the predicted representations and the target latent representations:

$$\mathcal{L}_{\text{RP}} = \sum_{i=1}^{k} \|z_i - f_{\theta_{\text{RP}}}(H_{n+i})\|_2^2, \quad (6)$$

where $f_{\theta_{\text{RP}}}$ is the representation predictor with parameter θ_{RP}, and we use a linear transformation shared across different positions.

Overall Training Objective The whole framework is jointly optimized as follows:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \mathcal{L}_{\text{AE}} + \alpha\mathcal{L}_{\text{RP}}, \quad (7)$$

where α is the coefficient of the latents prediction loss. By compelling the model to predict the abstract representations of the future response in advance, we can mitigate the Clever Hans cheat issue that arises from exposure to the ground-truth prefix.

Inference During inference, we simply append the planning tokens to the prefix, and the inference remains standard autoregressive decoding.

4 Experiments on Graph Path-finding

The graph path-finding task, as introduced by Bachmann and Nagarajan (2024), involves a unique structure known as a path-star graph $G(d, l, N)$. Each graph features a central node from which d distinct paths emerge, each comprising l nodes, including the central node. The parameter N represents the range of node values, randomly selected from the set $\{0, 1, ..., N-1\}$, and may exceed the total number of nodes in the graph. The input of language models includes all of the edges of the star-graph, the start node, and the end node. The objective is to accurately predict the sole correct path between the designated start and end nodes.

In particular, both the training and test graphs are derived from the same distribution, maintaining consistent topology characterized by fixed values of d, l, and N. This setup ensures that the observed failures are attributable to in-distribution errors rather than lack of compositional or length generalization capabilities. Given that each graph is uniquely labeled and features a randomized adjacency list, the model is required to deduce a general algorithmic solution. Following Bachmann and Nagarajan (2024), the dataset comprises 200,000 training samples and 20,000 test samples. The number of node values N is set as the product of l and d, facilitating a diverse range of graph instantiations.

4.1 Settings

Baselines We use the pretrained GPT2-Large and GPT2-Small (Radford et al., 2019) as the base models of our experiments[2]. We then compare our Semformer with the following baselines: (1) **Standard**, which uses standard teacher forcing training;

[2]We use the open-source resource at https://github.com/gregorbachmann/Next-Token-Failures.git

Model	G(2,20)	G(5,20)	G(5,30)	G(10,20)	G(15,15)	G(20,5)	G(30,5)	G(20,10)
				GPT2-Large				
Standard	49.2	20.1	19.8	10.1	6.8	4.8	3.0	4.9
Teacher-less	1.7	97.8	0.0	0.0	0.0	99.9	99,8	1.8
Multi-token	51.0	19.6	20.0	10.1	6.8	99.9	3.3	4.9
BoW	**100.0**	**99.9**	87.9	85.3	99.0	99.9	99.9	**99.9**
Pause	49.9	20.0	19.7	9.7	6.9	5.0	3.2	4.8
Semformer	99.9	**99.9**	**99.2**	**99.6**	**99.5**	**100.0**	**100.0**	**99.9**
				GPT2-Small				
Standard	49.6	19.7	19.9	9.8	6.7	4.9	3.2	4.8
Teacher-less	0.0	0.0	0.0	0.0	0.0	5.0	99.5	0.0
Multi-token	50.2	19.8	20.3	10.1	5.0	4.9	3.3	4.9
BoW	**99.9**	95.1	82.7	10.3	82.3	99.9	**99.9**	4.9
Pause	50.0	19.9	19.9	10.0	6.6	5.0	3.3	5.0
Semformer	**99.9**	**99.5**	**99.0**	**98.0**	**99.1**	**100.0**	99.6	**99.9**

Table 1: Accuracies on the graph path-finding test sets. The setting G(d,l) is characterized by the degree of the node at the center d and the length of each path l, respectively. The number of node values N is the product of l and d, omitted for simplicity. The results for Standard and Teacher-less are obtained by running the code released by Bachmann and Nagarajan (2024), and the other baselines are re-implemented.

