Superposed Decoding: Multiple Generations from a Single Autoregressive Inference Pass

Ethan Shen[◇] **Alan Fan**[◇] **Sarah Pratt**[◇] **Jae Sung Park**[◇] **Matthew Wallingford**[◇]
Sham Kakade[†] **Ari Holtzman**[‡] **Ranjay Krishna**[◇] **Ali Farhadi**[◇] **Aditya Kusupati**[◇]
[◇]University of Washington [†]Harvard University [‡]University of Chicago
`{ethans03, kusupati}@cs.washington.edu`

Abstract

Many applications today provide users with multiple auto-complete drafts as they type, including GitHub's code completion, Gmail's smart compose, and Apple's messaging auto-suggestions. Under the hood, language models support this by running an autoregressive inference pass to provide a draft. Consequently, providing k drafts to the user requires running an expensive language model k times. To alleviate the computation cost of running k inference passes, we propose Superposed Decoding, a new decoding algorithm that generates k drafts at the computation cost of one autoregressive inference pass. We achieve this by feeding a superposition of the k most recent token embeddings from the drafts as input to the next decoding step of the language model. At every inference step we combine the k drafts with the top-k tokens to get k^2 new drafts and cache the k most likely options, using an n-gram interpolation with minimal compute overhead to filter out incoherent generations. Our experiments show that k drafts from Superposed Decoding are at least as coherent and factual as Nucleus Sampling and Greedy Decoding respectively, while being at least $2.44\times$ faster for $k \geq 3$. In a compute-normalized setting, user evaluations demonstrably favor text generated by Superposed Decoding over Nucleus Sampling. Code and more examples open-sourced at https://github.com/RAIVNLab/SuperposedDecoding.

1 Introduction

Commercial surveys find that 80% of e-commerce websites provide autocomplete as a feature [20]. With the proliferation of large language models, autocomplete drafts are now ubiquitous in an even wider range of applications. Examples include short draft suggestions in Gmail Smart Compose [7] and code snippets in GitHub Copilot [6]. These options provide users with the ability to consider different phrasings and increase the likelihood of having at least one suggestion that reflects their intent. While language models (LMs) [33] now power these multiple suggestions in most settings, each additional suggestion necessitates another inference pass, making it computationally expensive.

Language models use autoregressive inference to generate a plausible sequence of next tokens for a given prefix [39]. The next token generated depends on the prefix and the previously generated tokens. Decoding algorithms like Greedy Decoding, Top-k Sampling [11], Beam Search, and Nucleus Sampling [17] present various ways of obtaining generations during autoregressive inference. For a prefix, Greedy (maximum likelihood) Decoding picks the most probable token at every autoregressive timestep, eventually generating only one auto-completion suggestion. Instead of making a greedy choice at every timestep, we can also sample from the top-k most probable tokens to generate the sequence [11]. Also leveraging the top-k most probable tokens is Beam Search, which picks the most probable k beams from the k^2 possible drafts at every timestep. Alternatively, Nucleus Sampling [17], particularly effective at generating natural-sounding text, grows generations by sampling from the

Preprint. Under review.

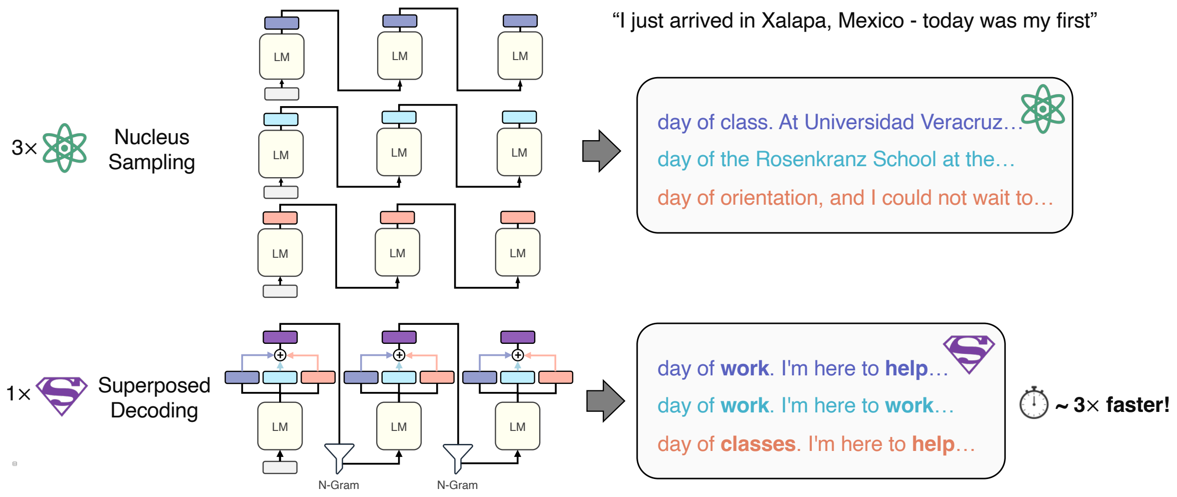

"I just arrived in Xalapa, Mexico - today was my first"
3×
Nucleus Sampling
LM
day of class. At Universidad Veracruz…
day of the Rosenkranz School at the…
day of orientation, and I could not wait to…
1×
Superposed Decoding
N-Gram
day of work. I'm here to help…
day of work. I'm here to work…
day of classes. I'm here to help…
~ 3× faster!

Figure 1: To generate multiple k auto-complete suggestions for a prefix using an LM, the existing decoding methods like Nucleus Sampling need k inference passes. In contrast, Superposed Decoding can generate k suggestions at the cost of a single inference pass while being as coherent and factual.

collection of the smallest subset of tokens that form a preset probability mass (p) at every timestep. Top-k Sampling, Beam Search, and Nucleus Sampling offer the benefit of generating multiple suggestions. However, this comes at the cost of requiring multiple autoregressive inference passes.

We introduce Superposed Decoding (SPD) (Figure 1), a decoding algorithm that can generate mutiple (k) high-quality generations using only a *single* autoregressive inference pass. At each autoregressive timestep during inference, Superposed Decoding feeds in the superposition (weighted combination) of the embeddings corresponding to the k most recent drafted tokens (Section 3.1). After selecting the top-k output tokens, SPD expands the existing k drafts with them, resulting in k^2 potential new drafts. Each draft has a probability score assigned to it, which is further smoothed using a combination of various n-gram models ($n \in [2, 6]$) [37, 24]. N-gram interpolation for smoothing helps improve coherency by selecting the most probable and coherent continuations (top-k drafts) for the next autoregressive step (Section 3.2). The n-gram interpolation is computationally inexpensive and allows flexibility to change domains of interest during inference (eg., programming, healthcare, finance, etc.). Superposed Decoding's effectiveness can be attributed to the apparent linearity of representations in language models [34, 22] which we concurrently discovered (Section 3.3).

Our experiments demonstrate that Superposed Decoding on Llama-2-7B [40] can generate $k \geq 1$ drafts that are as coherent, in terms of perplexity, as Nucleus Sampling (Section 4.1). However, SPD achieves this with a single inference pass, making it at least $2.44\times$ faster than any other standard decoding methods for $k \geq 3$ (Section 4.3). The ability to generate multiple drafts at the cost of one also increases the coverage significantly for fact-based evaluation tasks like TriviaQA and Natural questions – where SPD increases the chance of generating the correct answer by at least 5% when using 3 drafts (Section 4.2). Finally, through an extensive human evaluation for a wide range of prefixes, we show that SPD generations are as preferred as Nucleus Sampling in direct comparison (1v1) while outperforming by up to 20% in a compute normalized setting (3v1 and 3v2) (Section 4.4).

Superposed Decoding can help improve user experience by offering significant computational efficiency while maintaining accuracy for various writing tasks that often benefit from multiple short draft suggestions. Additionally, Superposed Decoding is extremely generalizable, works across different language models like Mistral 7B, and is capable of generating long-form content reliably, all while being nearly as diverse in suggestion as Nucleus Sampling (Section 5).

2 Related Work

Decoding algorithms determine how tokens are selected from a language model's next token distribution. This is typically done greedily with Beam Search or Greedy Decoding, or stochastically with Nucleus Sampling [17] or Top-k Sampling [11]. Greedy Decoding operates by selecting the token with the highest probability at each timestep. However, locally optimal decisions can be globally suboptimal. On the other extreme, an exhaustive exploration of all token combinations is intractable with time complexity $\Theta(|\mathcal{V}|^T)$, where $\mathcal{V}$ is the vocabulary size and T the total number of timesteps. A practical compromise is Beam Search, which continually caches the top-B most likely generations, where B is the number of beams or drafts. Thus, Beam Search is guaranteed to find more likely generations than Greedy Decoding while avoiding its drawbacks.

Alternatively, Nucleus Sampling and Top-k Sampling decode probabilistically. To avoid degeneration, Top-k Sampling truncates the probability mass to the k most probable tokens, whereas Nucleus Sampling truncates the probability mass to the smallest possible set of words whose cumulative probability exceeds probability p. Because of Nucleus Sampling's propensity to produce unique and unrepetitive outputs, it has become the standard [3, 21, 41], though Greedy Decoding is still favored for tasks such as short question answering [40].

