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

Gleb Mezentsev
AIRI
Skoltech
mezentsev@airi.net

Ivan Oseledets
AIRI
Skoltech
oseledets@airi.net

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 trained input embedding. In this work, we explore whether autoregressive decoding is essential for such reconstruction. We show that frozen LLMs can generate hundreds of accurate tokens in just one token-parallel forward pass, when provided with only two learned embeddings. This reveals a surprising and underexplored multi-token generation capability of autoregressive LLMs. We examine these embeddings and characterize the 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 – suggesting the potential to train a practical encoder. The existence of such representations hints that multi-token generation may be natively accessible in off-the-shelf LLMs via a learned input encoder, eliminating heavy retraining and helping to overcome the fundamental bottleneck of autoregressive decoding while reusing already-trained models.

1 Introduction

Large language models are typically trained to generate text in an autoregressive manner – they predict one token at a time based on the previously generated context. Several attempts aim to change this. However, they either require an additional model for candidate generation (Leviathan et al., 2023), substantial additional fine-tuning of autoregressive LLM (Cai et al., 2024; Stern et al., 2018; Gloeckle et al., 2024) or full model retraining (Ghazvininejad et al., 2019; Austin et al., 2021; Li et al., 2022). This leaves an open question – is it possible to reuse autoregressively pretrained LLM for multi-token generation with minimal to no additional training. We discover a previously undocumented phenomenon, that can help us to achieve this goal.

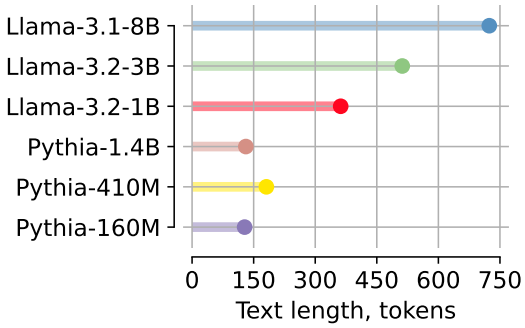

Figure 1: One pass, many tokens. Each dot shows the maximum exact reconstruction length in a single non-autoregressive forward pass with frozen weights, conditioned only on two learned embeddings – evidence of hidden multi-token capabilities.

We found that for any given text of reasonable length, there exists a latent one-vector representation of this text, such that, if a frozen pretrained LLM is conditioned on this representation, it accurately generates the whole text in a single forward pass, without any iterative decoding. In this work, we demonstrate this phenomenon, investigate what those compressed representations encode and whether this finding reveals anything about LLMs' parallel generation capabilities.

Our contribution is as follows:

1. We show that LLMs can reconstruct arbitrary sequences of hundreds of tokens from as few as two learned input embeddings, with one of them being universal for all texts.

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 empirically characterize learned representations – analyze their information content and embedding-space geometry.

The code is available at this GitHub page.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 22880–22889
November 4-9, 2025 ©2025 Association for Computational Linguistics

2 Related Work

The most direct influence for our work is a paper by Kuratov et al. (2025), which showed that frozen LLM can reconstruct an arbitrary sequence of tokens $T = [t_1, \dots, t_N]$ if given a sequence of special, so-called memory tokens $[mem_1, \dots, mem_K]$. The embeddings for these tokens are trained by optimizing a causal language modeling objective over a concatenated input sequence $Z = [mem_1, \dots, mem_K, t_1, \dots, t_N]$ 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 text length and can be as low 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 trained from different random initializations for the same text end up far apart. Moreover, linear interpolations between those embeddings produce 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 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 frozen LLMs to improve their performance on new tasks. Several speculative (Xia et al., 2023) and parallel (Santilli et al., 2023) decoding approaches utilize a similar mechanism for multiple token prediction using decoder architectures. 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 prior work, we show that a frozen LLM can generate accurate multi-token sequences in one forward pass without additional LLM training or iterative decoding.

3 Method

To adopt the approach from Kuratov et al. (2025) to the 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 drops dramatically when using only one (see Section 4).

There are many ways to arrange two vectors into 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.

