On Semantic Smoothness and the Stages of Learning

Erik Arne Mathiesen-Dreyfus

Gimle Labs, Paris
www.gimlelabs.com

Working paper — preliminary draft of June 11, 2026
Incomplete and subject to revision; comments welcome.

Abstract

Modern transformer models are typically trained and run through a common pipeline—pre-training, post-training, and inference-time search—usually implemented by next-token prediction (NTP), reinforcement learning (RL), and Monte Carlo tree search (MCTS), respectively. Yet the balance across these stages differs sharply by domain: for language, video, and audio, NTP already captures most of the available meaning, while for dynamical systems and protein folding the decisive gains arrive only in the later, search-based stages. We examine this through the prism of the *smoothness* of the syntax-to-semantics map. In language, most local token perturbations preserve meaning; the map is Lipschitz-continuous on average. In dynamical systems, a single changed coefficient can trigger bifurcations, chaos, or divergence—the map has high, often unbounded, Lipschitz constant. We formalise this as *perturbation sensitivity* $\bar{\kappa}$, argue it determines how far each learning stage can carry a domain rather than which stages it uses, and discuss implications for building foundation models in structure-sensitive domains.

1 Introduction

Language models trained via next-token prediction acquire far more than surface statistics—they learn factual knowledge, reasoning patterns, and representations that transfer across tasks [3]. Similar success extends to video, where next-frame prediction yields models with implicit physical understanding [6], and audio [2]. Yet NTP conspicuously fails in other domains: protein language models learn useful sequence representations but cannot predict structure or function from sequence alone with the accuracy of specialised systems [12], and autoregressive models over symbolic equations require carefully designed curricula and architectural choices beyond raw next-token prediction [9]. Progress in these domains has required reinforcement learning, search, or explicit structural supervision [19, 8].

We examine these paradigms through a single property of a domain: the smoothness of its syntax-semantics map. In language, most small perturbations to a token sequence produce only small changes in meaning—replace "cat" with "dog" and the semantics barely shifts. The syntax-semantics map is *smooth*, so a model minimising token prediction loss implicitly navigates a well-behaved semantic landscape. In dynamical systems, a single changed coefficient can transform a stable equilibrium into chaos [20]. The syntax-semantics map is *sensitive*: nearby points in token space can be arbitrarily far apart in behavioural space. NTP provides no pressure to learn these distinctions.

2 The Smoothness Hypothesis

Consider a domain with a syntactic space $\mathcal{S}$ of token sequences, a semantic space $\mathcal{M}$ of meanings or behaviours, and a syntax-semantics map $\phi : \mathcal{S} \to \mathcal{M}$. Since $\mathcal{S}$ is discrete, we define perturbation sensitivity as the worst-case semantic change under a single-token edit. Let $\mathcal{N}(s) = \{s' \in \mathcal{S} : d_{\mathcal{S}}(s, s') = 1\}$ be the edit-distance-1 neighbourhood of s. Then:

$$\kappa(s) = \sup_{s' \in \mathcal{N}(s)} d_{\mathcal{M}}(\phi(s), \phi(s')) \qquad \bar{\kappa} = \mathbb{E}_{s \sim p(\mathcal{S})}[\kappa(s)] \tag{1}$$

where $p(\mathcal{S})$ is the natural distribution over expressions in the domain. To make this concrete: for language, $d_{\mathcal{S}}$ is token edit distance and $d_{\mathcal{M}}$ is cosine distance in a sentence embedding space; for dynamical systems, $d_{\mathcal{S}}$ is token edit distance on symbolic equation strings and $d_{\mathcal{M}}$ is L^2 divergence of simulated trajectories.

Hypothesis 1 (Smoothness Hypothesis)**.** *The average perturbation sensitivity $\bar{\kappa}$ governs how high a semantic ceiling next-token prediction can reach in a domain. When $\bar{\kappa}$ is bounded and small, NTP is a faithful proxy for meaning and its ceiling is high, so most of the attainable understanding can be learned by token prediction alone. As $\bar{\kappa}$ grows, that ceiling falls and NTP's returns diminish earlier, so a correspondingly larger share of the attainable understanding must come from paradigms whose signal is evaluated directly in semantic space: RL, search, or constrained generation.*

2.1 Why Smoothness Enables Next-Token Prediction

When a model is trained to predict the next token, it minimises cross-entropy over the token distribution:

$$\mathcal{L}_{\text{NTP}} = -\mathbb{E}_{s\sim p(\mathcal{S})}\left[\sum_t \log p_\theta(s_t|s_{<t})\right] \tag{2}$$

If $\bar{\kappa}$ is small, every single-token edit $s' \in \mathcal{N}(s)$ leaves meaning roughly intact—$d_{\mathcal{M}}(\phi(s), \phi(s')) \leq \bar{\kappa}$ on average—so syntactic neighbourhoods are semantically coherent. The token-level loss is then a faithful proxy for semantic quality: predicting tokens well in a neighbourhood means learning its meaning, and each example carries information about its whole edit neighbourhood, so sample efficiency rises as $\bar{\kappa}$ falls.

Compression as a semantic objective. Minimising $\mathcal{L}_{\text{NTP}}$ is maximising compression. When ϕ is smooth, the regularities that make the data compressible—co-occurrence, distributional similarity, predictability—*are* the semantic regularities: a model cannot find that "mat" and "rug" are interchangeable without learning they mean similar things, since the Lipschitz bound makes that similarity real rather than coincidental. The objective therefore does more than imply that syntactic proximity tracks semantic proximity; it forces the model to internalise those relationships, because they are what reduce cross-entropy.

