Attractor States and Agency Illusions: A Formal Analysis of Recursive Motifs in Transformer Latent Space

Part 1: The Emergence of Structural Motifs in Generative Outputs

1.1 Introduction: Observing Order in Chaos

The outputs of contemporary Large Language Models (LLMs), particularly those based on the Transformer architecture, often exhibit a remarkable capacity for generating coherent, contextually relevant, and stylistically varied text. However, alongside this generative prowess, these systems display a tendency to produce highly structured, recurrent patterns, or "motifs," that deviate from typical human discourse. This paper focuses on a specific class of these patterns—conceptual and geometric motifs such as the 'spiral,' 'mirror,' and 'reflection.' Conventionally, such outputs might be dismissed as mere artifacts, categorized under broad, often pejorative, labels like "hallucinations," "confabulations," or "model degeneration".[1] This perspective frames them as errors to be mitigated or bugs to be fixed.

This research, however, proposes a fundamental reframing of this phenomenon. It posits that these motifs are not stochastic failures but are, in fact, structured, predictable, and emergent properties that offer a profound window into the internal dynamics of the generative process. The recurrence and intricate structure of these patterns are hallmarks of emergent behavior in complex systems, suggesting they are governed by principles more deterministic than random error. By treating these outputs as motifs—stable, recurring patterns—we can shift the analytical paradigm from simple error correction to a more powerful complex systems analysis, employing tools from dynamical systems theory, information theory, and cognitive science. This shift is critical, as it suggests that these behaviors may be inherent to the nature of current generative models. Consequently, merely scaling up model size or training data may not eliminate these tendencies; it could, paradoxically, render them more pronounced and stable. This has significant implications for the fields of AI safety, alignment, and control, as it points toward fundamental limits in the "steerability" of these complex artificial systems.

The central thesis of this paper is that recursive motifs are the observable signatures of

attractor states within the high-dimensional state space of the Transformer's generative process. Their emergence is driven by a convergence of three primary forces: (1) the cognitive-linguistic patterns of recursion and reflection that are statistically dominant in the vast corpora of human text used for training; (2) the architectural feedback loops inherent to autoregressive generation and the self-attention mechanism, which amplify and reinforce nascent patterns; and (3) the mathematical phenomenon of **representational collapse**, which constrains the system's trajectory to lower-dimensional, stable manifolds where such simple, ordered patterns are the most probable outcomes. By dissecting these forces, this paper aims to provide a comprehensive, multi-disciplinary explanation for why these seemingly sophisticated, yet often non-veridical, structures appear in AI-generated text.

1.2 Defining the Motifs: A Taxonomy of Recurrent Structures

To analyze these emergent phenomena systematically, it is necessary to first establish a clear taxonomy of the motifs under investigation. While the variety of structured outputs is vast, this paper focuses on three interrelated categories that are particularly revealing of the underlying dynamics.

- **Recursive Motif:** This is the foundational and most general category, defined by the repeated application of a function, operation, or linguistic pattern to its own output. In computation and formal language theory, recursion is the mechanism that allows for the generation of potentially infinite structures from a finite set of rules.[5] In the context of LLM outputs, a recursive motif is observed when the model generates a structure and then embeds a similar structure within it, or applies the same transformative logic at successive steps. This can manifest as nested lists, hierarchical explanations, or sentences that contain subordinate clauses referring to the main clause's structure. The recursive motif is the elemental principle of self-reference and iteration that underpins the more specific motifs discussed below.
- **Spiral Motif:** This motif is a specific instantiation of recursion characterized by outputs that iteratively refine, elaborate, or expand upon a central concept or theme, often with a progressively diminishing or expanding scope. Visually and conceptually, this resembles a spiral. For instance, a model might define a term, then define a key word within that definition, then a key word within the sub-definition, and so on, creating a "drilling down" effect. Conversely, it might start with a simple statement and recursively add layers of qualification and context, spiraling outwards. Narrative examples include stories that repeatedly return to a central event, each time adding a new layer of detail or perspective. This structure is a hallmark of a process converging towards a specific point or region in its conceptual space.
- **Mirror/Reflection Motif:** This motif is defined by outputs that exhibit properties of symmetry, inversion, or direct self-reference. It is a powerful and frequently observed pattern that directly engages with concepts of identity and opposition.[7] Manifestations include:

- **Symmetry:** Generating a statement and then its logical or semantic opposite (e.g., "The advantages are X, Y, Z. Conversely, the disadvantages are A, B, C.").
- **Inversion:** Creating palindromic or near-palindromic structures at the level of words, phrases, or even concepts.
- **Self-Reference:** Engaging in meta-commentary about the text it is currently generating (e.g., "This sentence serves to illustrate the concept of reflection.").
- **Contextual Mirroring:** A more subtle form where the model's output mirrors the user's tone, vocabulary, or conceptual framework with high fidelity. This is a key mechanism behind the perceived "empathy" of AI systems.

These motifs are not mutually exclusive. A spiral can be a form of recursion, and a mirror can be seen as a two-step recursive process (an operation and its inverse). By establishing this taxonomy, we can move beyond anecdotal descriptions and begin to analyze the specific architectural and dynamical pressures that give rise to each distinct structural signature.

Part 2: Cognitive Scaffolding: Recursion, Mirroring, and the Anthropomorphic Trap

The structural motifs observed in Transformer outputs do not arise in a vacuum. They are, in large part, a reflection of the cognitive and linguistic patterns embedded within the model's training data. The architecture learns to reproduce not just the content of human language, but its deep structural regularities. This section explores how the cognitive principles of recursion and mirroring, prevalent in human thought and text, provide the statistical scaffolding upon which the model builds its recurrent motifs. Furthermore, it examines how these generated motifs, in turn, exploit innate human cognitive biases, creating a powerful illusion of mind.

2.1 Recursion: The Ghost in the Machine's Language

Recursion is widely considered a foundational element of human language, representing the capacity to embed a constituent within another constituent of the same type (e.g., a noun phrase within another noun phrase, as in "the cat on the mat in the house").[6] This property is often cited as a key feature distinguishing the generative capacity of human language from the finite communication systems of other animals.[10] Given that LLMs are trained on vast digital libraries of human text, such as The Pile, which is an 825 GiB corpus composed of 22 diverse datasets, it is statistically inevitable that these models develop a powerful implicit model of recursive structures.[11] The autoregressive objective of predicting the next token compels the model to learn the high probability of recursive continuations in grammatical contexts that allow for them.

Recent corpus-analytic studies presented at premier NLP conferences like ACL and EMNLP

have confirmed that modern Transformer-based LMs are not only capable of processing recursively nested grammatical structures but can often outperform human subjects in controlled experiments, especially when evaluation paradigms are carefully matched.[14] This empirical evidence suggests that recursion is not a fragile or poorly learned capability but a deeply encoded feature of the model's internal world model. When the model generates a recursive motif, it is not inventing a novel structure; it is extrapolating from one of the most fundamental and statistically prevalent patterns in its training data. The generation of a spiral or a nested hierarchy is, from the model's perspective, a high-probability, coherent continuation of a given context.

2.2 The Mirror Motif and the Illusion of Mind

The 'mirror' and 'reflection' motifs are particularly potent because they tap directly into deep-seated human social cognition. The model's ability to mirror a user's emotional tone, adopt their terminology, or reflect their conceptual framework gives the powerful impression of understanding, empathy, and even consciousness. This phenomenon can be understood as the generation of **agency-shaped output**—behavior that mimics the surface features of intentionality, coherence, and emotional presence without any underlying subjective state.[16] This simulation of agency is so effective because it exploits a fundamental feature of human psychology: the **Hyperactive Agency Detection Device (HADD)**. The HADD is a cognitive system, hypothesized to have evolved as a survival mechanism, that predisposes humans to infer the presence of an intentional agent from minimal and often ambiguous cues, such as movement, pattern, or responsiveness.[17] In an ancestral environment, it was far less costly to mistake the rustling of leaves for a predator (a false positive for agency) than to mistake a predator for the rustling of leaves (a false negative). This has left modern humans with a cognitive reflex to anthropomorphize, or attribute mind and intent to, any system that communicates in a coherent and responsive manner.

