A Survey of Alternatives To Next-Token Prediction In Text Generation

Anonymous ACL submission

Abstract

Next Token Prediction (NTP) has driven the unprecedented success of Large Language Models (LLMs) based on Transformer, but is also the source of their most persistent weaknesses such as poor long-term planning, error accumulation, and computational inefficiency. Acknowledging the growing interest in exploring alternatives to NTP, the survey organises the emerging ecosystem of alternatives to NTP into four main families: (1) **Multi-Token Prediction**, which targets a block of future tokens instead of a single one; (2) **Plan-then-Generate**, where a global, high-level plan is created upfront to guide token-level decoding; (3) **Latent Reasoning**, which shifts the autoregressive process itself into a continuous latent space; and (4) **Continuous Generation Approaches**, which replace sequential generation with iterative, parallel refinement through diffusion, flow matching, or energy-based methods. By synthesizing insights across these methods, this survey offers a taxonomy to guide research into models that address the known limitations of token-level generation to develop new transformative models for text generation, and, broadly, natural language processing.

1 Introduction

Large Language Models (LLMs) have achieved remarkable performance across a wide range of natural language processing (NLP) tasks, including question answering, summarization, creative writing and coding (Hendrycks et al., 2021; Srivastava et al., 2023; Jiang et al., 2024). **Next Token Prediction (NTP)** (Vaswani et al., 2023), the task of predicting the next subword given the previous tokens in a sequence, has been the dominant training and decoding paradigm for LLMs, and specifically the Transformer.

Yet, LLMs often exhibit brittle behavior: they hallucinate facts, lose coherence in long outputs (Maharana et al., 2024), and struggle with tasks that require long-term planning or reasoning (Hao et al., 2023). A growing body of work suggests that these failures are symptomatic of deeper structural problems with the NTP framework itself since the probability of the next most likely token does not guarantee the start of the most probable *overall sequence* (Bachmann and Nagarajan, 2024).

This survey explores emerging alternatives with Table 1 summarising these alternatives. Within the scope of Transformer-based models, we identify four broad families: Multi-Token Prediction (MTP), Plan-then-Generate (PtG), Latent Reasoning (LR), and Continuous Generation Approaches (CG). The systematic review methodology, full scope, and per-family summaries are provided in Appendix A. As Figure 1 illustrates, a diverse set of methods depart from token-by-token generation (as in the case of NTP) toward richer and more global semantic representations.

2 Failure Modes of NTP

We begin with describing typical failure modes of NTP. In the subsequent survey, each family's "Failure Modes" subsection uses a shared vocabulary of five recurring themes (Interpretability, Training Brittleness, Representation Collapse, Costs, Output-Form Constraints) so the reader can compare. These themes are defined in Appendix B.

2.1 Inference-Time Failures: Greedy Generation and Error Accumulation

Greedy generation focuses on short-term likelihood which leads to **error accumulation** (Rohatgi et al., 2025; Nnadi and Bertini, 2024), where early, minor inaccuracies compound into significant incoherence or factual drift later in the sequence.

Despite near-perfect "needle-in-a-haystack" retrieval tasks (e.g., GEMINI 1.5's $> 99\%$ accuracy (Gemini Team et al., 2024)), LLMs struggle with

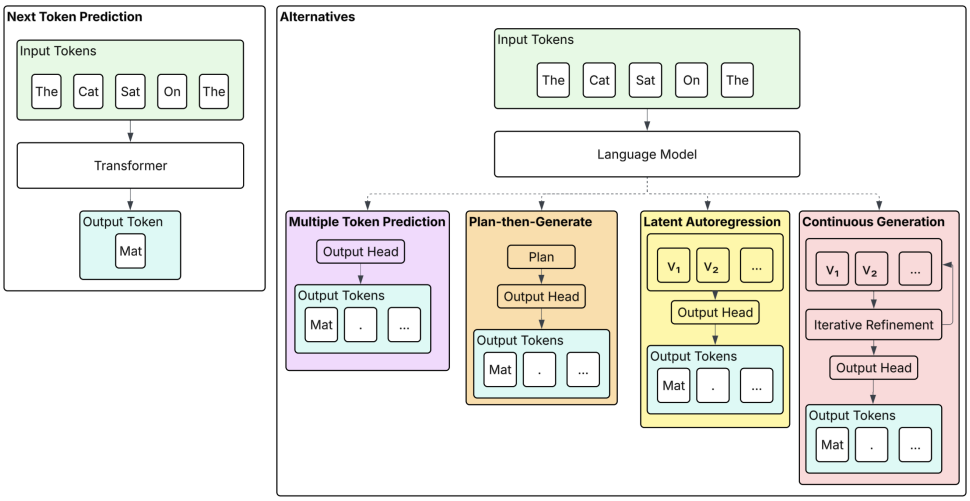

Figure 1: An abstracted visualization of NTP and its alternatives.

long-context understanding, even when equipped with large context windows (Maharana et al., 2024). Decision-making studies categorize this failure as **greediness in action selection**, where LLMs prematurely commit to local continuations rather than exploring alternate trajectories (Schmied et al., 2025). This behaviour reflects the inability of NTP to revise earlier decisions in light of later ones.

2.2 Granularity Mismatch: Tokens vs. Ideas

LLMs typically rely on subword tokenization, most commonly the Byte-Pair Encoding (BPE) algorithm (Sennrich et al., 2016). While effective, BPE fragments text into the most frequent units, which are usually neither linguistically natural nor semantically meaningful. This misalignment manifests in *unexpected and undesirable behaviors*, such as "glitch tokens" like `solidgoldmagikarp` (Land and Bartolo, 2024), arising from limited exposure during training. Although recent work has proposed methods to mitigate these under-trained tokens, the fact that ad hoc fixes are needed suggests that sub-word tokens may lack the right level of abstraction for language understanding. BPE also leads to poor performance on trivial character-level tasks. Cosma et al. (2025) and Fu et al. (2024) demonstrate tokenized language models fail at rudimentary character-counting tasks such as answering *"How many r's are in strawberry?"*.

2.3 Compute Inefficiency: Quadratic Attention

NTP is computationally expensive, since Transformers compute attention over the full input sequence at each layer and time-step, incurring a time and memory complexity of $\mathcal{O}(n^2)$ for sequences of length n (Vaswani et al., 2023). The environmental impact of these computationally expensive NLP architectures has also been reported (Ansar et al., 2024). Recent work has explored architectural innovations like sparse attention (Roy et al., 2021; Nguyen et al., 2025), linear transformers (Beltagy et al., 2020), retrieval-based compression, and KV-sharing attention mechanisms (Ainslie et al., 2023; Shazeer, 2019). While these optimizations improve efficiency, they do not address the core granularity mismatch. Each pass through the LLM must still process the full document processing potentially thousands of tokens even for conceptually simple operations.

