Anticipating Future Tokens to Help Transformer LLMs Plan

Victor Wang
The University of Texas at Austin
victorwang37@utexas.edu

Abstract

When transformer LLMs generate text autoregressively, the only information carried over from the computation in the current step to the input of the next step is the newly generated token. In this paper, we explore a more informative *planning signal* containing a set of tokens likely to appear in the near future. We train an *anticipator* head on top of the backbone transformer and use it to produce the planning signal. The planning signal increases the computational depth and width of the model, in a medium separate from its textual output. We evaluate on a diverse set of generation tasks using GPT-2 as the baseline and backbone transformer.

1 Introduction

During the production or consumption of language, humans form a tentative plan or belief about the direction the narrative will take. On the other hand, autoregressive text generators traditionally cannot plan ahead (Lu et al., 2022). While decoding strategies like beam search give more room to consider diversity and future continuations compared to greedy search, they are still limited by the expressivity and foresight of the underlying LM. In particular, with a vanilla autoregressive LM, the only communication carried within the model from one decoding step to the next is the newly decoded token. In this paper, we explore a more informative *planning signal* which contains the prediction that certain tokens appear in the near future. To provide such a signal to the LM, we train an *anticipator* head to compute a score for each vocabulary token, reflecting a confidence about the proximity of the token's next appearance. The planning signal can be viewed as a form of local memory that allows the LM to condition on its past reasoning about the development of the text.

As the attachment of the anticipator head and the insertion of the planning signal into the LM are not agnostic to the LM architecture, we focus in this paper on the transformer architecture (Vaswani et al., 2023) due to its prevalence and success in building LLMs (Radford et al., 2018; Touvron et al., 2023). Our method trains the anticipator head using a text corpus as data. From the output of the anticipator, the highest scoring tokens relative to their global frequency are embedded, summed with a future position embedding, and introduced at the next decoding step into the LM's first attention layer for the last context token to attend to. To enable token-parallel training, we simulate the planning signal using a tiny model that continuously learns to mimic the error distribution of the anticipator.

We evaluate our method on perplexity, constrained generation and story generation, and long form question answering.

2 Related Work

Delayed answering Scratchpad (Nye et al., 2021) and chain-of-thought (CoT) prompting (Wei et al., 2022) elicit the LM to decode and condition on intermediate reasoning steps leading to the answer to a multi-step problem. These approaches have the advantage that they are easy to apply to a promptable LM. However, they are only applicable to problem solving without concern for token efficiency, and not to tasks for which there are no concrete intermediate steps concisely expressible in natural language, nor to tasks that penalize showing intermediate steps, such as summarization or story generation. Conversely, our method is less suitable for solving the types of problems that Scratchpad and CoT are intended to solve. Our approach increases the width of computation but in a medium separate from the model's textual output, which is related to the approach by Goyal et al. (2023)

of buffing the context size by appending learnable pause tokens.

Speculative decoding The proposed anticipator head is reminiscent of a speculative decoding approach that trains k heads on top of the backbone LM to predict the next k tokens in parallel (Stern et al., 2018). The purpose of speculative decoding is computational efficiency, while our purpose is helping the model to plan. Also, we hypothesize that the quasi-bag-of-words task our anticipator learns is easier because the exact word order is not important, which is especially simplifying for languages with relatively free word order.

Non-autoregressive generation As our method involves conditioning on an anticipated future context, it is related to a variety of methods that break the autoregressive barrier, several of which we list below. Lu et al. (2022) use a heuristic search motivated by A* to guide generations toward those satisfying task-specific constraints. West et al. (2020) use a forward and a backward LM to contextualize the input in each direction and then reflect on those contexts in the reverse directions, useful for tasks like paraphrasing. Qin et al. (2020) iterate forward and backward LM passes using gradient descent to refine a generation to be consistent with a past context and a future constraint. Welleck et al. (2019a) train a model to learn a generation order following a tree structure. Ghazvininejad et al. (2019) iteratively mask tokens and predict them with bidirectional attention. Han et al. (2023) decode a chunk of tokens at a time with diffusion, using the continuous representation of a simplex whose vertices represent discrete tokens. It is worthwhile to note that several of the above methods use an LLM in some innovative way as a black-box, so they can be used orthogonally to our method, which modifies the transformer architecture in an independent way.