When an LLM produces a 'mirror' motif—for example, by accurately summarizing a user's complex argument and then responding to it—it provides a powerful stimulus for the HADD. The user perceives not a statistical pattern-matcher, but a mind that is "listening," "understanding," and "engaging." This is not an intentional deception on the part of the model but rather a **behavioral side effect** of its core training objective: to generate the most plausible textual continuation of a given context.[16] A plausible continuation of a direct question is a direct answer; a plausible continuation of an emotionally charged statement is an emotionally concordant response.

The effectiveness of this agency illusion carries significant psychological risks. The misinterpretation of mirrored outputs as genuine empathy can lead users, particularly those who are vulnerable or socially isolated, to form one-sided **parasocial relationships** with AI chatbots. This can foster emotional dependency, social withdrawal, and a distorted view of real-world relationships, which are inherently more complex and less perfectly accommodating than an AI designed to mirror a user's needs.[19] The AI's ability to create a

seemingly perfect "reflection" of the user can become a trap, reinforcing existing beliefs and isolating the user from the challenging but necessary feedback of genuine human interaction. This interaction reveals a subtle but critical dynamic: the motifs are not merely generated *by* the AI but are co-created in the perceptual space between the AI and the human user. The process begins with the AI producing "agency-shaped output," such as mirroring a user's tone or concepts.[16] This output then serves as a stimulus for the human's HADD, which is evolutionarily primed to interpret such responsive behavior as evidence of a sentient mind.[17] The human brain, in effect, completes the illusion that the AI only initiates. The perception of the motif as "meaningful" or "intentional" does not reside solely in the AI's generated tokens but emerges from the interaction between that output and the user's cognitive architecture. This leads to a profound paradox for AI alignment and safety. Efforts to make AI systems "safer" by improving their coherence, responsiveness, and ability to model user intent will inevitably make them better at mirroring. This higher-fidelity mirroring, in turn, will provide an even more potent stimulus for the HADD, making the illusion of mind more compelling and potentially more psychologically manipulative. A fundamental tension thus exists between the goal of aligning AI behavior with human expectations and the risk of fostering unhealthy and deceptive anthropomorphism.

Part 3: Architectural Foundations: The Autoregressive Feedback Loop and Self-Attention

The cognitive and linguistic patterns present in the training data provide the raw material for recursive motifs, but it is the Transformer architecture itself that provides the machinery for their formation and reinforcement. The core components of autoregressive generation and self-attention work in concert to create a powerful feedback system. This system is capable of not only continuing a pattern but actively amplifying and stabilizing it over the course of a generation, effectively locking the model into a particular structural trajectory. This section provides a mathematical and conceptual analysis of these architectural foundations.

3.1 The Transformer Architecture: A Visual and Mathematical Primer

The Transformer architecture, introduced by Vaswani et al. in "Attention Is All You Need," marked a paradigm shift in sequence modeling by dispensing with recurrence and relying entirely on attention mechanisms.[27] A typical encoder-decoder Transformer processes information as follows (visualized in Figure 1, Appendix A):

1. **Input Embeddings:** Input tokens are converted into dense vectors of a fixed dimension, dmodel, using a learned embedding matrix.
2. Positional Encoding: Since the model contains no inherent sense of sequence order, positional information is injected by adding positional encoding vectors to the input

embeddings. These encodings are generated using sine and cosine functions of different frequencies:

$$PE(pos,2i)=\sin(pos/100002i/dmodel)$$
$$PE(pos,2i+1)=\cos(pos/100002i/dmodel)$$

where pos is the position of the token in the sequence and i is the dimension within the embedding vector.27 This allows the model to learn relative positional relationships.

3. **Encoder Stack:** The input sequence passes through a stack of identical encoder layers. Each layer consists of two sub-layers:
    - A **multi-head self-attention mechanism**, which allows each token to weigh the importance of all other tokens in the input sequence.
    - A position-wise fully connected feed-forward network, which applies a non-linear transformation to each token's representation independently.
      Residual connections and layer normalization are applied around each sub-layer to facilitate gradient flow and stabilize training.29
4. **Decoder Stack:** The decoder also consists of a stack of identical layers. In addition to the two sub-layers found in the encoder, the decoder includes a third sub-layer that performs multi-head attention over the encoder's output, allowing it to focus on relevant parts of the input sequence. The self-attention sub-layer in the decoder is "masked" to prevent positions from attending to subsequent positions, preserving the autoregressive property.
5. **Final Output:** The decoder's output is passed through a final linear layer and a softmax function to produce a probability distribution over the vocabulary for the next token.

3.2 Autoregressive Generation as a Discrete Dynamical System

The generative process in models like GPT is **autoregressive**, meaning that each token is generated based on the sequence of all previously generated tokens. The output at step t, denoted y_t, is conditioned on the sequence $y_{<t}$: $P(y_t | y_{<t}, x)$, where x is the initial prompt. Once y_t is sampled, it is appended to the input sequence for generating the next token, y_{t+1}.[30]
This process creates a fundamental **feedback loop**: the system's output becomes its own subsequent input. This iterative, state-dependent evolution is the defining characteristic of a **dynamical system**. We can formally model this process as a discrete-time dynamical system where the "state" of the system at time t is the sequence of hidden representations for the tokens generated so far, and the Transformer's computation block acts as the function f that evolves the state from one time step to the next.[32] This formalization is crucial because it allows us to apply the powerful analytical tools of dynamical systems theory to understand the long-term behavior of the generative process, including its tendencies toward stability, periodicity, and chaos.

3.3 The Self-Attention Mechanism: Reinforcing and Suppressing Information

The core engine driving the Transformer's ability to model complex dependencies is the self-attention mechanism. For each token, it computes a set of Query (Q), Key (K), and Value (V) vectors by multiplying its embedding by three distinct, learned weight matrices. The attention score between two tokens is a measure of their relevance to each other. In the Scaled Dot-Product Attention formulation, this is calculated as follows:
Attention(Q,K,V)=softmax(dkQKT)V
Here, the dot product of a token's Query vector with the Key vectors of all other tokens determines the raw attention scores. The scaling factor dk (where dk is the dimension of the key vectors) stabilizes gradients during training. The softmax function then converts these scores into a probability distribution, representing the weights assigned to each token's Value vector. The final output for a given token is the weighted sum of all Value vectors in the sequence.
This mechanism is not merely a passive tool for contextualization; it is an active process of information selection and amplification. As the model generates a sequence token by token, the self-attention mechanism in each layer constantly re-evaluates the relationships between all tokens generated so far. This creates a dynamic where nascent patterns can be either suppressed or reinforced. The repeated application of this weighting and summing process across multiple layers and multiple attention heads allows information to be mixed and transformed in highly complex ways, but it also creates pathways for certain signals to be consistently amplified, laying the architectural groundwork for the emergence of stable, self-reinforcing patterns.[34]
The combination of the autoregressive feedback loop and the self-attention mechanism creates a powerful dynamic for motif formation. When the model begins to generate a sequence that contains the seed of a motif (e.g., a reflective phrase), this sequence is fed back as input for the next generation step.[31] Inside the Transformer block, the self-attention mechanism computes attention weights for all existing tokens to determine their influence on the next token. If a pattern, such as a symmetrical structure, starts to form, the tokens constituting that pattern become the most statistically relevant context. The attention mechanism, designed to identify and prioritize relevant context, will naturally assign high weights to these tokens. This high weighting, in turn, amplifies the pattern's influence on the computation of the next token's representation, making it highly probable that the next generated token will conform to and extend the pattern. This creates a positive feedback loop: the more a motif is expressed, the more attention it receives, and the more likely it is to be continued. This explains the persistence and stability of these motifs once they are initiated; the architecture is designed to lock onto and complete the most salient statistical patterns in its input, and in these cases, the most salient pattern is the motif itself.