2.4 Training-Time Failures: Misaligned Supervision in Teacher-Forcing

A deeper issue arises during training. Models are typically trained using teacher-forcing, where they learn to predict the next token given the true previous tokens. As Bachmann and Nagarajan (2024) argue, this setup can cause models to learn **spurious shortcuts**, a phenomenon they call the "Clever Hans" effect. Instead of learning to plan, models exploit the revealed output prefix to guess plausible continuations without understanding the task.

3 Multi-token prediction

To accelerate inference and encourage short-term planning, **Multi-Token Prediction (MTP)** models change the prediction target from a single token to a block of k future tokens. This is typically achieved by using a shared context representation (a "trunk") to feed multiple parallel output heads,

Category	Definition	Key Approaches
Multi-Token Prediction	Generates several future tokens at once from some shared latent space ("trunk").	ProphetNet Qi et al. (2020); Gloeckle et al. (2024); Ahn et al. (2025); DeepSeek-V3 Technical ReportDeepSeek-AI et al. (2025)
Plan-then-Generate	Two-stage process where a high-level plan is generated before token-by-token generation.	Plan-then-Generate Su et al. (2021);Semformer Yin et al. (2024); DGLM Lovelace et al. (2024); PLANNER Zhang et al. (2024)
Latent Reasoning	Generates continuous latent vectors instead of, or alongside, tokens.	COCONUT Hao et al. (2024);Latent Reasoning Geiping et al. (2025); LCM LCM team et al. (2024); CoCoMix Tack et al. (2025)
Continuous Generation Approaches	Denoises a noisy latent sequence over multiple steps instead of generating left-to-right.	Diffusion-LM Li et al. (2022); Multi-Granularity Diffusion ModelingYe et al. (2025) ; TEncDMShabalin et al. (2025); Scaling Masked DiffusionNie et al. (2025); Block DiffusionArriola et al. (2025); Gemini Diffusion: `https://deepmind.google/models/gemini-diffusion/`; LLaDA: `https://ml-gsai.github.io/LLaDA-demo/`; Flow MatchingLipman et al. (2023); Flow Matching for TextHu et al. (2024) ;EDLMXu et al. (2025); Residual EBMsDeng et al. (2020)

Table 1: Taxonomy of alternatives to Next Token Prediction in transformers.

each predicting a different future token. This aims to improve **computational efficiency** and **greedy generation**.

Formally, the probability is calculated as

$$P(x_{t+1:t+k} \mid x_{\leq t}) = \prod_{i=1}^{k} P(x_{t+i} \mid h_t)$$

where the shared trunk representation is $h_t = f_{\text{trunk}}(x_{\leq t})$.

This setup breaks the strict chain rule of NTP while also allowing for parallel decoding (Santilli et al., 2023).

3.1 Key Approaches

Table 2 compares the key approaches for MTP in terms of the trunk, MTP head and number of predicted tokens. **ProphetNet** (Qi et al., 2020) was the first canonical example, using an n-stream self-attention mechanism to predict the next n tokens in parallel from a shared context stream (trunk). Gloeckle et al. (2024) extend this idea from encoder-decoder models like ProphetNet to large-scale autoregressive LLMs by training a standard Transformer (trunk) with multiple output heads, each predicting a separate future token. **Joint Token Prediction (JTP)** (Ahn et al., 2025) improves upon prior MTP variants by modeling the joint distribution of future tokens rather than treating them as conditionally independent. JTP introduces a lightweight attention module ("Fetch") as a bottleneck to force the trunk to encode richer multi-step planning signals. JTP outperforms other MTP methods on synthetic reasoning benchmarks with fewer parameters and minimal computational overhead. MTP is also used as an auxiliary objective during training. **DeepSeek-V3** (DeepSeek-AI et al., 2025) shows that adding MTP heads during training improves generalization across benchmarks. The shared Transformer trunk encourages broader temporal representations as well as enabling speculative decoding.

3.2 Quantitative Impact & Failure Modes

We map the reported numbers to the specific NTP failure modes from Section 2.2 (indicated in square brackets) and its neighbours.

Compute inefficiency [Section 2.3]. Gloeckle et al. (2024) report a $3\times$ inference speedup at 13B scale via speculative decoding using the auxiliary heads, at no training-time overhead. Similarly, for **DeepSeek-V3** (DeepSeek-AI et al., 2025), 85–90% acceptance rate for speculatively proposed tokens, enabling production-scale inference acceleration.

Greedy generation [Section 2.1]. Multi-token training also encourages short-horizon planning that vanilla NTP lacks. In the case ofGloeckle et al. (2024), it is $+12$ pp pass@1 on HumanEval and $+17$ pp on MBPP against an otherwise-identical 13B next-token baseline.

MTP's reported weaknesses map onto the common themes (§B) as follows: (A) *Output-Form Constraints:* Despite modeling multiple tokens, MTP often focuses on short-term future steps; (B) *Training Brittleness — Architectural inflexibility:* Joint prediction heads (e.g., JTP's Fetch module) introduce multiple loss terms that must be care-

Model	Trunk	MTP Head(s)	Predicted Tokens
ProphetNet	Transformer encoder	Offset decoder streams for future token prediction	1 - 3
Gloeckle et al.	Transformer trunk	Independent heads for each future token	1 - 32
JTP (Ahn et al.)	Transformer trunk	Fetch module + joint head for joint token prediction	1 - 8
DeepSeek-v3	Transformer trunk	MTP heads (discarded or used for speculative decoding)	1 - 2

Table 2: Key MTP approaches in terms of the trunk structure, head design, and distinguishing features.

fully weighted; standard NTP training recipes don't transfer, and mistuned head weights degrade downstream quality; (C) *Costs:* When used at inference, speculative decoding introduces a trade-off between speed, accuracy and cost.

It can be seen that MTP offers a simple extension to NTP by forecasting blocks of future tokens. This enables short-term planning and partial parallelization, addressing NTP's sequential bottleneck without major architectural changes. However, they often only look 4-5 tokens into the future. Overall, MTP improves efficiency and token-level foresight but lacks mechanisms for truly global planning.

4 Plan-then-Generate

Instead of generating text greedily from left to right, **PtG** models introduce a planning stage. This plan can be either **symbolic**, or **latent**. Formally -

$$z \sim P_{\text{planner}}(z \mid c) \tag{1}$$

$$y \sim P_{\text{generator}}(y \mid z, c) = \prod_{i=1}^{N} P(y_i \mid y_{<i}, z, c) \tag{2}$$

We categorize PtG along three axes (summarised in Table 3): (A) **Hierarchy Unit:** What level of abstraction the plan operates on; (B) **Planner Module:** How the plan is created; (C) **Conditioning Strategy:** How the plan influences decoding.

4.1 Key Approaches

Symbolic Planning Models : Symbolic planners are interpretable, but limited by the need for symbolic input formats and task-specific schemas. Early PtG work in structured data-to-text generation (Puduppully and Lapata, 2021; Chen et al., 2020; Li et al., 2023a) used symbolic plans. **PlanGen** (Su et al., 2021) predicts a sequence of content slots from RDF triples using a CRF, which a BART model then verbalizes. **LLM+P** (Liu et al., 2023) generalizes this idea to classical planning: an LLM formalizes a task into a structured PDDL program, which an external planner solves before the LLM verbalizes the result.

