ChatGPT

Prompting as a Control Surface in Transformer LLMs

Prompt interpretation inside the transformer stack

A deployed large language model (LLM) is (mechanistically) a conditional distribution over token sequences: given an input context, it produces a probability distribution for the next token, then repeats that process autoregressively. This framing is not a metaphor; it is the training and inference interface of mainstream transformer LLMs ("predict the next token" under an autoregressive likelihood objective). [1]

A prompt therefore cannot be read as "instructions in English" in the human sense. It is first compiled into **tokens**, then mapped to vectors, then processed by a stack of attention/MLP blocks. What looks like a single sentence can become dozens of discrete token-events, each of which can receive attention and contribute to intermediate representations. [2]

Tokenization is the first control bottleneck

Most modern LLM pipelines use subword/byte-level tokenization families derived from Byte Pair Encoding (BPE) and related algorithms. The key mechanical point is that "the prompt" is not the character string you wrote; it is the **sequence of token IDs** after encoding. BPE-style tokenization was popularized in neural NLP for open-vocabulary handling by segmenting rare words into subword units, and practical production tokenizers emphasize reversibility/losslessness and efficiency. [3]

Implications for prompting (verified where the tokenizer behaves this way; the downstream behavioral claims are inferred from the autoregressive interface):

- **Control granularity is token-level.** If two prompt variants differ by only a few characters but tokenize differently, they become different conditioning sequences; downstream divergences are therefore not surprising. [4]
- **"Formatting" is not cosmetic.** Newlines, punctuation, Markdown fences, and delimiter strings alter token patterns and attention structure, and prompt sensitivity to small variations is explicitly measured in recent work. [5]
- **Length is not measured in characters.** Context limits (and truncation behavior in applications) operate on tokens, not characters; production tooling exposes token counts for this reason. [6]

A minimal ASCII pipeline view:

```
raw text prompt
   ↓ (tokenizer: BPE/SentencePiece/variant)
token IDs: t1, t2, ... , tN
   ↓ (embedding + positional encoding)
vectors:  x1, x2, ... , xN
   ↓ (transformer layers: attention + MLP)
contextual vectors: h1, h2, ... , hN
```

```
   ↓ (linear head)
logits over vocab for next token
   ↓ (decoding: greedy / sampling / constraints)
next token → append → repeat
```

The transformer component of this pipeline is not optional: the original transformer formulation eliminates recurrence and convolution, using attention mechanisms plus feed-forward sublayers and positional signals to process sequences. [7]

Context windows are not “memory”; they are the domain of conditioning

A context window is best treated as the maximum length of the conditioning sequence visible at inference. Anything outside it is not an internal “background memory” unless externally re-injected (retrieval, summarization, or other system scaffolding). This is a structural property of transformer inference: the model computes next-token probabilities from the provided token prefix. [8]

Even when a model supports long contexts, empirical work shows that *using* long contexts is brittle. In long-context retrieval and multi-document QA tasks, performance often drops when relevant information is placed in the middle of the context (“lost in the middle”), with higher performance when the relevant span is near the beginning or end. [9]

This matters for prompting because many “prompt failures” are actually **placement failures**: the constraint exists but is positioned such that it is weakly utilized during generation. The existence of positional encoding schemes designed to model distance/position interactions (e.g., rotary position embeddings) further underlines that sequence position is an active variable in attention computations, not a human-level formatting detail. [10]

Attention weighting is the mechanical locus of “what the model uses”

Self-attention computes weighted combinations of token representations, and multi-head attention provides multiple parallel weighting patterns; this is the core transformer operation. [7]

However, translating “attention” into “importance” is not generally valid (inferred; widely discussed but not uniquely settled). What is stable and directly relevant to prompting is:

- There is no hard architectural separation between “instructions” and “data.” Both are tokens in the same sequence, competing to influence representations through attention and subsequent nonlinear transformations. This becomes security-relevant in prompt injection and agent settings, where untrusted retrieved text can function as instruction-like material. [11]
- Long-context fragility suggests that attention and/or downstream components do not robustly preserve mid-context constraints across diverse tasks and setups, even when models nominally accept long sequences. [9]