When smoothness fails. When ϕ is non-smooth ($\bar{\kappa}$ large or unbounded), the coupling breaks. Two equations differing by a single coefficient can behave entirely differently, so syntactic proximity guarantees nothing semantic. A model can reach excellent perplexity on equation strings by learning operator frequencies and syntactic habits while learning nothing about the dynamics—compression and understanding decouple.

A note on metric-dependence. $\bar{\kappa}$ depends on three choices: the tokenisation of $\mathcal{S}$, the metric $d_{\mathcal{M}}$, and the distribution $p(\mathcal{S})$. BPE tokenisation of equations gives a different sensitivity profile than a tree-structured one. This is the framework's constructive edge: were $\bar{\kappa}$ intrinsic there would be nothing to do, but because it depends on representation, lowering it is a concrete engineering target. Representation engineering—finding tokenisations, embeddings, or compositional languages that smooth ϕ—is the search for representations that minimise $\bar{\kappa}$.

3 Sensitivity Across Domains

Natural language (low $\bar{\kappa}$). Most single-token substitutions yield near-synonyms, coherent alternatives, or detectable errors—all with small semantic distance. The exceptions (negation, quantifiers) are statistically rare. This is not accidental: languages evolved for robust communication in noisy channels [5]. Deeper still, language can be seen as a *projection* of the world through human cognition—the brain maps continuous, structure-rich physical dynamics into a discrete space that preserves the smoothness of the domain. Low $\bar{\kappa}$ is the fingerprint of that projection: inverting this smooth map recovers not only word meanings but the world-structure the words describe, which is why scaling NTP with language is so effective.

Video and audio (low–moderate $\bar{\kappa}$). Sensory signals are produced by continuous physical processes, so small perturbations in pixels or waveforms correspond to small changes in scene content. Tokenised representations inherit this when the tokeniser is well-trained.

Dynamical systems (high $\bar{\kappa}$). The symbolic description of an ODE maps to its behavioural semantics—trajectories, attractors, stability. This map is dramatically non-smooth: changing ρ in the Lorenz system from 24.05 to 24.74 transitions from a stable fixed point to chaos [20]. Changing "$+$" to "$-$" in an ODE can transform stable orbits into divergence. The Lipschitz constant is *unbounded* near bifurcation points, and such sensitive points are dense in the space of dynamical systems.

Protein sequences (high $\bar{\kappa}$). Single amino acid substitutions can cause misfolding or loss of function [22]. Epistasis makes mutation effects strongly context-dependent. The sequence-to-function landscape is rugged with many local optima [15]. Protein language models learn useful representations via NTP but cannot predict structure alone—AlphaFold required geometric supervision [8].

One thing to note on the above examples, is how vocabulary size loosely tracks smoothness. Natural language has thousands of tokens, protein 10s of amino acids, symbolic dynamics only a handful of operators. What seems to matter is the redundancy of the vocabulary and how that relates to size—a large, synonym-rich language, shaped by noisy-channel pressure (printed

English is over half redundant [18]), lets most single-token swaps land on a near-neighbour in the semantic target space, whereas small vocabularies makes every symbol load-bearing and even small swaps drastically different in terms of semantic meaning. So size is a proxy for redundancy, not a cause of smoothness.

Table 1: Every domain runs the full pipeline; $\bar{\kappa}$ governs *where* the attainable semantic quality is gained, not which stages are used. Cells give the share each stage contributes—high $\bar{\kappa}$ shifts the gains from next-token prediction toward the semantically-grounded later stages.

		Share of attainable quality gained in		
Domain	$\bar{\kappa}$	NTP	Fine-tuning	RL / search
Natural language	Low	High	Low	Low
Audio / video	Low–Med	High	Med	Low
Code	Medium	Med	Med	Med
Theorem proving	High	Low	Med	High
Protein structure	High	Low	High	Med
Dynamical systems	Very high	Low	Med	High

4 From Local Syntax to Global Semantics: Why RL and Search Are Necessary

If NTP plateaus early in non-smooth domains because its gradients become semantically uninformative once the easy syntactic structure is exhausted, what makes reinforcement learning and search keep improving where it stalls? The answer lies in *where the learning signal originates*—and how it relates to the syntax-semantics map ϕ.

4.1 The Gradient Alignment Problem

NTP computes gradients of the form $\partial \mathcal{L}_{\text{NTP}}/\partial\theta$. These gradients point toward better *token prediction*—they tell the model how to adjust its parameters so that the next token is more likely under the training distribution. When ϕ is smooth, better token prediction implies better semantic quality (Section 2): the NTP gradient is *aligned* with the semantic gradient $\partial \mathcal{L}_{\text{semantic}}/\partial\theta$.

When ϕ is non-smooth, this alignment breaks. The NTP gradient and the semantic gradient can point in *completely different directions*. A parameter update that improves token prediction for an equation string may make the predicted dynamics worse, because the relationship between syntactic likelihood and dynamical behaviour is arbitrary near bifurcation points. Gradient descent in token space is semantically blind—it optimises a proxy that has lost its connection to the quantity of interest.