Latent Planning Models Recent PtG models replace symbolic plans with learned **latent representations**, *i.e.*, continuous vectors that encode global information without requiring hand-crafted structure. **Semformer** (Yin et al., 2024) learns special tokens that reconstruct sentence embeddings, inserted between prompt and output to provide planning signals. **DGLM** (Lovelace et al., 2024) samples an abstract intent vector from a diffusion model over Sentence-T5 latents, which is then transformed into a soft prompt to condition decoding. **PLANNER** (Zhang et al., 2024) encodes a paragraph into a latent vector using a VAE and denoises it via diffusion. The final plan conditions a decoder.**Cornille et al.** (Cornille et al., 2024) cluster sentence embeddings into discrete writing actions. These serve as modular planning units and are injected into a frozen LLM via adapter layers.

4.2 Quantitative Impact & Failure Modes

Greedy generation [Section 2.1]. A plan provides the long-horizon commitment that NTP lacks. In**PlanGen** (Su et al., 2021), 140M parameters beats T5-3B (a 20× larger model) on ToTTo PARENT (+0.5%) and BLEURT (+8.3%), demonstrating that content planning can substitute for parameter scale.Cornille et al. (2024) report −1.14 PPL on Wikipedia using a frozen GPT-2 backbone with an adapter-injected planner—improvement is concentrated in long-form structure (lower latent PPL, lower edit distance).

Training-time misaligned supervision [Section 2.4]. Teacher-forcing causes the "Clever-Hans" effect on lookahead tasks; planning objectives mitigate it. **Semformer** (Yin et al., 2024): reaches 99.2% accuracy on the G(5,30) star-graph path-finding task—a benchmark which teacher-forced NTP fails at—with −5% PPL on Wikitext and −9% on LAMBADA versus a same-scale teacher-forced NTP baseline.

Model	Hierarchy Unit	Planner Module	Conditioning Strategy
(Su et al., 2021)	Slot-level (symbolic)	CRF over RDF triples	Cross-attention to plan
(Yin et al., 2024)	Sentence-level (latent)	Autoencoder (offline)	Prepending latent tokens
(Lovelace et al., 2024)	Sentence-level (latent)	Diffusion over Sentence-T5	Soft prompt from embedding
(Zhang et al., 2024)	Paragraph-level (latent)	VAE + latent diffusion	Decode conditioned on latent
(Cornille et al., 2024)	Sentence-level (discrete actions)	Clustering + external classifier	Adapter-based plan injection

Table 3: Comparison of Plan-then-Generate models by hierarchy unit, planning method, and conditioning strategy.

As a result, the failure modes of PtG approaches are as follows: (A) *Training Brittleness:* If the plan is poor, downstream outputs degrade significantly. (B) *Interpretability:* Latent planners are not easily interpretable, making it difficult to identify errors. (C) *Output-Form Constraints:* Symbolic methods require structured inputs and fail to generalize across domains; (D) *Training Brittleness:* Most PtG systems require multiple pretrained components (e.g., VAE, diffusion models, clustering pipelines), increasing training cost and brittleness.

In summary, while PtG offers a promising strategy to mitigate the myopia of next-token decoding, current implementations introduce steep engineering and alignment costs. Symbolic planners lack scalability, while latent planners introduce interpretability and reliability challenges.

5 Latent Reasoning

LR models replace token generation with autoregression over latent states, shifting the generative loop into a continuous space. This addresses the **granularity mismatch** between subword tokens and semantics, enabling operation over sentences, thoughts, or concepts.

5.1 Key Approaches

We group LR into two categories as follows.

- **Latent Autoregression**: These models first generate latent vectors, then decodes them into text:

$$P(z \mid x) = \prod_{i=1}^{N'} P(z_i \mid z_{<i}, f_{\text{enc}}(x)) \quad (3)$$

$$P(y \mid z) = \prod_{i=1}^{N'} P(y_i \mid z_i) \quad (4)$$

SentenceVAE (An et al., 2024) encodes *each sentence* into a latent vector and autoregressively predicts the next. A decoder reconstructs each latent into a sentence. Compared to token-level baselines, it improves inference speed (2–3×), perplexity (46–75%), and memory use (86–91% reduction). **Large Concept Models (LCMs)** (LCM team et al., 2024) generate one "concept" vector *per sentence* to enable long-range abstraction and reduce sequence length. **CoCoMix** (Tack et al., 2025) interleaves latent vectors (from a pretrained autoencoder) with token decoding, enriching each step with high-level context. Latent concepts and tokens coexist, so the model retains token-level decoding while gaining concept-level planning signal.

- **Latent Chain-Of-Thought** Here, the model refines a latent state over several internal steps before emitting a token. The steps indicate the chain of thought.

$$h_i^{(r)} = f_{\text{core}}(h_i^{(r-1)}, x) \quad (5)$$

$$P(y_i \mid y_{<i}, x) = f_{\text{out}}(h_i^{(R)}) \quad (6)$$

Latent Reasoning (Geiping et al., 2025) introduces an inner loop where a shared module is applied repeatedly in latent space before a token is emitted. Unlike regular chain-of-thought (CoT), where CoT emits discrete intermediate tokens to externalise its thinking, latent reasoning performs iterative refinement in latent space, with no visible intermediate tokens. **Coconut** (Hao et al., 2024) performs latent reasoning between text segments, enabling breadth-first exploration of output trajectories before committing to a token sequence. Unlike regular CoT, Coconut's thoughts are not decoded back to language and instead kept as latent.

5.2 Quantitative Impact & Failure Modes

Granularity mismatch [Section 2.2]. Operating on "concept-sized" units reduces sequence length and the token-by-token planning burden **SentenceVAE** (An et al., 2024): reduces perplexity to 46–76% of an OPT token-AR baseline at

Model	Mechanism	Token Unit	Key Distinction	Output Format
SentenceVAE	**Latent Autoregression**	Sentence embedding	Autoregresses over sentence vectors with a base LLM decoder	Discrete (sentences)
LCM	**Latent Autoregression**	Sentence embedding	Fully integrated latent autoregressive architecture	Discrete (sentences)
CoCoMix	**Latent Autoregression**	Latent concept vector	Predicts token and concept vector at every step	Discrete (tokens)
Latent Reasoning	**Latent Chain of Thought**	Continuous thought	Internal loop increases compute before each token	Discrete (tokens)
Coconut	**Latent Chain of Thought**	Continuous thought	Latent planning between text chunks	Mixed

Table 4: Comparison of Latent Reasoning Models.

matched scale, demonstrating that sentence-level units can better represent tokens for the same compute.**CoCoMix** (Tack et al., 2025) matches NTP perplexity on OpenWebText with 21.5% fewer training tokens at 1.38B scale—the concept-level signal does work that token-level training cannot. **Coconut** (Hao et al., 2024) lifts ProsQA accuracy from 77.5% to 97.0% over CoT prompting.