What a prompt is as a computational object

From the model’s perspective, a prompt is neither a “request” nor an “instruction.” It is a **conditioning program** that modifies the probability field over continuations. This is a direct consequence of the autoregressive objective: the model is trained to assign high probability to token sequences resembling its training distribution given a prefix, and at inference it generates by sampling/decoding from that conditional distribution. [1]

A prompt can be decomposed into three computational roles (conceptual decomposition; grounded in the conditional-generation interface, but the decomposition itself is inferred):

The prompt as a probability shaper

At time step k, the model represents $P(t_{k+1} \mid t_{\leq k})$. Changing the prompt changes $t_{\leq k}$, which changes the entire conditional distribution. Because transformer LLM capabilities scale and generalize in ways consistent with improved cross-entropy modeling under scaling laws, "prompting" is best viewed as steering within a learned distribution, not invoking a symbolic interpreter. [12]

The prompt as an output-space reducer

Constraints like "answer in JSON," "use only these labels," or "cite sources" function by making large parts of the continuation space low-probability relative to a smaller manifold of acceptable outputs. This is not guaranteed compliance; it is a shift in likelihood mass over continuation classes. The dependence of output quality and diversity on decoding strategy (greedy vs sampling, nucleus sampling, etc.) shows that *even with fixed model weights* the realized continuation depends strongly on how probability mass is truncated/selected. [13]

The prompt as a role-positioning mechanism

Role text ("you are X") is computationally just more tokens, but instruction-tuned / RLHF-trained models have learned correlations between such prefixes and downstream response styles. Instruction tuning and RLHF are explicitly used to make models follow written instructions more reliably; without that training, "role" text would be primarily a style-conditioning prefix rather than a control authority. [14]

This is the place where many user intuitions break: role text *can* shift response distribution, but its effect size depends on how the model was trained to treat such cues. The empirical result that persona system prompts often do not improve objective task performance (and can slightly harm it) is consistent with role prompting being a weak control lever for correctness, even if it affects surface style. [15]

The real power of the prompter and its actual limits

"Control" in prompting can be partitioned into what is **mechanically available** via conditioning versus what is merely **socially implied**.

What control humans actually have

You control the conditioning evidence. Prompting is a way of injecting tokens that reweight the model's posterior over continuations. Few-shot and instruction-based task specification via text is explicitly demonstrated in large autoregressive LMs, which can perform tasks "without gradient updates" by conditioning on demonstrations/instructions in-context. [16]

You control the training-aligned hooks if they exist. Instruction tuning (e.g., Jason Wei [17] et al.'s FLAN) and RLHF (e.g., Long Ouyang [18] et al.) aim to make models respond to instruction-like text as a privileged signal. These methods change how much weight the model places on "do X" style directives relative to generic continuation. [14]

You control decoding parameters in many deployments. Output randomness/degeneration is not only a "model" phenomenon; it is an interaction between the model distribution and the decoding

algorithm. Nucleus sampling was proposed precisely because maximizing likelihood via greedy decoding can produce bland/repetitive text, and truncating the unreliable tail of the distribution can improve quality and diversity. [13]

In tool-using systems, you can control the environment. Prompting can be augmented by external actions (retrieval, tool execution, constrained APIs). The ReAct paradigm explicitly demonstrates gains from interleaving reasoning traces and external actions, reducing hallucination and enabling interaction with knowledge sources. [19]

What control humans think they have but don't

You do not directly control internal computation. The prompter supplies tokens; the transformer computes forward pass activations. Claims like "think harder" are at best indirect attempts to bias the model toward generating longer intermediate traces (which themselves become additional tokens). The fact that chain-of-thought prompting improves reasoning accuracy indicates that *token-level scaffolding* can change outcomes, but it does not imply the user can directly set the model's "effort knob" in a general way. [20]