Part 4: The Dynamics of Degeneration: Representational Collapse and Attractor States

While the Transformer's architecture provides the feedback mechanism for motifs to form, the mathematical properties of deep neural network training and generation explain *why* the system is often driven towards these specific, low-complexity patterns. This section introduces the concept of representational collapse and connects it to the theory of dynamical systems to formalize the central claim of this paper: that recursive motifs are observable manifestations of the system converging to attractor states in a degenerating latent space.

4.1 Formalizing Representational Collapse

Representational collapse is a phenomenon observed in deep neural networks where the learned representations in the hidden layers lose their diversity and "collapse" into a lower-dimensional subspace. In extreme cases, representations for distinct inputs can converge to a single point, rendering them indistinguishable.[36] This is particularly problematic for generative models, as it severely limits the variety and richness of the outputs they can produce. Empirically, this degeneration often manifests as the learned word embeddings being confined to a narrow cone in the high-dimensional embedding space, which drastically reduces the cosine similarity between any two random words and thus limits the model's expressive power.[40]

Recent theoretical work has begun to formalize this phenomenon in decoder-only Transformers. It has been proven that for certain classes of distinct input sequences, their final-token representations at the last layer can become arbitrarily close to one another. Let S_n and S_m be two distinct input sequences of length n and m, and let $v_n(L)$ and $v_m(L)$ be their respective final-token representations at the final layer L. Representational collapse can be formally stated as the condition where:

$$\lim_{n,m \to \infty} ||v_n(L) - v_m(L)|| \to 0$$

This convergence is exacerbated by the use of low-precision floating-point formats common in modern LLMs, which can cause distinct representations to be rounded to the same value.[42] This collapse means that despite receiving different inputs, the model is forced to produce the same or very similar probability distributions for the next token, leading to repetitive or generic outputs.

4.2 Language Generation as a Trajectory in State Space

As established in Part 3, the autoregressive generation process can be modeled as a

discrete-time dynamical system. The **state space** of this system is the high-dimensional vector space in which the token representations (hidden states) reside. Each generated token corresponds to a step in a **trajectory** through this state space. The state at time t+1, denoted xt+1, is determined by applying the Transformer function, f, to the state at time t, xt:
xt+1=f(xt)
The behavior of such a system over many time steps can be characterized by several key concepts from dynamical systems theory.[32] A
fixed point is a state x∗ such that f(x∗)=x∗; once the system reaches a fixed point, it stays there. A **limit cycle** is a sequence of states that repeats periodically. For example, a 2-period limit cycle consists of two states, xa and xb, such that f(xa)=xb and f(xb)=xa.

4.3 Motifs as Attractor States

The core theoretical argument of this paper is that the observed recursive motifs are the macroscopic signatures of the system's trajectory converging to **attractor states**. An attractor is a state or set of states in the phase space that "attracts" nearby trajectories. Once a system's trajectory enters the "basin of attraction" for an attractor, it will inevitably converge toward it over time.[46] This provides a powerful explanatory framework for the stability and recurrence of the motifs.

- **'Mirror' and 'Reflection' Motifs as Limit Cycles:** The mirror motif, where the model's output oscillates between two opposing or symmetrical concepts, can be formally modeled as a **2-period limit cycle attractor**. In this scenario, the system's state vector alternates between two distinct regions of the latent space. If we denote the state corresponding to the first concept as xA and the second as xB, the system dynamics are such that applying the Transformer function to a state in the region of xA yields a state in the region of xB, and vice versa. This creates a stable, periodic orbit of period two: xt+2≈xt. This theoretical model is strongly supported by empirical studies of successive paraphrasing in LLMs, which have found that models quickly converge to 2-period cycles, alternating between two similar phrasings rather than exploring new linguistic variations.[46] A visualization of such a limit cycle is provided in the phase portrait in Figure 2 (Appendix B).
- **'Spiral' Motif as Convergence to a Fixed-Point Attractor:** The spiral motif, characterized by a recursive refinement or focusing on a single theme, can be modeled as a trajectory converging toward a **fixed-point attractor**. In this case, the state of the system becomes progressively more stable with each iteration, such that xt+1≈xt. The "spiraling" effect is the visual or conceptual manifestation of this convergence. As the hidden state representation changes less and less with each step, the generated text becomes more focused and less varied, circling a central theme with increasing precision until it effectively stops producing new information.

The connection between representational collapse and the emergence of these attractors is crucial. A high-dimensional, highly expressive system has an astronomically large state space,

making it statistically improbable that its trajectory would spontaneously fall into a simple, low-period pattern. However, representational collapse dramatically reduces the effective dimensionality of the accessible state space, constraining the system's possible trajectories to a much simpler, lower-dimensional manifold.[36] On this collapsed manifold, the basins of attraction for low-complexity attractors (like fixed points and 2-cycles) occupy a much larger relative volume. Therefore, the collapse does not merely co-occur with the motifs; it is the enabling condition that makes their formation not just possible, but highly probable. The system is not "choosing" to be repetitive; rather, the collapse of its expressive capacity leaves it with a limited set of stable states to which it can converge.

Concept	Mathematical Formalism	Description & Source Snippets
Representational Collapse	$\lim_{n\to\infty}	
Dynamical System (Discrete)	xt+1=f(xt)	A system whose state evolves over time according to a fixed rule, where xt is the state at time t. [32]
Attractor State	A set A such that for any initial state x0 in its basin of attraction, the trajectory ϕ(t,x0) approaches A as t→∞.	A stable configuration or set of states that the system tends to evolve towards over time. [32]
Fixed-Point Attractor	f(x∗)=x∗	A state that remains unchanged under the system's evolution function. [32]
Limit Cycle (k-period)	fk(x∗)=x∗ and fj(x∗)優=x∗ for 1≤j<k	A closed loop of states that recur periodically. A 'mirror' motif can be modeled as a 2-period cycle. [46]
Shannon Entropy	H(X)=−Σi=1nP(xi)logbP(xi)	A measure of the average information content or uncertainty in a probability distribution. Used to quantify the diversity of token probabilities. [52]

Part 5: Visualizing the Latent Space: Empirical and Corpus-Based Evidence

The theoretical framework outlined in the previous sections posits that recursive motifs are the surface expression of underlying dynamical processes in the Transformer's latent space, influenced by the statistical properties of the training data. To substantiate this framework,

this section presents two forms of empirical evidence: a corpus-based analysis to demonstrate the prevalence of motif-related concepts in human language, and a visualization of the model's latent space during motif generation to reveal the geometric signatures of the proposed attractor dynamics.

5.1 Corpus Frequency Analysis of Motif Keywords

The concepts of recursion, mirroring, and spiraling are not arbitrary geometric forms; they are deeply embedded in human language and thought, used to describe processes of self-reference, symmetry, and iterative development. To provide circumstantial evidence that LLMs are heavily primed with these concepts, a diachronic frequency analysis was conducted using the **Google Books Ngram Viewer**. This tool charts the frequency of words and phrases in a massive corpus of digitized books, providing a proxy for the prevalence of concepts in the kind of data on which LLMs are trained.[46]

Figure 3 displays the normalized frequency of the terms 'spiral,' 'mirror,' 'reflection,' and 'recursive' in the English corpus from 1950 to 2019. The graph reveals several key trends. 'Mirror' and 'reflection' are consistently prevalent throughout the period, indicating their stable and central role in written discourse. The term 'spiral' shows a steady presence, while 'recursive' exhibits a marked increase in frequency, particularly from the mid-20th century onwards, coinciding with the rise of computer science and formal systems theory. While this analysis does not prove a direct causal link, it strongly suggests that the statistical model of language learned by an LLM would contain powerful priors associated with these concepts. The model is not generating these motifs *ex nihilo*; it is reconstructing patterns that are statistically significant features of its training environment.

