

[Home](#) [Blog](#) [NCM vs LLM: What Next-Concept Modeling Really Changes](#)[SCIENCE](#) [GENERATIVE AI](#) 10 min reading

NCM vs LLM: What Next-Concept Modeling Really Changes



By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

Summarize this article with:

[Preferences](#)

[Reject](#)

[Accept](#)

Powered by  Consent Pro

SUMMARY

Next-Concept Modeling (NCM), from Stanford's Hazy Research and Together AI, generates text from coarse structure to fine detail rather than token by token. It reports 10–31× faster generation than strong diffusion language models. This article explains how NCM works, what its benchmarks actually measure, how it compares with today's LLMs, and what it means for teams building AI products.

Next-Concept Modeling (NCM) is a new text-generation framework from Stanford's Hazy Research lab and Together AI, published on arXiv on October 9, 2026. Instead of predicting one token at a time, it builds text from coarse structure to fine detail. In the authors' experiments, it matched or beat strong diffusion language models while generating 10–31× faster and training on roughly 4–5× fewer tokens.

Those numbers quickly turned into “NCM vs LLM” headlines. The comparison is worth making, just not the way most headlines make it. NCM's results come from small research models benchmarked against diffusion models, not against GPT-, Claude- or Mistral-class systems. Here is what the approach changes, what the benchmarks really show, and what it means for teams building on language models today.

Three Ways to Generate Text

Most large language models are **autoregressive**. They predict the next token, append it to the sequence, and repeat until the response is complete. Prompt processing can run in parallel, but each output token depends on the ones before it, so decoding remains sequential.

Diffusion language models take a different route. They start from a corrupted sequence and refine many positions in parallel over several iterations. This reduces the number of sequential steps, but each iteration usually produces a full sequence. The number of iterations is cut.

Next-Concept Modeling explores a third way. It repeatedly refines a full token sequence, adding detail and hierarchy from coarse to fine.

By clicking "**Accept**", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

AUTOREGRESSIVE**One token at a time**

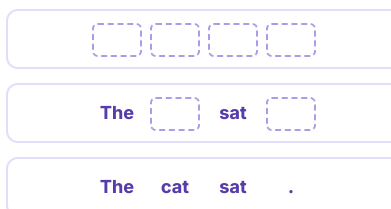
Each new token depends on the tokens generated before it.

SEQUENTIAL DECODING**Core trade-off**

Generation is inherently sequential across output tokens.

DIFFUSION LM**Refine the sequence**

Multiple positions are progressively refined across iterations.

ILLUSTRATIVE REFINEMENT STEPS**Core trade-off**

Parallel updates, but potentially many full-sequence passes.

NEXT-CONCEPT MODELING**Build from coarse to fine**

Generate successive representation levels, predicting codes in parallel within each level.

ILLUSTRATIVE HIERARCHY**Core idea**

Sequential across levels; parallel within each level.

Conceptual illustration, not a representation of the exact number of tokens, refinement steps or hierarchy levels in any particular model.

NCM does more than generate several tokens at once. It changes *what* is generated, a learned hierarchy of discrete codes rather than tokens, and *the order* in which the output takes shape.

How Next-Concept Modeling Works

NCM adapts an idea from image generation to text. That idea is next-scale prediction, popularized by VAR, which won a NeurIPS 2024 best paper award: generate a low-resolution version first, then add detail. Text is harder than images in this respect, because there is no natural way to “downsample” a sentence. The coarse representations have to be learned.