You do not get a guaranteed instruction/data boundary. Prompt injection research demonstrates that adversarial text can hijack behavior in systems that treat untrusted text as part of the same instruction-bearing sequence. OWASP classifies prompt injection as a top risk category in LLM applications, and indirect prompt injection attacks formalize the "blurred line between data and instructions" in LLM-integrated applications. [21]

You do not reliably get correctness from "tone," "confidence," or "emotion." The most defensible version of this claim is empirical: persona conditioning in system prompts does not typically improve objective accuracy and can slightly degrade it, even though persona attributes can measurably influence predictions. This supports a mechanistic view where persona/tone tokens mostly move outputs along stylistic submanifolds rather than enforce stronger correctness constraints. [15]

Where authority comes from in deployed chat systems

In many commercial systems, authority is implemented as an **instruction hierarchy** (system/developer/user), and models are trained and/or scaffolded to follow that hierarchy. OpenAI's public Model Spec formalizes levels of authority (platform > developer > user > guideline) and states that higher-authority instructions override lower ones. [22]

This "authority" is not something a base transformer inherently possesses; it is an alignment/training and product-layer construct. The existence of research explicitly training models to prioritize privileged instructions ("instruction hierarchy") is evidence that instruction priority is learned and reinforced, not automatically guaranteed by architecture. [23]

Why prompts fail as a distributional pathology

Prompt failure is often described as "the model ignored me." Mechanistically, many failures are better described as: **the prompt did not sufficiently concentrate probability mass on the desired continuation class given competing constraints and defaults**.

Under-specification produces high-entropy continuations

If the prompt does not sharply define:

- the task (what is being produced),
- the evaluation criteria (what counts as correct),
- the output manifold (format/schema),
- the admissible evidence (sources, allowed assumptions),

then the model's continuation is pulled toward generic, high-frequency completions shaped by pretraining and alignment. Instruction-following tuning exists precisely because "bigger" models are not automatically better at matching user intent without additional alignment, implying that defaults matter. [24]

A minimal example of under-specification (snippet is illustrative; analysis is inferred from the conditional-generation framing):

```
Describe the transformer.
```

This does not define depth, audience, evidence standards, or whether to focus on architecture vs history vs applications, so the continuation distribution remains broad. Prompt sensitivity measurements show that even small rephrasings can lead to different outputs under such broad task definitions. [5]

Over-specification can create constraint interference

Over-specification is not "too much detail" in the human sense; it is **too many constraints whose conjunction is hard to satisfy** within the model's learned continuation space.

Typical mechanical signatures:

- **Conflicting constraints** ("be concise" + "provide exhaustive citations" + "no bullets" + "include examples") create a satisfiability problem where any completion violates something. Instruction hierarchy frameworks exist because models are otherwise vulnerable to conflict resolution failures under competing instructions. [25]
- **Context placement failure**: critical constraints buried in the middle of a long prompt may be weakly used, consistent with long-context degradation findings ("lost in the middle"). [9]
- **Constraint dilution**: adding many low-authority stylistic cues (persona, tone) can compete with or distract from task constraints, and persona prompts do not reliably improve objective accuracy. [15]

A concrete over-specification fragment (illustrative):

```
You are a skeptical philosopher, but also a friendly tutor.
Write a rigorous proof, but keep it informal.
Use no jargon, but include mechanistic details.
```

The failure mode is not “confusion” but a forced compromise: the model samples a point in output space that partially satisfies each constraint, producing an answer that is neither maximally formal nor maximally informal (inferred; consistent with constraint-conjunction framing). [26]

Prompt injection is the extreme failure mode of instruction/data entanglement

In LLM-integrated applications (agents, RAG), untrusted data is often concatenated into the prompt context. Indirect prompt injection work shows that retrieved content can function as “arbitrary instruction” because the model processes it as part of the same token sequence, enabling “remote” control via data channels. [27]

This is not a bug in natural language; it is a consequence of the model’s lack of a native instruction/data type system at the token level. OWASP’s categorization of prompt injection as a top LLM application risk is consistent with this structural vulnerability. [28]

Why prompts succeed as entropy control and compute steering

Successful prompting is often described as “clarity.” Mechanistically, it is closer to: **reducing the effective entropy of the response distribution while allocating tokens to intermediate structure that the model can condition on**.

