Exploring the Latent Capacity of LLMs for One-Step Text Generation

Anonymous ACL submission

Abstract

A recent study showed that large language models (LLMs) can reconstruct surprisingly long texts – up to thousands of tokens – via autoregressive generation from just one specially trained input embedding. In this work, we explore whether such reconstruction is possible without autoregression.

We show that frozen LLMs can generate hundreds of accurate tokens in just one forward pass, when provided with only two learned embeddings. This reveals a surprising and underexplored capability of LLMs – multi-token generation without iterative decoding.

We investigate the behaviour of these embeddings and provide insight into the type of information they encode. We also empirically show that although these representations are not unique for a given text, they form connected and local regions in embedding space – a property that suggests the potential of learning a dedicated encoder into that space.

1 Introduction

Large language models (LLMs) are typically trained to generate text in an autoregressive manner – they predict one token at a time based on the previously generated context.

Recent work by Kuratov et al. (2025) demonstrated that LLMs can autoregressively generate an arbitrary text starting from a single, specially trained input embedding corresponding to that text. This raises an intriguing question: is autoregressive generation an essential part of such reconstruction? Can LLMs reconstruct accurate multi-token sequences from some compressed representation in a single forward pass, without any iterative generation, and if so, how?

In this work, we aim to find out whether this is possible and to to understand, what those compressed representations encode and whether it reveals anything about LLMs' parallel generation capabilities.

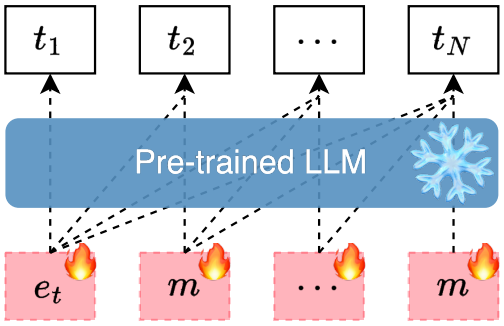

Figure 1: Two "proto-tokens" (trainable embeddings) are fed into frozen, pre-trained LLM and optimized in such a way, that the LLM predicts an arbitrary target sequence in a single forward pass.

Our contribution is as follows:

1. We show that LLMs can reconstruct arbitrary sequences from as few as two learned input embeddings, achieving perfect reconstruction of sequences of up to several hundred tokens.
2. We identify key design aspects for such a setup, that enable this generation, including the critical importance of input token arrangement.
3. We study how the reconstruction capability varies with the model size and the nature of the target sequence (e.g. natural vs synthetic text).
4. We conduct several experiments which shed some light on the nature of the representations, the structure of their embedding space, and the possibility to replace their direct optimization with parameterized encoding.

2 Related Work

The most direct influence for our work is a paper by Kuratov et al. (2025), which showed that frozen LLMs can reconstruct an arbitrary text (a sequence of tokens) $T = [t_1, \ldots, t_N]$ if given a set of special, so-called memory tokens $[m_1, \ldots, m_K]$. The embeddings for these tokens are trained by optimizing a causal language modeling objective (next-token

prediction cross-entropy loss) over a concatenated input sequence $X = [m_1, \ldots, m_K, t_1, \ldots, t_N]$ passed through a frozen LLM. In the case of perfect next-token prediction accuracy (which could be achieved for reasonable text length), this allows the model to autoregressively predict the whole text starting from the memory tokens. The number of memory tokens controls the maximum available text length and can be as few as one.

Although surprisingly long (up to 1568 tokens) texts could be compressed even into a single memory token, the authors note that the embeddings optimized for the same text from different initializations often lie far apart. Moreover, linear interpolations between those embeddings produce very poor reconstruction accuracy, suggesting that the solution space lacks desirable smoothness and locality qualities, which are important for learning a practical encoder that could replace the direct optimization.