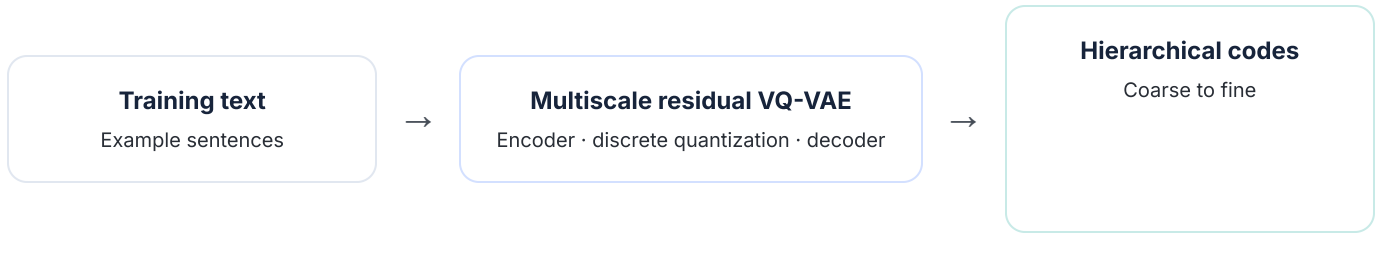
Stage 1: Learn a hierarchy. The researchers train a multiscale residual vector-quantized variational autoencoder (VQ-VAE) that converts text into discrete codes spread across several levels. Early levels hold only a few codes and capture broad structure. Later levels are longer, and each one encodes what the earlier levels left unexplained. A decoder turns the completed hierarchy back into text. Once trained,

Stage 2: Learn to generate the hierarchy. The model learns to generate the hierarchy levels before it. At inference time, it generates codes within a level in parallel. The decoder

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

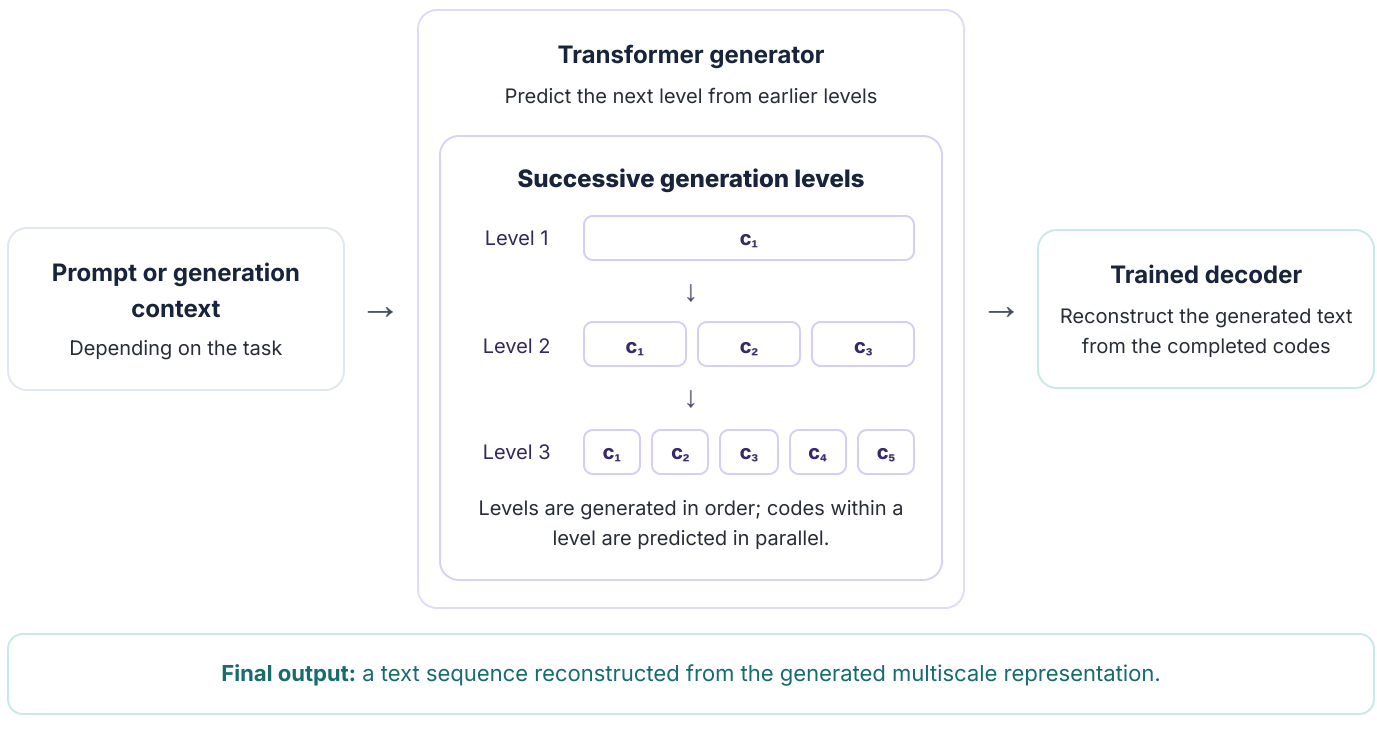
01 Build the multiscale tokenizer

Training the representation used by NCM



02 Generate the hierarchy

The inference process: successive levels, parallel codes within each level



Conceptual overview of the architecture. The illustrated levels and code counts are schematic, not the model's exact configuration. The tokenizer is trained separately and then used by the generation system.