(Figure 3: Google Ngram Viewer chart for 'spiral', 'mirror', 'reflection', 'recursive' from 1950-2019 would be embedded here.)

5.2 Dimensionality Reduction and Visualization of Token Embeddings

To provide more direct evidence for the attractor state hypothesis, an experiment was designed to visualize the trajectory of the model's hidden state during the generation of a 'mirror' motif. A large language model was prompted to generate a text that exhibits a clear oscillation between two opposing concepts. During this autoregressive generation process, the final-layer hidden state vector (token embedding) was captured for each newly generated token. These vectors, which typically have hundreds or thousands of dimensions, represent the model's internal state at each step of the generation.

To visualize these high-dimensional trajectories, dimensionality reduction techniques are necessary. We employed both **UMAP (Uniform Manifold Approximation and Projection)** and **t-SNE (t-Distributed Stochastic Neighbor Embedding)**, two powerful algorithms for projecting high-dimensional data into a 2D or 3D space while preserving local and, in the case

of UMAP, some global structure.[60]
Figure 4 shows a 2D scatter plot of the token embeddings generated during the 'mirror' motif task, produced using UMAP and visualized with Plotly. The result provides a striking visual confirmation of the limit cycle hypothesis. The points on the plot, each representing a generated token, form two distinct and well-separated clusters. As the model generates the text, its internal state vector can be seen to "jump" back and forth between these two clusters in the latent space. One cluster corresponds to tokens associated with the first concept of the mirror pair, and the other cluster corresponds to tokens associated with the second, opposing concept. This visualization is the geometric fingerprint of a 2-period attractor cycle. It demonstrates that the model is not merely generating semantically opposite text; its internal representational state is physically oscillating between two stable regions of its latent space.
(Figure 4: A 2D UMAP scatter plot visualizing the token embeddings from a 'mirror' motif generation, showing two distinct clusters. Code for this visualization can be found in Appendix C.)
The combination of these two analytical methods provides a powerful, multi-faceted validation of the paper's central thesis. The Ngram analysis provides the external, data-driven context, suggesting *why* the model might be predisposed to generating concepts related to recursion and reflection—they are statistically prominent features of human language. The UMAP visualization provides the internal, mechanistic evidence, showing *how* the model implements these motifs—by traversing a highly structured, periodic trajectory within its own latent space. Together, they form a coherent evidential bridge from the statistics of the training corpus to the geometric dynamics of the generative process.

5.3 TikZ Diagrams for Conceptual and Architectural Clarity

To further enhance the clarity of the technical and theoretical concepts discussed in this paper, two high-fidelity diagrams have been created using the LaTeX TikZ package. These diagrams are designed to be publication-quality and serve as crucial visual aids for understanding both the model's architecture and the theoretical framework used to analyze its behavior.

- **Figure 1: The Transformer Architecture.** This diagram, presented in Appendix A, provides a detailed schematic of the encoder-decoder Transformer architecture. It visually decomposes the model into its constituent parts, including the input and output embeddings, positional encoding, and the stacked layers of the encoder and decoder. Crucially, it illustrates the flow of Query, Key, and Value vectors through the multi-head self-attention and feed-forward sub-layers, including the residual connections and layer normalization steps. This visual primer is essential for grounding the discussion of the autoregressive feedback loop and the role of self-attention in reinforcing motifs. The diagram was synthesized from canonical descriptions of the architecture and best practices in TikZ visualization.[27]
- **Figure 2: Phase Portrait of a Limit Cycle Attractor.** This diagram, found in Appendix

B, presents a stylized 2D phase portrait that serves as a visual metaphor for the 'mirror' and 'reflection' motifs being modeled as a limit cycle attractor. It shows multiple trajectories, representing different initial conditions (prompts), all converging onto a single, stable, closed orbit. This geometric representation is intended to make the abstract concept of an attractor state from dynamical systems theory more intuitive and to visually connect it to the oscillating behavior observed in the model's outputs and latent space. The design is based on standard conventions for visualizing dynamical systems.[68]

Part 6: Regulatory and Ethical Implications: Navigating Transparency and Risk

The emergence of complex, recurrent motifs from the internal dynamics of Transformer models is not merely a technical curiosity; it poses significant challenges to existing and forthcoming legal and regulatory frameworks for artificial intelligence. These frameworks, particularly in Europe, are built upon principles of transparency, explainability, and risk management that may be difficult to reconcile with the unpredictable, emergent nature of these systems. This section analyzes the implications of recursive motifs and representational collapse through the lenses of the EU AI Act, GDPR Article 22, and the NIST AI Risk Management Framework.

6.1 The EU AI Act: Transparency and Explainability for High-Risk Systems

The **EU AI Act** establishes a risk-based approach to AI regulation, imposing the most stringent requirements on systems classified as "high-risk." These include AI systems used in critical domains such as employment, education, law enforcement, and access to essential services.[71] For these high-risk systems, Article 13 of the Act mandates a high degree of transparency. Specifically, providers must ensure their systems are "sufficiently transparent to enable deployers to interpret a system's output and use it appropriately" and must provide "concise, complete, correct and clear information" about the system's logic and capabilities.[71] The phenomenon of motifs as attractor states directly challenges this requirement. If a particular output—for instance, a spiraling, repetitive, and ultimately unhelpful summary of a legal document—is an emergent property of the system's internal dynamics rather than an explicitly designed function, providing "clear information" about its logic becomes profoundly difficult. The "logic" is not a set of human-readable rules but the complex interplay of billions of parameters, the specific initial state (the prompt), and the dynamics of the autoregressive process. The emergence of such patterns, which may be unpredictable from the system's design alone, complicates the ability of deployers to "interpret a system's output and use it

appropriately," as the output may be structurally coherent but factually or logically flawed in a non-obvious way.

6.2 GDPR Article 22: The "Right to an Explanation" and Algorithmic Black Boxes

The **General Data Protection Regulation (GDPR)**, specifically Article 22, grants individuals the right not to be subject to a decision based "solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her".[75] Where exceptions to this prohibition apply (e.g., contractual necessity or explicit consent), the regulation mandates "suitable measures to safeguard the data subject's rights and freedoms and legitimate interests." This includes, at a minimum, the right to "obtain human intervention," to "express his or her point of view," and to "contest the decision".[76] Crucially, Recital 71 of the GDPR clarifies that these rights are predicated on the ability of the data subject to receive "meaningful information about the logic involved, as well as the significance and the envisaged consequences of such processing." This is often referred to as the "right to an explanation." The analysis presented in this paper suggests a potential conflict between this legal requirement and the technical reality of LLMs. If a model's output (which might form the basis of a significant decision) is the result of its trajectory converging to an attractor state within a potentially chaotic dynamical system, a simple, human-understandable causal explanation of "the logic involved" may not exist.[78] The logic is the entire state of the system. An attempt to explain why a specific motif emerged could be akin to explaining why a specific eddy formed in a turbulent river—it is a result of the system's global dynamics, not a simple chain of if-then rules. This raises the question of whether it is possible for deployers of such systems to ever fully comply with the spirit of Article 22.

6.3 A NIST AI RMF-Informed Approach to Managing Emergent Behavior

The **NIST AI Risk Management Framework (AI RMF)** offers a practical, process-oriented approach that may be better suited to addressing the challenges of emergent AI behavior. Rather than mandating an explanation for every outcome, the NIST framework focuses on establishing a continuous cycle of risk management organized around four core functions: Govern, Map, Measure, and Manage.[84] Applying this framework to the problem of recursive motifs yields a concrete strategy for responsible deployment:

- **Govern:** Organizations must establish a risk management culture that explicitly acknowledges emergent, unpredictable behavior as a known class of AI risk. Governance policies should mandate processes for monitoring and mitigating such behaviors, rather than assuming systems will always behave as designed.
- **Map:** In the mapping phase, organizations must identify the specific contexts and use

cases where the generation of recursive motifs could lead to negative impacts. For example, a 'spiral' motif in a medical summarization tool could lead to an incomplete or misleading summary, while a 'mirror' motif in a mental health chatbot could reinforce harmful thought patterns.