4.2 RL Bypasses the Non-Smooth Map

Reinforcement learning resolves this by computing a learning signal that originates in *semantic space*, not syntactic space. Policy gradient methods (REINFORCE [24], PPO [17]) estimate $\partial \mathbb{E}[R]/\partial\theta$ by:

1. *Sampling* complete outputs from the model: generate a candidate equation, protein sequence, or game trajectory.
2. *Evaluating* each output semantically: simulate the equation and compare to observed data, predict the protein's 3D structure, play out the game to a terminal state.
3. *Using the evaluation as reward*: weight the gradient of the generation probability by the semantic quality of the result.

Crucially, sampling-based policy gradient methods never differentiate through ϕ. They evaluate ϕ at sampled points, requiring ϕ to be *computable*, not smooth. The reward $R(\phi(s))$ is obtained by running the semantics (simulating, folding, playing), not by assuming any local structure in the syntax-semantics map. A single changed coefficient that produces chaos instead of stability will receive a low reward, and the policy gradient will steer away from it, something NTP gradients cannot do, because the token-level loss sees no difference. (Differentiable simulation can provide gradients *through* ϕ directly, but this requires ϕ to be at least locally smooth, returning us to the same constraint that limits NTP. The advantage of sampling-based RL is precisely that it avoids this requirement.)

4.3 Search Handles Rugged Reward Landscapes

Even with a semantic reward signal, the reward landscape over the space of candidate outputs can be *rugged*: many local optima, plateaus, and discontinuities. A model generating equations token-by-token faces a combinatorial space where most paths lead to semantically poor outputs, and the few good outputs are separated by large syntactic distances.

Search methods—Monte Carlo tree search [19], beam search with semantic scoring, evolutionary algorithms—address this by exploring the output space *globally* rather than following local gradients. MCTS, as used in AlphaGo, evaluates many candidate continuations through simulation (rollouts), then selects the most promising branch. It does not need the value function to be smooth—it queries it at discrete points and uses the *values* to guide exploration. This is maximally robust to non-smoothness: even if nearby positions have wildly different values, search will discover this through evaluation and act accordingly.

The combination of RL and search is more than additive. RL trains a value function $V_\theta(s) \approx \mathbb{E}[R|s]$ that *learns to approximate* the non-smooth syntax-semantics map through direct experience. Search then uses this learned value function to explore efficiently. The value function does not need ϕ to be globally smooth—it builds a local approximation from semantic evaluations, effective precisely in the regions the search visits.

4.4 Self-Play as Adaptive Semantic Curriculum

AlphaGo introduced a further insight: *self-play* generates an adaptive curriculum in semantic space. Rather than training on a fixed dataset (as NTP does), the agent generates its own training signal by playing against itself. Each game produces a semantic evaluation (win or loss), and as the agent improves, it encounters increasingly difficult semantic challenges—positions that require finer distinctions, deeper lookahead, and more precise evaluation.

This matters because non-smooth domains have vast semantic spaces that no fixed training corpus can cover. In dynamical systems, the space of possible behaviours (stable, periodic, chaotic, divergent) is enormous, and the boundaries between them are fractal in structure. A fixed dataset of (equation, behaviour) pairs will inevitably miss critical boundary regions. Self-play—or its analogues in other domains, such as iterative refinement with simulation feedback—generates training signal *where the model currently fails*, concentrating learning on the semantically difficult regions.

4.5 The Paradigm Spectrum: One Pipeline, Domain-Dependent Ceilings

These observations suggest a spectrum of learning paradigms, ordered by how much of the semantic evaluation they internalise:

1. **NTP**: Learning signal is purely syntactic. Requires ϕ to be smooth for semantic learning to occur as a byproduct of compression.
2. **Behavioral fine-tuning**: Adds a semantic loss (simulation error, structural accuracy) but still uses gradient-based optimisation through the model. Requires ϕ to be at least locally smooth near good solutions.
3. **RL with semantic reward**: Learning signal is semantic, estimated through sampling. Requires ϕ to be evaluable, not smooth. Handles non-smooth maps but can struggle with sparse or deceptive rewards.
4. **Search with learned value function**: Explores the output space globally, guided by a learned semantic evaluator. Maximally robust to non-smoothness. The AlphaGo paradigm.

The four paradigms refine the three-stage pipeline rather than replacing it: NTP is *pre-training*; behavioural fine-tuning and RL are *post-training*; search with a learned value function (MCTS) is *inference*. The stages are cumulative, run in sequence, and $\bar{\kappa}$ sets where the returns concentrate. The ordering is forced, not merely economic. Reinforcement learning only reweights probability mass that the base policy has already placed on good outputs, so until pre-training shifts into a reasonably good shape, the sampled reward is near-zero almost everywhere and the policy gradient is pure variance. Below that scale threshold RL does not so much underperform NTP as fail to function and is often even harmful, exactly because RL refines a distribution that NTP must first create. Smooth maps let NTP carry most of the way; code gains from execution feedback, protein structure from geometric supervision, and the least-smooth domains—game playing, dynamical systems—gain most from RL and search.