A Translation Example

The authors illustrate the process with an example. The model would write the English sentence, which is readable. Each later level adds and refines the translation, which closely matches the reference translation. The process is perfectly steady, since quality rises as the model refines the translation. The movement is from rough draft to finished translation. The final level on the [Hazy Research blog](#).

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

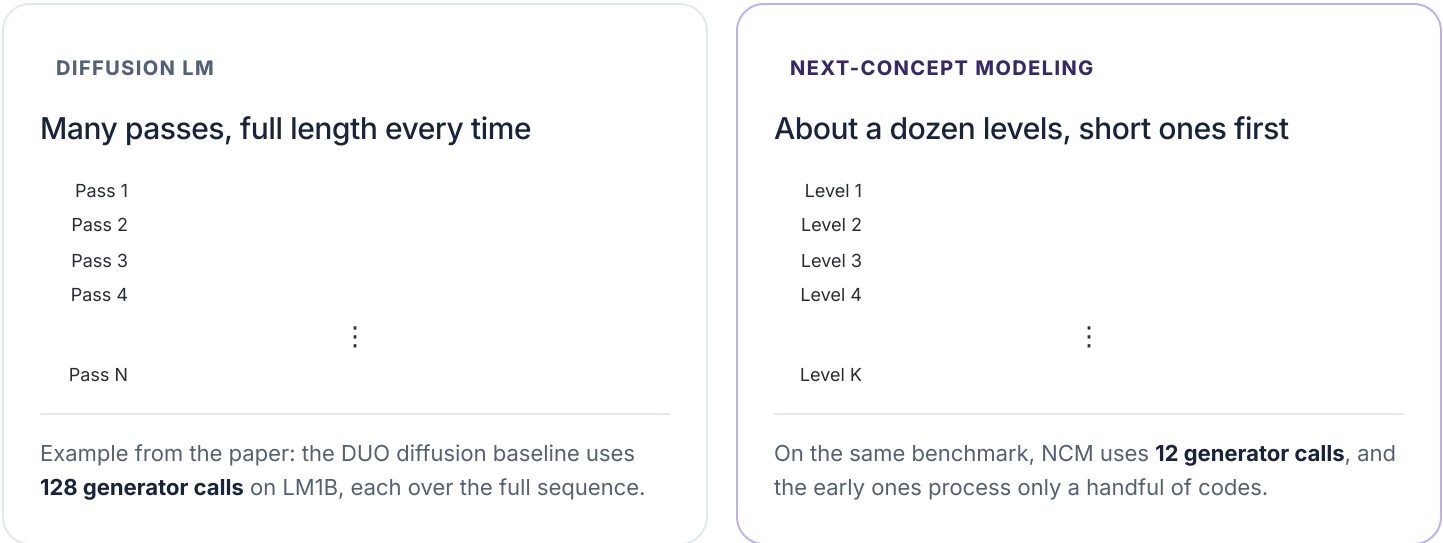
Why It's Faster: Fewer Steps

step.

WHERE THE SPEED COMES FROM

Fewer steps × cheaper steps

Each bar represents one generator pass. Its width represents how much of the sequence that pass has to process.



Fewer sequential steps

×

Lower average cost per step

=

Faster wall-clock generation

Schematic. The final levels are still full length, and every output passes through the whole hierarchy, including short ones.

What “Concept” Means Here

The word *concept* needs some care. In NCM, it refers to the information held in the coarse levels of a learned hierarchy of codes. It is not a claim that the model reasons like a human or works with human-readable ideas. The coarse levels appear to capture useful structure, which is an interesting architectural property, but it should not be read as a statement about intelligence.