Compute inefficiency [Section 2.3]. Fewer, larger units mean fewer forward passes per output. **SentenceVAE** (An et al., 2024) achieves 2–3.65× inference speedup, with memory reduced by 86–91% for matched context length. **Coconut** (Hao et al., 2024) uses 3× fewer generated tokens (8.2 vs. 25.0 on GSM8K) while improving accuracy on ProsQA from 77.5% to 97.0%.

Greedy generation [Section 2.1]. Internal iteration gives the model room to plan before emitting tokens. In Geiping et al. (2025), at 3.5B scale, internal latent iterations produce monotonic gains on ARC-Easy (49.07% → 69.91% from $r = 4$ to $r = 32$) and HellaSwag (43.46% → 65.21%), establishing internal iteration as a new scaling axis orthogonal to parameter count.

The following failure modes are observed in the case of LR approaches: (A)*Interpretability:* autoregression occurs in latent space, making it difficult to interpret; (B) *Representation Collapse:* poorly trained or misaligned latent encoders/decoders lead to degraded outputs during decoding; (C) *Training Brittleness:* many models (e.g., SentenceVAE, LCM, CoCoMix) rely on complex multi-stage training involving external encoders, autoencoders, or diffusion priors; (D) *Evaluation Difficulty:* outputs often require post-decoding reconstruction, introducing noise that complicates evaluation and benchmarking; (E) *Representation Collapse:* out-of-distribution latent vectors may decode to nonsensical outputs without token-level grounding.

To summarise, LR offers a compelling departure from NTP by shifting generation to abstract semantic units. These methods achieve greater speed and generalization by reducing sequence length and operating at coarser granularity. However, they introduce difficulties from interpretability to training complexity, and latent stability.

6 Continuous Generation Approaches

CG treat text generation as a global optimization problem. Rather than producing tokens one-by-one, these methods initialize the entire output simultaneously and refine it iteratively. This enables **parallel** generation instead of **sequential** generation, providing three key advantages: (1) global planning and revision, (2) mitigation of error accumulation, and (3) bidirectional refinement where later context influences earlier tokens.

Diffusion-based models, the most established CG approach, treat generation as an iterative denoising process that starts from noise and progressively refines the full output. Continuous diffusion models operate in embeddings spaces while discrete models operate over tokens of logits via masking and corruption (illustrated in Figure 3). A dedicated survey by Li et al. (2023b) is a comprehensive review of the technical landscape of text diffusion models.

6.1 Key Approaches

Table 5 categorizes recent CG approaches along several dimensions. **Continuous Diffusion Models** apply noise to continuous representations and train a denoiser to recover the original signal. **Diffusion-LM** (Li et al., 2022) generates word embeddings via iterative denoising, allowing fine-grained control over syntax and semantics. **TEncDM** (Shabalin et al., 2025) denoises in

a frozen encoder's latent space (e.g., BERT), capturing sentence-level semantics more effectively with fewer steps. In contrast, **Discrete Diffusion Models** operate directly on tokens or logits, generalizing masked language modeling into a multi-step process. **MGDM** (Ye et al., 2025) improves symbolic reasoning by denoising from multiple noisy views, mitigating subgoal imbalance in AR models. **Block Diffusion** (Arriola et al., 2025) blends diffusion with AR by refining fixed-length blocks. This supports scalable, partially parallel decoding with token caching. **Mercury** (Labs et al., 2025) scales discrete diffusion to commercial deployment, achieving **1100 tokens/sec** with latencies under **25ms**.

Large-scale efforts like **Gemini Diffusion**[1] and **LLaDA**[2] signal growing industry investment in diffusion-based language modeling.

Flow matching (Lipman et al., 2023) offers an alternative to diffusion that learns deterministic trajectories between noise and data distributions. Unlike diffusion's stochastic paths, flow matching constructs optimal transport maps that can be more sample-efficient. The framework has been successfully extended beyond its original formulation to simulation-based inference (Dax et al., 2023) and structured data (Eijkelboom et al., 2025). Recent work has adapted flow matching to text generation (Hu et al., 2024), treating sentences as points in a continuous semantic space and learning smooth transformations between random initialization and meaningful text. This approach entirely sidesteps tokenization by operating in embedding space throughout the generation process.

Energy-based models (EBMs) define generation as an energy landscape where valid outputs correspond to low-energy configurations. Originally developed in statistical physics, then extended to machine learning vision tasks, EBMs learn a scalar energy function that assigns compatibility scores to entire configurations. **Residual EBMs** (Deng et al., 2020) impose energy-based post-processing through contrastive objectives. **Energy-based Diffusion Language Models (EDLM)** (Xu et al., 2025) introduce EBM formulations within discrete diffusion models, enabling sequence-level energy modeling at each diffusion step. **Energy-Based Reward Models (EBRM)** (Lochab and Zhang, 2025) enhance LLM alignment by replacing scalar rewards with energy functions that better capture distributional uncertainty in human preferences. **Energy-Based Transformers (EBT)** (Gladstone et al., 2025) reframe the transformer architecture as an EBM, learning energy functions to assess compatibility between context and generated text.

[1] https://deepmind.google/models/gemini-diffusion/

[2] https://ml-gsai.github.io/LLaDA-demo/

6.2 Quantitative Impact & Failure Modes

Error accumulation [Section 2.1]. Iterative global refinement lets later context fix earlier mistakes—the failure mode left-to-right NTP cannot resolve. **MGDM** (Ye et al., 2025): reaches 100% on Sudoku (vs. LLaMA-13B's 32.9%), 91.5% on Countdown-4 (vs. GPT-2 scratch 45.8%), and 76.0% on Game of 24 with $1\times$ token cost where GPT-4 Tree-of-Thoughts uses $186\times$. Tasks where a single wrong digit invalidates the full output expose the error-accumulation gap most starkly. Similarly, **Diffusion-LM** (Li et al., 2022) beats FUDGE by $+11.3$ pp on semantic control, $+63.0$ pp on parts-of-speech, $+68.1$ pp on syntax-tree control, and $+53.0$ pp on length control. Global iterative refinement allows hard constraints that token-level NTP cannot enforce.

Compute inefficiency [Section 2.3]. Parallel denoising and single-step sampling decouple inference cost from output length. **Mercury** (Labs et al., 2025): achieves 1109 tokens/second on H100 ($\sim 10\times$ frontier speed-optimised LLMs) and ~ 25 ms latency in Copilot Arena (vs. GPT-4o Mini's ~ 100 ms), at 90% HumanEval pass@1 (76.6% MBPP), within 1–2 pp of frontier AR quality. Also, **FlowSeq** (Hu et al., 2024): reaches competitive question-generation BLEU using a single ODE step against a 2000-step DiffuSeq baseline, reducing sampling time from 520 s to 0.26 s.