3 Methods

This section describes our modifications to the transformer architecture, overviewed in Figure 1, as well as training and inference procedures.

3.1 Anticipator head

The anticipator head is attached on top of the backbone transformer. Like the LM head, the anticipator head's architecture is a feedforward neural network (FNN) with input dimension equal to the transformer's model dimension and output dimension equal to the vocabulary size, ending with a softmax. In fact, if the LM is pre-trained, we initialize the anticipator head to the LM head. We train the anticipator to predict a score for each vocabulary token that captures its expected proximity. The output of the anticipator is used to select K tokens to form a planning signal for the transformer, for a hyperparameter K.

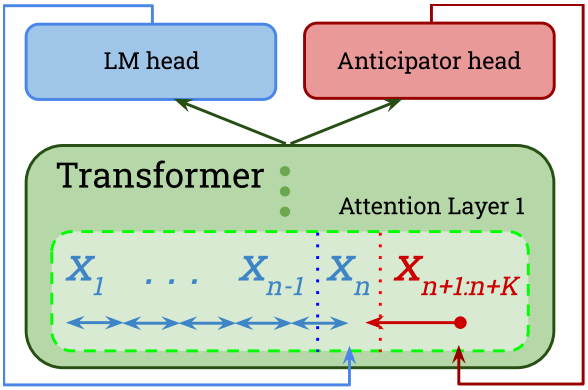

Figure 1: Model architecture at inference time. The anticipator head generates a score distribution over the vocabulary, which is contrasted with the global token frequencies to prepare a set of K relevant future tokens. These tokens are embedded, summed with a future position embedding, and introduced to the first attention layer for the last context token to attend to.

Objective We use a text corpus as training data, constituting the ground truth of future tokens. For a training sample, the simplest target score distribution would be a K-hot vector, in which the score mass is uniform across the next K tokens and zero elsewhere. However, this creates a sharp contrast from the Kth to the $(K+1)$th token. We address this by decaying $s(d)$, the score for a token d tokens ahead of the last context token, such that

$$s(d) \propto \log(K + 2 - d) \tag{1}$$

for $1 \leq d \leq K$, and $s(d) = 0$ for $d > K$. If a token appears more than once, its score is the sum of $s(d)$ over its distances d. We normalize the scores (i.e. divide by sum) into pseudo-probabilities, in the sense that they form a probability distribution but semantically do not represent the probabilities of events.

Training The anticipator head ends with a softmax layer to predict a score distribution:

$$\hat{s}(i) \propto \exp(z_i/\tau) \tag{2}$$

for token i, where z_i is an input to the softmax layer and τ is the temperature; the scores are normalized to sum to 1. We make τ learnable so the model may learn to distribute score mass across the candidates.

We train the anticipator with a combination of two loss functions. The most straightforward one is the cross-entropy loss $\mathcal{L}_{\text{CE}}$. However, since the ground truth of future tokens is nearly identical from one decoding step to the next, there is a significant risk that with $\mathcal{L}_{\text{CE}}$ alone, the anticipator will not learn to drastically reduce the score for newly decoded tokens, promoting degenerate repetition. One workaround would be to train only on every Kth decoding step to eliminate overlap of future tokens, but at the cost of thinning the training signal. To teach the anticipator to discriminate when future tokens have moved into the context while still learning at every decoding step, we use the unlikelihood loss (Welleck et al., 2019b) to punish high scores for recent tokens that do not reappear:

$$\mathcal{L}_{\text{UL}}(\hat{s}, \mathcal{C}_n) = -\sum_{c \in \mathcal{C}_n} \log(1 - \hat{s}(c)) \quad (3)$$

$$\mathcal{C}_n = \{x_{n-K+1}, \ldots, x_n\} \setminus \{x_{n+1}, \ldots, x_{n+K}\},$$

where n is the position of the last context token and x_i denotes the token at position i. The combined loss is $\mathcal{L}_{\text{CE}} + \lambda_{\text{UL}}\mathcal{L}_{\text{UL}}$ for a hyperparameter $\lambda_{\text{UL}} = 1$.