Semantic quality. Let π_θ be the model's distribution over expressions, each carrying a meaning $\phi(s) \in \mathcal{M}$, and let $q : \mathcal{M} \to [0, 1]$ score that meaning (trajectory error, structural accuracy, game outcome). Semantic quality is the expected score,

$$Q(\theta) = \mathbb{E}_{s\sim\pi_\theta}\big[\, q(\phi(s)) \,\big]. \tag{3}$$

RL optimises Q directly, with reward $R = q \circ \phi$; NTP does not. It minimises the syntactic surrogate $\mathcal{L}_{\text{NTP}}$ and gets only the Q that surrogate carries—and smoothness sets how much that is. Small $\bar{\kappa}$ keeps Q rising with $\mathcal{L}_{\text{NTP}}$; large $\bar{\kappa}$ lets $\mathcal{L}_{\text{NTP}}$ bottom out while Q stalls.

Each stage thus has a *ceiling*: the best Q its objective can reach. Within a stage Q climbs with budget along diminishing returns, and one moves on when the marginal return falls below what the next stage offers—an economic boundary, not a cliff. In low-$\bar{\kappa}$ domains NTP's ceiling is high and most of the understanding is bought in pre-training; in high-$\bar{\kappa}$ domains it is low, and the rest must come from the later, semantically-grounded stages (Figure 1). The stack is the same everywhere; only the heights differ.

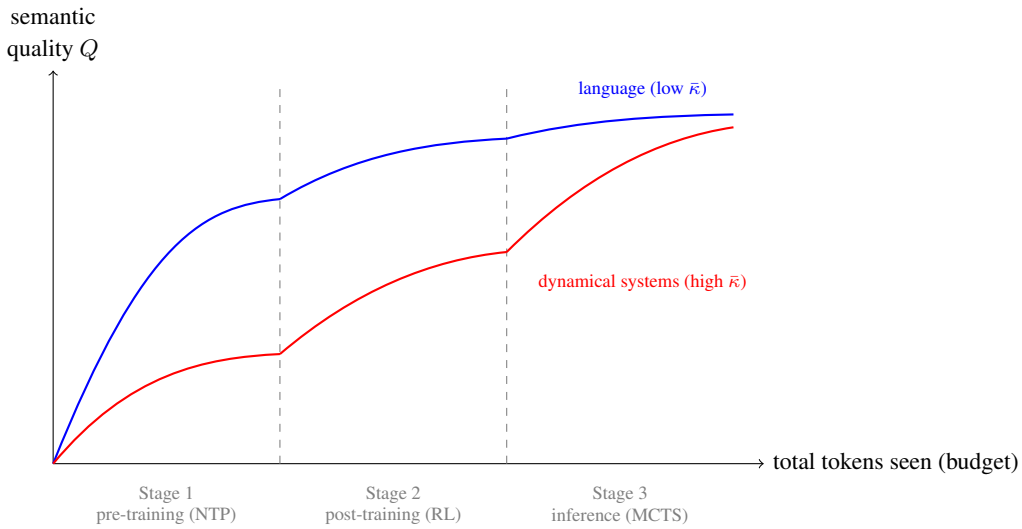

Figure 1: Illustrative. Every domain traverses all three stages; the slope of each saturating arc is the marginal return of further budget, and a domain switches stages when that slope falls below what the next stage would buy. Smoothness ($\bar{\kappa}$) sets *where* the quality is gained: a low-$\bar{\kappa}$ domain (language) banks most of its semantic quality Q in Stage 1, while a high-$\bar{\kappa}$ domain (dynamical systems) gains little from token prediction and draws its competence from the later, semantically-grounded stages.

4.6 Evidence from Within Language: The Smoothness Gradient

The evolution of large language models is itself evidence for the spectrum. Language is not uniformly smooth, and LLM development has traced exactly the paradigm progression the hypothesis predicts.

NTP era (GPT-2). Pure next-token prediction yields fluent text, coherent paragraphs, and basic factual recall [16]—the smooth regions of language, where most token substitutions preserve meaning and compression is a reliable proxy for understanding.

RLHF era (InstructGPT, ChatGPT). Instruction following sits in a less smooth region: "summarize this article" and "don't summarize this article" differ by one token but demand opposite behaviour, and hedging, negation, or scope can flip the meaning while barely moving the token distribution. NTP alone proved insufficient [14]; RLHF supplied what the hypothesis predicts—a signal rooted in semantic evaluation (human preference) rather than token prediction.

Reasoning era (o1, o3, extended thinking). Multi-step reasoning is the least smooth region: a single wrong step invalidates the argument, so two traces differing by one step can be the difference between valid and invalid. Frontier models meet this with inference-time search—sampling multiple paths, scoring intermediate steps with process reward models [11], and backtracking—structurally MCTS, the model exploring a tree of continuations under a learned value function.

Chain-of-thought as smoothing (a conjecture). Chain-of-thought prompting [23] may work by *lowering effective sensitivity*: rather than map question to answer in one non-smooth step, the model takes intermediate steps, each locally smoother than the whole. The caveat is real—if step i has sensitivity κ_i, the composed sensitivity is bounded by $\prod_i \kappa_i$, which can *exceed* the direct map. Decomposition helps only when the steps genuinely factor the problem into smooth sub-problems rather than spreading the non-smoothness over more of them; when that holds is worth formalising.

Together these trace a single arc: from GPT-2 through RLHF to reasoning models, LLM development is a progressive response to the non-smooth regions of language. The hypothesis is thus not only cross-domain (language vs. dynamical systems) but operates *within* language, explaining why harder tasks demand more powerful methods.