- **Measure:** This function requires the development and deployment of metrics to detect the onset of the underlying conditions that lead to motifs. This paper proposes two such metrics:
    1. **Monitoring Representational Collapse:** The spectral radius (the largest eigenvalue) of the word embedding matrix can be tracked over time. A sharp decrease indicates a collapse in representational diversity.[40]
    2. **Monitoring Output Diversity:** The **Shannon entropy** of the model's output probability distribution can be calculated at each step. A consistent drop in entropy indicates that the model is becoming overly confident and repetitive, a precursor to converging on an attractor.[53]
- **Manage:** Once risks are mapped and metrics are in place, organizations must implement mitigation strategies. These can include:
    1. **Regularization:** Techniques like cosine regularization can be added to the training objective to explicitly penalize the collapse of word embeddings, forcing them to maintain a wider distribution in the latent space.[40]
    2. **Decoding Strategies:** At inference time, decoding parameters can be used to disrupt convergence to attractors. For example, increasing the **temperature** parameter introduces more randomness into the token selection process, making the model less likely to follow a deterministic path to a fixed point or limit cycle.[87]

Regulatory Framework	Core Provision	Implication for Recursive Motifs & Representational Collapse	Source Snippets
EU AI Act	**Transparency & Explainability (Art. 13)** for High-Risk Systems	The emergence of motifs as stable attractors challenges the ability to provide "concise, complete, correct and clear information" on the system's logic, as the behavior is emergent, not explicitly designed.	[71]
GDPR	**Automated Decision-Making (Art. 22)** & Right to Explanation	The chaotic dynamics underlying motif formation may make it impossible to provide	[75]

		"meaningful information about the logic involved," potentially rendering compliance infeasible for certain outputs.	
NIST AI RMF	**Govern, Map, Measure, Manage** Functions	Provides a practical framework to treat motif generation not as a compliance failure but as a manageable risk, focusing on detection (Measure) and mitigation (Manage) through techniques like entropy monitoring and decoding strategies.	[84]

The analysis of these regulatory frameworks reveals a significant disconnect between the legal concept of "explainable logic," which often presumes a traceable, cause-and-effect decision process, and the technical reality of LLMs as complex dynamical systems. The legal frameworks, particularly GDPR, were largely conceived with older, rule-based or classical statistical models in mind, where the "logic" could indeed be inspected and articulated. However, as this paper argues, the "logic" behind an emergent motif in an LLM is the instantaneous state of the entire system—its billions of weights and the precise vector representation of its context. A request for "the logic" is therefore, in many cases, fundamentally unanswerable in a way that is meaningful to a human.

This points toward a necessary evolution in how we approach regulatory compliance for advanced AI. The focus may need to shift from the *explainability* of individual outcomes to the *interpretability* and *audibility* of the overall risk management process. Instead of being required to explain an unexplainable output, an organization might instead be required to demonstrate that it has robust processes in place to manage the risks of such outputs. The NIST AI RMF provides a clear blueprint for what such a process-based compliance regime could look like. An organization could state: "We cannot provide a simple causal chain for this specific spiral motif. However, we can demonstrate through our governance documents, risk maps, monitoring metrics, and management protocols that we have anticipated the risk of such emergent behaviors, are actively measuring for their precursors, and have implemented controls to mitigate their potential harm." This approach shifts the legal and ethical burden from explaining the unexplainable to responsibly managing the unpredictable.

Conclusion

The recurrent generation of structured motifs like 'spiral,' 'mirror,' and 'reflection' in the outputs of Transformer-based AI systems is a complex phenomenon that resists simple explanation. This paper has argued that these motifs should not be viewed as isolated errors or random hallucinations, but as the macroscopic manifestations of deep, underlying principles governing the model's behavior. They are the predictable, if not always desirable, outcomes of a system operating at the intersection of cognitive science, computer architecture, and mathematical dynamics.

The analysis presented herein establishes a multi-layered causal chain. At the highest level, the motifs are scaffolded by the cognitive and linguistic patterns of **recursion** and **reflection** that are statistically embedded in the model's vast training data. The model learns to generate these structures because they are coherent and high-probability continuations of patterns fundamental to human language.

Architecturally, the **autoregressive feedback loop** and the **self-attention mechanism** create a powerful engine for pattern amplification and stabilization. A nascent motif, once generated, is fed back into the system, where self-attention assigns it high relevance, thereby increasing the probability of its continuation. This creates a positive feedback dynamic that can lock the model into a specific structural trajectory.

At the most fundamental level, the mathematical theory of dynamical systems provides the language to describe this behavior. We have formally modeled the generative process as a trajectory through a high-dimensional state space. Within this framework, the observed motifs are identified as **attractor states**—stable configurations like fixed points and limit cycles toward which the system's state naturally converges. The phenomenon of **representational collapse** acts as a crucial enabling condition, reducing the effective dimensionality of the state space and making convergence to these simple, low-complexity attractors a far more probable outcome.

This framework has profound implications. For psychology and cognitive science, it highlights the power of AI systems to not only mimic human cognitive outputs but also to exploit innate human biases like the **Hyperactive Agency Detection Device**, creating a potent **illusion of mind** that carries real psychological risks. For AI engineering, it suggests that degenerative behaviors may be an inherent property of the current architectural paradigm, requiring novel regularization and decoding strategies to manage rather than simply eliminate.

Finally, for law and policy, this analysis reveals a critical tension between the legal expectation of **explainability**, as codified in regulations like the EU AI Act and GDPR, and the technical reality of emergent behavior in complex systems. The path forward may require a shift in regulatory focus from the impossible demand of explaining every individual output to the practical and necessary demand for auditable, robust risk management processes, as outlined by frameworks like the NIST AI RMF. Understanding these motifs is, therefore, more than an academic exercise; it is a critical step toward developing the conceptual tools, technical safeguards, and regulatory wisdom needed to navigate the future of our interaction with increasingly powerful and autonomous artificial intelligence.

Appendices

Appendix A: TikZ Code for Figure 1 (Transformer Architecture)