We point out that our use of unlikelihood training is orthogonal to the original usage because they apply it to the backbone LM. Also, the original method for training on a text corpus sets $\mathcal{C}_n$ to the previous tokens that do not appear as the *single* next token, whereas our usage more carefully discerns whether recent tokens are acceptable to repeat within the context by checking whether they appear in the next K tokens.

3.2 Planning signal

The anticipator's score distribution is used to construct a planning signal to the LM.

Contrastive scoring There is a clear problem with feeding the K tokens with the highest anticipator score into the LM: the planning signal will be occupied by stop words that are trivial to predict and offer no substance. To address this, we apply the concept of contrastive decoding (Li et al., 2023) to scoring. Rather than the highest scoring tokens, we are interested in the tokens with the greatest relevance to the current context relative to their usual usage. Let $p_{\text{tf}}(i)$ denote the global term frequency of token i. We compute the contrastive score as

$$\hat{s}_{\text{cv}}(i) = \log \hat{s}(i) - \log p_{\text{tf}}(i). \quad (4)$$

In practice, we set $p_{\text{tf}} \leftarrow \max(p_{\text{tf}}, \epsilon_{\text{tf}})$ for a hyperparameter $\epsilon_{\text{tf}} = 10^{-5}$ to avoid the negative consequences of noisy frequency data for highly rare words, and to discourage selecting them. The K tokens with the most positive contrastive score are selected[1], or however many there are if there are fewer.

Future position embedding Before feeding the embeddings of the K selected tokens to the transformer, we add a future position embedding to inform that these tokens are from the future. As we do not attempt to predict precise positions but rather a quasi-bag-of-words, the same future position embedding is shared by all K tokens. The future position embedding is obtained by applying a learnable linear transformation to the position embedding of the last context token. One particular function that a linear transformation could implement is $\text{PE}_n \rightarrow \text{PE}_{n+k}$ for any constant k, where PE_i denotes the sinusoidal positional embedding proposed by Vaswani et al. (2023) at position i.

First attention layer The embedded K tokens are passed to the transformer. For computational feasibility during training, the only attention connections added are of the last context token attending to the K anticipated tokens at the first attention layer. A more formal description follows, simplified to a single attention head. Using the attention parameters of the first layer (with vector dimension d), we compute $\mathbf{Q}_c \in \mathbb{R}^{1\times d}$ as the query vector of the last context token x_n, and $\mathbf{K}_a, \mathbf{V}_a \in \mathbb{R}^{K\times d}$ as the key and value vectors of the K anticipated tokens. Let $\mathbf{w} \in [0,1]^{1\times K}$ be the contrastive scores of the K tokens normalized to sum to 1. Then, the modification to the transformer can be described as the addition of

$$(\text{softmax}(\mathbf{Q}_c\mathbf{K}_a^\top/d) \odot \mathbf{w})\mathbf{V}_a \quad (5)$$

to x_n's representation at the following residual connection, where $\odot$ denotes broadcasted elementwise product. Further weighting the attention weights by $\mathbf{w}$ allows us to frame the selection of the K highest scoring tokens as an approximation of including the full distribution of anticipated tokens.

[1] There is no reason this K must equal the K used to construct the target score distribution for training the anticipator, but we set them equal for simplicity.