(2)**Teacher-less**, which predicts multiple future tokens at once (i.e., non-autoregressive generation) (Bachmann and Nagarajan, 2024); (3) **Multi-token**, which predicts the following multiple tokens using different output heads (Gloeckle et al., 2024); (4) **BoW**, which predicts bag-of-words of the target sequence; (5) **Pause**, (Goyal et al., 2024) which appends planning tokens and learns them only using the language modeling loss.

Hyper-parameters We train all models using a batch size of 32 for a maximum of 100 epochs. The AdamW optimizer is employed with a learning rate set at 1e-5. For Semformer, the number of planning tokens is set to 4 and the coefficient α is set to 1.0 by default. We use the language model as the encoder and the decoder is set to 6 layers to enhance training efficiency. In more challenging configurations, such as G(10,20), while an α of 1.0 remains effective, increasing it to 10.0 significantly accelerates convergence. For Multi-token, we employ a three-token strategy. For Pause, we insert a number of planning tokens equivalent to those used in Semformer. For the BoW approach, we predict the bag-of-words from the average pooled representations of the planning tokens. The regularization coefficient for BoW is set to 0.1 through a grid search.

4.2 Main Results

The evaluation results are presented in Table 1. Overall, Semformer achieves near-perfect performance across all the graph configurations. The standard Transformer encounters significant challenges in learning the planning task accurately, due to the Clever Hans cheat learned by teacher forcing. In particular, the accuracy for predicting the first node following the start node is approximately $1/d$. Once the first node after the start node is provided, the model demonstrates a high level of accuracy in generating the entire corresponding path (Bachmann and Nagarajan, 2024).

The non-autoregressive Teacher-less models avoid the pitfalls of the cheat to fit the training data. They demonstrate impressive performance on configurations such as G(5,20), G(20,5), and G(30,5) when using the GPT2-Large. However, these models encounter difficulties with the longer responses, which can lead to significant challenges in fitting the training data and results in complete failure (i.e., accuracy 0.0) during test phases. The Multi-token approach does not offer particular advantages and only works on G(20,5) with the shortest target path. The difference between Multi-token and Semformer is that Semformer is trained to predict the complete semantic planning of the target while Multi-token is only trained to predict local

Graph	Accuracy
G(21,10)	60.2
G(23,10)	22.1
G(25,10)	2.8
G(40,10)	99.8
G(10,40)	10.0

Table 2: Performance of Semformer under more challenging settings.

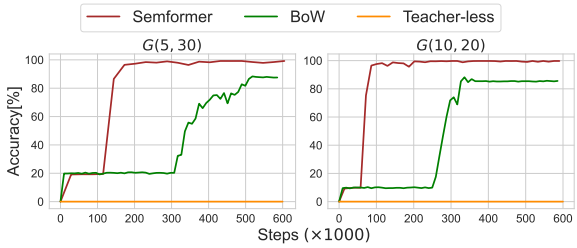


Figure 3: Convergence curves of Teacher-less, BoW, and our Semformer on tasks G(5,30) and G(10,20).

Model	10,20	20,10
NonAE	9.6	4.7
NonAE(ema)	9.6	5.0
AE	99.6	99.9

Table 3: Encoder design. The use of an autoencoder works better than using the language model itself as the encoder.

future tokens. In particular, Pause does not learn to solve this problem. This indicates that simply increasing computing capacity may not be enough to learn lookahead skills effectively, echoing the theoretical research on the competencies of filler tokens (Malach, 2024; Wies et al., 2023). The BoW method can be regarded as a simplified variant of Semformer. It disregards the sequence dependency of the target and only considers surface token information. When integrated with GPT2-Large, BoW achieves commendable results in some settings due to the enforcement of predicting the overall nodes in the target path. Nevertheless, it underperforms in scenarios involving longer target sequences, such as G(5,30) and G(10,20).

We also explore the impact of model size by employing GPT2-Small, which is approximately one-sixth the size of GPT2-Large. Remarkably, our Semformer still maintains nearly 100% accuracy scores without modification to the hyperparameters, while the performance of other baseline models declines. For instance, in configurations such as G(5,30) and G(10,20), the performance of BoW deteriorates to the level of random guessing, exhibiting the underlying limitation in the simple token prediction.