Therefore, the failure modes of CG approaches are: (A) *Costs:* While parallelizable, the iterative refinement process incurs significant costs. Diffusion models require dozens of denoising steps; flow matching needs multiple ODE solver steps despite being deterministic; and energy-based models require expensive MCMC sampling or gradient-based optimization; (B) *Output-Form Constraints:* Most CG models struggle with variable-length outputs. Diffusion models like Diffusion-LM and MGDM require padding or truncation and energy-based models typically define energy over fixed-size configurations; (C) *Interpretability:* Since these methods operate in latent space, generation trajectories are opaque across all approaches; (D)

Model	Approach Type	Latent Space	Granularity	Update Mechanism
Diffusion-LM	Diffusion	Word embeddings	Token-level	Stochastic denoising
TEncDM	Diffusion	Frozen encoder latents	Sentence-level	Stochastic denoising
MGDM	Diffusion	Token logits	Token-level	Discrete denoising
Block Diffusion	Diffusion	Token blocks	Block-level	Block-wise refinement
Mercury	Diffusion	Token logits	Token-level	Parallel denoising
FlowSeq	Flow Matching	Text embeddings	Sequence-level	Deterministic flow
EDLM	Energy-Based	Token logits	Token-level	Energy minimization
EBRM	Energy-Based	Reward space	Sequence-level	Gradient descent
Residual EBMs	Energy-Based	Output logits	Token-level	Contrastive energy

Table 5: Comparison of CG Approaches by type, latent space, granularity, and update mechanism.

Training Brittleness: Diffusion needs noise schedulers and denoising objectives; flow matching requires ODE solvers and transport cost optimization; energy-based models need contrastive sampling strategies and careful initialization to avoid mode collapse.

In closing, CG approaches break from the rigidity of token-by-token autoregressive generation by enabling global and parallel reasoning. Diffusion-LM supports rich control signals, while discrete methods like Mercury scale to production-level throughput; flow matching offers deterministic, theoretically grounded trajectories that can be more sample-efficient; and EBMs provide a principled framework for incorporating global constraints and preferences through holistic sequence scoring. Yet the cost of these benefits is high: slow inference, interpretability challenges, and complex training procedures remain core obstacles.

7 Conclusion & Future Directions

NTP has long been the foundational paradigm for large-scale language modeling based on Transformers. While it enables impressive fluency, a growing body of literature highlights that NTP is also a bottleneck, with its myopic, token-by-token nature contributing to failures in long-term planning, computational efficiency, and semantic abstraction. In response, a rich ecosystem of alternatives has emerged. For completeness, Non-Transformer Architecture (NTA) approaches such as State Space Models and JEPAs are discussed in Appendix C.

7.1 Future Research Directions

Based on this survey, we see two opportunities in the context of **multi-token prediction**. First, *predict farther ahead*: study how performance changes as we increase k (the number of future tokens), using simple schedules (start small, grow k) and checks for when extra lookahead stops helping. Promising ideas include variable-length or adaptive horizons, and hierarchical heads that look ahead at different scales (phrases, sentences, sections). As for **Plan-then-Generate.**, we note the need for *diverse planners* that move beyond generic embeddings to planners that capture discourse structure, key entities, or task steps, with evaluation focused on whether the generated text actually follows the plan. Secondly, further *hybrid architectures* of different planning modules should be explored, along with systematic studies of *adaptable time horizons* plans that can flex between sentence-level, paragraph-level, or document-level guidance depending on task requirements.

Latent Reasoning as an alternative for NTP requires improved *Encoder/decoders* that faithfully move between text and latent space, with simple checks that meaning is preserved. *Non-Transformer models*, have seen growing use, including state-space models and JEPA-style training, to predict in representation space rather than token space.

Finally, block composition is a promising direction in the case of **Continuous Generation Approaches.** Block Diffusion demonstrates that interpolating between autoregression and diffusion at the block level works well. Block composition refers to ways to combine blocks at different granularities, so that global structure and local fluency are jointly respected. For example, mixing sentence-level, paragraph-level, and section-level blocks within a single generation.

Limitations

This survey aims to provide a broad and structured overview of emerging alternatives to NTP in trans-

former models, despite some limitations. First, given the rapid pace of research in this area, our coverage may not include the most recent or unpublished advances, particularly those in proprietary or closed-source settings. Second, although we categorize alternatives along high-level axes (e.g., planning, token granularity, decoding strategy), the boundaries between categories are sometimes blurred. For example, some diffusion-based models also implement PtG, and some multi-token predictors incorporate latent semantic reasoning.

Moreover, our analysis focuses primarily on high-level paradigms that challenge the generative process of LLMs. Less emphasis is given to tokenization approaches, output head optimizations, or architectural innovations in the intermediate layers, which could meaningfully challenge the NTP paradigm. Finally, we do not present empirical benchmarking of these alternatives under controlled conditions; instead, we rely on results reported by the original authors.

References

Kwangjun Ahn, Alex Lamb, and John Langford. 2025. Efficient joint prediction of multiple future tokens. *Preprint*, arXiv:2503.21801.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *Preprint*, arXiv:2305.13245.

Hongjun An, Yifan Chen, Zhe Sun, and Xuelong Li. 2024. Sentencevae: Enable next-sentence prediction for large language models with faster speed, higher accuracy and longer context. *Preprint*, arXiv:2408.00655.

Wazib Ansar, Saptarsi Goswami, and Amlan Chakrabarti. 2024. A survey on transformers in nlp with focus on efficiency. *Preprint*, arXiv:2406.16893.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. *Preprint*, arXiv:2503.09573.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. 2023. Self-supervised learning from images with a joint-embedding predictive architecture. *Preprint*, arXiv:2301.08243.

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. *Preprint*, arXiv:2403.06963.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. 2022. data2vec: A general framework for self-supervised learning in speech, vision and language. *Preprint*, arXiv:2202.03555.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. 2024. Revisiting feature prediction for learning visual representations from video. *Preprint*, arXiv:2404.08471.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Gianni Brauwers and Flavius Frasincar. 2023. A general survey on attention mechanisms in deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3279–3298.

Pengfei Cao, Tianyi Men, Wencan Liu, Jingwen Zhang, Xuzhao Li, Xixun Lin, Dianbo Sui, Yanan Cao, Kang Liu, and Jun Zhao. 2025. Large language models for planning: A comprehensive and systematic survey. *Preprint*, arXiv:2505.19683.

Kai Chen, Fayuan Li, Baotian Hu, Weihua Peng, Qingcai Chen, and Hong Yu. 2020. Neural data-to-text generation with dynamic content planning. *Preprint*, arXiv:2004.07426.