Generating multiple drafts linearly scales the number of inference passes in these decoding methods. Multi-token prediction [13, 35] addresses this by pre-training an LM having n independent softmax heads that predict n future tokens in parallel. When using multi-token prediction with speculative decoding, exact inference is $3\times$ faster. In a similar vein, Medusa [4] reduces the number of decoding steps by adding extra decoding heads, using a tree-based attention mechanism to generate candidates while simultaneously verifying them in each decoding step. Through this, Medusa achieves a $2.2\times$ reduction in inference latency while maintaining generation quality. However, multi-token prediction requires re-training and Medusa requires additional fine-tuning for the extra decoding heads.

Superposed Decoding (SPD), on the other hand, can be easily integrated **out of the box** without any additional training on any language model. Further, Superposed Decoding is complementary to Medusa and multi-token prediction, as well as other efficiency techniques like speculative decoding [29, 25, 5], quantization [9, 31, 18], pruning [12, 38, 27], and architectural optimizations [36, 2, 1, 42, 26, 10].

3 Superposed Decoding (SPD)

Given a text input as a prefix, Superposed Decoding uses an autoregressive LM f_θ to produce k viable completion drafts in one inference pass.

First Timestep. Let x denote a generated token in the vocabulary $\mathcal{V}$ and $M = (x_1, \ldots, x_m)$ represent an initial prefix of m tokens. Our goal is to generate k unique drafts starting from the prefix. The next token distribution at the first timestep $m+1$ is: $p_\theta(x_{m+1}|x_1, \ldots x_m) = p_\theta(x_{m+1}|M)$.

For the first timestep, each of the k drafts is initialized as the prefix so we use the same next token distribution for all drafts. We grow draft d_i by greedily selecting the i^{th} most probable token, making:

$$d_i = (M, x^i_{m+1})$$

where x^i_t is the token at timestep t for the i^{th} draft. We also track each draft's probability p_i as its score, which is initially the probability of its first token $p_\theta(x^i_{m+1}|M)$. Consequently, the set of drafts D after the first timestep is:

$$D = \begin{pmatrix} (M, x^1_{m+1}) \\ \vdots \\ (M, x^k_{m+1}) \end{pmatrix} \text{ with probabilities } P = \begin{pmatrix} p_\theta(x^1_{m+1}|M) \\ \vdots \\ p_\theta(x^k_{m+1}|M) \end{pmatrix} \tag{1}$$

Next Timesteps. As the input to the LM at timestep t, we construct $\tilde{x}_{t-1}$, which is a superposition (weighted linear combination) of the embeddings for the most recent token x_{t-1} of the k drafts. This means that $\tilde{x}_{m+1}$, the input to the LM at the second timestep $m+2$, is the superposition of the tokens x^i_{m+1} for $i = 1, \ldots, k$. We use this *superposed embedding* as a single and accurate approximation for the most recent token across all k drafts at the $(t-1)^{\text{th}}$ timestep. We run autoregressive inference over $\tilde{x}_t$ once for all k drafts instead of the usual once per draft, allowing us to formulate inference as:

$$p_\theta(x_t|M, \tilde{x}_{m+1}, \ldots, \tilde{x}_{t-1})$$

Because each draft contains its own contextual clues and syntax, we cannot blindly use the distribution of x_t for each draft. Instead, we interpolate the superposed distribution $p_\theta(x_t|M, \tilde{x}_{m+1:t-1})$

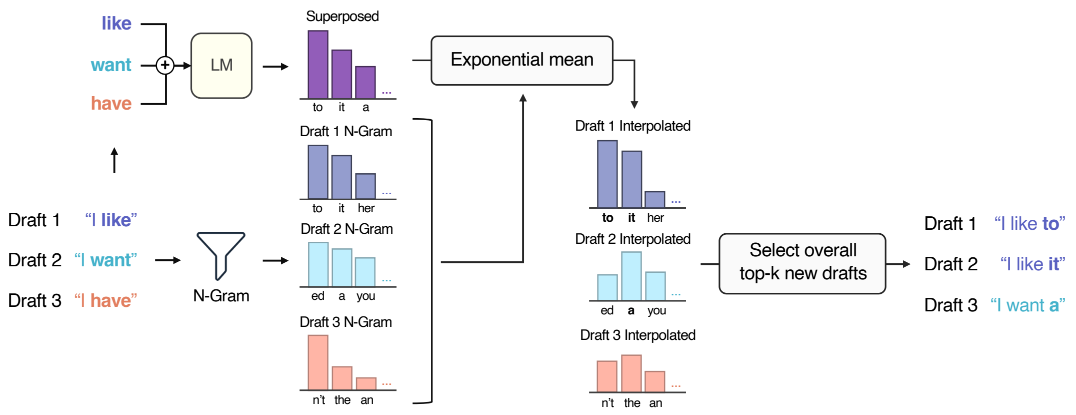


Figure 2: Superposed Decoding relies on feeding a superposed token embedding – based on the most recent tokens from the current k drafts – as the input during the auto-regressive inference step. This generates k^2 new drafts using the existing k drafts and the top-k output tokens at the new timestep. Finally, keep the top-k drafts after filtering with an n-gram interpolation to improve coherency.

with a draft-specific next token distribution from an n-gram model to get a final distribution $p_f(x_t^i|M, x_{m+1:t-1})$ that is unique to each draft. Next, we rank the draft options by the joint probability of their respective previous draft p_i and their selected token. We choose the top k options as drafts for the next timestep and update their probabilities. This gives:

$$D = \begin{pmatrix} (M, x_{m+1}^1, \ldots, x_t^1) \\ \vdots \\ (M, x_{m+1}^k, \ldots, x_t^k) \end{pmatrix} \text{ with probabilities } P = \begin{pmatrix} p_\theta(x_{m+1}^1|M) \prod_{s=m+2}^t p_f(x_s^1|M, x_{m+1:s-1}^1) \\ \vdots \\ p_\theta(x_{m+1}^k|M) \prod_{s=m+2}^t p_f(x_s^k|M, x_{m+1:s-1}^k) \end{pmatrix} \tag{2}$$

We continue generation until the maximum generation length or stop token is reached. In the following sections, we break down the process in detail. We also present pseudocode in Appendix A.

3.1 Token Superposition

During training, language models learn a representation (token embedding) $z \in \mathcal{R}^d$ for every token $x \in \mathcal{V}$. We leverage this representation at timestep t to construct $\tilde{x}_t$, weighing the representation for x_{t-1}^i by a coefficient γ_i.

$$\tilde{x}_t = \sum_{i=1}^{k} \gamma_i \cdot z_{t-1}^i \text{ where } \sum_{i=1}^{k} \gamma_i = 1 \tag{3}$$

The performance of $\tilde{x}_t$ is highly dependent on choosing the appropriate weight for each embedding z_{t-1}^i. We find that the best strategy is to set γ_i to the normalized probability of draft i such that

$$\gamma_i = \frac{p_i}{\sum_{j=1}^{k} p_j} \tag{4}$$

where p_i is the probability of the i^{th} draft, introduced in Section 3. This allows us to directly tie the weight of a token to the likelihood that it will be preserved in future timesteps, reducing drift between the superposed embeddings and the drafts they represent.

3.2 N-Gram Interpolation

We construct each draft's n-gram distribution $p_{\text{ngram}}(x_t^i|M, x_{m+1}^i, \ldots, x_{t-1}^i)$ by interpolatively smoothing the next token distributions over a set of n-gram models using weights λ, where $n \in [2, 6]$.

$$p_{\text{ngram}}(x_t^i|M, x_{m+1}^i, \ldots, x_{t-1}^i) = \sum_{n=2}^{6} \lambda_n \cdot p_{n\text{-gram}}(x_t^i|M, x_{m+1}^i, \ldots, x_{t-1}^i) \tag{5}$$

We base our interpolation weights on weights for RedPajama found by Liu et al. [30], with additional tuning (specifics in Appendix B). However, domain-specific n-gram models can easily be swapped in for specific tasks such as code generation [19]. We use the exponential mean of p_θ and p_{ngram} as our final distribution p_f, where α is a hyperparameter controlling the impact of the n-gram distribution:

$$p_f(x_t^i|M, x_{m+1:t-1}^i) = p_\theta(x_t|M, \tilde{x}_{m+1:t-1})^{1-\alpha} \cdot p_{\text{ngram}}(x_t^i|M, x_{m+1:t-1}^i)^\alpha \tag{6}$$

This means that when generating, we only consider tokens appearing in both the Superposed Decoding and n-gram distributions. If there is no overlap between the distributions, then we instead calculate

$$p_f(x_t^i|M, x_{m+1:t-1}^i) = \delta \cdot p_\theta(x_t|M, \tilde{x}_{m+1:t-1})^{1-\alpha} \tag{7}$$

without interpolation, where δ is a penalty term that disincentivizes selecting an uninterpolated draft for the next timestep. This approach is the backbone of Superposed Decoding (Equation 2) and allows us to create context-aware next-token distributions for *each* draft with only *one* inference pass.