Ambiguity collapse is constraint sharpening

A prompt succeeds when it collapses latent degrees of freedom that would otherwise be resolved by defaults.

A ceiling-level fragment (illustrative) tends to specify:

- **output specification** (schema/format),
- **evidence specification** (what to cite, what to treat as uncertain),
- **failure behavior** (what to do when evidence is missing),
- **evaluation hook** (how the answer will be judged).

This mirrors what deployed prompt engineering guidance emphasizes: using message roles / higher-authority instructions and monitoring prompt performance across model versions with evals because behavior varies across model snapshots. [29]

An explicit “veto path” (illustrative):

```
If a claim is not supported by the provided sources, label it as INFERRED or
SPECULATIVE.
Do not present it as verified.
```

Mechanistically, this does not guarantee truth; it creates a local policy inside the continuation space that penalizes unsupported assertions by making them incompatible with the specified output manifold (inferred). [30]

Eliminating low-effort completions by forcing intermediate structure

Chain-of-thought prompting demonstrates that providing (or eliciting) intermediate reasoning steps can substantially improve performance on multi-step tasks. This is empirical: the chain-of-thought prompting paper reports significant gains on arithmetic/commonsense/symbolic reasoning when intermediate reasoning is included. [31]

Zero-shot CoT shows that even a short prefix like "Let's think step by step" can improve reasoning in some settings, indicating that small prompt fragments can act as triggers for a different continuation regime. [32]

Two crucial ceiling-level caveats follow from the literature:

- The benefit is not monotonic or universally benign; critiques note bias and fallibility modes in step-by-step prompting regimes. [33]
- "Better reasoning traces" are still **generated text**, not guaranteed internal proofs; self-consistency improves reliability by sampling multiple reasoning paths and selecting the most consistent answer, explicitly treating reasoning as stochastic output requiring aggregation. [34]

Mechanistic interpretation: prompts can induce in-context "learning-like" computation

A deeper explanation for why examples and structured traces work is emerging from in-context learning research: transformers can implement learning algorithms *in the forward pass* on some tasks, and mechanistic work links in-context learning to attention circuitry such as induction heads. [35]

This supports a mechanistic (still incomplete) view: a sequence of demonstrations is not just "more information," it can induce an internal computation resembling parameter-update behavior within activations rather than weights. The claim is domain-limited in the cited work (regression-style tasks) but establishes plausibility that in-context structure can change computation class, not only style. [36]

Implicit content in prompts and the model's inferred latent variables

A ceiling-level prompt always contains more than its literal text because the model must infer latent variables that the prompt does not (and often cannot) fully specify.

The prompt transmits an evaluation function, even if you never state one

Examples:

- A prompt that cites papers and uses terms like "mechanistic" implicitly signals that the completion will be judged by technical correctness and grounding, not solely fluency. (Inferred; grounded in the fact that instruction-tuned models respond to distributional cues learned during training.) [37]
- A prompt that includes a schema implies that violating the schema is a high-cost error, concentrating probability mass on schema-valid continuations (inferred; consistent with output-space reduction framing). [13]

Risk tolerance and refusal style are not "tone"; they are aligned defaults

RLHF and related alignment methods explicitly optimize for behaviors described as helpfulness and reductions in toxicity, producing models preferred by humans even at smaller parameter counts. This implies that many "defaults" users experience are learned policies, not emergent linguistic politeness. [38]