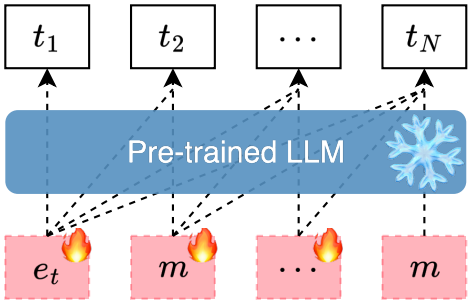

Figure 2: Two "proto-tokens" (trainable embeddings) are fed into frozen, pretrained LLM and optimized in such a way that LLM predicts an arbitrary token-sequence in a single forward pass. e_t is trained for each text separately, while m could be shared across texts.

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: $Z = [e, m, m, \dots, m]$ – 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 $T = [t_1, t_2, \dots, t_N]$ and the frozen LLM's output for the input sequence:

$$L_{CE} = -\sum_{i=1}^{N} log\mathbb{P}_{LM}(t_i \mid e, \underbrace{m, \dots, m}_{i-1}) \quad (1)$$

The prediction is obtained using standard causal attention masking, so that the predicted probabilities for the token t_i depend on the first i input "proto-tokens" (see Figure 2).

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

$$C_{tokens} = \sum_{i=1}^{N} \mathbb{1}(LM(Z_{[1:i]}) = t_i) \quad (2)$$

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:

$$H_{LM} = -\sum_{i=1}^{N} log\mathbb{P}_{LM}(t_i \mid t_{<i}) \quad (3)$$

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 tasks. We optimize the parameters of a degree-two 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:

$$\phi_\pi(\tau) = (1-\tau)^2 p_1 + 2\tau(1-\tau)\pi + \tau^2 p_2 \quad (4)$$

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

We want to find the value of π that minimizes the cross-entropy loss along the curve. To do that, we minimize the expectation of the cross-entropy loss with respect to a uniform distribution of τ:

$$l_\pi = \mathop{\mathbb{E}}_{\tau \sim \mathcal{U}[0,1]} \sum_{i=1}^{N} -log\mathbb{P}_{LM}(t_i \mid \phi_\pi(\tau)) \quad (5)$$

To do that, we iteratively sample $\tau \sim \mathcal{U}[0,1]$ and optimize l_π with respect to π using Adam optimizer. This optimization under the uniform distribution over τ 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 $T = 1$.

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

[2] https://archiveofourown.org/

Training details The embeddings of the proto-tokens are initialized from standard normal distribution and optimized via 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 an early stopping at a perfect reconstruction accuracy. 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 and report the best results. Although, the exact reconstruction capacity could be under-estimated, we believe that, given he exploratory nature of this work, it is more important to demonstrate and characterize the phenomenon itself, rather than to achieve the precise upper bound on reconstruction capacity. All models are trained using PyTorch framework and Transformers library. Each experimental run is done on a single A100 or H100 80GB GPU with gradient accumulation enabled where necessary.

Proto-token arrangement To select the best way to arrange two proto-tokens as input for 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 token-level reconstruction accuracy change 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. In the second-to-last scheme the LLM is trained to predict the first text token t_1 for the proto-token e, while with the last one, the prediction for proto-token e is not guided and t_1 is a target for the first copy of m.

Interestingly, having two proto-tokens is essential. The one-token scheme fails to reconstruct even 2-token text, while best two-token schemes reconstruct 256-token texts almost perfectly.

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 strongly attend to the initial tokens in the sequence even when they are not relevant. So, it is possible, that in order to successfully decode "information" proto-token, LLM needs a distinguishable "sink" proto-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, a second token serves a structural or mechanistic purpose 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, splitting 256 texts from the PG-19 into groups of varying 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-shared mode - 256. The results are averaged over 10 random seeds. The selection of 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	$0.98_{\pm 0.08}$	$0.90_{\pm 0.17}$	$0.86_{\pm 0.20}$
m	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 with one of proto-tokens shared within groups for different group sizes. "max" indicates that for every text, maximum accuracy across ten random seeds is averaged across texts, while "avg" denotes averaging across both seeds and texts.

Sharing either token yields comparable performance if provided with a sufficiently large number of restarts (random seeds), but the required number of restarts increases significantly with group size.

		Share m	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.