3.3 Superposed Decoding Semantically Approximates Beam Search

Superposed Decoding relies on the ability of the underlying model to preserve the linear relationship between $\tilde{x}_t$ and its component vectors z_{t-1}^i [22]. More formally, if $\tilde{x}_t$ is the input to the LM f_θ, then $f_\theta(\tilde{x}_t) = \sum_{i=1}^k \gamma_i \cdot f_\theta(z_{t-1}^i)$ should also be true (γ_i defaults to draft probability in SPD). As long as this holds, the combination of a superposed embedding and n-gram smoothing should allow us to generate completions that resemble those from methods such as Beam Search.

We test this linearity by computing the cosine similarity between a set of superposed embeddings $\{\tilde{x}\}$ and the linear combination of their component embeddings across 20 timesteps on Llama-2-7B. At each timestep, we first use Beam Search to generate tokens for three beams. We then manually input the superposed embedding of the three tokens into a model using Superposed Decoding . Finally, we measure the alignment between $f_\theta(\tilde{x}_t)$ and $\sum_{i=1}^k \gamma_i \cdot f_\theta(z_{t-1}^i)$ using cosine similarity $\cos(a, b)$ as:

$$\cos(f_\theta(\tilde{x}_t), \sum_{i=1}^k \gamma_i \cdot f_\theta(z_{t-1}^i)) \tag{8}$$

We compute the cosine similarities for ten randomly sampled batches of ten prefixes each from the OpenWebText training split and plot the mean cosine similarities across batches, as well as the standard deviation (Figure 3). We find that Llama-2-7B is sufficiently linear up to 10 timesteps across all layers. However, this linearity is imperfect, and Superposed Decoding and Beam Search eventually diverge over time. Owing to this, we identify 10 timesteps as the optimal generation length. We also show how linearity changes through the layers within each timestep in Appendix H.

4 Experiments

We evaluate Superposed Decoding by analyzing generation quality, factuality, and latency. We demonstrate that Superposed Decoding improves over Nucleus Sampling and Greedy Decoding in generation perplexity (Section 4.1), fact-based benchmarks (Section 4.2), and wall-clock time (Section 4.3). Finally, we conduct a user study highlighting that users prefer Superposed Decoding over Nucleus Sampling in a compute-normalized setting (Section 4.4). We also include qualitative examples of Superposed Decoding generations in Figure 4, with more in Appendix G.

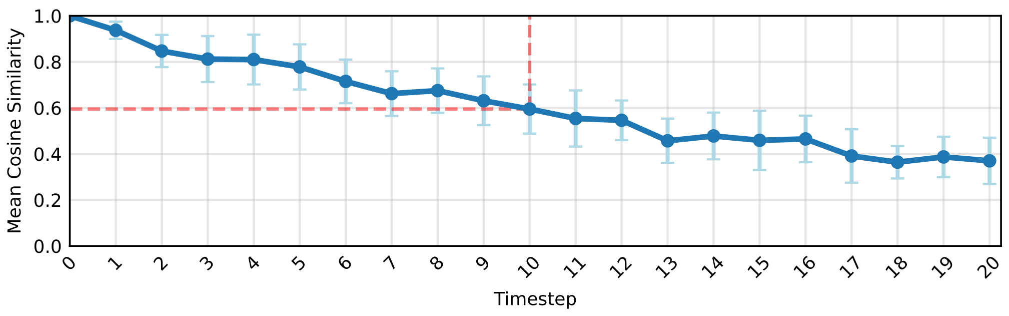
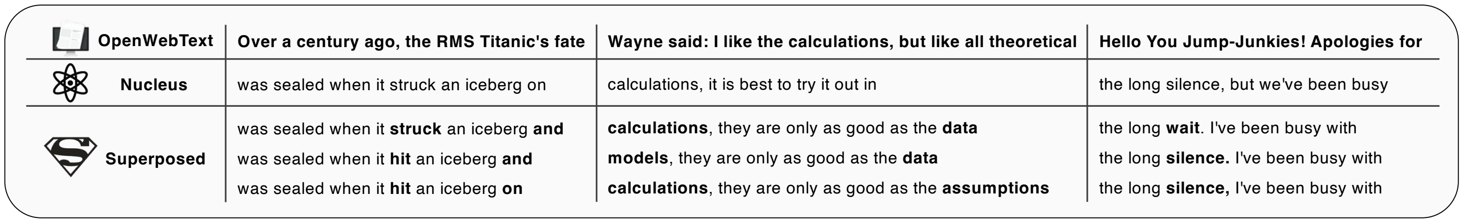


Figure 3: Llama-2-7B maintains the linear relationship between superposed embeddings and the component token embeddings, with mean cosine similarity ≥ 0.6 for the first 10 timesteps.



OpenWebText	Over a century ago, the RMS Titanic's fate	Wayne said: I like the calculations, but like all theoretical	Hello You Jump-Junkies! Apologies for
Nucleus	was sealed when it struck an iceberg on	calculations, it is best to try it out in	the long silence, but we've been busy
Superposed	was sealed when it **struck** an iceberg **and**	**calculations**, they are only as good as the **data**	the long **wait**. I've been busy with
	was sealed when it **hit** an iceberg **and**	**models**, they are only as good as the **data**	the long **silence.** I've been busy with
	was sealed when it **hit** an iceberg **on**	**calculations**, they are only as good as the **assumptions**	the long **silence,** I've been busy with

Figure 4: Qualitative text generations in a compute-normalized setting for Superposed Decoding and Nucleus Sampling with prefixes sampled from OpenWebText. See Appendix G for more.

We implement Superposed Decoding on the base pre-trained version of Llama-2-7B [41] for the majority of our experiments, running on a single A40 GPU. We do not retrain or fine-tune any part of the model. For perplexity evaluations, we use Llama-2-70B over eight A40 GPUs.

For n-gram interpolation, we construct n-gram models using 200,000 documents (roughly 200 million tokens) randomly sampled from the RedPajama dataset, an open-source replication of Llama-2's training dataset [8]. We represent each n-gram model internally as a frequency table, storing each unique n-gram and its corresponding count. This enables faster lookup and is the basis behind the compute reduction that we offer. Our n-gram models require approximately 14 GB of disk storage overall, split 57 MB, 433 MB, 2.15 GB, 4.7 GB, and 6.8 GB for $n = 2$ to 6. While we interpolate up to $n = 6$ in this work, we note that in practice the benefits of n-gram interpolation saturate past $n = 4$, suggesting that the number of n-gram models can be lowered to reduce storage (Appendix C).

4.1 Generation Quality

We test generation quality on OpenWebText's test split [14], which consists of 5,000 web-scraped documents. For each document, we use the first 15 tokens as the prefix and generate $k = 3$ drafts of 10 tokens. We decide to focus on short generations because drafts are typically a short-form task. Before running experiments, we identify the optimal interpolation weight α and temperature τ by optimizing for the lowest average perplexity across three drafts on the validation split. We list the specific hyperparameter values that we use in Appendix B.

We only evaluate one draft of the baselines in order to match the compute used by Superposed Decoding. We find that while Nucleus Sampling and Greedy Decoding outperform Superposed Decoding on a per-draft basis, the average best perplexity from Superposed Decoding is **5% lower** than that of Nucleus Sampling. From the point of view of a user, this means for each prefix, at least one draft can be expected to be on par with Nucleus Sampling and Greedy Decoding. The other drafts all come free of additional compute. In the following Sections 4.2 and 4.4, we show that this diversity is beneficial for both factuality and human preference.

Table 1: Superposed Decoding is natural-sounding and has lower average perplexity than Nucleus Sampling, which typically approximates human writing.

	Nucleus	Beam/Greedy	N-Gram	Superposed Decoding			
Draft #	-	-	-	1	2	3	Best
Avg Perplexity	5.17	3.77	152.75	5.03	7.97	10.05	4.63

4.2 Fact-Based Evaluation

Next, we test the ability of Superposed Decoding to generate not only coherent but also accurate completions. We assess this using exact match precision (P@k) for $k = 1, 2, 3$ on TriviaQA [23] and Natural Questions [28], two common benchmarks for short answer generations. We decide not to use multiple choice datasets such as MMLU [16] and OpenBookQA [32] because multiple choice questions are trivial when using multiple drafts, unfairly advantaging Superposed Decoding.

In Figure 5, we show that Superposed Decoding outperforms Nucleus Sampling and Beam Search at P@1 in a zero-shot setting, with three drafts providing accuracy gains of up to 2.72% in TriviaQA and 1.69% in Natural Questions. These results demonstrate that Superposed Decoding substantially increases the likelihood of getting a factually correct generation in addition to one that is coherent.