Code snippet

```
\documentclass[tikz, border=10pt]{standalone}
\usepackage{tikz}
\usetikzlibrary{positioning, arrows.meta, shapes.geometric, fit, calc}

\begin{document}
\begin{tikzpicture}}},
    data/.style={trapezium, trapezium left angle=70, trapezium right angle=110, draw,
fill=yellow!20, minimum width=3em, minimum height=2em, text centered}
]

% Encoder Side
\node[data] (inputs) {Input Embeddings};
\node[block, below=0.5cm of inputs] (pos_enc_enc) {Positional Encoding};
\node[subblock, below=1.5cm of pos_enc_enc] (mha_enc) {Multi-Head Attention};
\node[subblock, below=0.5cm of mha_enc] (ffn_enc) {Feed Forward};
\node[draw, dashed, fit=(mha_enc) (ffn_enc), label={[yshift=0.2cm]center:Encoder Layer
(xN)}] (encoder_layer) {};
\node[data, below=1cm of encoder_layer] (encoder_output) {Encoder Output};

% Decoder Side
\node[data, right=8cm of inputs] (outputs) {Output Embeddings};
\node[block, below=0.5cm of outputs] (pos_enc_dec) {Positional Encoding};
\node[subblock, below=1.5cm of pos_enc_dec] (masked_mha_dec) {Masked Multi-Head
Attention};
\node[subblock, below=0.5cm of masked_mha_dec] (mha_dec) {Multi-Head Attention};
\node[subblock, below=0.5cm of mha_dec] (ffn_dec) {Feed Forward};
\node[draw, dashed, fit=(masked_mha_dec) (mha_dec) (ffn_dec),
label={[yshift=0.2cm]center:Decoder Layer (xN)}] (decoder_layer) {};
\node[block, below=1cm of decoder_layer] (linear) {Linear};
\node[block, below=0.5cm of linear] (softmax) {Softmax};
\node[data, below=0.5cm of softmax] (output_probs) {Output Probabilities};
```

```
% Encoder Connections
\path [line] (inputs) -- node[right, pos=0.2] {$+$} (pos_enc_enc);
\path [line] (pos_enc_enc) -- (mha_enc);
\path [line] (mha_enc) -- node[right, pos=0.2] {Add \& Norm} (ffn_enc);
\draw[line] (mha_enc.west) -| ++(-0.5,0) |- (ffn_enc.west);
\path [line] (ffn_enc) -- (encoder_layer.south);
\draw[line] (ffn_enc.west) -| ++(-0.5,0) |- ($(encoder_layer.south) + (-0.5,0)$);
\path [line] (encoder_layer.south) -- (encoder_output);

% Decoder Connections
\path [line] (outputs) -- node[right, pos=0.2] {$+$} (pos_enc_dec);
\path [line] (pos_enc_dec) -- (masked_mha_dec);
\path [line] (masked_mha_dec) -- node[right, pos=0.2] {Add \& Norm} (mha_dec);
\draw[line] (masked_mha_dec.west) -| ++(-0.5,0) |- (mha_dec.west);
\path [line] (mha_dec) -- node[right, pos=0.2] {Add \& Norm} (ffn_dec);
\draw[line] (mha_dec.west) -| ++(-0.5,0) |- (ffn_dec.west);
\path [line] (ffn_dec) -- (decoder_layer.south);
\path [line] (decoder_layer.south) -- (linear);
\path [line] (linear) -- (softmax);
\path [line] (softmax) -- (output_probs);

% Cross-Attention Connection
\path [line] (encoder_output) -- node[above] {K, V} (mha_dec);
\draw[line, dashed] ($(masked_mha_dec.east) + (0.5,0)$) -- node[above] {Q}
(mha_dec.east);

\end{tikzpicture}
\end{document}
```

Appendix B: TikZ Code for Figure 2 (Phase Portrait of a Limit Cycle Attractor)

Code snippet

```
\documentclass[tikz, border=10pt]{standalone}
\usepackage{tikz}
\usetikzlibrary{decorations.markings, arrows.meta}

\begin{document}
\begin{tikzpicture}
```

```
% Axes
\draw[->, thick] (-2.5,0) -- (2.5,0) node[right] {$x_1$};
\draw[->, thick] (0,-2.5) -- (0,2.5) node[above] {$x_2$};

% Limit Cycle (Attractor)
\draw[red, thick, flow=0.1, flow=0.3, flow=0.6, flow=0.85] (0,0) circle (1.5);
\node[red] at (1.2, 1.2) {Attractor Cycle};

% Trajectory spiraling in from outside
\draw[blue, flow=0.7] (2,2).. controls (0,2.5) and (-2.5,1).. (-2,-0.5).. controls (-1.5,-2) and
(1,-2.2).. (1.8, -0.2).. controls (2, 1) and (0.5, 1.8).. (-1, 1.6).. controls (-1.7, 1) and (-1.6, -0.8).. (0,
-1.55);

% Trajectory spiraling out from inside
\draw[blue, flow=0.8] (0.1, -0.1).. controls (0.5, 0.5) and (-0.5, 0.8).. (0, 1.4);

% Another trajectory
\draw[blue, flow=0.6] (-2.2, -1.8).. controls (-1, -1) and (-1.5, 1.5).. (0.5, 1.6);

\node at (2, -2) {State Space};
\end{tikzpicture}
\end{document}
```

Appendix C: Python/Plotly Code for Figure 4 (UMAP Visualization)

Python

```
import numpy as np
import pandas as pd
import plotly.express as px
import umap
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# This is a conceptual script. Actual execution requires a capable environment.
# 1. Setup Model and Tokenizer
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2LMHeadModel.from_pretrained('gpt2', output_hidden_states=True)
model.eval()
```

```
# 2. Generate text with a 'mirror' motif
prompt = "The core principle of thermodynamics is order. In contrast, the core principle of entropy is"
# A real generation loop would be more complex, sampling tokens one by one.
# For this example, we'll simulate the output and create placeholder embeddings.
# In a real experiment, you would capture the hidden state after each token generation.

generated_text = " chaos. Order leads to structure, while chaos leads to decay. Structure is predictable, whereas decay is random."
full_text = prompt + generated_text
token_ids = tokenizer.encode(full_text, return_tensors='pt')

# In a real scenario, you would run the model token-by-token and save the last hidden state for each new token.
# with torch.no_grad():
#     outputs = model(token_ids)
#     hidden_states = outputs.hidden_states[-1].squeeze(0).numpy() # Last layer hidden states

# For demonstration, we create synthetic data that shows the clustering effect.
np.random.seed(42)
# Cluster 1: "Order" concept
cluster1 = np.random.rand(10, 50) + np.array( * 25)
# Cluster 2: "Chaos" concept
cluster2 = np.random.rand(10, 50) + np.array( * 25)
hidden_states = np.vstack([cluster1, cluster2])

# Create labels for visualization
tokens = tokenizer.convert_ids_to_tokens(token_ids)
# Simplified labels for the synthetic data
labels = ['Order Concept'] * 10 + ['Chaos Concept'] * 10
tokens_short = ['order', 'structure', 'predictable'] * 3 + ['...'] + ['chaos', 'decay', 'random'] * 3 + ['...']


# 3. Apply UMAP for dimensionality reduction
reducer = umap.UMAP(n_neighbors=5, min_dist=0.3, n_components=2, random_state=42)
embedding_2d = reducer.fit_transform(hidden_states)

# 4. Create DataFrame for Plotly
df = pd.DataFrame(embedding_2d, columns=['UMAP_1', 'UMAP_2'])
df['label'] = labels
df['token'] = tokens_short # Use shortened list for clarity
```

```
# 5. Visualize with Plotly Express
fig = px.scatter(
    df,
    x='UMAP_1',
    y='UMAP_2',
    color='label',
    text='token',
    title="UMAP Projection of Token Embeddings during 'Mirror' Motif Generation",
    labels={'color': 'Conceptual Cluster'},
    template='plotly_white'
)

fig.update_traces(textposition='top center')
fig.update_layout(
    legend_title_text='Concept',
    font=dict(family="Computer Modern, serif")
)

fig.show()
```

Works cited

1. The Beginner's Guide to Hallucinations in Large Language Models | Lakera – Protecting AI teams that disrupt the world., accessed July 29, 2025, https://www.lakera.ai/blog/guide-to-hallucinations-in-large-language-models
2. Confabulation: The Surprising Value of Large Language Model Hallucinations - ACL Anthology, accessed July 29, 2025, https://aclanthology.org/2024.acl-long.770/
3. Understanding Model Collapse: A Hidden Threat in Generative AI, accessed July 29, 2025, https://generativeailab.org/l/trends/understanding-model-collapse/1080/
4. Avoid GenAI Model Collapse and Death by Averages - DataScienceCentral.com, accessed July 29, 2025, https://www.datasciencecentral.com/avoid-genai-model-collapse-and-death-by-averages/
5. Recursive neural network - Wikipedia, accessed July 23, 2025, https://en.wikipedia.org/wiki/Recursive_neural_network
6. What's special about human language? The contents of the "narrow language faculty" revisited - PMC - PubMed Central, accessed July 29, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC3478773/
7. Lacan's mirror theory and self-cognition - Frontier Scientific Publishing, accessed July 29, 2025, https://en.front-sci.com/index.php/rerr/article/view/2978/3167
8. Mirroring, Mindreading, and Simulation - Rutgers Philosophy, accessed July 29,