Depending on the proto-token being shared, we can build different intuitions behind the function of the shared tokens and the method itself.

If the 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 the m-token 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.

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 and changes depending on the nature of the text, 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_{LM} which correspond to the longest prefix for which at least 0.99 token-level accuracy is achieved – we treat such sequences as successfully reconstructed. In addition to a scheme with a shared m token, we also run a scheme with m not shared, to eliminate the effect of the insufficient number of random initializations. While our results in Section 4, suggest that m, can in principle, 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 with the best results across datasets presented in Figure 1.

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

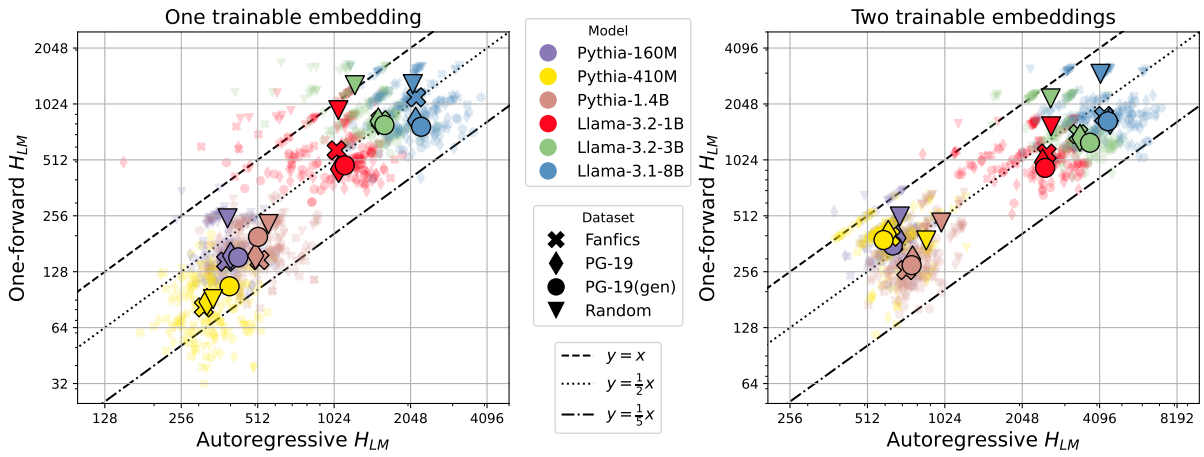

Figure 3: Maximum language information (H_{LM} for a maximum text prefix that is accurately reconstructed) compressed for different models and datasets. In the left plot, a single [mem] token is used in the autoregressive setting, and in the non-autoregressive one, m proto-token is shared between all texts within each model. In the right plot, two [mem] tokens are used and m 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.

The natural text source (unseen, seen or generated) does not 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 "proto-tokens" do not "store" tokens directly, but encode some more high-level representations, using language modeling capabilities of LLM. However, we also can not 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 3, where the amount of the 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 difference is just two-fold, suggesting that autoregressive decoding approximately doubles the "effective" information density in natural text.

Although less information-dense, our one-forward method achieves significantly higher decoding throughput in the context of text reconstruction – outperforming its autoregressive counterpart by a factor of 279 on average (Figure 4). This dramatic difference is due to the number of forward passes. While an obvious downstream task is still to be found, such speed could matter in many settings where fast decoding is particularly important.

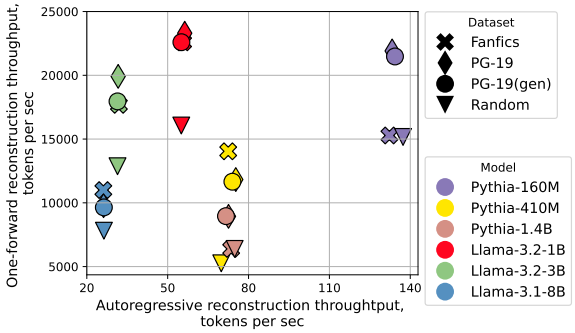

Figure 4: Reconstruction throughput for autoregressive and non-autoregressive setups. For each model-dataset pair, the throughput equals to a maximum losslessly compressible length divided by the reconstruction time.