It also helps to separate NCM from a (LCM), published in December 2024 sentence autoregressively in SONAR learns its own hierarchy of discrete c level. The names are similar, but the

What the Benchmark

The paper evaluates NCM on uncon summarization and Sudoku. The con matched parameter scales.

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

The headline results come from different tasks and metrics. Read each one in the context of its own experiment.

TASK OR METRIC	REPORTED RESULT	HOW TO INTERPRET IT
LM1B Unconditional generation	21× faster	Faster than the strongest diffusion baseline, with better generative perplexity at that comparison point.
OpenWebText Unconditional generation	10× faster	Competitive quality with the evaluated diffusion baselines in much less generation time.
XSum Conditional summarization	31× faster	Matches the strongest diffusion baseline's ROUGE-L score.
Training efficiency LM1B and OpenWebText	~4–5× fewer tokens	Includes both tokenizer and generator training. Counts token exposures, not compute (FLOPs) or total training cost.
Sudoku Constraint-based reasoning	86.1% of hard boards	Compared with 81.3% for the strongest baseline (FMLM+), using 12 generator calls instead of 16. NCM also leads on easy (99.5%) and medium (96.5%) boards.

How to read these numbers: Speedups are wall-clock comparisons against diffusion language models of similar size, measured on a single NVIDIA H100 GPU. All results are reported by the authors and have not yet been independently replicated.

Source: [Next-Concept Modeling: Fast, Parallel Coarse-to-](#)

The three speed figures come from t into one universal claim. The 21× res is from OpenWebText, and the 31× re

The authors also measure **wall-cloc** matters because a step that process the full sequence, and step counts a

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

workload actually costs.

The **Sudoku** results stand out because every cell must satisfy row, column and box constraints at the same time. The authors suggest that coarse-to-fine generation helps with this kind of global planning, since later levels can revise the board as a whole. That makes structured tasks a promising direction to explore, but it does not show that NCM is a better general reasoner.

One more finding matters for practitioners. Because NCM samples quickly, the authors use the time they save to generate **several candidates and pick the best one**, using a technique called minimum Bayes risk (MBR) decoding. This improves translation and summarization scores while keeping total sampling time competitive with the diffusion baselines. Trading speed for more candidates is a pattern production teams already use with conventional LLMs.

NCM vs LLM: Where It Actually Stands

NCM is often discussed as a direct rival to GPT-, Claude- or Mistral-class systems. A more useful comparison looks at how each approach generates text, and how mature each one is.

NCM VS LLM

Three approaches, three levels of maturity

The real difference today is not only how text is generated, but how much evidence and infrastructure exists around each approach.

DIMENSION	AUTOREGRESSIVE LLMs	DIFFUSION LANGUAGE MODELS	NCM
Generation	Sequential next-token prediction	Iterative refinement of the full sequence	Hierarchical, coarse-to-fine generation
Parallelism	Limited across dependent output tokens	Many positions	All codes within a
Role in the NCM paper	Not part of the head-to-head speed comparison; autoregressive baseline appears on Sudoku		

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

		deployed systems	preliminary runs up to 725M
Production ecosystem	Mature tooling and hosted APIs	Varies by model and provider	Research code and task-specific checkpoints
Main open question	How to keep lowering latency and cost	How to cut refinement steps without losing quality	Whether the gains hold at larger scale and across broader tasks

Based on the NCM paper and authors' materials, October 2026.

The main limitation is scale. Most of NCM’s experiments use generators of roughly 100 million parameters, and the scaling runs up to 725 million parameters are described as preliminary. The design also has known constraints:

- It depends on a tokenizer that is trained separately and then frozen.
- It uses a fixed hierarchy depth, so even a short continuation goes through every level.
- The authors report occasional artifacts at the transitions between coarse levels.

The research team’s next steps address these points directly: further scaling, training the tokenizer and generator together, and adaptive computation that would let the model spend less effort on easier outputs.

In short, NCM shows that a different generation strategy can deliver large efficiency gains in specific settings. It has not yet shown a general advantage over frontier LLMs. That would require evidence at much larger scale, across many kinds of workloads, and under real serving conditions.