When Semformer Falls Short. We further evaluate Semformer under more challenging conditions to identify scenarios where its performance may falter (Table 2). In fact, (Bachmann and Nagarajan, 2024) specifically excludes out-of-distribution testing scenarios to better control variables. We test on G(21,10), G(23,10), and G(25,10) using the Semformer model trained on G(20,10) to simulate out-of-distribution conditions. The results indicate that the model has some extrapolation ability on G(21,10), but fails in more different situations.

Additionally, we conduct experiments on G(40,10) and G(10,40) to assess performance on larger graphs. Since the sequence length in these settings exceeds the maximum length of GPT2 (i.e., 1024), we switch the backbone to pythia-410m[3]. In the case of the same number of nodes, the graph with a longer path is more difficult. Our model can achieve 98% accuracy on G(40,10), but it failed to learn successfully on G(10,40).

4.3 Analysis

4.3.1 Convergence of Different Models

We choose the graph setting G(5,30) and G(20,10), then display the accuracies with training steps in Figure 3. Teacher-less fails on both tasks, yielding an accuracy of 0. Semformer achieves peak accuracy in less than 50,000 steps. In contrast, BoW requires over 4 times more training steps than Semformer to converge, and fails to attain perfect accuracy on both tasks. These results demonstrate that our framework provides a highly efficient supervisory signal to learn the lookahead skill.

4.3.2 Ablations of Autoencoder

Encoder Design An alternative method is using the language model itself as the encoder to induce the latent planning representations instead of specially training an autoencoder. Concretely, the language model takes the concatenation of the target sequence and the planning tokens as input, and the latents are obtained by stacking a linear transformation on the contextualized representations. We also attempt to using the exponential moving average trick to generate the encoder, which has shown effectiveness in contrastive learning (He et al., 2020).

[3]https://huggingface.co/EleutherAI/pythia-410m

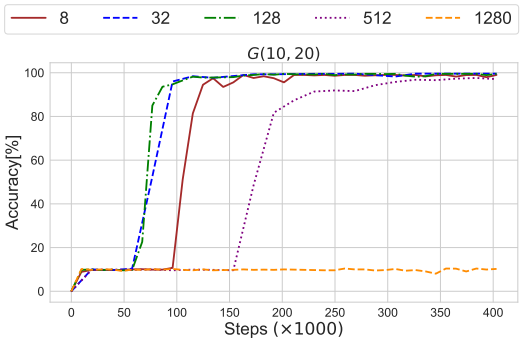


Figure 4: Convergence curves of models with different latent dimensions.

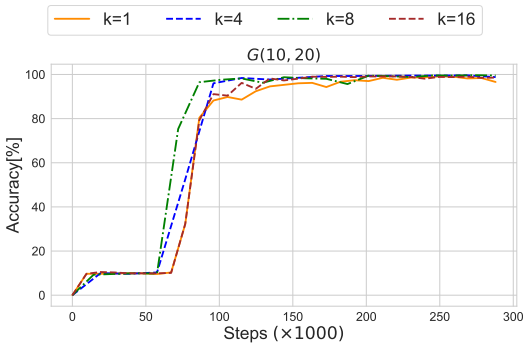


Figure 5: Convergence curves of models with different numbers of planning tokens.

The results in Table 3 demonstrate the advantage of using a separately trained autoencoder, which can learn more meaningful and structured abstract representations than the simple encoding of the input information.

Decoder Layers The number of decoder layers in the autoencoder used in the main results is 6, and we further investigate its influence on performance. We choose a challenging setting, G(10,20), and use GPT2-Large as the base model. For configurations with 1, 3, 6, and 12 layers, the test accuracy scores all exceed 99% and the model with 6 decoder layers converges slightly faster than the others. This result is reasonable since a one-layer Transformer decoder can achieve satisfactory performance on language reconstruction (Montero et al., 2021), and reconstructing a path sequence is simpler than reconstructing natural language.

Latent Dimension Figure 4 reveals the impact of the latent dimension. Dimension reduction helps both the final accuracy and convergence speed, and using relatively lower dimensions such as 32 is more effective than using higher ones. Although the model with a latent dimension of 512 successfully performed the task, it requires a significantly longer time to converge. When using the same dimension as the model, we remove the linear transformation and this leads to poor performance, indicating the benefits of using compressed representations.