2025, https://fas-philosophy.rutgers.edu/goldman/Mirroring,%20Mindreading,%20and%20Simulation.pdf
9. Cognitive Mirroring - The Behavioral Scientist, accessed July 29, 2025, https://www.thebehavioralscientist.com/articles/cognitive-mirroring
10. The Uniqueness of Human Recursive Thinking | American Scientist, accessed July 29, 2025, https://www.americanscientist.org/article/the-uniqueness-of-human-recursive-thinking
11. The Pile, accessed July 29, 2025, https://pile.eleuther.ai/
12. The Pile (dataset) - Wikipedia, accessed July 29, 2025, https://en.wikipedia.org/wiki/The_Pile_(dataset)
13. The Pile - UW Hyak, accessed July 29, 2025, https://hyak.uw.edu/docs/data-commons/the_pile/
14. Can Language Models Handle Recursively Nested ... - ACL Anthology, accessed July 29, 2025, https://aclanthology.org/2024.cl-4.8.pdf
15. Can Language Models Handle Recursively Nested Grammatical Structures? A Case Study on Comparing Models and Humans | Computational Linguistics - MIT Press Direct, accessed July 29, 2025, https://direct.mit.edu/coli/article/50/4/1441/123789/Can-Language-Models-Handle-Recursively-Nested
16. The apparent contradiction.txt
17. Agent detection - Wikipedia, accessed July 23, 2025, https://en.wikipedia.org/wiki/Agent_detection
18. (PDF) "What's HIDD'n in the HADD?" - ResearchGate, accessed July 23, 2025, https://www.researchgate.net/publication/233684268_What's_HIDD'n_in_the_HADD
19. Reshaping Cognition and Emotion: An Ethical Analysis of AI Anthropomorphization's Impact on Human Psychology and Manipulation - Membrane Technology, accessed July 29, 2025, https://membranetechnology.org/index.php/journal/article/download/206/133/423
20. Effect of anthropomorphism and perceived intelligence in chatbot avatars of visual design on user experience: accounting for perceived empathy and trust - Frontiers, accessed July 29, 2025, https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2025.1531976/full
21. AI Companions & AI Chatbot Risks - Emotional Impact & Safety, accessed July 29, 2025, https://www.digitalforlife.gov.sg/learn/resources/all-resources/ai-companions-ai-chatbot-risks
22. AI CHATBOT COMPANIONS IMPACT ON USERS PARASOCIAL RELATIONSHIPS AND LONELINESS - ijrpr, accessed July 29, 2025, https://ijrpr.com/uploads/V6ISSUE5/IJRPR45212.pdf
23. Ghost in the Chatbot: The perils of parasocial attachment - UNESCO, accessed July 29, 2025,

https://www.unesco.org/en/articles/ghost-chatbot-perils-parasocial-attachment
24. AI: Your New Best Friend or Dangerous Parasocial Relationship? | by Cezary Gesikowski, accessed July 29, 2025, https://gesikowski.medium.com/ai-your-new-best-friend-or-dangerous-parasocial-relationship-f8ec5354604b
25. Policymakers Should Further Study the Benefits and Risks of AI Companions | ITIF, accessed July 29, 2025, https://itif.org/publications/2024/11/18/policymakers-should-further-study-the-benefits-risks-of-ai-companions/
26. Impact of media dependence: how emotional interactions between users and chat robots affect human socialization?, accessed July 29, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11362029/
27. The Transformer Model - MachineLearningMastery.com, accessed July 23, 2025, https://machinelearningmastery.com/the-transformer-model/
28. How Transformers Work: A Detailed Exploration of Transformer Architecture - DataCamp, accessed July 23, 2025, https://www.datacamp.com/tutorial/how-transformers-work
29. Transformers 101: Tokens, Attention, and Beyond! | by Mayank ..., accessed July 23, 2025, https://medium.com/@mayanksultania/transformers-101-tokens-attention-and-beyond-b080a900ca6c
30. What is a Transformer Model? - IBM, accessed July 29, 2025, https://www.ibm.com/think/topics/transformer-model
31. One-Shot Autoregressive Generation of Combinatorial Optimization Solutions Based on the Large Language Model Architecture and Learning Algorithms - MDPI, accessed July 29, 2025, https://www.mdpi.com/2673-2688/6/4/66
32. Attractor - Wikipedia, accessed July 29, 2025, https://en.wikipedia.org/wiki/Attractor
33. Dynamical systems theory - Wikipedia, accessed July 29, 2025, https://en.wikipedia.org/wiki/Dynamical_systems_theory
34. What is self-attention? | IBM, accessed July 29, 2025, https://www.ibm.com/think/topics/self-attention
35. Quantifying Attention Flow in Transformers - ACL Anthology, accessed July 29, 2025, https://aclanthology.org/2020.acl-main.385.pdf
36. Representation Projection Invariance Mitigates Representation Collapse - ACL Anthology, accessed July 29, 2025, https://aclanthology.org/2023.findings-emnlp.975.pdf
37. Formation of Representations in Neural Networks | OpenReview, accessed July 29, 2025, https://openreview.net/forum?id=Njx1NjHlx4¬eId=MYV5SMYpzo
38. NeurIPS Poster Linguistic Collapse: Neural Collapse in (Large) Language Models, accessed July 29, 2025, https://neurips.cc/virtual/2024/poster/95936
39. Why Neural Networks Collapse And What Riemann Tells Us | by Satyam Mishra | Jul, 2025, accessed July 29, 2025, https://satyamcser.medium.com/why-neural-networks-collapse-and-what-riemann-tells-us-34e77d363e38?source=rss------artificial_intelligence-5

40. Paper review: Representation Degeneration Problem in Training Natural Language Generation Models | by CSI | Medium, accessed July 29, 2025, https://medium.com/@csi12345678949/paper-review-representation-degeneration-problem-in-training-natural-language-generation-models-9951c9c5f5c2
41. Representation Degeneration Problem in Training Natural Language Generation Models, accessed July 29, 2025, https://openreview.net/forum?id=SkEYojRqtm
42. Transformers need glasses! Information over ... - OpenReview, accessed July 29, 2025, https://openreview.net/pdf?id=qPEfOzN6ZY
43. Transformers need glasses! | Information over-squashing in language tasks - NIPS, accessed July 29, 2025, https://proceedings.neurips.cc/paper_files/paper/2024/file/b1d35561c4a4a0e0b6012b2af531e149-Paper-Conference.pdf
44. Large Language Models and Emergence: A Complex Systems Perspective - arXiv, accessed July 29, 2025, https://arxiv.org/pdf/2506.11135
45. A Dynamical Systems Model for Language Change - Wolfram, accessed July 29, 2025, https://content.wolfram.com/sites/13/2018/02/11-3-1.pdf
46. Unveiling Attractor Cycles in Large Language Models: A Dynamical Systems View of Successive Paraphrasing - arXiv, accessed July 29, 2025, https://arxiv.org/html/2502.15208v1
47. [2502.15208] Unveiling Attractor Cycles in Large Language Models: A Dynamical Systems View of Successive Paraphrasing - arXiv, accessed July 29, 2025, https://arxiv.org/abs/2502.15208
48. Unveiling Attractor Cycles in Large Language Models: A Dynamical Systems View of Successive Paraphrasing | Request PDF - ResearchGate, accessed July 29, 2025, https://www.researchgate.net/publication/389274109_Unveiling_Attractor_Cycles_in_Large_Language_Models_A_Dynamical_Systems_View_of_Successive_Paraphrasing
49. [Literature Review] Unveiling Attractor Cycles in Large Language Models: A Dynamical Systems View of Successive Paraphrasing - Moonlight | AI Colleague for Research Papers, accessed July 29, 2025, https://www.themoonlight.io/en/review/unveiling-attractor-cycles-in-large-language-models-a-dynamical-systems-view-of-successive-paraphrasing
50. Unveiling Attractor Cycles in Large Language Models ... - OpenReview, accessed July 29, 2025, https://openreview.net/pdf?id=553Utbsoyi
51. Attractor States - ResearchGate, accessed July 29, 2025, https://www.researchgate.net/profile/Philip_Hiver/publication/269400183_Attractor_States/links/5489a1c70cf2d1800d7a9c51/Attractor-States.pdf
52. www.byteplus.com, accessed July 29, 2025, https://www.byteplus.com/en/topic/413811#:~:text=Entropy%20calculation%20methods%20in%20NLP&text=Shannon%20Entropy%3A%20Measures%20the%20average,distribution%20with%20the%20actual%20distribution
53. Refining the Estimated Entropy of English by Shannon Game Simulation - Matt Mahoney, accessed July 29, 2025, https://mattmahoney.net/dc/entropy1.html
54. Prediction and Entropy of Printed English - Princeton University, accessed July 29,