While we do not introduce the method as a practical way of compressing or generating texts, but rather as a demonstration of interesting phenomenon, we still measure the training time across models and text lengths to demonstrate the full picture. Training time (Table 4) scales roughly linearly with sequence length, with around 10 seconds for length 32 and around 200 seconds for length 512.

Model | N	32	64	128	256	512
Pythia-160M	6	25	68	–	–
Pythia-410M	10	21	106	–	–
Pythia-1.4B	10	66	129	–	–
Llama-3.2-1B	11	15	27	87	–
Llama-3.2-3B	14	18	28	78	261
Llama-3.1-8B	16	20	29	60	215

Table 4: Proto-token training time in seconds for different models and sequence lengths averaged over datasets.

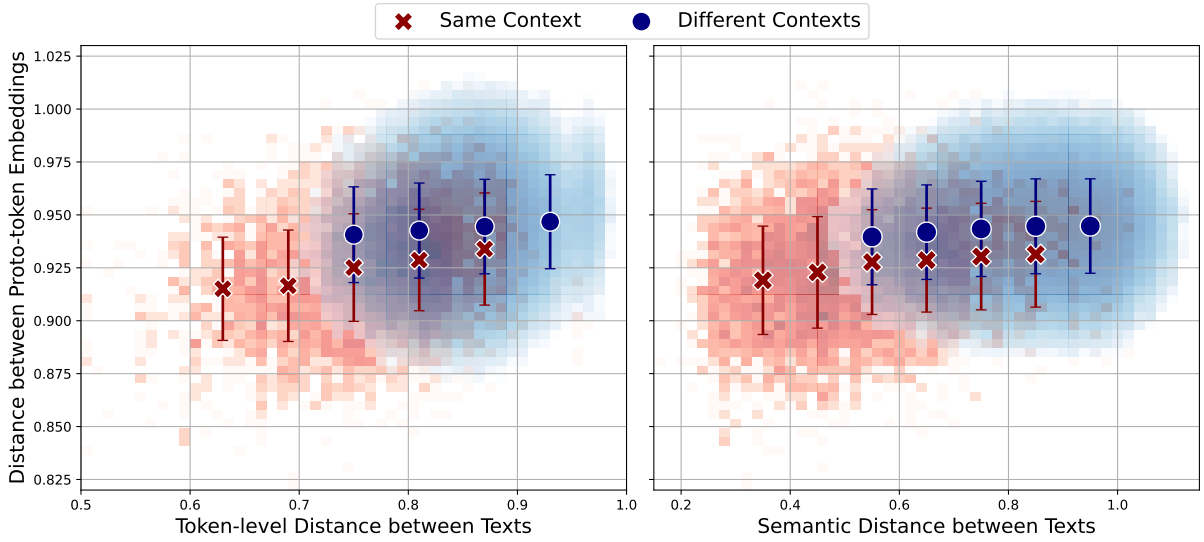

Figure 5: We compare proto-token embedding distances for same context text pairs and different-context text pairs. Token-level distance is measured as cosine distance between TF-IDF embeddings. Semantic distance is measured as cosine distance between semantic text embeddings (see Section 3 for details).

Proto-tokens interpretation We examine the information encoded in proto-tokens and the implications this has for potential practical applications. In worst case scenario, they directly encode target tokens (imagine a vector containing token_ids). If so, the entire "language generation" effort happens during encoding, making decoding irrelevant for accelerated inference – though the approach could still be useful as a context-compression tool. The alternative is that proto-tokens encode a compressed representation of a prefix which, when the model generates from it, produces the observed suffix. In that case, the hard work of text generation is done during decoding, which is more promising from the point of view of accelerated inference. All the intermediate options are also possible.

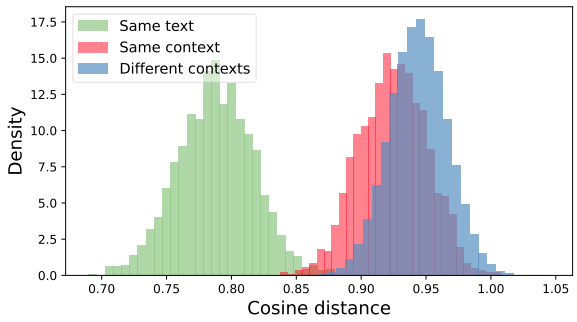

Figure 6: Cosine embedding distances for different pairings of proto-tokens. We select 50 contexts from PG19 and for each context, generate 10 continuation texts. We find one solution for each of the first 9 generations and 10 different-seed solutions for the last generation.