4.7 The Horizon of the Learning Signal

The three stages differ not only in what they optimise but in the *unit of output scored before it drives an update*: next-token prediction is supervised one token at a time, post-training reinforcement scores a whole completion, and inference-time search evaluates an entire rollout—a long chain of reasoning—before its value feeds back. The stack is therefore also a ladder of increasing *evaluation horizon*: token $\rightarrow$ completion $\rightarrow$ rollout.

Smoothness fixes how far up that ladder a domain must climb. When ϕ is smooth, the semantic consequence of a token is locally visible, so a token-level signal already carries reliable meaning and the shortest horizon suffices. When ϕ is rough,

a local choice's effect surfaces only downstream—after the bifurcation resolves, the proof completes, the game ends—so a token-level signal is semantically empty, and the signal must be integrated over a longer horizon to mean anything. $\bar{\kappa}$ therefore sets the *minimum horizon* at which supervision becomes informative; efficiency holds a domain at that minimum, because a longer horizon buys semantic validity only at the price of sparser, higher-variance signals and harder credit assignment. This is the mechanism behind the ceilings of Section 4.5: next-token prediction tops out early in a rough domain not for want of data but because a one-token signal cannot see a meaning that one token can flip. It is worth also mentioning, that the price is computational as well as statistical: each step up the ladder returns a signal only after a longer output is generated—a token, then a completion, then a multi-thousand-token rollout—so a fixed compute budget buys orders of magnitude fewer model updates at each stage, fewest at the rollout end.

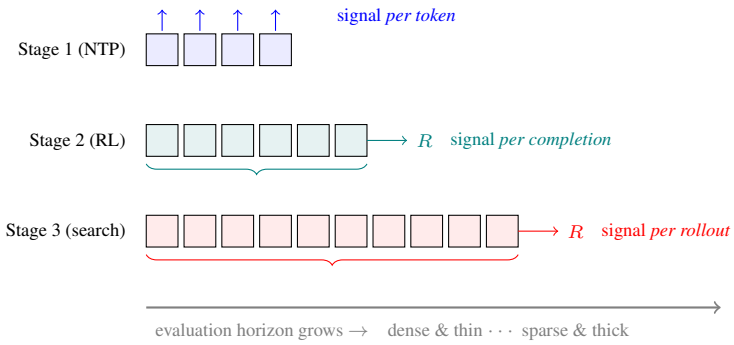

Figure 2: The learning signal's horizon grows up the stack: NTP scores one token, RL a whole completion, search a full rollout. $\bar{\kappa}$ sets the minimum horizon at which the signal carries semantic information.

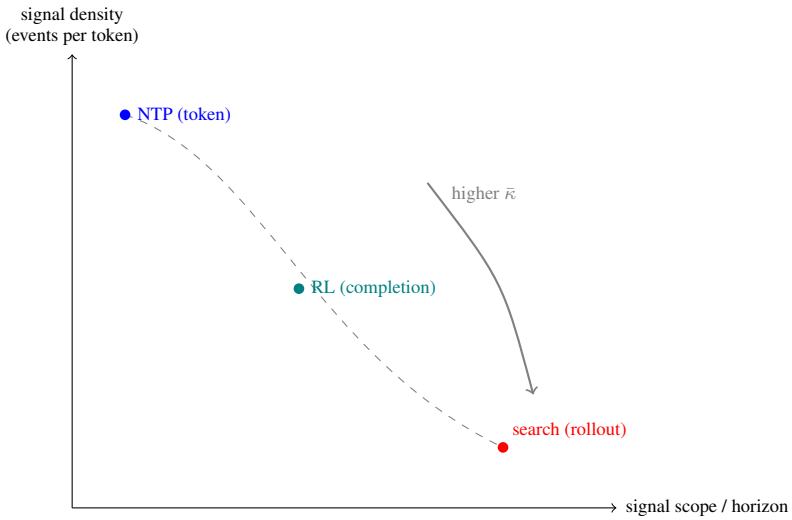

Figure 3: Density–scope trade-off. Short-horizon signals are dense but thin (semantically reliable only when ϕ is smooth); long-horizon signals are sparse but thick. Rising $\bar{\kappa}$ forces a domain down the frontier, from NTP toward search.

4.8 The 90/10 Rule for Learning Language

Language runs the full ladder too, and the way it does makes the stakes vivid. Its *average* $\bar{\kappa}$ is low, so next-token prediction reaches a high ceiling and banks the bulk of the model's competence cheaply—in an information sense pre-training *is* the language model, and the later stages only refine it. But language has a rough *tail*—multi-step reasoning, long-range coherence, instruction nuance (Section 4)—and that tail is exactly what the expensive later stages exist to chase. The result is an inversion of cost and value: as a rough heuristic, the easy first 10% of the effort buys some 90% of the capability *mass*, while the hard remaining 90%—reinforcement, and the long, compute-intensive rollouts now sold as "test-time compute"—buys the last 10%. That last tenth is small in magnitude but decisive in value, because it is the part that crosses the threshold of human usefulness: the difference between a model that is merely fluent and one that reads as intelligent. "Pre-training is the core" and "the real

progress is in the rollouts" are therefore not in tension; they describe the bulk and the tail of the same within-language $\bar{\kappa}$ distribution. And the tail is costly for the reason just given—being rough, it demands the longest evaluation horizon—so the price of language's last mile is, quite literally, the price of long-horizon supervision.