Our work also relates to efforts in prompt-tuning and its variants (Lester et al., 2021; Liu et al., 2024; Li and Liang, 2021). Most similarly, Lester et al. (2021) train task-specific soft tokens to condition the frozen model to improve their performance in new tasks. Finally, several speculative (Xia et al., 2023) and parallel (Santilli et al., 2023) decoding utilize a similar mechanism for multiple token prediction using decoder models. More specifically, they add special [PAD] or [MASK] tokens at the end of the current context in order to make a prediction for several tokens into the future at once. Critically, in these works either special training or multiple generative iterations are required.

Unlike the prior work, we show that LLMs can generate accurate multi-token sequences in one forward pass without any additional training or iterative decoding.

3 Method

To adopt the approach from Kuratov et al. (2025) to a non-autoregressive case, we replace all input tokens of the LLM with specially trained "proto-tokens" and predict the target token sequence in one forward pass. In practice, "proto-tokens" are just trainable vectors that are not tied to any real items in the vocabulary. The main difference between regular tokens and these "proto-tokens" is that "proto-tokens" encode multiple tokens at once and only produce human-readable text after passing through the LLM. Our goal is to identify the smallest possible number of such "proto-tokens" needed for accurate reconstruction. Interestingly, we find that it is essential to have at least two – the performance degrades significantly when using only one (see Section 4).

There are many ways to arrange two vectors as an input sequence of arbitrary length. We report results for different variants later in the paper, but here we describe the arrangement that is used in the majority of the experiments.

Exact scheme We introduce two "proto-tokens" e and m with trainable embeddings of dimension d_{model} (model input embedding dimension) and construct the input sequence as follows: $Z = [e, m, m, \ldots, m]$ – one copy of token e is followed by $N-1$ copies of token m, where N is the target text length. We then train the vectors by optimizing cross-entropy loss between the target sequence $T = [t_1, t_2, \ldots, t_N]$ and the frozen LLM's output for the input sequence. The prediction is made using standard causal attention masking, so that the prediction for the token t_i depends on the first i input "proto-tokens" (see Figure 1).

Metrics Our main evaluation metric is the number of correctly reconstructed tokens in a generated sequence defined as:

$$C_{tokens} = \sum_{i=1}^{N} \mathbb{1}(LM(Z_{[1:i]}) = t_i) \qquad (1)$$

Additionally, we measure the amount of information contained in the reconstructed token sequence from the perspective of causal language modeling with a given LLM. Specifically, we compute the cross-entropy between the compressed sequence and LLM's autoregressive probability distribution:

$$H_{LM} = \sum_{i=1}^{N} -log\mathbb{P}_{LM}(t_i|t_{<i}) \qquad (2)$$

This quantity measures how uncertain a model is about the compressed text, that is, how much information it contains.

Solution space connectivity To gain insights into the structure of the solution space of our problem, we analyze whether different proto-token embeddings obtained for the same text but from different random initializations are connected. We adopt a technique from (Garipov et al., 2018) which is used to find paths connecting different minima of

the loss function in computer vision problems. We optimize the parameters of a degree-one Bezier curve, connecting two solutions, to maximize reconstruction accuracy along the curve. The curve is parameterized by a control point π in the following way:

$$\phi_\pi(t) = (1-t)^2 p_1 + 2t(1-t)\pi + t^2 p_2 \quad (3)$$

Here, p_1 and p_2 are the two original solutions that we aim to connect.

The expectation of the cross-entropy loss function under the uniform distribution over $t \in [0, 1]$ (4) is minimized by iteratively sampling $\tilde{t} \in [0, 1]$ and making a gradient step, effectively obtaining unbiased estimate of the gradient of l_π:

$$l_\pi = \int_0^1 \sum_{i=1}^{N} -log\mathbb{P}_{LM}(t_i|\phi_\pi(t))dt \quad (4)$$

This acts as a more tractable alternative to direct optimization under the uniform distribution along the curve itself.

Token sequences similarity In Section 4, we aim to measure the similarity between two token sequences in order to control for this similarity. To measure token-level similarity we use the cosine distance between TF-IDF embeddings of two sequences. To measure semantic similarity we use cosine-distance between semantic sequence embeddings obtained from a MiniLM model fine-tuned[1] for the semantic sentence embedding.