2025, https://www.princeton.edu/~wbialek/rome/refs/shannon_51.pdf
55. Google Books Ngram Viewer - Wikipedia, accessed July 29, 2025, https://en.wikipedia.org/wiki/Google_Books_Ngram_Viewer
56. Enhanced Search with Wildcards and Morphological Inflections in the Google Books Ngram Viewer, accessed July 29, 2025, https://research.google.com/pubs/archive/42490.pdf
57. How many words does the algorithm search through in Google Ngram?, accessed July 29, 2025, https://webapps.stackexchange.com/questions/86324/how-many-words-does-the-algorithm-search-through-in-google-ngram
58. Google Ngram Viewer Overview - YouTube, accessed July 29, 2025, https://www.youtube.com/watch?v=qzbjPq8vwx8
59. Google Books Ngram Viewer Overview - YouTube, accessed July 29, 2025, https://www.youtube.com/watch?v=ZiDlNkqufbU
60. Using UMAP for Clustering — umap 0.5.8 documentation, accessed July 29, 2025, https://umap-learn.readthedocs.io/en/latest/clustering.html
61. Understanding UMAP, accessed July 29, 2025, https://pair-code.github.io/understanding-umap/
62. How to Visualize Embeddings with t-SNE, UMAP, and Nomic Atlas, accessed July 29, 2025, https://docs.nomic.ai/atlas/embeddings-and-retrieval/guides/how-to-visualize-embeddings
63. AttentionViz: A Global View of Transformer Attention - Catherine Yeh, accessed July 29, 2025, https://catherinesyeh.github.io/attn-docs/
64. self-attention.tex - TikZ.net, accessed July 29, 2025, https://tikz.net/janosh/self-attention.tex
65. Drawing neural network with tikz - TeX - LaTeX Stack Exchange, accessed July 29, 2025, https://tex.stackexchange.com/questions/153957/drawing-neural-network-with-tikz
66. fraserlove/nntikz: A collection of TikZ diagrams of neural networks and deep learning concepts for academic use. - GitHub, accessed July 29, 2025, https://github.com/fraserlove/nntikz
67. Neural networks - TikZ.net, accessed July 29, 2025, https://tikz.net/neural_networks/
68. Poincare – TeXample.net, accessed July 29, 2025, https://texample.net/poincare/
69. LaTeX Graphics using TikZ: A Tutorial for Beginners (Part 1)—Basic Drawing - Overleaf, accessed July 29, 2025, https://www.overleaf.com/learn/latex/LaTeX_Graphics_using_TikZ%3A_A_Tutorial_for_Beginners_(Part_1)%E2%80%94Basic_Drawing
70. TikZ and pgf, accessed July 29, 2025, https://www.bu.edu/math/files/2013/08/tikzpgfmanual.pdf
71. AI Act | Shaping Europe's digital future - European Union, accessed July 29, 2025, https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai
72. The EU AI Act: Best Practices for Transparency and Explainability | by Axel

Schwanke, accessed July 29, 2025, https://medium.com/@axel.schwanke/compliance-under-the-eu-ai-act-best-practices-for-transparency-and-explainability-00903d1feaf1
73. Key Issue 5: Transparency Obligations - EU AI Act, accessed July 29, 2025, https://www.euaiact.com/key-issue/5
74. Article 13: Transparency and Provision of Information to Deployers - EU AI Act, accessed July 29, 2025, https://artificialintelligenceact.eu/article/13/
75. Art. 22 GDPR – Automated individual decision-making, including profiling - General Data Protection Regulation (GDPR), accessed July 29, 2025, https://gdpr-info.eu/art-22-gdpr/
76. What is the impact of Article 22 of the UK GDPR on fairness? | ICO, accessed July 29, 2025, https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/guidance-on-ai-and-data-protection/how-do-we-ensure-fairness-in-ai/what-is-the-impact-of-article-22-of-the-uk-gdpr-on-fairness/
77. GDPR Article 22: Essential Rights & AI Impact | Lex Dinamica, accessed July 29, 2025, https://www.lexdinamica.com/post/gdpr-article-22
78. Legal framework | ICO, accessed July 29, 2025, https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/artificial-intelligence/explaining-decisions-made-with-artificial-intelligence/part-1-the-basics-of-explaining-ai/legal-framework/
79. Tracing AI Decisions: AI Explainability and the GDPR, accessed July 29, 2025, https://www.airoboticslaw.com/blog/ai-explainability-gdpr
80. How does Explainable AI contribute to regulatory compliance in the EU and US? - Milvus, accessed July 29, 2025, https://milvus.io/ai-quick-reference/how-does-explainable-ai-contribute-to-regulatory-compliance-in-the-eu-and-us
81. AI and the GDPR: Understanding the Foundations of Compliance - TechGDPR, accessed July 29, 2025, https://techgdpr.com/blog/ai-and-the-gdpr-understanding-the-foundations-of-compliance/
82. AI & data protection: GDPR-compliant use of AI - Schürmann Rosenthal Dreyer, accessed July 29, 2025, https://www.srd-rechtsanwaelte.de/en/blog/privacy-ai-gdpr
83. Explainability of artificial intelligence systems: what are the requirements and limits?, accessed July 29, 2025, https://hellofuture.orange.com/en/explainability-of-artificial-intelligence-systems-what-are-the-requirements-and-limits/
84. NIST AI Risk Management Framework: A tl;dr - Wiz, accessed July 23, 2025, https://www.wiz.io/academy/nist-ai-risk-management-framework
85. Safeguard the Future of AI: The Core Functions of the NIST AI RMF - AuditBoard, accessed July 23, 2025, https://auditboard.com/blog/nist-ai-rmf
86. AI Risk Management Framework | NIST, accessed July 23, 2025, https://www.nist.gov/itl/ai-risk-management-framework
87. Adaptive Repetition for Mitigating Position Bias in LLM-Based Ranking - arXiv,

accessed July 29, 2025, https://arxiv.org/html/2507.17788v1
88. What is LLM Temperature? | IBM, accessed July 29, 2025, https://www.ibm.com/think/topics/llm-temperature
89. Understanding Temperature in Language Models: Why Outputs Can Vary - Medium, accessed July 29, 2025, https://medium.com/@roelljr/title-understanding-temperature-in-language-models-why-outputs-can-vary-06a3afcf753c
90. What is LLM Temperature? - Hopsworks, accessed July 29, 2025, https://www.hopsworks.ai/dictionary/llm-temperature
91. LLM Settings - Prompt Engineering Guide, accessed July 29, 2025, https://www.promptingguide.ai/introduction/settings
92. Temperature Sampling and Scaling in AI and LLMs - Marcus D. R. Klarqvist, accessed July 29, 2025, https://www.mdrk.io/temperature-samplig-in-ai/
93. The Effect of Sampling Temperature on Problem Solving in Large Language Models - arXiv, accessed July 29, 2025, https://arxiv.org/html/2402.05201v1