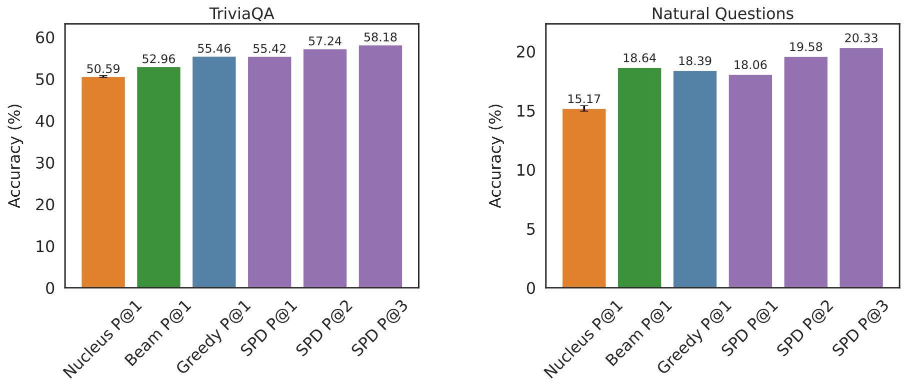


Figure 5: Superposed Decoding is as accurate as Greedy Decoding for P@1 and increases the fact-based coverage using multiple drafts (P@2,3) on TriviaQA (**left**) and Natural Questions (**right**).

4.3 Latency

Superposed Decoding presents a significant *theoretical* reduction in latency, but it is important to investigate how well these savings translate to real-world settings. In Figure 6, we show that the only additional cost incurred by Superposed Decoding is due to dictionary lookups, barring which Superposed Decoding has near-constant compute cost. Even so, the total cost of Superposed Decoding is significantly lower than other decoding methods, with Superposed Decoding $\mathbf{2.44}\times$ faster on three drafts and **3.54**$\times$ faster on eight compared to Nucleus Sampling, the second fastest.

It is important to note that Superposed Decoding results are obtained using unoptimized code. Our n-gram models are implemented using single-threaded lookup on Python dictionaries, and we do not cache any lookup results. This has an enormous, visible impact. In addition, Liu et al. [30] propose the use of suffix arrays for n-gram models, allowing a near-instantaneous lookup of arbitrary-length n-grams in trillion-token corpora. These techniques open up multiple avenues for additional speedup.

4.4 Human Evaluation

While perplexity and factuality are good proxies for textual quality, real human evaluation remains the best measure of coherency. Indeed, perplexity is an imperfect evaluator of long-form language modelling [43] while factuality metrics are inherently biased towards Greedy Decoding methods [40]. Consequently, to verify our findings, we conduct a random study asking human respondents to rank Superposed Decoding and Nucleus Sampling generations based on how well they complete a provided prefix. We compare against Nucleus Sampling because n-gram distributions from Nucleus Sampling are demonstrably the closest to those of humans [17].

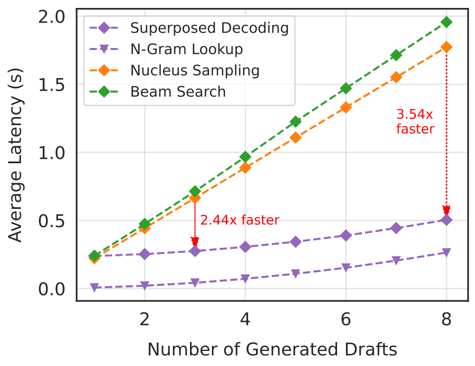


Figure 6: Average wall clock time to generate 10 token long drafts from a 15 token prefix on OpenWebText. Superposed Decoding is significantly faster for all $k > 1$, with n-gram lookup costs being a major factor for $k \geq 4$, which can be optimized further.

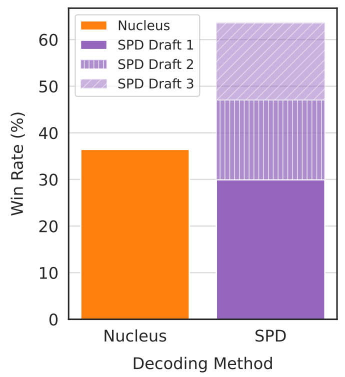


Figure 7: Drafts are ordered by the probability they obtain during generation using SPD, which wins over Nucleus Sampling in a compute-normalized setting.

Setup. We conduct our study using Amazon Mechanical Turk [15]. First, we randomly sample 1,000 prefixes from the OpenWebText test set, truncating to the first 15 tokens as in Section 4.1. Next, we manually remove prefixes that are grammatically incorrect, such as text from website toolbars. From the remaining prefixes, we generate three Superposed Decoding drafts and one Nucleus Sampling draft in a compute-normalized setting, randomizing their order. Finally, we filter out prefixes with duplicate or unparseable generations (e.g. emojis). After preprocessing, 707 prefixes are left.

It is important to note that one Nucleus Sampling generation is 20% less expensive than three Superposed Decoding drafts, shown in Section 4.3. However, two Nucleus Sampling drafts disadvantages three Superposed Decoding drafts by 60%. Therefore, we decide to run our main survey using the first setup, which is closest to equal compute, but also conduct smaller surveys in 2v3 and 1v1 settings.

Results. We define a Superposed Decoding "win" as when one of the Superposed Decoding drafts is ranked first. As shown in Figure 7, we find that Superposed Decoding generations are preferred approximately $\mathbf{63.6}\%$ of the time . If every Superposed Decoding draft was consistently the same quality as Nucleus Sampling, then we would expect the rankings to be uniform, resulting in a 75% win rate. However, this is not the case, suggesting that Nucleus Sampling is more reliable than any *individual* draft, but the *aggregate* of drafts provides a diversity that is highly valuable to users.

We run the two smaller-scale iterations of the study with 100 prefixes each. In the 2v3 setting, we ask users to rank two nucleus drafts and three superposed drafts to investigate whether the benefits of Superposed Decoding persist even when compute favors Nucleus Sampling. In the 1v1 setting, users choose between one nucleus draft and the lowest perplexity superposed draft, removing any potential bias caused by unequal numbers of drafts. As shown in Figure 12, in both situations, users still prefer Superposed Decoding over Nucleus Sampling $\mathbf{60.6}\%$ and $\mathbf{51.4}\%$ of the time respectively. While the surveys have higher variance due to their small size, they cement Superposed Decoding as a strong alternative to Nucleus Sampling. We show details on the additional studies in Appendix F.1 and present our survey page in Figure 13.

5 Further Analysis and Ablations

5.1 Results on Mistral 7B

We also implement Superposed Decoding on pre-trained Mistral 7B [21] and conduct the same experiment as Section 4.1 with one change. In Section 4.1, it was possible to evaluate the perplexity of the 10 generated tokens exactly because the generating model (Llama-2-7B) and the evaluation model (Llama-2-70B) use the same tokenization. This is not the case for Mistral 7B and Llama-2-70B. Consequently, we calculate perplexity for Mistral 7B over all tokens but the first five, ensuring that the entire generation is included. While this approach also includes several tokens from the initial prefix in perplexity calculations, they are redundant across generations, thus preserving relative ordering.

Table 2 compares the resulting perplexities. Like with Llama-2-7B, the average best draft perplexity using Superposed Decoding is lower than that of Nucleus Sampling, demonstrating that Superposed Decoding is adaptable to other language models out of the box.

5.2 Textual Analysis

Next, we extensively investigate the diversity and repetition of Superposed Decoding in order to better understand its properties.

Repetition within Generations. Repetition is a well-documented issue in all decoding methods but is most prevalent when decoding greedily [17]. We explore to what extent, if any, Superposed Decoding degenerates as well. To measure repetition, we calculate the proportion of unique unigrams, bigrams, and trigrams in each generation. The lower the uniqueness, the higher the repetition. In

Table 2: Superposed Decoding generalizes across language models like Mistral 7B as shown here with similar results on coherency, as Llama-2-7B, when evaluated using Llama-2-70B.

	Nucleus	Superposed Decoding			
Draft #	-	1	2	3	Best
Avg Perplexity	11.42	11.34	12.74	13.63	10.87

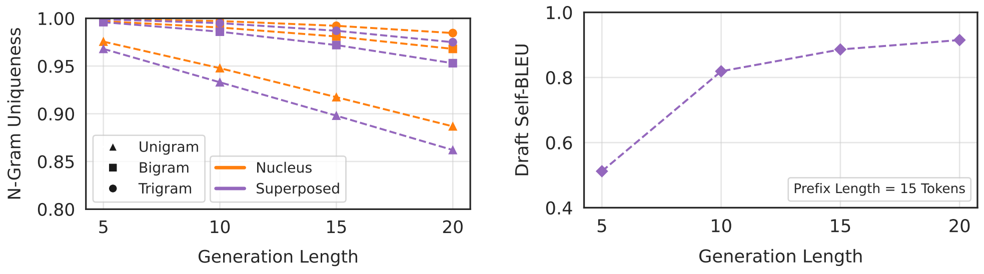


Figure 8: **Left:** Minimal difference in repetition for Superposed Decoding and Nucleus Sampling in short generations. **Right:** Generation length is an effective knob for adjusting the diversity of drafts. Both experiments use a prefix length of 15 tokens.

Figure 8, we plot results for Superposed Decoding and Nucleus Sampling for several generation lengths. Because drafting is usually a short-form task, we only consider generation lengths up to 20 tokens. We find that Superposed Decoding does not repeat significantly more than Nucleus Sampling in this range, suggesting that degeneration is not an issue in most use cases. This is especially true for bigrams and trigrams, which are more reliable indicators of degeneration than unigrams. However, we qualitatively observe that repetitions become frequent after 100 tokens. To address this, we propose Superposed Decoding Resets, which we explain in Section 5.3.