4 Experiments and results

We test the ability of different LLMs of varying sizes to generate a predefined text from different sources in a non-autoregressive (parallel) mode. Moreover, we compare different ways to feed our trainable "proto-tokens" into LLM. We also try to understand the structure of the solution space by examining the relations of solutions for different problems.

Models We use six models for all experiments: three Pythia (Biderman et al., 2023) models of sizes 160M, 410M, and 1.4B, and three Llama-3 (Grattafiori et al., 2024) models of sizes 1B, 3B, and 8B.

Data Four text sources are used in the experiments to explore the possible connection between reconstruction performance and the text nature.

A set of random texts is generated by sampling from the top 100,000 words of the GloVe vocabulary (Pennington et al., 2014), to evaluate performance on unnatural texts.

To assess generation performance on natural but unseen texts, we use a collection of fanfiction texts from AO3 library [2], with a publication date cutoff of October 2024, which is later than the end of training for all models. For data processing details, see Kuratov et al. (2025).

The performance on seen natural texts is evaluated using PG-19 dataset (Rae et al., 2019) – a part of a dataset used for training Pythia models.

Finally, we include a set of model-specific generated texts. Specifically, for each model and each context text from PG-19 dataset, a suffix of the same length is generated as autoregressive continuation. The generation is done via multinomial sampling with sampling temperature $T = 1$.

Training details The embeddings of the proto-token are initialized randomly from a standard normal distribution and optimized using AdamW optimizer (Loshchilov and Hutter) with 0.01 learning rate, β_1, β_2 set to 0.9 and a weight decay of 0.01. The embeddings are trained for 5000 iterations with early stopping if perfect reconstruction accuracy is achieved. This number of iterations is often insufficient for convergence, but due to limited computational resources, we are unable to increase it. Instead, we aggregate results across multiple sequences. All models are loaded and trained using PyTorch framework and the Hugging Face Transformers library. Each experimental run is done on a single A100 or H100 80GB GPU with gradient accumulation enabled where necessary.

The code is available at this page[3].

Proto-token arrangement To select the best way to arrange two proto-tokens as input to an LLM for the main experiments, we conduct test runs on a single dataset-model pair for the variety of arrangements. For each arrangement, the same 50 texts from the PG-19 are selected, and the Llama-3.2-1B model is trained on prefixes of these texts at lengths: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024], to assess how token-level reconstruction accuracy

[1] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[2] https://archiveofourown.org/

[3] https://anonymous.4open.science/r/OneStep-91DDa

changes with respect to sequence length N. A representative selection of results is presented in Table 1.

Arrangement	$N=1$	$N=2$	$N=4$	$N=256$
$[e]_{\times N}$	$1.00_{\pm 0.00}$	$0.45_{\pm 0.31}$	$0.17_{\pm 0.18}$	$0.01_{\pm 0.01}$
$[e]_{\times (N/2)}[m]_{\times (N/2)}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.12_{\pm 0.13}$	$0.01_{\pm 0.01}$
$[e,m]_{\times (N/2)}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.17_{\pm 0.34}$
$[e][m]_{\times N}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.97_{\pm 0.15}$
$[e][m]_{\times (N-1)}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.99_{\pm 0.10}$

Table 1: Reconstruction accuracies for different input token arrangements across varying sequence lengths. Subscripts indicate the number of copies for each proto-token. The last two schemes differ as follows: in one the LLM is trained to predict the first text token t_1 for the proto-token e, while for the other the prediction for proto-token e is not guided and t_1 is a target prediction for the first copy of m instead.

Interestingly, having two trainable tokens is essential for the performance – the scheme with one trainable token fails to reconstruct even 2-token text, while best two-token schemes can reconstruct 256-token texts almost ideally.

Moreover, the way these two tokens are arranged is also important, with the best results obtained when the first token e is followed by $N-1$ copies of the second token m. This asymmetrical arrangement and critical necessity for two tokens suggest possible variation in functions of e and m. It is possible, that while one of them mostly incorporates language information, the role of the other one is mainly structural or mechanistic. This could be related to the phenomenon of "attention sinks" – Xiao et al. (2023) showed that LLMs are strongly attend to the initial tokens in the sequence even when they are not relevant. Moreover, adding a placeholder token as an attention sink could largely improve the performance of window-attention based models, which do not see the initial tokens by design. So, it is possible, that in order to successfully decode "proto-tokens", LLM needs a distinguishable token, which can be used as attention sink.