In deployed systems with explicit behavioral specs, the instruction hierarchy and safety boundaries are formalized as higher-authority constraints (platform/developer/user). That means the model is expected to resolve conflicts by authority level, not by rhetorical force. [39]

"Models respond to what the prompter is" is a claim about distributional classification

A non-anthropomorphic restatement:

- The model estimates, from cues in the prompt, what class of interaction this resembles in its training distribution (technical report, casual chat, adversarial jailbreak, etc.), then generates continuations typical of that class. (Inferred; consistent with instruction tuning/RLHF shaping response distributions and the observed sensitivity of outputs to framing.) [40]

This explains why: - asserting authority via "I am your boss" is often weak unless reinforced by higher-authority roles or system scaffolding, and
- persona prompts can change surface features without reliably improving correctness on objective tasks. [41]

Single prompt vs sequence and positioning the model in an environment

A single "perfect prompt" is structurally inferior in many real tasks because prompting is better modeled as a **control loop** than a one-shot command.

Why sequential interaction dominates one-shot prompting

Each additional turn appends tokens, updating the conditioning context. In Bayesian language: you are iteratively adding evidence, shrinking the posterior over acceptable continuations (inferred analogy; grounded in the conditional-generation interface and in-context learning framing). [42]

Empirically and operationally:

- Prompt sensitivity and model snapshot variation mean a single prompt cannot robustly cover all failure cases; monitoring prompts with evals and pinning model snapshots is recommended in production guidance precisely because behavior shifts across versions. [43]
- Long-context brittleness ("lost in the middle") suggests that piling everything into one giant prompt can degrade utilization of crucial constraints, strengthening the case for dialogue-driven re-centering of constraints and summaries. [9]

A control-loop schematic (conceptual):

```
specify constraints → sample output → inspect error → add targeted constraint
        ↑                                                    ↓
        └──────────────────── converge distribution over acceptable outputs
________┘
```

Role assignment vs capability activation

Saying "You are X" adds tokens; it does not automatically install a new capability. The most direct empirical evidence is that system personas do not improve objective accuracy in a large evaluation, although persona attributes can perturb predictions. [15]

Capability activation instead comes from:

- **training hooks** (instruction tuning, RLHF) that make certain instruction patterns effective, [44]
- **in-context computation** enabled by demonstrations and structure, potentially implementable by transformer circuits (e.g., induction heads), [45]
- **environment constraints** (tools, retrieval, validators) that change the effective task by adding external state and post-processing filters. [46]

Instructing behavior vs shaping computation

A useful ceiling-level distinction (inferred but grounded):

- **Instructing behavior**: adding tokens that bias the output style or policy, relying on alignment and learned instruction-following patterns. [47]
- **Shaping computation**: structuring the context (demonstrations, intermediate traces, tool-interaction protocols, self-consistency sampling) so that the model performs different internal operations during the forward pass and decoding. Evidence: chain-of-thought prompting gains, self-consistency as decoding aggregation, and mechanistic in-context learning results. [48]

This distinction is also where "depth > speed" becomes computational rather than philosophical: generating intermediate structure consumes tokens and changes subsequent conditioning; sampling multiple paths and aggregating (self-consistency) spends more compute to approximate marginalization over reasoning trajectories. [49]

WHAT ACTUALLY MATTERS IN PROMPTING (AT THE CEILING)

At the ceiling, prompting is not "asking nicely." It is **distribution control under architectural constraints**. [50]

A prompt matters to the extent that it:

- **Compiles into token patterns that strongly condition the continuation**, not to the extent that it sounds authoritative to a human. [51]
- **Concentrates probability mass onto a narrow output manifold** (schemas, admissible evidence rules, explicit failure behaviors), shrinking response entropy. [13]
- **Allocates tokens to intermediate structure that the model can condition on** (demonstrations, step-by-step traces, tool protocols), thereby changing the effective computation and not merely the surface tone. [52]