Diversity across Drafts. To measure diversity, we apply Self-BLEU [44] on drafts and then calculate the average across prefixes. We compute Self-BLEU by calculating the BLEU score of each draft with the other $k - 1$ drafts as references. Hence, a low Self-BLEU signifies high diversity, while a high Self-BLEU suggests low diversity. After calculating Self-BLEU for varying prefix lengths, generation lengths, and numbers of drafts, we find that generation length is the most impactful hyperparameter for diverse drafts. We plot Self-BLEU against generation length in Figure 8, demonstrating that shorter generations significantly increase diversity.

5.3 Superposed Decoding Resets

To reduce long-form repetition in Superposed Decoding, we propose resetting superposed tokens every s timesteps, where s is a user-selected hyperparameter. On each reset step we sample one of the k drafts and restart draft generation. Resetting helps Superposed Decoding escape repetitive loops while reestablishing linearity, which Figure 3 shows to deteriorate over time. This is similar to a user selecting one of the k drafts while typing. We find that resetting noticeably improves long-form generation quality (examples in Appendix I) and leave further investigation for future work.

Further, we also ablate on prefix length, generation length, and number of drafts. Figures 9, 10, and 11 in Appendix E highlight that Superposed Decoding is robust to all three hyperparameters, with lower perplexity than Nucleus Sampling in nearly all settings tested.

6 Discussion and Conclusion

While we demonstrate that Superposed Decoding is an effective technique for multiple draft generation, Superposed Decoding is limited by the quality of the n-gram models used, which are essential for maintaining coherence. In addition, while Superposed Decoding drafts are *syntactically* diverse, they are not often *semantically* diverse. We suspect that the orthogonality of token embeddings discovered by Jiang et al. [22] is a potential solution. While our initial experiments did not show diversity gains, we believe that orthogonality is promising and is a logical next step for future work.

In conclusion, we present **Superposed Decoding**, a novel decoding method that superposes token embeddings to generate multiple generations at the cost of one. We demonstrate that Superposed Decoding improves on Nucleus Sampling in terms of generation quality and human preference. The plurality of choices from Superposed Decoding also leads to better performance on common benchmarks. We envision that the latency reduction from Superposed Decoding will make it practical to apply large pre-trained language models on drafting tasks where compute is often a barrier for deployability. Finally, Superposed Decoding can be deployed in messaging applications using n-grams personalized to each user, where the number of n-grams can be reduced to save storage without compromising performance.

Acknowledgments

We are grateful to Jiacheng (Gary) Liu for his assistance in using infinigram for initial experimentation. We also thank Alisa Liu, Vivek Ramanujan, Reza Salehi, Prateek Jain, and Yejin Choi for their feedback. We also acknowledge the computing resources and support from HYAK at the University of Washington, FAS RC at Harvard University & Kempner Institute, Gradient AI and research GCP credit award from Google Cloud and Google Research. Ali Farhadi acknowledges funding from the NSF awards IIS 1652052, IIS 17303166, DARPA N66001-19-2-4031, DARPA W911NF-15-1-0543, and gifts from Allen Institute for Artificial Intelligence and Google.

References

[1] M. Adnan, A. Arunkumar, G. Jain, P. J. Nair, I. Soloveychik, and P. Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference, 2024.

[2] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[4] T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads, 2024.

[5] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[6] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[7] M. X. Chen, B. N. Lee, G. Bansal, Y. Cao, S. Zhang, J. Lu, J. Tsay, Y. Wang, A. M. Dai, Z. Chen, T. Sohn, and Y. Wu. Gmail smart compose: Real-time assisted writing, 2019.

[8] T. Computer. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

[9] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale, 2022.

[10] Devvrit, S. Kudugunta, A. Kusupati, T. Dettmers, K. Chen, I. Dhillon, Y. Tsvetkov, H. Hajishirzi, S. Kakade, A. Farhadi, P. Jain, et al. Matformer: Nested transformer for elastic inference. *arXiv preprint arXiv:2310.07707*, 2023.

[11] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation, 2018.

[12] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023.

[13] F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve. Better & faster large language models via multi-token prediction, 2024.

[14] A. Gokaslan and V. Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[15] V. Harinarayan, A. Rajaraman, and A. Ranganathan. Hybrid machine/human computing arrangement. United States Patent.

[16] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding, 2021.

[17] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration, 2020.

[18] W. Huang, Y. Liu, H. Qin, Y. Li, S. Zhang, X. Liu, M. Magno, and X. Qi. Billm: Pushing the limit of post-training quantization for llms, 2024.

[19] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt. CodeSearchNet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019.

[20] B. Institute. 9 ux best practice design patterns for autocomplete suggestions. 2023. URL https://baymard.com/blog/autocomplete-design. Accessed: 2024-05-18.

[21] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.

[22] Y. Jiang, G. Rajendran, P. Ravikumar, B. Aragam, and V. Veitch. On the origins of linear representations in large language models, 2024.

[23] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017.

[24] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[25] S. Kim, K. Mangalam, S. Moon, J. Malik, M. W. Mahoney, A. Gholami, and K. Keutzer. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36, 2024.

[26] N. Kitaev, Łukasz Kaiser, and A. Levskaya. Reformer: The efficient transformer, 2020.

[27] A. Kusupati, V. Ramanujan, R. Somani, M. Wortsman, P. Jain, S. Kakade, and A. Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, pages 5544–5555. PMLR, 2020.

[28] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[29] Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding, 2023.

[30] J. Liu, S. Min, L. Zettlemoyer, Y. Choi, and H. Hajishirzi. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. *arXiv preprint arXiv:2401.17377*, 2024.

[31] S. Ma, H. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, and F. Wei. The era of 1-bit llms: All large language models are in 1.58 bits, 2024.

[32] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018.

[33] e. a. OpenAI. Gpt-4 technical report, 2024.

[34] K. Park, Y. J. Choe, and V. Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

[35] W. Qi, Y. Yan, Y. Gong, D. Liu, N. Duan, J. Chen, R. Zhang, and M. Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training, 2020.

[36] N. Shazeer. Fast transformer decoding: One write-head is all you need, 2019.

[37] W. Shi, J. Michael, S. Gururangan, and L. Zettlemoyer. Nearest neighbor zero-shot inference. In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:249152130.

[38] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter. A simple and effective pruning approach for large language models, 2024.

[39] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[40] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.

[41] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[42] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity, 2020.

[43] Q. Zhang, D. Ram, C. Hawkins, S. Zha, and T. Zhao. Efficient long-range transformers: You need to attend more, but not necessarily at every layer, 2023.

[44] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu. Texygen: A benchmarking platform for text generation models. *SIGIR*, 2018.

A Superposed Decoding Pseudo-Code

Algorithm 1 Superposed Decoding algorithm generating k drafts.

```
 1: Input: Prefix $M = (x_1, \dots, x_m)$, generation length $G$
 2: Initialize $t \leftarrow m + 1$
 3: Initialize $D \leftarrow \{M\}_k$                                  ▷ Draft store
 4: while $t \neq m + G$ do
 5:     if $t = m + 1$ then                                           ▷ First timestep
 6:         $p_\theta \leftarrow f_\theta(M)$
 7:         for $i = 1$ to $k$ do
 8:             $x^i_{m+1} \leftarrow \text{Largest}(p_\theta, i)$          ▷ Take the $i^{\text{th}}$ most probable token
 9:             $D_i \leftarrow \{D_i \cup x^i_{m+1}\}$
10:         end for
11:         $\tilde{x}_{m+1} \leftarrow \text{Superpose}(x^1_{m+1}, \dots, x^k_{m+1})$    ▷ Superpose the selected tokens
12:     else
13:         $p_\theta \leftarrow f_\theta(M, \tilde{x}_{m+1:t-1})$
14:         $p_\theta \leftarrow \text{KeepTopK}(p_\theta)$                    ▷ Keep only the top $k$ tokens
15:         for $i = 1$ to $k$ do
16:             $p_{\text{ngram}} \leftarrow \text{N-Gram}(D_i)$            ▷ Calculate the n-gram distribution of draft $i$
17:             $p_f \leftarrow \text{Interpolate}(p_\theta, p_{\text{ngram}})$
18:             for $x^i_t \in p_f$ do
19:                 $D_{tmp} \leftarrow \{D_i \cup x^i_t\}$               ▷ Create new draft option
20:                 $D \leftarrow \{D \cup D_{tmp}\}$                  ▷ Add draft option to draft store
21:             end for
22:         end for
23:         $D \leftarrow \text{TopK}(D)$                              ▷ Keep the top $k$ drafts
24:         $\tilde{x}_t \leftarrow \text{Superpose}(x^1_t, \dots, x^k_t)$      ▷ Superpose most recent tokens from drafts
25:     end if
26:     $t \leftarrow t + 1$
27: end while
28: return $D$
```

B Hyperparameter Choices

α and τ. We find that the best performing hyperparameters for coherent generation on Llama-2-7B are $\alpha = 0.54$ and $\tau = 0.06$, which we use in Section 4.1, 4.3, and 4.4. The only exception is in Section 4.2, where we disable n-gram smoothing.