Token sharing In the previous section, we showed that the quality of reconstruction is very dependent on having two separate proto-tokens as an input. This observation, led us to hypothesize that, if a second token plays some structural or mechanistic purposes and does not contain information about the sequence itself. In that case, the second token could be shared between texts, reducing the number of optimized parameters, and simplifying the training process of the potential encoder.

To test this hypothesis, we run the same optimization process, but splitting 256 texts from the PG-19 dataset into groups of different sizes $S_g \in [1,4,16,64,256]$ and sharing either e or m within each group. We selected the maximum length of the text that can be losslessly compressed in a non-sharing mode - 256. The selection of the results is presented in Table 2.

Shared	Agg	$S_g=1$	$S_g=16$	$S_g=256$
e	max	$1.00_{\pm 0.00}$	$0.99_{\pm 0.01}$	$0.99_{\pm 0.02}$
	avg	$0.98_{\pm 0.08}$	$0.90_{\pm 0.17}$	$0.86_{\pm 0.20}$
p	max	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.01}$
	avg	$0.98_{\pm 0.07}$	$0.86_{\pm 0.19}$	$0.83_{\pm 0.18}$

Table 2: Reconstruction accuracy for schemes where one of the trainable tokens is shared within a group across different group sizes. "max" aggregation indicates that for every text, maximum accuracy across ten random seeds is selected and then averaged across texts, while "avg" denotes averaging across both seeds and texts.

Our experiments show that either of the tokens can be share and both options produce approximately the same results, if provided with sufficiently large number of initializations (seeds), but the number of starts needed increases significantly with the group size.

Depending on the proto-token being shared, we can build different intuitions behind the function of the shared tokens and the method itself. Is e-token is shared, which is located in the very beginning of the input sequence, the analogy that comes to mind is prompt-tuning (Lester et al., 2021), where a set of prompt embeddings is trained in order to improve performance in some specific task. In our case, a shared token e could be viewed as an "instruction" saying what an LLM should do with the upcoming embeddings (m-tokens) - decode different pieces of information for different positions. If m is shared, then training and prediction scheme resembles some of the speculative decoding approaches (Xia et al., 2023), where a number of special "[mask]" tokens are appended at the end of the sequence and the prediction for all them is then done in parallel. For all other experiments, unless stated otherwise, we use scheme with sharing m token between texts and random seeds and e token being unique for each text/seed pair.

		Share p	Pythia			Llama		
			160M	410M	1.4B	3.2-1B	3.2-3B	3.1-8B
Random	C_{tokens}	False	90	92	90	256	362	512
		True	45	22	45	181	256	256
	H_{LM}	False	$507.5_{\pm 105.9}$	$377.1_{\pm 133.1}$	$470.7_{\pm 103.1}$	$1551.3_{\pm 159.5}$	$2193.4_{\pm 190.2}$	$2974.4_{\pm 298.3}$
		True	$247.9_{\pm 32.0}$	$91.1_{\pm 30.8}$	$231.0_{\pm 37.9}$	$947.7_{\pm 155.0}$	$1292.2_{\pm 217.4}$	$1309.4_{\pm 234.6}$
Fanfics	C_{tokens}	False	128	128	131	362	512	724
		True	45	45	45	181	288	362
	H_{LM}	False	$358.9_{\pm 73.3}$	$395.4_{\pm 97.8}$	$261.0_{\pm 56.4}$	$1107.6_{\pm 129.1}$	$1408.4_{\pm 179.5}$	$1763.3_{\pm 280.2}$
		True	$145.0_{\pm 26.2}$	$82.3_{\pm 28.1}$	$147.9_{\pm 29.7}$	$576.4_{\pm 90.4}$	$835.9_{\pm 121.7}$	$1112.8_{\pm 168.6}$
PG-19	C_{tokens}	False	128	167	128	362	512	724
		True	45	32	64	181	256	362
	H_{LM}	False	$388.4_{\pm 66.4}$	$408.8_{\pm 96.3}$	$298.4_{\pm 77.4}$	$993.8_{\pm 183.4}$	$1346.0_{\pm 218.4}$	$1659.8_{\pm 344.5}$
		True	$156.0_{\pm 33.9}$	$88.1_{\pm 30.3}$	$156.0_{\pm 30.2}$	$456.5_{\pm 56.5}$	$826.1_{\pm 117.6}$	$832.3_{\pm 171.0}$
PG-19 (gen)	C_{tokens}	False	128	181	128	362	512	724
		True	45	32	64	181	362	362
	H_{LM}	False	$354.1_{\pm 72.0}$	$379.2_{\pm 82.6}$	$277.6_{\pm 71.3}$	$927.3_{\pm 103.4}$	$1266.6_{\pm 125.9}$	$1653.1_{\pm 211.4}$
		True	$153.0_{\pm 17.8}$	$106.9_{\pm 38.5}$	$197.1_{\pm 39.3}$	$478.7_{\pm 85.7}$	$788.6_{\pm 130.8}$	$771.7_{\pm 143.0}$