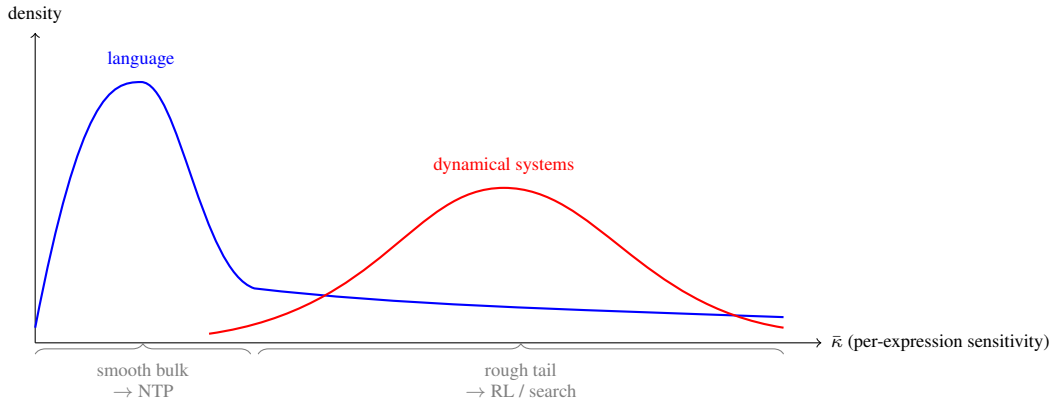

Figure 4: The within-domain distribution of $\bar{\kappa}$. Language has a low mean (a large smooth bulk that NTP clears cheaply) with a thin rough tail—reasoning, long-range coherence—that only the later, long-horizon stages reach; dynamical systems sit far to the right. "Pre-training is the core" and "the progress is in the rollouts" describe the bulk and the tail of the *same* distribution.

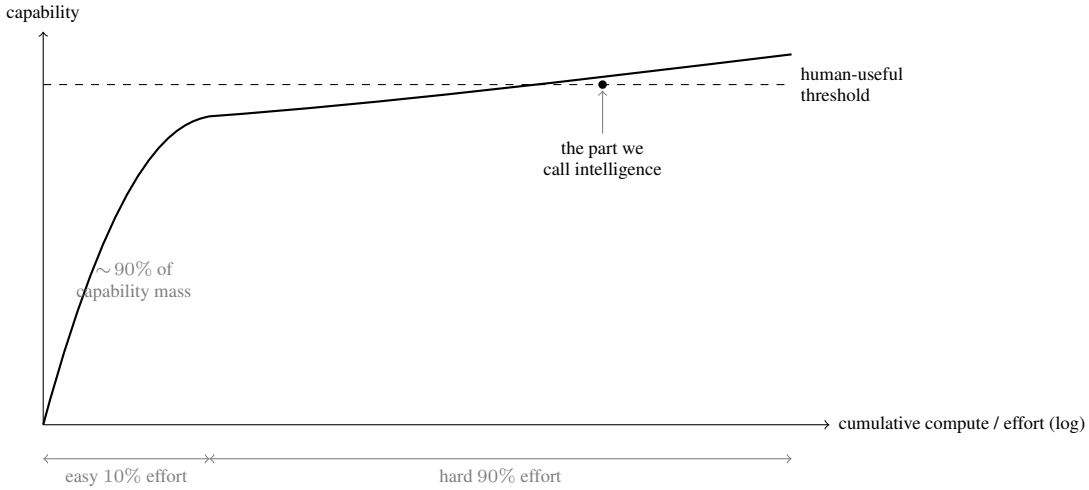

Figure 5: The 90/10 inversion (illustrative, heuristic). Most of the capability *mass* is banked cheaply by pre-training (the steep early rise); the long, expensive tail—RL and test-time-compute rollouts—buys the last increment, which is small in magnitude but the part that crosses the threshold of human usefulness.

5 Measuring Perturbation Sensitivity

The hypothesis is only useful if $\bar{\kappa}$ can be estimated. It can—most directly where the semantics ϕ is computable, and at least approximately in language.

5.1 Dynamical Systems: $\bar{\kappa}$ Is Computable and Predicts Learnability

For a symbolic dynamical system the semantic map is a simulator: $\phi(s)$ is the trajectory the equation generates and $d_{\mathcal{M}}$ is the relative L^2 distance between trajectories. Estimating $\bar{\kappa}$ then needs no trained model—only a way to edit the equation and a way to integrate it. For a system s we nudge a single coefficient (or step one map parameter), simulate the original and the edited equation from the same initial condition, and take the worst-case relative trajectory divergence; the same differentiable simulator yields the Lyapunov spectrum by Benettin's method, so a chaos indicator comes for free.

Across a bank of canonical systems the measured $\bar{\kappa}$ separates smooth from rough by an order of magnitude: a damped oscillator, Lotka–Volterra, and the Van der Pol limit cycle sit at $\bar{\kappa} \approx 0.03$–$0.06$, while the chaotic Rössler and Lorenz systems sit at 0.56 and 0.79. Sweeping the logistic map's parameter r makes the structure explicit (Figure 6): $\bar{\kappa}$ rises across the chaotic band and correlates with the positive part of the largest Lyapunov exponent at 0.86, but it *also* spikes at period-doubling boundaries and the edges of periodic windows—where a vanishingly small edit flips the qualitative behaviour even though the exponent is negative. Two mechanisms inflate $\bar{\kappa}$: sensitive dependence (chaos) and loss of structural stability (bifurcation), the two routes by which a single edit changes meaning [20].