Number of Planning Tokens We choose the task setting G(10,20) to examine the effect of the number of planning tokens. As shown in Figure 5, the number of planning tokens does not have a particularly significant impact on the final accuracy. This may be because the suffix length is short (<50) and the model capacity is sufficient. The number of tokens influences the speed of the convergence, and the model converges fastest with $k = 8$.

4.3.3 Attention Visualization

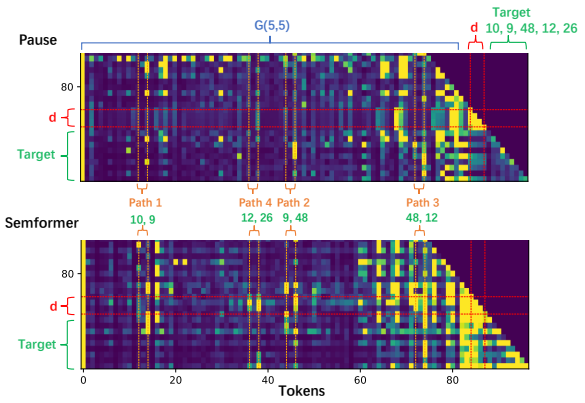
We conduct a visualization of the attention weights to see what information is captured by these tokens (Figure 6). We select the graph setting G(5,5) and the pretrained model is GPT2-Large. For each layer, we average the attention weights from all the attention heads, and observe a shift in the attention distribution in the 28th layer of Semformer. The planning tokens are successful in capturing the paths leading to the answer. Moreover, the answer tokens not only concentrate on their context but also allocate sufficient attention to the planning tokens. This contrasts with the Pause model, where the planning tokens fail to capture the correct paths, and the attention from the answer tokens to the planning tokens is insignificant.

5 Experiments of Pretraining

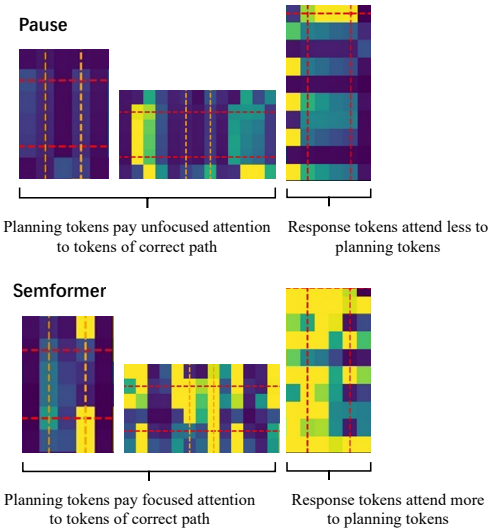
In this section, we extend the proposed model to pre-training, and validate its effectiveness in terms of perplexity, in-context learning, and supervised fine-tuning.

5.1 Setting

We train a Transformer language model with the same configuration as GPT2, totaling 125M parameters. The corpus is the public version of OpenWebText. We use a sequence length of 1,024, and the batch size is 512. For each sequence, we randomly split it into a suffix and a prefix, ensuring that the prefix contains at least 128 tokens. Following (Hewitt et al., 2023), we set the gradient steps to 100,000 and it approximately runs 6 epochs. The optimizer is AdamW with a learning rate of 6e-4 and a warmup of 5,000 steps. The number



(a) Attention Visualization of Complete Sequence



(b) Attention Visualization of Special Tokens

Figure 6: Visualization of Pause and Semformer's attention weights.

of planning tokens and the latent dimension are set to 16 and 64, respectively. The two numbers are set empirically and we do not tune them. For the coefficient of the regularization α, we select it from $\{0.1, 0.5, 1.0\}$ according to the perplexity on Wikitext (Merity et al., 2017), and find that the model achieves lowest perplexity with $\alpha = 0.5$. In addition to our proposed model, we also train a vanilla Transformer model and a model without latent representation prediction (i.e., Pause) using the identical hyper-parameters.