Table 3: Maximum reconstruction capacities for different models on different datasets.

Generation capacity We already see, that similar to autoregressive mode (Kuratov et al., 2025), LLMs can generate fairly long sequences in just one forward pass. To characterize this capability, and understand how it scales with model size, we run the optimization process for text prefixes of the predefined lengths [4, 5, 8, 11, 16, 22, 32, 45, 64, 90, 128, 181, 256, 362, 512, 724, 1024, 1448]. We report the maximum values of C_{tokens}, and H_{max} which correspond to the longest prefix for which at least 0.99 token-level accuracy is achieved – we treat such sequences as successfully predicted. In addition to a scheme with a shared p token, we also run a scheme with p not shared, to eliminate the effect of the insufficient number of random initializations. While our results in Section 4, suggest that p, can in principal, be shared without any quality drop, we also note that the optimization process is highly sensitive to initialization, especially when the proto-tokens are shared. The results are presented in Table 3.

Larger models in Llama family show greater reconstruction capabilities than the smaller ones of their family, while the situation with Pythia model-family is less obvious, with all the models showing approximately the same performance. Llama 1B model is also able to reconstruct almost three times larger sequence compared to Pythia model of the same size.

The source of the natural language (unseen / seen / generated) doesn't seem to have any systematic influence on the quality of reconstruction in terms of the number of tokens, while for unnatural random texts the generation capacity is significantly worse. This suggests that our "proto-tokens" do not "store" text tokens directly, but encode some more high-level representations, using language modeling capabilities of LLM. However, we also can't say that the compressibility of the text is determined by its likelihood under the sequential language model. In fact, we observe the opposite trend: lower total information content H_{LM} is compressed for less-information dense texts, such as generated by the LLM itself.

This difference is highlighted in Figure 2, where the amount of the language-information contained in trainable tokens is compared to autoregressive setup. The performance for unnatural texts is very similar and sometimes even identical, while for natural texts, the difference in capacity can be up to five times lower. However, more often the performance is just two times lower in non-autoregressive