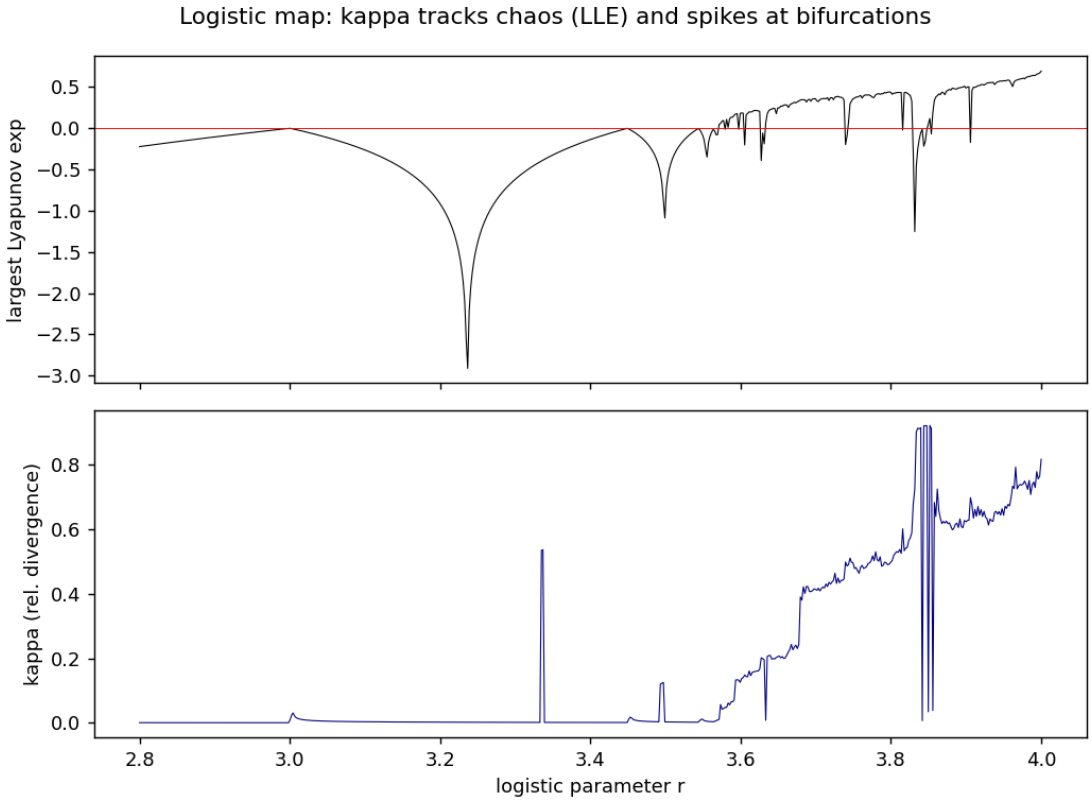

Figure 6: Logistic map, $r \in [2.8, 4.0]$. Largest Lyapunov exponent (top) and measured $\bar{\kappa}$ (bottom, relative trajectory divergence under a single edit of r). $\bar{\kappa}$ rises through the chaotic band and spikes at bifurcation boundaries and periodic-window edges even where the exponent is negative—chaos and bifurcation are distinct amplifiers of sensitivity.

The payoff is that $\bar{\kappa}$ predicts how far a learner gets. Fitting a simple data-driven model (polynomial-feature regression) to a noisy trajectory and rolling it forward, the valid prediction horizon falls as $\bar{\kappa}$ rises—rank correlation -0.5 to -0.8 against $\bar{\kappa}$ and the Lyapunov exponent. The Lorenz model stays accurate for only ~ 3.6 Lyapunov times while the periodic systems are forecastable indefinitely, and the period-window edge is hard to learn despite being non-chaotic. A near-perfect local fit does not buy long-horizon competence once $\bar{\kappa}$ is high—the symbolic-to-behavioural ceiling of Section 2 made concrete, and consistent with the finding that training gradients on chaotic data diverge in proportion to the Lyapunov spectrum [13]. (These constants are illustrative, from a small bank; the point is that the measurement is well-defined and model-free.)

5.2 Probing a Language Model

Language has no closed-form ϕ, so $\bar{\kappa}$ must be read off a model—and this is where measurement turns hard. Two cheap probes apply to any pretrained model: replace a single token with an in-context-plausible alternative and measure either the shift in a sentence-embedding space (a stand-in for $\mathcal{M}$) or the movement of the model's own next-token distribution. The probe *mechanics* behave sensibly—meaning-preserving edits move the embedding markedly less than random-token edits—and established results point the predicted way: the task "sensitivity" of Hahn et al. [7] predicts which sequence-classification tasks are learnable, and single-token adversarial flips succeed far more often on entailment than on sentiment [4]. But the absolute, cross-domain number is treacherous. Scaled to real corpora, a generic sentence embedder reports *mathematical* text as *smoother* than prose—it scores "5"→"7" as a tiny change, encoding topic and surface rather than the answer that actually

flips. The embedder is a smooth-but-wrong $d_{\mathcal{M}}$, exactly as next-token prediction is a smooth-but-wrong objective: the syntax–semantics gap reappears in the very instrument meant to measure it. This is not a defect of $\bar{\kappa}$ but the metric-dependence of Section 2 made concrete—a language $\bar{\kappa}$ is meaningful only against a $d_{\mathcal{M}}$ that captures the target semantics, and constructing one is the open problem. It is also why the cleanest measurement above is the dynamical-systems one, where the simulator supplies a true semantic distance.