N-Gram Interpolation. The weights we use for interpolation are $[0.01, 0.04, 0.15, 0.18, 0.12]$ for $n = 2$ to $n = 6$.

C Perplexity Evaluation with Fewer N-Grams

We evaluate perplexity for Superposed Decoding on the OpenWebText test split by interpolating only n-grams from $n = 2$ to $n = 4$, requiring only 2.64 GB of storage. We use the same n-gram interpolation weights as in Section 3 but retune α and τ to 0.55 and 0.1 respectively. In Table 3, we show that the perplexity of the average best generation is almost identical to the average Nucleus Sampling perplexity. This demonstrates that Superposed Decoding can be effectively implemented with minimal storage overhead in storage-constrained devices like smartphones.

D Standard Deviations for Perplexity Evaluation

We calculate the standard deviations across the 5000 OpenWebText prefixes used for evaluation for Llama-2-7B and Mistral 7B and show them below in Tables 4 and 5. We note that Superposed Decoding has higher standard deviation than Nucleus Sampling and Beam Search when implemented

Table 3: Superposed Decoding is highly competitive with Nucleus Sampling even when only n-grams from $n = 2$ to $n = 4$ are interpolated to calculate p_{ngram}.

	Nucleus	Beam/Greedy	N-Gram	Superposed Decoding			
Draft #	-	-	-	1	2	3	Best
Avg Perplexity	5.17	3.77	152.75	5.83	8.38	9.97	5.18

on Llama-2-7B but that the standard deviations equalize on Mistral 7B. Despite this high variance, human evaluation results in Section 4.4 affirm that Superposed Decoding is consistently competitive in coherency.

Table 4: Standard deviation of Llama-2-7B generation perplexity calculated on OpenWebText test split in Section 4.1.

	Nucleus	Beam/Greedy	N-Gram	Superposed Decoding			
Draft #	-	-	-	1	2	3	Best
Avg Perplexity	4.82	3.31	305.00	8.90	14.44	20.58	8.24

Table 5: Standard deviation of Mistral 7B generation perplexity calculated on OpenWebText test split in Section 5.1.

	Nucleus	Superposed Decoding			
Draft #	-	1	2	3	Best
Avg Perplexity	10.19	9.92	11.53	12.53	9.38

E Further Ablations

In this section, we present further ablations studying the impact of number of drafts, prefix length, and generation length on Superposed Decoding perplexity (average best). We find that for scenarios such that the number of drafts $k \geq 3$, Superposed Decoding consistently outperforms nucleus sampling (Figure 9). This is generally expected as the higher the number of drafts, the higher the likelihood that at least one draft is good quality. Similarly, we find that Superposed Decoding is robust to prefix length and generation length. Indeed, the patterns of Superposed Decoding given longer prefix and generation lengths mimic those of Nucleus Sampling in the same situation, with consistently similar perplexities suggesting good performance (Figures 10, 11).

F Human Evaluation

F.1 Additional Results

We run the same experiment as in Section 4.4 two more times at a smaller scale. First, we compare two Nucleus Sampling drafts and three Superposed Decoding drafts. This scenario results in a significant compute advantage for Nucleus Sampling. Still, we find that Superposed Decoding wins the majority of the time, with a win rate of 60.6%. However, this is slightly lower than the 64% win rate in the main study.

Next, we compare one Nucleus Sampling draft and the lowest perplexity Superposed Decoding draft of three. This mimics the perplexity evaluation in Section 4.1 but in a human setting. We find that Superposed Decoding is still ranked first the majority of the time, with a 51% win rate.

For cost efficiency, we run both studies on only 100 random prompts, with five responses per prompt.

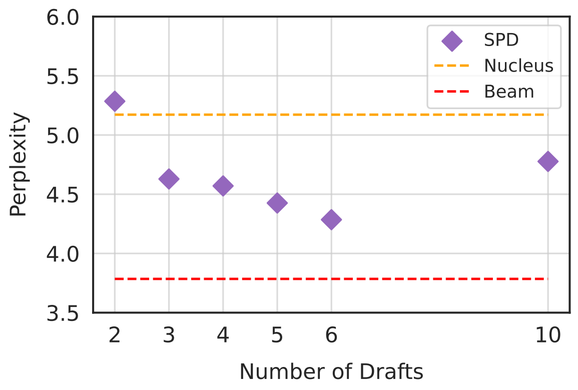


Figure 9: Average perplexity of one draft of Nucleus Sampling and Beam Search compared against average best perplexity of SPD using drafts $k = \{2, 3, 4, 5, 6, 10\}$ on OpenWebText. SPD outperforms Nucleus Sampling for all values of k tested.

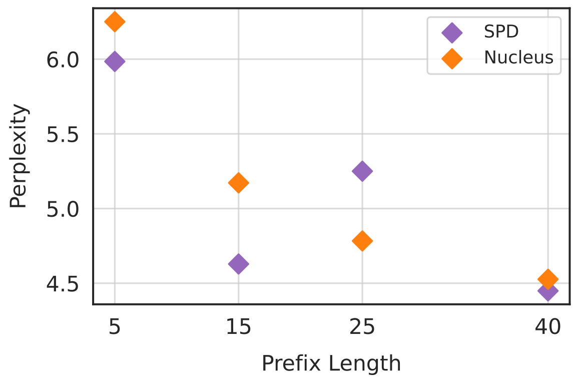


Figure 10: Average perplexity of one draft of Nucleus Sampling compared against average best perplexity of SPD for prefix lengths 5, 15, 25, 40. We find that SPD and Nucleus Sampling consistently have similar perplexities.

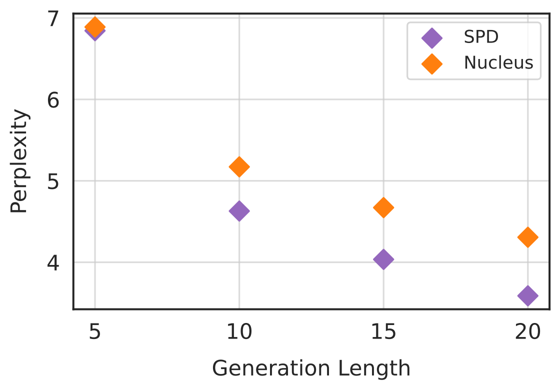


Figure 11: Average perplexity of one draft of Nucleus Sampling compared against average best perplexity of SPD for generation lengths 5, 10, 15, 20. We find that SPD is robust for a wide range of short form generation lengths.

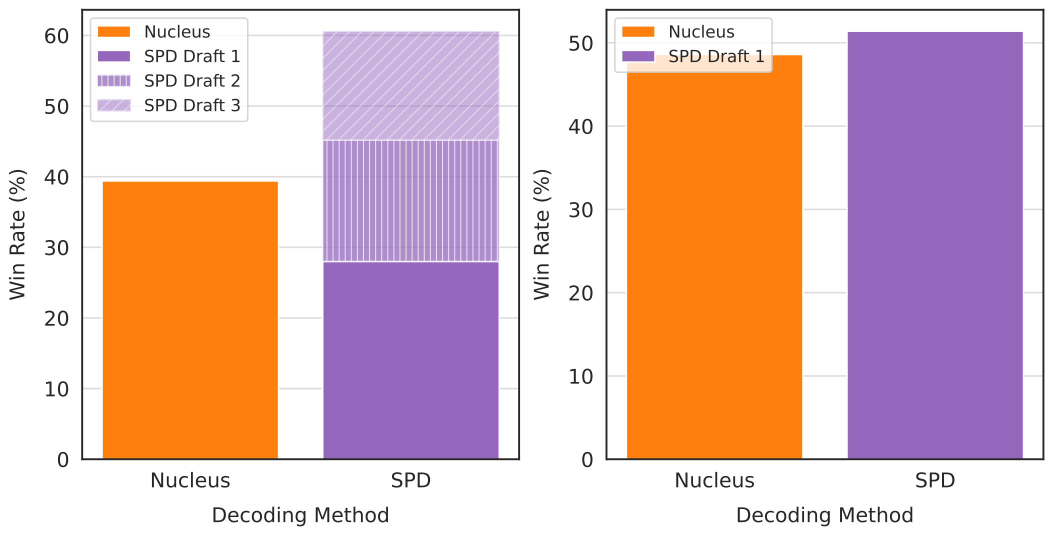


Figure 12: **Left**: Study results comparing two Nucleus Sampling drafts and three Superposed Decoding drafts. **Right**: Study results comparing one Nucleus Sampling draft & the lowest perplexity Superposed Decoding draft out of three generations. Note that uncertainty is higher for both iterations due to smaller sample size.

F.2 Mechanical Turk Survey

We show a screenshot of our study page in Figure 13. We compensate workers at an approximate rate of $15 per hour.

G Example Generations

In Figure 14, we show 12 additional example generations comparing Superposed Decoding and Nucleus Sampling in a compute-normalized setting. Prefixes are sampled from OpenWebText and truncated to the first 15 tokens, and each generated completion is 10 tokens in length.