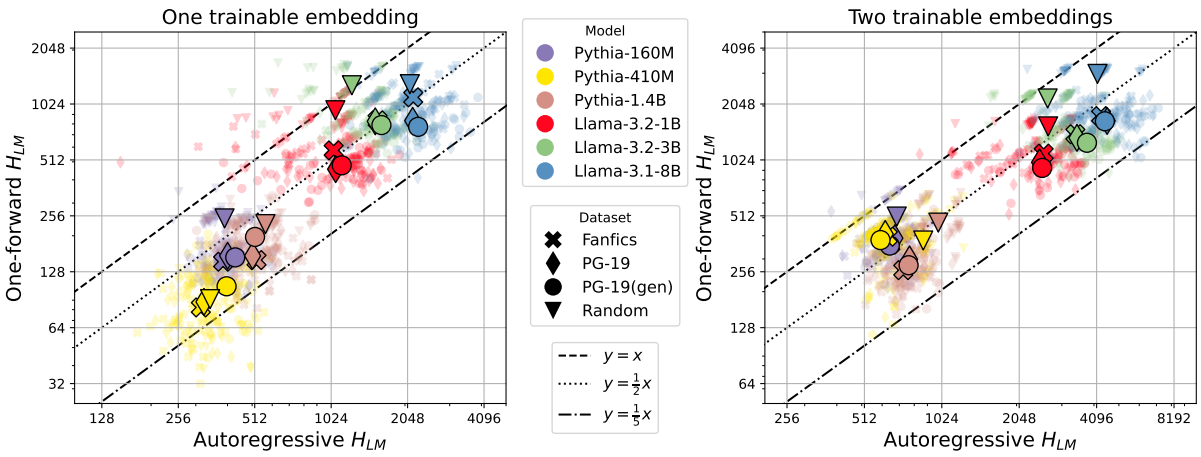

Figure 2: Maximum language information (H_{LM} for a maximum text prefix that is accurately reconstructed) compressed for different models and datasets. In the left plot, a single [mem] token is used in the autoregressive setting, and in the non-autoregressive one, p proto-token is shared between all texts within each model. In the right plot, two [mem] tokens are used and p proto-tokens are not shared. Each small point on the plots represents a single text, larger points indicate the average within each (model, dataset) pair.

case, suggesting that autoregressive decoding approximately doubles the information density that could be decoded for natural texts.

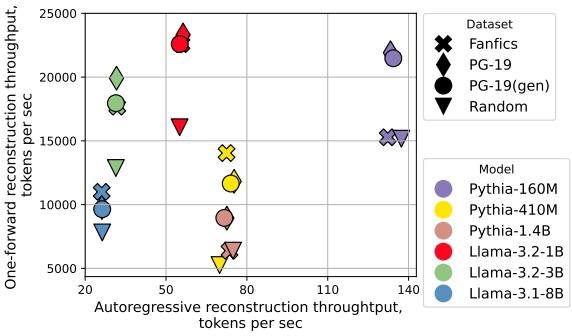

Figure 3: Reconstruction throughput comparison between autoregressive and non-autoregressive setups. For each (model, dataset) pair, we compress the texts of maximum losslessly compressible length. To measure execution time we use PyTorch profiling tools.

Although less information-dense, our one-forward method achieves significantly higher decoding – outperforming its autoregressive counterpart by a factor of 279 on average in the from the point of view of reconstruction throughtput (Figure 3). This dramatic difference is primary due to the number of forward passes.

Proto-tokens interpretation We investigate what type of information is encoded in proto-tokens, and the implications this has for a potential practical applications. In worst case scenario, they directly encode target tokens (imagine a vector just containing token_ids). In that case, the whole work of "language generation" should be done when encoding the text to this vector, which renders the "decoding" useless from the point of view of potential accelerated inference, though it could still be useful a context-compression tool. The opposite option, is that proto-tokens encode some compressed representation of possible prefix sequence, that can lead to such suffix, if the generation process is applied. In that case, the hard work of text generation is done, when the proto-tokens are decoded, which is more promising from the point of view of accelerated inference. All the intermediate states such as semantic prefix representation, semantic suffix representation, or the combination of both are also possible.

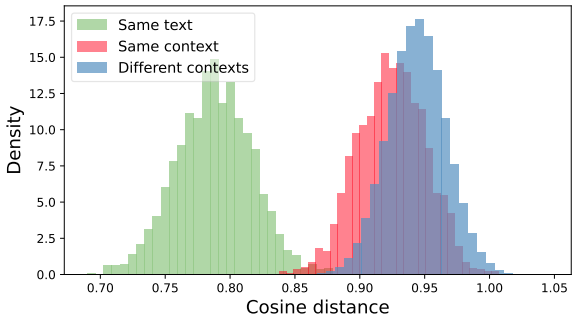

Figure 4: Pairwise cosine distances distribution for types kinds of proto-token embedding pairs. We select 50 contexts from PG19, for each context, generate 10 different continuation texts with sampling temperature 1. Then we find one solution for each of the first 9 generations and 10 solutions for the last generation.