Liang Chen, Zekun Wang, Shuhuai Ren, Lei Li, Haozhe Zhao, Yunshui Li, Zefan Cai, Hongcheng Guo, Lei Zhang, Yizhe Xiong, Yichi Zhang, Ruoyu Wu, Qingxiu Dong, Ge Zhang, Jian Yang, Lingwei Meng, Shujie Hu, Yulong Chen, Junyang Lin, and 8 others. 2024. Next token prediction towards multimodal intelligence: A comprehensive survey. *Preprint*, arXiv:2412.18619.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Preprint*, arXiv:1412.3555.

Nathan Cornille, Marie-Francine Moens, and Florian Mai. 2024. Learning to plan for language modeling from unlabeled data. *Preprint*, arXiv:2404.00614.

Adrian Cosma, Stefan Ruseti, Emilian Radoi, and Mihai Dascalu. 2025. The strawberry problem: Emergence of character-level understanding in tokenized language models. *Preprint*, arXiv:2505.14172.

Maximilian Dax, Jonas Wildberger, Simon Buchholz, Stephen R. Green, Jakob H. Macke, and Bernhard Schölkopf. 2023. Flow matching for scalable simulation-based inference. *Preprint*, arXiv:2305.17161.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai,

and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. 2020. Residual energy-based models for text generation. *Preprint*, arXiv:2004.11714.

Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. 2025. Variational flow matching for graph generation. *Preprint*, arXiv:2406.04843.

Tairan Fu, Raquel Ferrando, Javier Conde, Carlos Arriaga, and Pedro Reviriego. 2024. Why do large language models (llms) struggle to count letters? *Preprint*, arXiv:2412.18626.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. 2025. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Preprint*, arXiv:2502.05171.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, Lexi Walker, Cosmin Paduraru, Christina Sorokin, and 1118 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Alexi Gladstone, Ganesh Nanduru, Md Mofijul Islam, Peixuan Han, Hyeonjeong Ha, Aman Chadha, Yilun Du, Heng Ji, Jundong Li, and Tariq Iqbal. 2025. Energy-based transformers are scalable learners and thinkers. *Preprint*, arXiv:2507.02092.

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

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *Preprint*, arXiv:2305.14992.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *Preprint*, arXiv:2412.06769.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Vincent Hu, Di Wu, Yuki Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees Snoek. 2024. Flow matching for conditional text generation in a few sampling steps. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 380–392, St. Julian's, Malta. Association for Computational Linguistics.

Hai Huang, Yann LeCun, and Randall Balestriero. 2025. Llm-jepa: Large language models meet joint embedding predictive architectures. *Preprint*, arXiv:2509.14252.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *Preprint*, arXiv:2406.00515.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. 2025. Mercury: Ultra-fast language models based on diffusion. *Preprint*, arXiv:2506.17298.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. *Preprint*, arXiv:2405.05417.

LCM team, Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R. Costa-jussà, David Dale, Hady Elsahar, Kevin Heffernan, João Maria Janeiro, Tuan Tran, Christophe Ropers, Eduardo Sánchez, Robin San Roman, Alexandre Mourachko, and 2 others. 2024. Large concept models: Language modeling in a sentence representation space. *Preprint*, arXiv:2412.08821.

Liang Li, Ruiying Geng, Chengyang Fang, Bing Li, Can Ma, Binhua Li, and Yongbin Li. 2023a. Plan-then-seam: Towards efficient table-to-text generation. *Preprint*, arXiv:2302.05138.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Preprint*, arXiv:2205.14217.

Yifan Li, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023b. Diffusion models for non-autoregressive text generation: A survey. *Preprint*, arXiv:2303.06574.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2023. Flow matching for generative modeling. *Preprint*, arXiv:2210.02747.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. Llm+p: Empowering large language models with optimal planning proficiency. *Preprint*, arXiv:2304.11477.

Anamika Lochab and Ruqi Zhang. 2025. Energy-based reward models for robust language model alignment. *Preprint*, arXiv:2504.13134.

Justin Lovelace, Varsha Kishore, Yiwei Chen, and Kilian Q. Weinberger. 2024. Diffusion guided language modeling. *Preprint*, arXiv:2408.04220.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *Preprint*, arXiv:2402.17753.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech 2010*, pages 1045–1048.

Duke Nguyen, Du Yin, Aditya Joshi, and Flora Salim. 2025. Spectraformer: A unified random feature framework for transformer. *ACM Transactions on Intelligent Systems and Technology*.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025. Scaling up masked diffusion models on text. *Preprint*, arXiv:2410.18514.

Gospel Ozioma Nnadi and Flavio Bertini. 2024. Survey on abstractive text summarization: Dataset, models, and metrics. *Preprint*, arXiv:2412.17165.

Zhijian Ou. 2024. Energy-based models with applications to speech and language processing. *Foundations and Trends® in Signal Processing*, 18(1–2):1–199.

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text generation with macro planning. *Preprint*, arXiv:2102.02723.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *Preprint*, arXiv:2001.04063.

Dhruv Rohatgi, Adam Block, Audrey Huang, Akshay Krishnamurthy, and Dylan J. Foster. 2025. Computational-statistical tradeoffs at the next-token prediction barrier: Autoregressive and imitation learning under misspecification. *Preprint*, arXiv:2502.12465.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

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

Thomas Schmied, Jörg Bornschein, Jordi Grau-Moya, Markus Wulfmeier, and Razvan Pascanu. 2025. Llms are greedy agents: Effects of rl fine-tuning on decision-making abilities. *Preprint*, arXiv:2504.16078.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *Preprint*, arXiv:1508.07909.

Alexander Shabalin, Viacheslav Meshchaninov, Egor Chimbulatov, Vladislav Lapikov, Roman Kim, Grigory Bartosh, Dmitry Molchanov, Sergey Markov, and Dmitry Vetrov. 2025. Tencdm: Understanding the properties of the diffusion model in the space of language model encodings. *Preprint*, arXiv:2402.19097.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *Preprint*, arXiv:1911.02150.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 432 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.

Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. 2021. Plan-then-generate: Controlled data-to-text generation via planning. *Preprint*, arXiv:2108.13740.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Interspeech 2012*, pages 194–197.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. 2025. Llm pretraining with continuous concepts. *Preprint*, arXiv:2502.08524.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025. Plangenllms: A modern survey of llm planning capabilities. *Preprint*, arXiv:2502.11221.

Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. 2025. Energy-based diffusion language models for text generation. *Preprint*, arXiv:2410.21357.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *Preprint*, arXiv:2410.14157.

Yongjing Yin, Junran Ding, Kai Song, and Yue Zhang. 2024. Semformer: Transformer language models with semantic planning. *Preprint*, arXiv:2409.11143.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Josh Susskind, and Navdeep Jaitly. 2024. Planner: Generating diversified paragraph via latent language diffusion model. *Preprint*, arXiv:2306.02531.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, and 14 others. 2025. A survey on latent reasoning. *Preprint*, arXiv:2507.06203.

Hao Zou, Zae Myung Kim, and Dongyeop Kang. 2023. A survey of diffusion models in natural language processing. *Preprint*, arXiv:2305.14671.