What NCM Means for AI E

The broader lesson is that language-
How a model represents information
For teams evaluating any new appro

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

1 Latency

Does it cut response time without hurting the quality your task needs?

NCM today: large wall-clock gains over diffusion baselines at small scale; not yet tested against production LLM serving.

2 Cost

Do fewer generation steps lead to lower end-to-end serving costs once hardware, batching and infrastructure are included?

NCM today: fewer training tokens reported; serving cost at scale unknown.

3 Quality

Do benchmark gains hold on your actual inputs, edge cases and domains?

NCM today: strong on the tested tasks (translation, summarization, Sudoku); broad, open-ended use is unproven.

4 Availability

Is it served through a stable API with documentation, reliability and support?

NCM today: research code and task-specific checkpoints; no hosted inference provider yet.

On availability specifically, the [NCM GitHub repository](#) and [Hugging Face checkpoints](#) make the work easy to experiment with. However, each checkpoint is trained for one task (XSum, WMT14, Sudoku, LM1B, TinyStories or OpenWebText), and none is served by an inference provider. It is a research release, not a drop-in replacement for a hosted model.

This gap between “works in a paper” and “works in production” is not unique to NCM. It is the same gap we described in [The Gap in Shared Understanding of LLM Capabilities](#). The fastest method in a research paper is not automatically the right choice for every application, and the most powerful model is not always worth its cost either, as discussed in [The greenest AI call is the one you did not oversize](#).

The practical response is to keep model choice flexible. If your application calls models through a single API, adopting a faster approach when it becomes production-ready means changing a configuration, not rebuilding your integration. With Eden AI, for example, you can [list available models and their capabilities](#), [route requests by price, speed or latency](#), and set up [automatic fallback](#) so that trying something new covers the strategies in more detail.

NCM’s lasting contribution may be that it challenges the assumption that efficient prediction. Whether it changes production scales.

The next generation of AI systems is how well each strategy is matched

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

- [Next-Concept Modeling: Fast, Parallel Coarse-to-Fine Text Generation](#): the original paper.
- [NCM explained by Hazy Research](#): the authors' walkthrough, with an interactive translation example.
- [HazyResearch/ncm on GitHub](#) and [NCM checkpoints on Hugging Face](#): code, checkpoints and evaluation setup.
- [Large Concept Models: Language Modeling in a Sentence Representation Space](#): Meta's separate, sentence-level approach from 2024.

From Eden AI

- [LLM Routing Explained: Best Strategies for Cost, Speed, and Quality](#)
- [The Gap in Shared Understanding of LLM Capabilities](#)
- [Eden AI documentation](#): one API for 500+ models, with provider routing and fallback built in.

Last updated on October 21, 2026

Clément Moreau

Clément Moreau is a Developer Relations specialist at Eden AI, where he designs practical AI workflows and helps developers integrate models from more than 30 providers through a unified API. His expertise spans OCR, NLP, speech-to-text, computer vision, API design, model orchestration, performance monitoring, and cost optimization.



Similar articles

See all

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

Powered by  Consent Pro

10/21/2026 · Written by Clément Moreau

10/16/2026 · Written by Taha Zemmouri

SCIENCE GENERATIVE AI

The greenest AI call is the one you did not oversize

9/25/2026 · Written by Taha Zemmouri

LET'S START

Start building with Eden AI

A single interface to integrate the best AI technologies into your products.

Get your /

By clicking "**Accept**", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.

LLMs and expert AI models,
unified behind a single API with
full control over providers and
regions.



PLATFORM

- App
- Documentation
- Pricing
- API status
- Integrations
- AI Providers

RESOURCES

- AI Models
- Blog
- Reduce token costs
- Organization management
- Content Creators
- Contact us

CONTENT

- 10 Best OpenRouter Alternatives in 2026
- Best European AI Models Providers in 2026
- EU AI Endpoint: How to Keep AI Requests and Data in Europe
- Top 10 Explicit Content Detection APIs in 2026
- Best GDPR-Compliant AI Gateways in 2026
- How to Use OpenAI, Claude & Gemini in Europe Without GDPR Risk
- The Missing Ring in Europe's AI Sovereignty Chain

By clicking **"Accept"**, you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. View our [Privacy Policy](#) for more information.