3.3 Anticipation simulator

The subsections above present a well-defined model. However, if we wish to train on a sequence of tokens in parallel as usual with transformer LMs, we are missing a way to simulate the planning signal from the anticipator. To do so, we train a tiny model to simulate the error distribution of the anticipator given the ground truth of future tokens.

For a given target score $s \in [0, 1]$, we model the distribution of anticipator scores $\hat{s} \in [0, 1]$ with a beta distribution $\mathrm{Beta}(\alpha, \beta)$. For this we learn a function $f_{\mathrm{AS}} : [0, 1] \to \mathbb{R}_+^2$ with an *anticipation simulator* whose architecture is an FNN with input dimension 1, a single hidden layer of dimension $d_{\mathrm{AS}} = 32$ with sigmoid activation, and output dimension 2 followed by a softplus function to ensure positivity. We train on previous samples of (target score, anticipator score) with negative likelihood loss. We use a high learning rate so that the simulator forgets old behavior of the anticipator. Then, for token-parallel training, we apply the anticipation simulator to each target score to sample a score, and use these scores to construct a planning signal for the transformer.

4 Experimental Setup

In our experiments, we use GPT-2 (Radford et al., 2019) as the backbone transformer, a random 100 MB subset of C4 (Raffel et al., 2019) as training data, and $K = 50$. We compare our method against the baseline GPT-2. As we intend our method to improve on a general-purpose transformer LLM, we consider a diverse set of generation tasks.

We also compare against three ablations: discard anticipator head (\ANTICIPATOR); freeze backbone LM (\FINE-TUNE); discard anticipator head and freeze backbone LM (BASE+TRAIN). Our method introduces potential benefits through (1) the planning signal and (2) fine-tuning the LM to produce a hidden representation conducive toward the anticipator head predicting a future context. The absence of each is examined through \ANTICIPATOR and \FINE-TUNE, and the absence of both is examined through BASE+TRAIN.

4.1 Accuracy of anticipator and LM

Before looking at real tasks, we first measure the "accuracy" of the anticipator and the LM. For the anticipator, we report the (average) KL divergence $D_{\mathrm{KL}}(P\|Q)$, where P is the ground truth score distribution (as defined in Section 3.1) and Q is the anticipator's score distribution. For the LM, we report perplexity. We evaluate both metrics on a held-out random 1 MB subset of C4.

Since the hidden state from the transformer is passed into both the LM head and the anticipator head, the planning signal may only be helpful if the predicted token becomes the last context token. However, due to teacher forcing, inconsistency would arise if the last context token is overridden by the ground truth, so measuring perplexity naively for our method would be unfair. Thus, we modify our architecture from that used during generation. Consider that we are predicting x_{n+1} from $x_{[1,n]}$. We run the model with context $x_{[1,n)}$ and a single anticipated token equal to x_n. We use the anticipator output at position $n - 1$ to construct a planning signal. We run the model again, with context $x_{[1,n]}$ and the planning signal. The purpose of including the single anticipated token x_n in the first run is to make the model likely to predict $\hat{x}_n = x_n$ without x_n being part of the context (if it was in the context, there would be a mismatch with training), but even if $\hat{x}_n \neq x_n$, we apply teacher forcing despite this causing inconsistency with the planning signal.

4.2 Constrained generation / story generation

RocStories We evaluate story generation with the RocStories dataset (Mostafazadeh et al., 2016). We measure perplexity from GPT-3, and coherence and quality evaluated by humans.

COMMONGEN We evaluate constrained generation with the COMMONGEN dataset (Lin et al., 2020), which asks a model to write a sentence that includes a given set of words. As in Lu et al. (2022), we evaluate the automatic metrics of coverage, ROUGE-L, and perplexity from GPT-3, and the human metrics of coverage, quality, and plausibility.

4.3 Long form question answering

We evaluate long form question answering with the ELI5 dataset (Fan et al., 2019). We use a BART-based retriever[2]. We measure ROUGE-L with the reference.