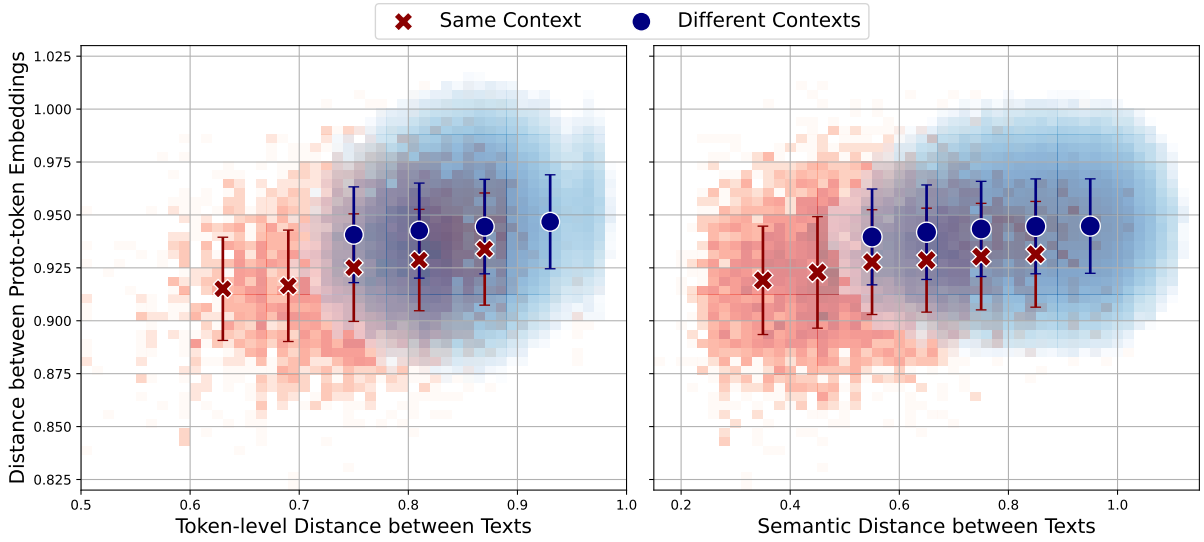

Figure 5: The comparison of the distances between proto-token embeddings for same-context text pairs and the distances between proto-token embeddings for different-context text pairs. Token-level distance is measured as cosine distance between TF-IDF embeddings. Semantic distance is measured as cosine distance between semantic text embeddings (see Section 3 for details).

We start by measuring the distances between three types proto-token embedding pairs: 1) corresponding to the same generated sequence, but different random seeds, 2) corresponding to the different texts but generated from the same context, 3) corresponding to the different texts generated from different contexts. As shown in Figure 4, that same-text solutions are almost always located closer to each other than different-texts solutions, which suggests a degree of locality in the learned representations. At the same time, same-context solutions are noticeably closer to each other than different-context ones. This may indicate that for that the encoded information at least partially reflects the potential context of the text, however we should be careful to account for the texts generated from the same context being more similar in general.

To do that, we measure pairwise distances between the generated texts, and examine whether the distance between the learned proto-token embeddings differ for a fixed distance between the texts. We use token-level measure of text similarity and semantic-level measure (see Section 3). For both measures, (see Figure 5) we observer that, given similar distances between texts, the proto-token embeddings are consistently closer when the text originate from the same context. We conclude that learned proto-tokens contain information beyond the information about the target sequence itself, that is somehow describes the potential context of the sequence.

Kuratov et al. (2025) raised the following concern about the structure of the solution space in autoregressive setup. Even though the same-text token embeddings are on average closer to each other than different-text token embeddings, they seem to be disconnected – a linear interpolation between two solutions does not yield a valid reconstruction. This could mean that the potential encoding to this space could be problematic as the same object could be mapped to disconnected regions. We find that in our non-autoregressive case, the linear interpolation between same-text solutions also does not produce a solution (Figure 6).