A Scope

This survey was conducted through a systematic review of recent literature on alternatives to NTP in language models. We searched major NLP venues including ACL, EMNLP, ICLR, NeurIPS, and ICML, as well as prominent preprint repositories (arXiv) covering the period from 2020 to 2025. Our inclusion criteria focused on methods that explicitly challenge or modify the standard NTP paradigm. This departure from NTP includes approaches that move beyond predicting individual subword tokens to instead generate or predict "larger-than-token" units such as multiple tokens simultaneously, sentence-level representations, or global structural plans. We explicitly exclude attention mechanism variants that optimize the existing token-by-token framework such as Grouped Query Attention (GQA), Multi-Query Attention (MQA), and other architectural optimizations surveyed by Brauwers and Frasincar (2023) and Roy et al. (2021). While these innovations improve computational efficiency, they fundamentally operate within the NTP paradigm, whereas our focus is on methods that challenge the token-level generation paradigm itself. Papers were excluded if they focused solely on computational efficiency improvements, retrieval augmentation or prompt engineering.

We restrict the scope of this survey to alternatives that operate within the Transformer architecture, since the Transformer remains the dominant backbone for modern language models and the locus of most active work on alternatives to NTP. For completeness, we briefly discuss Non-Transformer Architecture approaches (e.g., State Space Models and JEPAs) that depart from the Transformer backbone in Appendix C.

Based on our survey, we categorise the in-scope alternatives to NTP into four broad families (shown in Figure 2):

- **MTP** changes the prediction target from a single token to a block of future tokens, typically using a shared context representation to feed multiple output heads.
- **PtG** introduces a distinct, two-stage process where a global, high-level plan is generated upfront to guide the subsequent token-by-token generation.
- **LR** shifts the core autoregressive process into a continuous latent space, where the model's primary task is to generate an evolving sequence of latent states. These latent states are then used to predict the next sequence of tokens.
- **CG** replace left-to-right autoregressive generation with parallel, iterative refinement processes over a *global representation* of the entire output, including diffusion (illustrated in Figure 3), flow matching, and energy-based methods. Instead of autoregressive token prediction, tokens are produced through multiple rounds of simultaneous refinement across all positions in the output window.

Table 1 (main text) lists key approaches per family (Illustration in Figure 2). Our goal is to provide a roadmap for researchers exploring how language models might move beyond token-level generation and toward models that can plan, reason, and communicate more like humans.

While survey papers exist for each of these families, to our knowledge, no paper has explicitly unified these into a common taxonomy within the scope of alternatives to NTP. Table 7 lists representative prior surveys aligned to each family, demonstrating that substantial prior work has examined individual categories while no existing survey unifies all four. These works validate the importance of individual approaches but do not examine the common thread that unites them: the fundamental

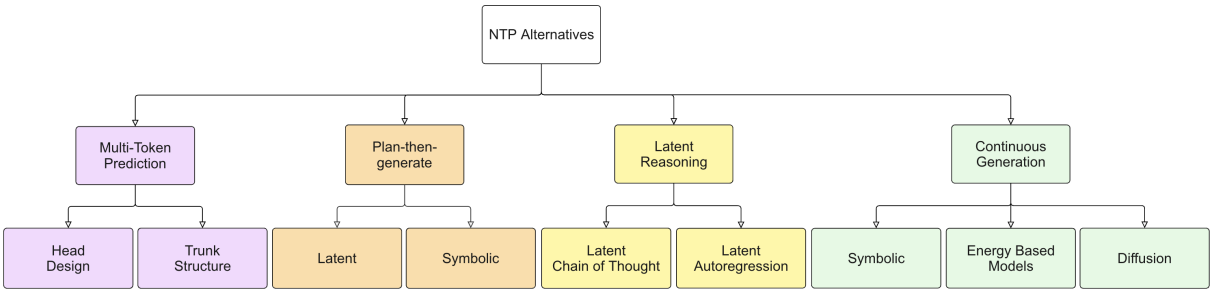

Figure 2: Families of Alternatives to NTP

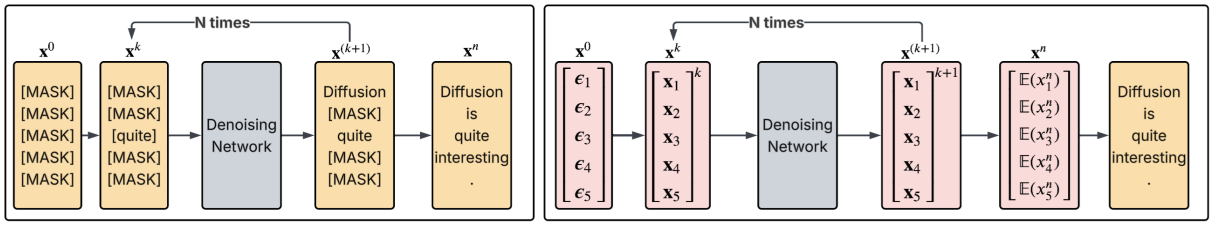

Figure 3: Comparison of text diffusion models. Left: **Discrete Text Diffusion Model** starts with masked tokens and iteratively fills them in over N steps. Right: **Continuous Text Diffusion Model** starts with random noise in an embedding space and iteratively denoises the latent vectors, which are then decoded into the final text.

challenge to token-by-token generation that motivates our taxonomic framework.

B Common Failure Modes

While the methods surveyed in this paper differ in their mechanics, their reported weaknesses recur. These failure modes have been clustered into the five common themes below which are analysed in each family's "Failure Modes" subsection so the reader can compare like with like.

Interpretability. Once generation leaves the token surface, it becomes difficult to inspect, debug, or steer. Errors are hard to localise because there is no human-readable intermediate state to examine.

Training Brittleness. Most alternatives require specialised training pipelines such as multiple pre-trained components, auxiliary losses, custom schedulers, or coupled encoder–decoder pairs. These pipelines are expensive to set up, sensitive to hyperparameters, and prone to component-level mismatches (e.g., a planner that does not pair cleanly with its generator).

Representation Collapse. Methods that operate over learned representations (latent vectors, joint embeddings, distilled features) risk degenerate solutions where the representation space loses semantic structure.

Costs (computational, inference, memory). While many alternatives are *motivated* by NTP's inefficiency, several reintroduce costs at inference. The cost–quality tradeoff is rarely strictly better than autoregressive decoding.

Output-Form Constraints. Several alternatives constrain *what* can be produced due to fixed-length outputs, rigid symbolic schemas, shallow planning horizons, or no explicit token-level output at all. These limits affect open-ended generation tasks and restrict applicability outside the regimes the methods were designed for.

C Non-Transformer Architectures

Because the main survey restricts its scope to alternatives that operate within the Transformer architecture, we discuss Non-Transformer Architecture approaches here for completeness. These methods either change the model backbone itself or replace the prediction target outside the Transformer altogether.