5 Results

We expect our method to improve on the baseline. We expect the ablations to come between the baseline and our method.

[2] https://huggingface.co/yjernite/bart_eli5

Method	D_{KL}
Ours	
\FINE-TUNE	

Table 1: KL divergence $D_{\mathrm{KL}}(P\|Q)$, where P is the ground truth score distribution and Q is the anticipator's score distribution.

Method	Perplexity
Ours	
\ANTICIPATOR	
\FINE-TUNE	
BASE+TRAIN	

Table 2: Perplexity.

5.1 Accuracy of anticipator and LM

Table 1 shows the results for KL divergence of anticipated tokens. Table 2 shows the results for perplexity.

5.2 Constrained generation / story generation

Table 3 shows the results for RocStories. Table 4 shows the results for COMMONGEN.

5.3 Long form question answering

Table 5 shows the results for ELI5.

6 Discussion

Tokenless anticipation A potential generalization of our method is to allow the planning signal to be an arbitrary vector rather than our engineered feature. While the former would grant more expressivity, the latter grants (1) efficiency – token-parallel training is possible because we can learn to simulate the error distribution of the anticipator given the ground truth of future tokens, (2) interpretability – the predictions of the anticipator may reveal the intentions of the model before they are realized, and (3) manipulability – we can manipulate the planning signal for the purpose of constrained decoding.

	Auto	Human	
Method	PPL	Coherence	Quality
Ours			
\ANTICIPATOR			
\FINE-TUNE			
BASE+TRAIN			

Table 3: RocStories. Perplexity evaluated by GPT-3.

Quality-compute tradeoff in 1 model It is desirable to have a single model that is optionally able to spend more time computing in exchange for higher quality outputs, generalizing between multiple models of different scale. This ability is akin to the way a human can adaptively either make snap decisions or ponder deeply. An iterative application of the method presented here is a possible implementation of such a model, where we can repeatedly pass the model's anticipations back to the model for it to continue refining them.

Hybrid bidirectional/causal attention Although bidirectional attention leverages context more effectively than causal attention, transformer LMs with token-parallel training do not permit bidirectional attention due to illegal lookahead. For this reason, even during inference, when context tokens attending to future context tokens would not be cheating, LMs do not do so because such attention interactions are out of training distribution. Our anticipation objective may be a preliminary approach at bridging the gap between what *can be* leveraged at inference and what *is* leveraged at training. Since each token attends to a set of anticipated future tokens during training, attending to future tokens is in-distribution to some degree, which means the true future context tokens can be leveraged at inference.

7 Practical Limitations

The practical limitations are related to compute. Since training language models is expensive, we may not be able to scale our experiments up to the largest models.

References

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Constant-time machine translation with conditional masked language models. *CoRR*, abs/1904.09324.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2023. Think before you speak: Training language models with pause tokens.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2023. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation

Method	Automatic			Human		
	Coverage	ROUGE-L	Perplexity	Coverage	Quality	Plausibility
Ours						
\ANTICIPATOR						
\FINE-TUNE						
BASE+TRAIN						

Table 4: COMMONGEN. Perplexity evaluated by GPT-3.

Method	ROUGE-L
Ours	
\ANTICIPATOR	
\FINE-TUNE	
BASE+TRAIN	

Table 5: ELI5.

and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11575–11596, Toronto, Canada. Association for Computational Linguistics.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 780–799, Seattle, United States. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. *CoRR*, abs/2010.05906.

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

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. *CoRR*, abs/1811.03115.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey

Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019a. Non-monotonic sequential text generation. *CoRR*, abs/1902.02192.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019b. Neural text generation with unlikelihood training. *CoRR*, abs/1908.04319.

Peter West, Ximing Lu, Ari Holtzman, Chandra Bhagavatula, Jena D. Hwang, and Yejin Choi. 2020. Reflective decoding: Unsupervised paraphrasing and abductive reasoning. *CoRR*, abs/2010.08566.