H Layer-Wise Linearity

In Figure 15, we show the cosine similarity between superposed embeddings $\{\tilde{x}\}$ and the linear combination of the component embeddings of three tokens, after each layer within the first five timesteps using Llama-2-7B. Within each timestep, superposed embeddings pass through an initial layer (Layer 1) that embeds token indices and weights the resulting embeddings using corresponding token weights. The weighted embedings then pass through 32 transformer blocks (Layers 2-33), then through a final linear projection layer (Layer 34) that produces the output superposed embedding of that timestep.

I Resetting

Here, we show examples of the Superposed Decoding Resetting technique described in Section 5.3 for generations of 200 tokens using prefixes from OpenWebText. Superposed Decoding Resetting is significantly less repetitive than non-reset Superposed Decoding. However, Nucleus Sampling, which is also shown, is qualitatively still the best performing decoding method in long form settings.

Prompt #1

Superposed Decoding with Resetting

Sydney architect Rana Abboud won a NAWIC Scholarship for 2019. Sydney-based architect and interior designer Rana Abboud has been awarded a $10,000 scholarship to study a Masters of Architecture at the University of New South Wales. The scholarship, awarded by the National Association of Women in Construction (NAWIC) is designed to help women in the construction industry to advance their careers. Rana Abboud, a graduate of the University of Sydney, is the founder of Rana Abboud & Associates. "I am honoured to be awarded this scholarship. I am passionate about design built for people and the environment. I am excited to be able to further my studies and continue to develop my skills and knowledge," she said. The NAWIC Scholarship is awarded to women who are studying or working in the construction industry. "The NAWIC Scholarship is a great opportunity for women to further their care and

Superposed Decoding without Resetting

Sydney architect Rana Abboud won a NAWIC Scholarship in 2017. The National Association of Women in Construction (NAWIC) has announced the winners of its 2017 scholarships. The NAWIC Scholarship Program is designed to encourage women to pursue careers in the construction industry. The 2017 scholarship recipients are: Rana Abboud, a student at the University of New South Wales, who is studying a Bachelor of Architecture. Megan Brennan, a student at the University of New South Wales, who is studying a Bachelor of Construction Management. Katee Cox, a student at the University of New South Wales, who is studying a Bachelor of Construction Management. Kate Dunn, a student at the University of New South Wales, who is studying a Bachelor of Construction Management. Kate Foster, a student at the University of New South Wales, who

Nucleus Sampling

Sydney architect Rana Abboud won a NAWIC Scholarship of $1000 for 'Significant Contribution to Career Growth'. Sydney architect Rana Abboud has won the National Association of Women in Construction (NAWIC) NSW/ACT Scholarship for 2017 for 'Significant Contribution to Career Growth'. She has also just been accepted to undertake a Master of Science in Architecture with Higher Degree by Research at Sydney University. The scholarship is one of eight awards presented by NAWIC to women to further their studies in the construction industry in Australia. "As a working mother, I am very grateful for the support I am receiving to further my professional and academic development," Abboud said. Rana Abboud (middle) with associates (L-R) Tara Abdul-Khalek and Aditi Rana. Abboud is a director of Pal Architecture in Alexandria,

Prompt # 2

Superposed Decoding with Resetting

NEW DELHI: Here's something ayurveda gurus have been telling for years. A new study has found that a combination of three herbs - including ashwagandha, guduchi and shatavari - can help in the treatment of diabetes. "The study was conducted to evaluate the efficacy and safety of a poly-herbal formulation containing three herbs - Ashwagandha (Withania somnifera), Guduchi (Tinospora cordifolia), and Shatavari (Asparagus racemosus) in inhibiting the progression of diabetes in alloxan-induced diabetic rats" said the study published in the journal 'Journal of Ayurveda and Integrative Medicine'. The study was conducted by research team from the Department of Pharmacology, Dr D. Patil College of Pharmacy, Pune, Maharashtra, and the Department of Pharm

Superposed Decoding without Resetting

NEW DELHI: Here's something ayurveda gurus have been saying for years: Eat your way to health. A new study has found that a diet rich in fruits and vegetables can help reduce the risk of developing type 2 diabetes. The study, published in the journal BMJ Open, found that a diet rich in fruits and vegetables can help reduce the risk of developing type 2 diabetes. The study, which was conducted by researchers from the University of Oxford, found that a diet rich in fruits and vegetables can help prevent the risk of developing type 2 diabetes. The study, which was conducted by researchers from the University of Oxford, found that a diet rich in fruits and vegetables can help prevent the risk of developing type 2 diabetes. The study, which was conducted by researchers from the University of Oxford, found that a diet rich in fruits and vegetables can help prevent the risk

Nucleus Sampling

NEW DELHI: Here's something ayurveda gurus have been telling us about for years - eating seasonal fruits and vegetables helps stay healthy. Here are 7 foods which can keep you healthy during the summer. Pomegranate is a summer fruit and if you wish to have optimum health, it is necessary to eat this fruit. The fruit can be eaten in raw form, or cut into pieces and mixed with honey and taken twice a day. The skin of the fruit is itself a health tonic and can be chewed. It contains natural bleaching properties. The fruit can also be squeezed and then the pulp mixed with milk, cumin seeds, one teaspoonful of honey and a pinch of ginger and drank thrice daily. Beets are a popular root vegetable. Beetroot juice may help increase exercise endurance, potentially because it increases blood flow, said WebMD. The best fruits

Prompt # 3

Superposed Decoding with Resetting

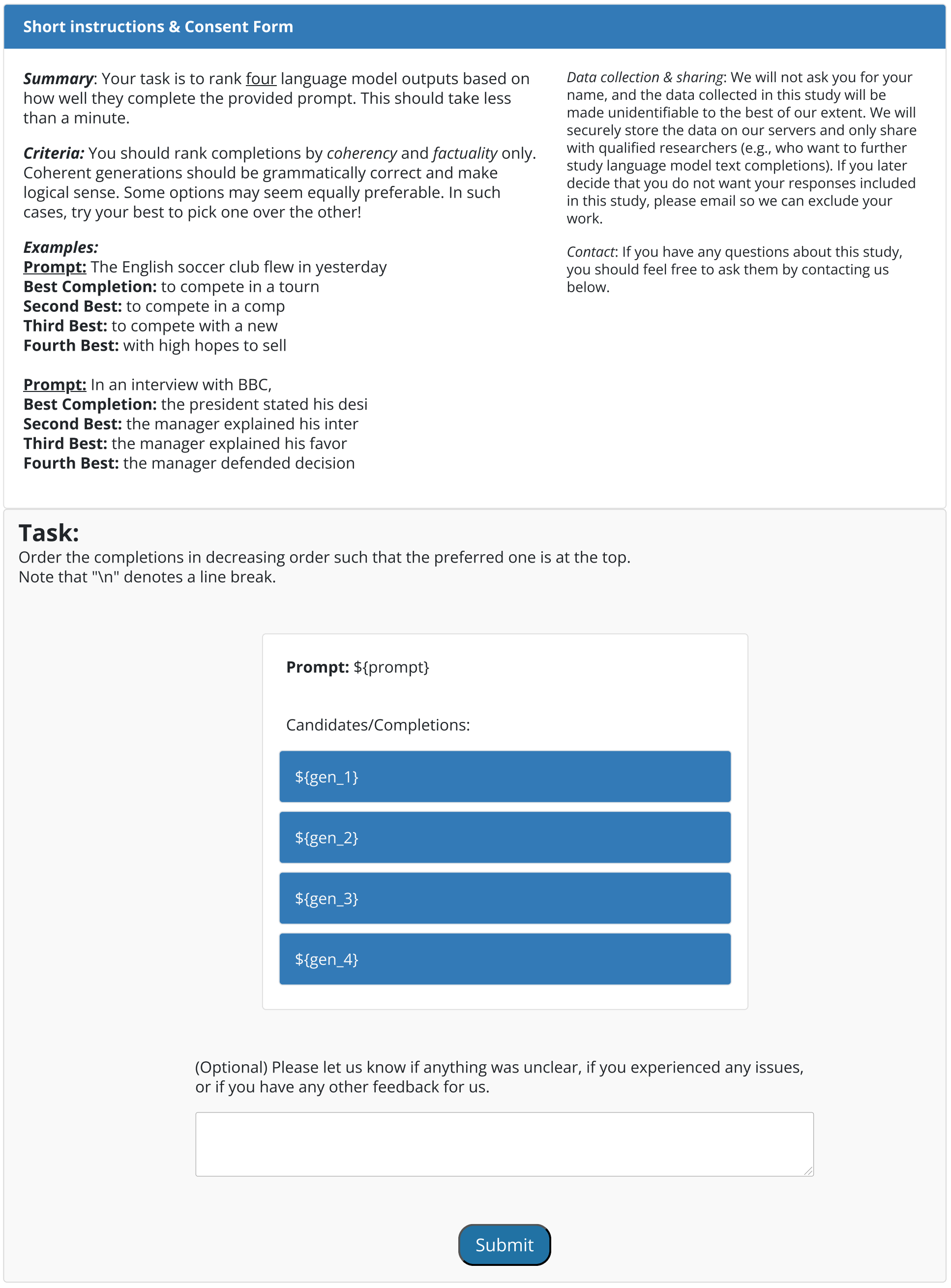
As part of a broad initiative to combat sexual harassment and assault, the University of California, Berkeley has launched a new website to help students, staff, and faculty report and respond to sexual harassment and assault. The new site, which went live on Monday, is the one-stop destination for all things related to sexual harassment and assault at UC Berkeley. It is the result of a collaboration between the Office of the Vice Chancellor of Student Affairs and the Office of the Vice Provost for Academic and Student Affairs. “This is a critical time for our campus and to come together to address sexual harassment and assault in our community,” said Vice Chancellor for Student Affairs, Steph Sutton-Wheeler, in a statement. “We are committed to providing a safe and inclusive environment for all members of our community and this new website is a critical step in that effort.” The site provides information on how to report sexual harassment and assault, as well as the for