We start by measuring the distances between three types of 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 6, the same-text solutions are almost always located closer to each other than different-texts solutions, which suggests 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 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 generated texts, and examine whether the distance between 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, (Figure 5) we observe that, given the same distances between texts, the proto-token embeddings are on average closer when the texts originate from the same context. We conclude that learned proto-tokens contain information beyond the information about the target sequence itself – they somehow partially describe the potential context of the sequence. However, we should note that, the ef-

fect of having the same context on the distance between proto-tokens is small, and the distributions for same-context distances and different-context distances heavily overlap. Our results suggest that proto-tokens still mostly contain information about the text itself with only a fraction of the information about the context preserved.

We also conducted a preliminary experiment on accessing the information contained in proro-tokens without first decoding them into text. We took 50 128-token context sequences from PG-19 dataset, generated 256-token continuations with Llama3.2-1B model and trained (e, m) pairs only for the first 128 tokens of the model continuations. Then we started the autoregressive generation of the same model from different combinations of proto-tokens and ⟨BOS⟩-token and visually compared the contents of the resulting token sequences with both contexts and model-continuations. In all cases, the resulting sequences either contain meaningless token-combinations or meaningful texts that are not related to either context or continuation. We conclude that the information from proto-tokens could not be accessed without decoding at least when they are used directly as an autoregressive generation context.

Proto-tokens embedding space structure Kuratov et al. (2025) raised the following concern about the structure of the solution space in the 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 7).

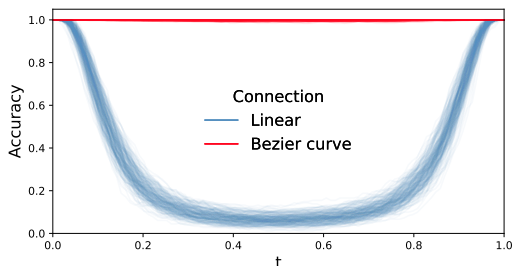

Figure 7: Pairwise interpolation accuracies between 10 solutions for 5 texts ($5 \times 10 \times 9/2$ pairs in total).

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 of tokens long. This significant leap in the performance, along the observation that one of the vectors can (in principle) be shared across many texts, suggests that proto-tokens play different functional roles during generation.

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 perform better.

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, with random-token sequences, 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, two times shorter than those in the autoregressive case, the efficiency of single-forward approach allows to achieve up to 279× greater generation throughput.

We also 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, on average, closer to each other than those from unrelated sequences. This indicates that learned representations capture some potential contextual information.

Finally, we discover that the embedding space of proto-tokens has very desirable structural properties – proto-tokens corresponding to the same text, form localized and connected regions with 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.

We view this work as an existence proof: certain text representations can elicit multi-token behavior in frozen, single-token LLMs. Making them practical requires training an encoder that maps text into these representations. As a future work, we plan to study decoders that either generate a suffix from an encoded prefix or reconstruct the entire text. Depending on the setup, such systems could enable multi-token or chunk-wise generation, learned compression or RAG (potentially by extracting information directly without decoding).

6 Limitations

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 yet, as direct proto-token optimization should be replaced with parametrized encoder for any practical application.

2. Architectural dependence: The method demonstrates different behavior across model families, suggesting some architectural dependence. As a result, 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.

4. Evidence, not bounds: Our 5k-step optimization budget may not always reach full convergence. Thus, our results should be read as evidence of feasibility (existence of one-pass decoding) rather than a precise capacity bound.

Acknowledgments

The work was supported by the grant for research centers in the field of AI provided by the Ministry of Economic Development of the Russian Federation in accordance with the agreement 000000C313925P4F0002 and the agreement with Skoltech №139-10-2025-033.

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.

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.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

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.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*.

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. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19323–19339, Vienna, Austria. Association for Computational Linguistics.

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.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343.

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.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31.

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*.