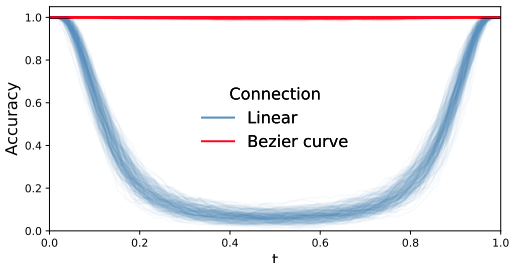

Figure 6: Pairwise interpolation accuracies between 10 solutions for 5 texts ($5 \times 10 \times 9/2$ pairs in total).

However, the solutions could be connected using quadratic Bezier curves (parabolic segments) lying inside "solution set". This means that even though

same-text solutions do not form a convex set, they form a connected set. In fact, our experiments show that the maximum ratio between Bezier curve length and the corresponding linear connection is only 1.2, indicating that the paths are nearly linear. These results demonstrate that the solution space is fairly well behaved, providing reasonable hope that an encoder model could be built to map into that space.

5 Discussion and Conclusions

In this paper, we demonstrate that frozen LLMs have a surprising ability to generate hundreds of accurate tokens in a single forward pass – without any iterative decoding – when provided with just two specially trained "proto-tokens".

We find that both the number and the arrangement of such tokens is crucial for enabling this generation capacity. Interestingly, with only one proto-token, LLMs are unable to generate more than a single token of text. In contrast, two properly arranged proto-tokens can enable the generation of sequences hundreds tokens long. This significant leap in the performance, along the observation that one of the vectors can (in principal) be shared across many texts, suggest that proto-tokens play different functional roles during generation. However, the precise nature of the role differentiation remains an open question.

We find that bigger model size does not universally imply better generation capacity. While larger models in Llama-3 family demonstrate improved reconstruction capacity, Pythia models show no such trend – larger models do not outperform smaller one. Whether this difference is connected to the architectural variations is an open question.

Additionally, we do not observe any consistent relationship between the source of the natural text and the reconstruction ability of LLMs. Surprisingly, Even for the texts generated by the LLM itself, the number of successfully reconstructed tokens is the same as for any other natural text. However, for the texts composed of random tokens, performance drops noticeably. This suggests that our reconstruction process does not fully leverage the language modeling capabilities of LLMs, and may instead mostly rely on low-level token patterns.

Although the reconstructed sequences in the non-autoregressive setting are, on average, about two times shorter than those in the autoregressive case, the computational efficiency of single-forward approach allows to achieve up to 279× greater generation throughput.

Despite this, we observe that proto-tokens encode more than just the target sequence. Embeddings of the "proto-tokens" corresponding to the different texts generated from the same context are significantly closer to each other than those from unrelated sequences. This indicates that the learned representations capture some contextual information.

Finally, we discover that the embedding space in which proto-tokens exist, has very desirable structural properties – proto-tokens corresponding to the same text, form localized and connected regions, enabling smooth transitions via quadratic interpolation. These findings suggest that it may be feasible to build an encoder capable of mapping into this space, opening the door to future work on non-autoregressive inference and representation learning.

6 Limitations

Although our paper demonstrates the surprising capability of LLMs to generate long sequences in a single forward pass from just two learned embeddings, several important limitations should be acknowledged:

1. Lack of immediate practical application: Most importantly, this work highlights an interesting quirk of LLMs and does not suggest any immediate practical implications or real-life usages for the method.

2. Architectural dependence: The method demonstrates different behavior across model families, suggesting some architectural dependence. As a results, our method may potentially not generalize to other model architectures.

3. Limited domain coverage: While we evaluate four different text sources , the results may not generalize beyond those explored in our experiments.

References

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yuri Kuratov, Mikhail Arkhipov, Aydar Bulatov, and Mikhail Burtsev. 2025. Cramming 1568 tokens into a single vector and back again: Exploring the limits of embedding space capacity. *arXiv preprint arXiv:2502.13063*.

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*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2024. Gpt understands, too. *AI Open*, 5:208–215.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. 2023. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12336–12355, Toronto, Canada. Association for Computational Linguistics.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925, Singapore. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.