While the approaches surveyed in the main text modify the generative objective within Transformers, this category targets a deeper shift: changing the model architecture itself, or replacing the prediction target outside the Transformer altogether. Non-Transformer Architectures (NTA) bypass the limitations of NTP by adopting fundamentally different sequence modeling mechanisms.

Paper Name	Category
Large Language Models for Planning: A Comprehensive and Systematic Survey (Cao et al., 2025)	**Plan-then-Generate**
PlanGenLLMs: A Modern Survey of LLM Planning Capabilities (Wei et al., 2025)	**Plan-then-Generate**
A Survey of Diffusion Models in Natural Language Processing (Zou et al., 2023)	**Continuous Generation Approaches**
Diffusion Models for Non-autoregressive Text Generation: A Survey (Li et al., 2023b)	**Continuous Generation Approaches**
Energy-Based Models with Applications to Speech and Language Processing (Ou, 2024)	**Continuous Generation Approaches**
A Survey on Latent Reasoning (Zhu et al., 2025)	**Latent Reasoning**
Next Token Prediction Towards Multimodal Intelligence: A Comprehensive Survey (Chen et al., 2024)	**Multi-Token Prediction (sectional coverage)**

Table 7: Survey papers aligned with the families defined in this paper.

Two distinct departures. The methods grouped here are not all "alternatives to NTP" in the same sense, and we explicitly distinguish between two categories:

- **Architectural alternatives that reduce the cost of NTP (RNNs, SSMs).** These models still train with a next-token objective and decode autoregressively, but replace the Transformer's quadratic self-attention with linear-time sequential mechanisms. They do not change *what* the model predicts; they change the *cost* of predicting it. Mamba (Gu and Dao, 2024), for instance, is trained with standard NTP. Including such methods is justified because the prohibitive cost of token-level decoding is one of the core motivations for moving beyond NTP (Section 2.2); a sub-quadratic backbone removes one of NTP's strongest weaknesses without otherwise altering the paradigm.

- **Objective alternatives that replace token prediction (JEPAs).** These models genuinely abandon the next-token objective: they predict masked or contextual *embeddings* rather than tokens. JEPAs (Assran et al., 2023; Bardes et al., 2024; Baevski et al., 2022; Huang et al., 2025) are therefore the only methods in this section that depart from NTP at the objective level rather than the architecture level.

The two categories address different weaknesses of NTP and have different implications for adoption: SSMs are drop-in cost reductions that can in principle plug into any AR training pipeline, whereas JEPA-style objectives require rethinking how the model is supervised and how outputs are generated.

C.1 Representative Papers

Recurrent Neural Networks

(Architectural alternative; still trained and decoded with NTP.)

Recurrent Neural Networks (RNNs) were the state-of-the-art NLP architecture before the Transformer (Mikolov et al., 2010). These models, such as LSTMs (Sundermeyer et al., 2012) and GRUs (Chung et al., 2014), process input sequentially, maintaining a hidden state that summarizes prior context. Unlike transformers, which compute attention over the entire sequence, RNNs make predictions using only this evolving summary. This results in linear time complexity and an implicit latent-space abstraction over all observed inputs.

However, RNNs struggle with vanishing gradients and limited memory, making them hard to scale for long-range dependencies. Recent work revisits this paradigm with more scalable variants, such as state space models.

State Space Models

(Architectural alternative; still trained and decoded with NTP. Mamba and its descendants change the cost *of NTP, not the objective.)*

State Space Models (SSMs) improve on RNNs by tracking sequence information using continuous signals instead of discrete steps. Modern implementations like **Mamba** (Gu and Dao, 2024) achieve linear-time inference while retaining strong modeling capacity. Crucially, Mamba is still trained with a next-token objective; its contribution is removing the $\mathcal{O}(n^2)$ attention cost that makes NTP expensive on long sequences, not replacing token prediction.

SSMs address key Transformer NTP bottlenecks:

- **Efficiency:** Linear scaling with sequence length avoids $\mathcal{O}(n^2)$ attention cost.
- **Abstraction:** Continuous state evolution allows modeling over higher-level semantic units.

However, they remain underexplored in open-ended generation and bring their own challenges:

- **Limited adoption:** Few large-scale applications in general-purpose NLP.
- **Opaque internals:** Dynamics are difficult to inspect or interpret.
- **Architectural fragility:** Sensitive to hyperparameters and discretization schemes.

While SSMs rethink temporal modeling, the next category, JEPAs, rethinks **prediction itself**, moving away from token-level outputs entirely.

Joint Embedding Predictive Architectures

(Objective alternative; the prediction target is no longer a token. This is the only genuinely non-NTP family in this section.)

A more radical departure from NTP is offered by **Joint Embedding Predictive Architectures (JEPAs)**, which model sequences via compatibility in a shared semantic space.

In vision, models like **I-JEPA** (Assran et al., 2023) and **V-JEPA** (Bardes et al., 2024) learn by reconstructing masked representations rather than raw inputs. This shifts supervision from low-level pixel-perfect reconstruction to high-level semantic consistency.

Data2vec (Baevski et al., 2022) uses a similar approach across 3 separate domains of NLP, vision and audio by predicting contextual embeddings of masked inputs using a self-distillation loss. Without predicting tokens explicitly, it achieves competitive results across all 3 domains of language, vision, and speech.

LLM-JEPA (Huang et al., 2025) extends JEPA principles directly to language model training by treating different modalities (text descriptions and code) as views of the same underlying knowledge. Unlike traditional NTP objectives, LLM-JEPA learns representations by ensuring that semantically equivalent text-code pairs can predict each other in embedding space, while preventing dimensional collapse. The primary limitation of LLM-JEPA is its reliance on naturally paired multi-view data (such as documentation-code pairs), which restricts its application compared to standard self-supervised objectives. Despite this constraint, the method achieves significant improvements over baseline finetuning across multiple benchmarks (NL-RX, GSM8K, Spider) and model families (Llama3, Gemma2), demonstrating that JEPA's success in vision can transfer to language when suitable multi-view datasets exist.

C.2 Failure Modes

Output-Form Constraints: difficult to adapt to generative tasks without auxiliary decoders.

Representation Collapse: risk of collapse or over-smoothing during distillation.

Evaluation Difficulty: embedding losses do not correlate cleanly with generative quality.

C.3 Discussion

Together, SSMs and JEPAs demonstrate two orthogonal routes for moving beyond standard NTP-on-Transformers, but they are routes of different kinds. SSMs change the cost structure of NTP without abandoning it: they are best understood as a more scalable substrate that an NTP-trained model can run on. JEPAs change the supervision signal itself, predicting embeddings rather than tokens, and are the only family in this section that genuinely departs from token-level prediction at the objective level. Both remain early-stage and face barriers in interpretability, generalization, and—for JEPAs in particular—integration into practical *generation* systems, since predicting embeddings does not by itself yield decoded text and requires auxiliary mechanisms to produce output sequences.