Superposed Decoding without Resetting

As part of a broad initiative to combat sexual harassment and assault, the University of California, Berkeley, has launched a new website that provides information about the university's policies and procedures for reporting and responding to sexual harassment and sexual violence. The website, which is available at http://sexualharassment.berkeley.edu, is designed to help students, faculty and staff understand their rights and responsibilities under the university's policies and procedures. The website includes information about the university's policies and procedures for reporting and responding to sexual harassment and sexual violence, as well as information about the university's policies and procedures for investigating and responding to sexual harassment and sexual violence. The website also includes information about the university's policies and procedures for investigating and responding to sexual harassment and sexual violence, including information about the university's policies and procedures for investigating and responding to sexual harassment and sexual violence in the workplace.

Nucleus Sampling

As part of a broad initiative to combat sexual harassment and misconduct on campus, a task force consisting of faculty, staff, students, and other members of the campus community will present a draft report with recommendations at the Board of Trustees meeting on Friday, October 27, 2017. Under the leadership of Executive Director for Human Resources Jesse Bozeman, the task force was formed to address concerns raised at the end of the Spring 2017 semester, which included a Student Government Association resolution and a student-driven, faculty-approved course called "Sexual Misconduct: Consent and Culture," taught by Professor Nancy Cantalupo and co-sponsored by Campus Life, Student Affairs, and Academic Affairs. The task force is comprised of the following members: Diana Ballin, Senior Lecturer in the Department of Anthropology, Chair; Patricia Hopkins, Dean of Students; Kelly McSweeney



Short instructions & Consent Form

Summary: Your task is to rank four language model outputs based on how well they complete the provided prompt. This should take less than a minute.

Criteria: You should rank completions by *coherency* and *factuality* only. Coherent generations should be grammatically correct and make logical sense. Some options may seem equally preferable. In such cases, try your best to pick one over the other!

Examples:
Prompt: The English soccer club flew in yesterday
Best Completion: to compete in a tourn
Second Best: to compete in a comp
Third Best: to compete with a new
Fourth Best: with high hopes to sell

Prompt: In an interview with BBC,
Best Completion: the president stated his desi
Second Best: the manager explained his inter
Third Best: the manager explained his favor
Fourth Best: the manager defended decision

Data collection & sharing: We will not ask you for your name, and the data collected in this study will be made unidentifiable to the best of our extent. We will securely store the data on our servers and only share with qualified researchers (e.g., who want to further study language model text completions). If you later decide that you do not want your responses included in this study, please email so we can exclude your work.

Contact: If you have any questions about this study, you should feel free to ask them by contacting us below.

Task:
Order the completions in decreasing order such that the preferred one is at the top.
Note that "\n" denotes a line break.

Prompt: ${prompt}

Candidates/Completions:

${gen_1}

${gen_2}

${gen_3}

${gen_4}

(Optional) Please let us know if anything was unclear, if you experienced any issues, or if you have any other feedback for us.

Figure 13: A screenshot of our Mechanical Turk template, which presents the respondent with a task summary, objective criteria for ranking generations, and two examples.

OpenWebText	**When I worked as a scout for the Carolina Panthers in the**	**As a kid, I played a game called The Sims, a**	**Over the next few years, NASA plans to get back to launching**
Nucleus	1990s, I would often	simulation of life in a suburban neighborhood. I	rockets from Cape Canaveral, Florida
Superposed	1990s, I was **always**	game where you **can** create your character and live **out**	astronauts from U.S. soil**.**
	1990s, I was **a**	game where you **could** create a character and live **out**	astronauts from U.S. soil**,**
	1990s, I was **responsible**	game where you **can** create your character and live **their**	astronauts from U.S. soil **for**

OpenWebText	**"Do you want an extended warranty with that?"**	**Synopsis A library for running tasks(jobs) on**	**Just this Tuesday I wrote calling on MUD to present a**
Nucleus	"No, thanks. I'm sure it	a cluster of nodes. It provides a simple API	plan to the community to resolve the situation with the
Superposed	"No, I don't **want an** extended	a cluster of **machines**. ## Installation	plan to the **public** for the future of the **M**
	"No, I don't **think an** extended	a cluster of **computers**. ## Installation	plan to the **public** for the future of the **site**
	"No, I don't **want any** extended	a cluster of **nodes**. ## Installation	plan to the **community** for the future of the **M**

OpenWebText	**As an undergraduate, I took a course titled "The Log**	**"Guess what? I have flaws. What are they?**	**In this gluten-free riff on banana cream pie**
Nucleus	ic of Scientific Discovery" with Karl Po	I'm a little too bossy,	, the banana flavor comes from a hom
Superposed	ic of Scientific Discovery" **with** the late	I'm not **sure**. I'm **working**	, the bananas are baked into **a cust**
	ic of Scientific Discovery" **in** the late	I'm not **sure**. I'm **not**	, the bananas are baked into **a cr**
	ic of Scientific Discovery" **from** the late	I'm not **perfect**. I'm **working**	, the bananas are baked into **the cust**

OpenWebText	**As most managers are probably starting to figure out, Midfielders**	**Martin Shkreli, the bad boy of the U.S.**	**Various ancient maps discovered throughout time reveal a slightly different reality than**
Nucleus	are the most important part of a team. This	pharmaceutical industry, was convicted	what we are used to seeing. The maps
Superposed	are the most important **part** of **the** team. They	pharmaceutical industry, has been **sent**	what we are used to. The **most famous**
	are the most important **players** of **the** team. They	pharmaceutical industry, has been **found**	what we are used to. The **world famous**
	are the most important **part** of **a** team. They	pharmaceutical industry, has been **arrested**	what we are used to. The **most interesting**

Figure 14: Additional qualitative text generations in a compute-normalized setting for Superposed Decoding and Nucleus Sampling with prefixes sampled from OpenWebText

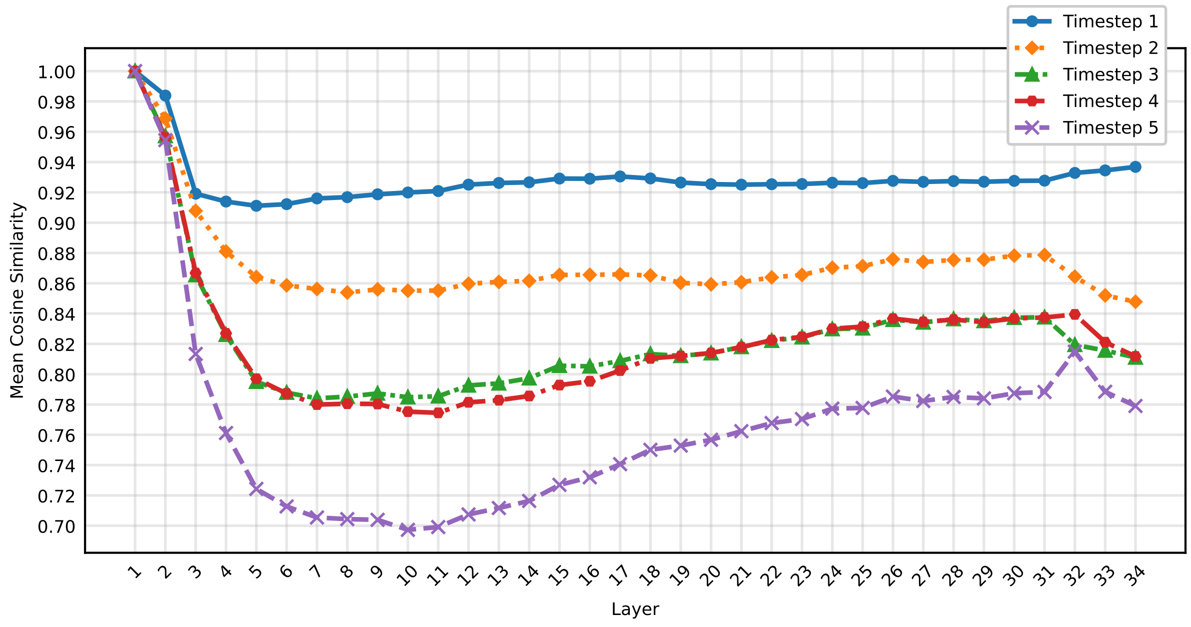


Figure 15: Layer-wise linearity analysis of the first five timesteps on Llama-2-7B with three tokens. The relationship between superposed embeddings and the component token embeddings is initially entirely linear; linearity then degenerates over the first few layers, but gradually recovers through the subsequent transformer block layers.