

Stanford researchers introduce Next-Concept Modeling, generating text coarse to fine in parallel

Source copy kept by wheresthe.ai for the story "Stanford researchers introduce Next-Concept Modeling, generating text coarse to fine in parallel" (<https://wheresthe.ai/d/us-stanford-next-concept-modeling-coarse-to-fine-text-generation-2026-10/>).

Source: <https://hazyresearch.stanford.edu/blog/2026-10-12-ncm>

Text saved 25 Oct 2026 (the page as published).

SHA-256 of the saved text: 7f452b6a045799e0d478e25f977993882d791cc542f5202513c936606e0450e6

Oct 12, 2026 · 10 min read Next-Concept Modeling: Fast, Parallel Coarse-to-Fine Text Generation Hermann Kumbong *, Ishan S. Khare *, Simran Arora †, Azalia Mirhoseini †, Christopher Ré †

Full NCM Team

TL;DR: Can we generate quality text in just a handful of parallel steps? We introduce Next-Concept Modeling (NCM), a new language modeling framework that generates text from coarse concepts to fine details. Unlike autoregressive models that generate one token at a time, or diffusion models that repeatedly refine the entire sequence, NCM predicts increasingly detailed representations of text in parallel. Across a wide range of tasks (unconditional generation, translation, summarization, and reasoning), NCM achieves competitive or better quality than strong diffusion baselines while generating text 10–31× faster and seeing 4–5× fewer training tokens.

 Paper

 GitHub

The Case for Non-Autoregressive Language Models

Language model inference has become a dominant computational workload, driving interest in faster alternatives to autoregressive generation, especially as models produce increasingly long outputs. Diffusion language models (DLMs) are a leading alternative: they refine multiple tokens in parallel, potentially accelerating generation compared with token-by-token decoding. This iterative refinement may also benefit tasks that require global planning. However, DLMs still require many sampling steps, process the full sequence at each step, and often need many passes over the training data.

We introduce Next-Concept Modeling (NCM), a new point on the non-autoregressive language modeling frontier built around coarse-to-fine generation. NCM learns to represent text as a hierarchy of discrete codes, then generates one level at a time, predicting all codes within each level in parallel. Generation takes only a few steps, and shorter sequences make early steps especially inexpensive. Across our benchmarks, NCM achieves better or comparable quality to strong diffusion baselines while generating substantially faster and using fewer training tokens.

NCM: Next-Concept Modeling at a Glance

The key idea behind NCM is simple: NCM first represents text as a hierarchy that grows from coarse to fine. Early levels of the hierarchy are short, holding just a few discrete codes that capture the broad structure of the text, while later ones are longer and add finer details. Below we show a sample generation trajectory with NCM. The model starts out with some high-level idea of what it wants to generate and then progressively adds detail to it.

German: Es besteht allerdings die Sorge, dass kleine und mittlere Unternehmen angreifbar gegenüber Hacking und Überwachung bleiben. Reference: However, there is concern that small and medium-sized companies remain vulnerable to hacking and surveillance.

Level 11 of 11 · decoded output after this level BLEU 77.8

However, there is concern that small and medium-sized enterprises will remain vulnerable to hacking and surveillance. Play ◀ Prev Next ▶

A single NCM generation on German-to-English translation, decoded level by level. BLEU is computed per level against the reference. In Stage 1, we train a multiscale tokenizer to represent text as a coarse-to-fine hierarchy of discrete codes. Building on VAR's approach to images, we make several changes for text: each level has its own

codebook, nested dropout encourages earlier levels to capture meaningful information without relying on later levels, and a token-level reconstruction objective promotes accurate text recovery.

In Stage 2, we freeze the tokenizer and train a transformer to predict each level from the cumulative reconstruction of earlier levels. We train on all levels in a single forward pass and corrupt a fraction of conditioning codes to improve robustness to generation errors. At inference, the model generates one level at a time, predicting all codes within each level in parallel. The tokenizer's decoder then converts the completed hierarchy into text. We provide additional details in our paper.

Evaluating NCM

We evaluate NCM on language modeling, machine translation, extreme summarization, reasoning (Sudoku), scaling (training and test time), inference efficiency, and training efficiency.

Language Modeling

Following standard practice in the diffusion language modeling literature, we evaluate NCM on three datasets, TinyStories, One Billion Word (LM1B), and OpenWebText, using generative perplexity, and report the mean per-sample unigram entropy. NCM achieves similar or better quality compared to the strongest baselines while using fewer function evaluations (NFE).

Generative perplexity on TinyStories, LM1B, and OpenWebText. NCM reaches comparable or better quality with 11× fewer function evaluations. Inference Efficiency

Efficiency in the diffusion language modeling literature is often reported through the number of function evaluations (NFE), which overlooks potentially substantial differences in the cost per evaluation. We therefore benchmark wall-clock generation time on a single NVIDIA H100 SXM 80GB GPU. For unconditional generation, NCM runs 21× faster on LM1B and 10× faster on OpenWebText than the strongest diffusion baselines, with better or comparable generative perplexity. For conditional generation, it matches the strongest diffusion baseline's ROUGE-L on XSum while running 31× faster.

Wall-clock generation time on an H100. Unconditional generation on LM1B, conditional generation on XSum, and reasoning on hard Sudoku. Conditional Generation

The growing consensus is that generative perplexity does not fully capture generative quality. We further evaluate NCM on machine translation (WMT14 De-En) and extreme summarization (XSum). On both tasks, NCM achieves similar or better performance than the strongest diffusion baselines while being up to 31× faster.

Reasoning

We hypothesize that coarse-to-fine generation supports global planning by establishing an overall structure before resolving finer details. We evaluate this intuition on Sudoku, where solutions must satisfy multiple constraints. NCM outperforms both autoregressive and diffusion baselines across all difficulty levels while requiring fewer function evaluations.

Completely solved Sudoku boards by difficulty. NCM degrades the least as puzzles get harder. The full comparison is in the paper. Scaling

We show preliminary scaling results for NCM: scaling the model parameter count consistently improves generative perplexity and unigram entropy. We also use NCM's fast sampling for test-time scaling. Minimum Bayes risk (MBR) decoding over multiple candidates improves BLEU on WMT14 and ROUGE on XSum. Although other methods can also benefit from MBR, NCM generates more candidates within the same sampling-time budget, enabling it to outperform diffusion baselines.

Model scaling on LM1B, and test-time scaling with MBR decoding on WMT14 and XSum. Lower Training Budget

NCM achieves these quality and inference gains with substantially fewer training tokens. On LM1B and OpenWebText, it uses approximately 4.9× and 4× fewer tokens than strong diffusion baselines, respectively, even when accounting for both tokenizer and generator training.

What's Next

We are excited that such a different way of generating text works at all, and produces strong early results. But we are even more excited about what comes next:

Further scaling. We showed initial scaling results, but how does NCM scale with even more parameters, FLOPs, and tokens?

Learning representations jointly with generation. Training the tokenizer and generator together could produce representations better suited to the generation task.

Adapting computation to difficulty. Instead of using a fixed hierarchy for every example, NCM could learn when to stop or allocate additional computation, potentially improving both speed and quality.

Feedback is always welcome! Feel free to reach out at kumboh@cs.stanford.edu .

Full Author List

Hermann Kumbong ^{*} , Ishan S. Khare ^{*} , Iddah Mlauzi , Andrew Woen , Charles Ding , E. Kelly Buchanan , Simran Arora [†] , Azalia Mirhoseini [†] , and Christopher Ré [†]

^{*} Equal contribution. [†] Equal advising.