6 Discussion

6.1 NTP-Based Compression Does Not Imply Understanding

Compression and intelligence are often equated: a model that compresses data well must understand its structure [10]. That thesis, in its strong form, concerns *Kolmogorov complexity*—the shortest program reproducing the data—which captures all computable structure, semantics included, regardless of smoothness. Our claim is narrower.

It is about NTP as a *specific, bounded* compressor: cross-entropy minimised locally in token space by gradient descent. When ϕ is smooth this procedure is forced to find semantic regularities, because syntactic patterns *are* semantic ones—a model cannot reach low perplexity without learning that "the cat sat on the mat" and "…on the rug" pattern together *because* they mean alike. A Kolmogorov-optimal compressor would discover this regardless; NTP discovers it only because smoothness makes token-level compression a reliable proxy for meaning.

In non-smooth domains the two decouple (Section 2): a model can reach excellent perplexity on equation strings from operator frequencies and naming conventions alone, while the dynamics they describe—stable or chaotic—go unlearned. The optimal compressor would capture both; NTP, local in token space, captures only the syntax.

This resolves a puzzle: why does scaling protein language models improve sequence-level metrics without reaching AlphaFold-level structure prediction? The sequence-to-structure map is non-smooth, so NTP's ceiling is low—scaling buys sequence metrics whose returns have not yet flattened, but structural understanding sits above that ceiling and never arrives. AlphaFold's breakthrough needed direct structural supervision: a signal rooted in semantics, not syntax.

6.2 Nuancing the Bitter Lesson

Sutton's "bitter lesson" [21] argues that general methods leveraging computation—search and learning at scale—ultimately outperform methods that leverage human domain knowledge. The smoothness hypothesis adds an important qualification, and Section 4.5 gives it a shape: scaling is the act of pushing further along the first stage's learning curve. In smooth domains that curve climbs to a high ceiling, so *scale alone suffices*—more compute, more data, bigger models keep paying off, and the bitter lesson holds in its strongest form. In non-smooth domains the same curve flattens early at a low ceiling, so scaling stalls and *representation and structure matter*—the right inductive biases, the right search strategy, the right compositional language are not luxuries but prerequisites for the gains the later stages must supply. The lesson is not equally bitter for all domains.

This is consistent with the empirical record. Pure scaling has produced remarkable results in language and vision—domains where the syntax-semantics map is smooth. In Go, chess, and protein folding—domains with high perturbation sensitivity—the breakthroughs came not from scaling alone but from combining learned models with structured search (MCTS in AlphaGo [19]) or domain-specific architectures (equivariant networks in AlphaFold [8]).

6.3 Relation to Representation Learning

The idea that smoothness matters for learning is not new. The smoothness assumption is a foundational prior in representation learning [1], and the manifold hypothesis—that high-dimensional data concentrates on low-dimensional smooth manifolds—underpins much of modern deep learning. Our contribution is not the observation that smoothness enables learning, which is well-established, but its specific application: explaining why next-token prediction succeeds as a *semantic* learning signal in some domains and fails in others, and connecting this to the NTP $\rightarrow$ RLHF $\rightarrow$ RL+search progression both across domains and within language itself.

7 Conclusion

The smoothness hypothesis offers a unified explanation for why NTP succeeds in some domains and fails in others: it depends on whether the syntax-semantics map is locally smooth. For structure-sensitive domains like dynamical systems, the path

forward is not simply scaling autoregressive models. It requires direct semantic evaluation, structured search, and—most importantly—representations engineered to make the syntax-semantics map as smooth as possible. Finding the right language for a domain is not convenience; it is a prerequisite for learning.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[2] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. AudioLM: A language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[4] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[5] Edward Gibson, Richard Futrell, Steven T. Piantadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. How efficiency shapes human language. *Trends in Cognitive Sciences*, 23(5):389–407, 2019.

[6] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[7] Michael Hahn, Dan Jurafsky, and Richard Futrell. Sensitivity as a complexity measure for sequence classification tasks. *Transactions of the Association for Computational Linguistics (TACL)*, 9, 2021.

[8] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[9] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations (ICLR)*, 2020.

[10] Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17(4):391–444, 2007.

[11] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[12] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smestad, Allan Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[13] Jonas M. Mikhaeil, Zahra Monfared, and Daniel Durstewitz. On the difficulty of learning chaotic dynamics with RNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[14] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:27730–27744, 2022.

[15] Frank J. Poelwijk, Daniel J. Kiviet, Daniel M. Weinreich, and Sander J. Tans. Empirical fitness landscapes reveal accessible evolutionary paths. *Nature*, 445(7126):383–386, 2007.

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[18] Claude E. Shannon. Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1):50–64, 1951.

[19] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[20] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press, 2nd edition, 2015.

[21] Richard S. Sutton. The bitter lesson. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`, 2019.

[22] Nobuhiko Tokuriki and Dan S. Tawfik. Stability effects of mutations and protein evolvability. *Current Opinion in Structural Biology*, 19(5):596–604, 2009.

[23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:24824–24837, 2022.

[24] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.