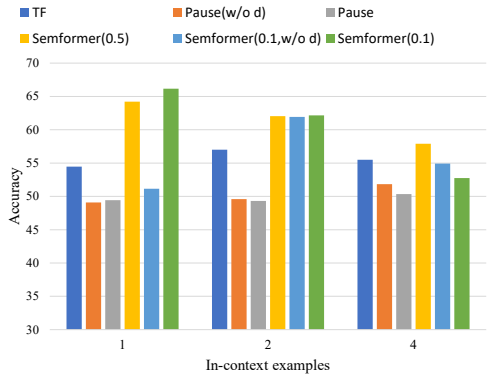
5.2 Results

Perplexity The perplexity scores are shown in Table 4. On the Wikitext test set, we simply insert planning tokens at the middle position of each sequence. Different from Wikitext, LAMBADA is dedicated to investigating the long-range dependencies in text (Paperno et al., 2016), and the perplexity

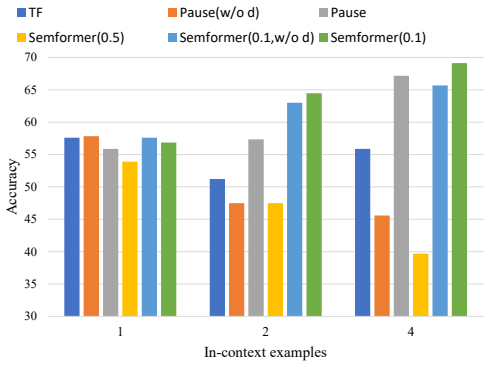
Model	Wikitext	LAMBADA
TF	37.5	42.5/32.1
TF-Pause	35.9	43.3/32.7
Semformer	**35.6**	**38.8/33.5**

Table 4: Language modeling performance measured by perplexity. For LAMBADA, we additionally report the accuracy followed by the perplexity score. The optimal results are highlighted in bold.

is only calculated on the tokens to predict. Similarly, Semformer achieves the lowest perplexity due to that the representation prediction encourages the model to predict the whole future semantic representations in advance. The performance gap between Semformer and TF-Pause has become more significant compared to that on Wikitext. Moreover, even without the tokens, our model achieves lower perplexity than the other two baselines (35.6 on Wikitext and 39.5 on LAMBADA), indicating that our framework also yields better representation learning.



(a) SST-2



(b) MRPC

Figure 7: In-context learning performance.

In-context Learning We select a single-sentence classification task, Stanford Sentiment Treebank Binary (SST-2) (Socher et al., 2013), and a paraphrase

Model	R-1	R-2	R-L
XSum			
TF	35.86	13.94	28.61
TF-Pause	35.85	13.85	28.60
Semformer	**36.47**	**14.37**	**29.07**
SAMSum			
TF	45.60	21.09	41.62
TF-Pause	46.74	21.96	42.54
Semformer	**46.93**	**22.29**	**42.72**
DialogSum			
TF	42.65	16.54	37.50
TF-Pause	42.17	16.47	37.09
Semformer	**43.18**	**16.59**	**38.02**

Table 5: Evaluation on abstractive text summarization.

identification task, Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), to investigate the performance of in-context learning (ICL)[4]. The results are presented in Figure 7, and we observed the following phenomena. Semformer performs best on both tasks, achieving an accuracy of 66.1 on SST-2 and 69.1 on MRPC. In contrast, the best TF model achieves 57.0 on SST-2 and 57.6 on MRPC. Specifically, when TF-Pause does not utilize planning tokens during inference, there was a significant decline in performance. However, the performance decrease of Semformer is not as pronounced when removing the planning tokens, demonstrating the improvement in representation learning due to the regularization. Furthermore, a larger coefficient of 0.5 is found inferior to a smaller one, i.e., 0.1. This may be because such classification tasks do not heavily rely on lookahead ability, and the model requires a balance between the use of context and the prediction of future information. Scaling up the model size to increase its capability could potentially mitigate this phenomenon, and we leave this as a future investigation.

Supervised Fine-tuning on Summarization In this section, we investigate the performance of supervised fine-tuning of the whole framework on abstractive summarization. We use XSum (Narayan et al., 2018), SAMSum (Gliwa et al., 2019), and DialogSum (Chen et al., 2021) for evaluation, and report ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004). We finetune each model on the training data of each task separately and select the checkpoints with highest ROUGE-L score on the individual validation set. The batch size is 128 and the learning rate is set to 5e-5. The value of α is set to 0.5, consistent with its setting during pretraining. We use beam search with a beam size 2 for all of the models. The results in Table 5 show that the Semformer outperforms the standard Transformer and TF-Pause, indicating that the mechanism of semantic planning modeling is beneficial for abstractive summarization.

6 Conclusion

In this paper, we presented Semformer which explicitly models semantic planning in addition to next-token prediction. Semformer introduces a sequence of trainable planning tokens to induce the planning within the internal computation, and the planning tokens in the language model are supervised by predicting the latent planning representations generated by an autoencoder. The results on the graph path-finding problem show that Semformer can achieve nearly perfect accuracy in such a minimal lookahead task, alleviating the shortcut learning caused by teacher forcing. Extending Semformer to a general pertaining on OpenWebtext demonstrates the advantages of the paradigm.

Future research will focus on validating our model with larger sizes and training corpus and exploring its application on reasoning-related tasks such as math and coding. Additionally, investigating hierarchical or block-wise prediction of semantic vectors presents a promising avenue for further exploration.

Acknowledgements

This work is funded by the National Natural Science Foundation of China Key Program under Grant Number 62336006. We would like to thank the anonymous reviewers for their helpful comments.

[4]https://github.com/EleutherAI/lm-evaluation-harness

7 Limitations

Due to limited computation resources, we only pretrain a language model with 125M. Whether our method can still outperform teacher forcing when combined with larger corpora and when the model size scales up to 1B or even larger needs to be verified in the future. In addition, we do not provide theoretical analysis to prove that the method can mitigate the bias in teacher forcing.

References

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 2296–2318. PMLR.

Yoshua Bengio. 2008. Neural net language models. *Scholarpedia*, 3(1):3881.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*.

Mikhail S. Burtsev and Grigory V. Sapunov. 2020. Memory transformer. *CoRR*, abs/2006.11527.

Yulong Chen, Yang Liu, and Yue Zhang. 2021. Dialogsum challenge: Summarizing real-life scenario dialogues. In *Proceedings of the 14th International Conference on Natural Language Generation, INLG 2021, Aberdeen, Scotland, UK, 20-24 September, 2021*, pages 308–313. Association for Computational Linguistics.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. 2023. Vision transformers need registers. *CoRR*, abs/2309.16588.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Li Du, Hongyuan Mei, and Jason Eisner. 2023. Autoregressive modeling with lookahead attention. *CoRR*, abs/2305.12272.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaïd Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15706–15734. PMLR.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

David Herel and Tomas Mikolov. 2024. Thinking tokens for language modeling.

John Hewitt, John Thickstun, Christopher D. Manning, and Percy Liang. 2023. Backpack language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9103–9125. Association for Computational Linguistics.

Hoyoun Jung and Kyung-Joong Kim. 2023. Discrete prompt compression with reinforcement learning. *CoRR*, abs/2308.08758.

Matti Kääriäinen. 2006. Lower bounds for reductions. In *Atomic Learning Workshop*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR 2014*.

Y LeCun. 2024. Do large language models need sensory ground ing for meaning and understanding? University Lecture.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4678–4699. Association for Computational Linguistics.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, Online. Association for Computational Linguistics.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Eran Malach. 2024. Auto-regressive next-token predictors are universal learners. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 34417–34431. PMLR.

R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L. Griffiths. 2023. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *CoRR*, abs/2309.13638.

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.

Giovanni Monea, Armand Joulin, and Edouard Grave. 2023. Pass: Parallel speculative sampling. *CoRR*, abs/2311.13581.

Ivan Montero, Nikolaos Pappas, and Noah A. Smith. 2021. Sentence bottleneck autoencoders from transformer language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1822–1831. Association for Computational Linguistics.

Jesse Mu, Xiang Li, and Noah D. Goodman. 2023. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. *CoRR*, abs/2404.15758.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 2401–2410. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Stephane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy. PMLR.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Hervé Jégou, and Armand Joulin. 2019. Augmenting self-attention with persistent memory. *CoRR*, abs/1907.01470.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010. Curran Associates Inc.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. 2024. Guiding language model math reasoning with planning tokens.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022.

Noam Wies, Yoav Levine, and Amnon Shashua. 2023. Sub-task decomposition enables learning in sequence to sequence tasks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Joshua M. Susskind, and Navdeep Jaitly. 2023. PLANNER: generating diversified paragraph via latent language diffusion model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.