- **Respects placement and context-window realities**, avoiding burying critical constraints where long-context utilization is empirically weak. [9]
- **Treats instruction hierarchy as a trained/scaffolded policy rather than a natural law**, and designs for conflict resolution rather than pretending conflicts won't occur. [53]
- **Assumes no intrinsic instruction/data boundary**, especially in agentic/RAG settings, and therefore designs explicit trust boundaries and mitigations rather than relying on delimiters or rhetoric. [54]
- **Uses sequential interaction as a control loop**, because a single prompt cannot robustly encode all constraints without dilution, sensitivity, and version drift. [55]

---

[1] [16] [26] [42] [50] [2005.14165] Language Models are Few-Shot Learners
https://arxiv.org/abs/2005.14165

[2] [4] [51] tiktoken is a fast BPE tokeniser for use with OpenAI's models.
https://github.com/openai/tiktoken?utm_source=chatgpt.com

[3] Neural Machine Translation of Rare Words with Subword Units
https://arxiv.org/abs/1508.07909?utm_source=chatgpt.com

[5] A Prompt Sensitivity Index For Large Language Models
https://aclanthology.org/2024.findings-emnlp.852.pdf?utm_source=chatgpt.com

[6] Tokenizer - OpenAI API
https://platform.openai.com/tokenizer?utm_source=chatgpt.com

[7] [8] Attention Is All You Need
https://arxiv.org/abs/1706.03762?utm_source=chatgpt.com

[9] Lost in the Middle: How Language Models Use Long Contexts
https://arxiv.org/abs/2307.03172?utm_source=chatgpt.com

[10] RoFormer: Enhanced Transformer with Rotary Position Embedding
https://arxiv.org/abs/2104.09864?utm_source=chatgpt.com

[11] [27] [54] [2302.12173] Not what you've signed up for
https://arxiv.org/abs/2302.12173?utm_source=chatgpt.com

[12] Scaling Laws for Neural Language Models
https://arxiv.org/abs/2001.08361?utm_source=chatgpt.com

[13] The Curious Case of Neural Text Degeneration
https://arxiv.org/abs/1904.09751?utm_source=chatgpt.com

[14] [37] [44] Finetuned Language Models Are Zero-Shot Learners
https://arxiv.org/abs/2109.01652?utm_source=chatgpt.com

[15] [41] When "A Helpful Assistant" Is Not Really Helpful: Personas in System Prompts Do Not Improve Performances of Large Language Models
https://arxiv.org/html/2311.10054v2

[17] [33] On Second Thought, Let's Not Think Step by Step! Bias and ...
https://hci.stanford.edu/publications/2023/shaikh-stepbystep.pdf?utm_source=chatgpt.com

[18] [24] [30] [38] [40] [47] Training language models to follow instructions with human feedback
https://arxiv.org/abs/2203.02155?utm_source=chatgpt.com

[19] [46] ReAct: Synergizing Reasoning and Acting in Language Models
https://arxiv.org/abs/2210.03629?utm_source=chatgpt.com

20 31 48 52 Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
https://arxiv.org/abs/2201.11903?utm_source=chatgpt.com

21 28 OWASP Top 10 for Large Language Model Applications
https://owasp.org/www-project-top-10-for-large-language-model-applications/?utm_source=chatgpt.com

22 39 53 Model Spec (2025/02/12)
https://model-spec.openai.com/2025-02-12.html

23 The Instruction Hierarchy:Training LLMs to Prioritize Privileged Instructions
https://arxiv.org/html/2404.13208v1

25 Training LLMs to Prioritize Privileged Instructions
https://openreview.net/forum?id=vf5M8YaGPY&utm_source=chatgpt.com

29 43 55 Prompt engineering | OpenAI API
https://developers.openai.com/api/docs/guides/prompt-engineering/

32 Large Language Models are Zero-Shot Reasoners
https://arxiv.org/abs/2205.11916?utm_source=chatgpt.com

34 49 Self-Consistency Improves Chain of Thought Reasoning in Language Models
https://arxiv.org/abs/2203.11171?utm_source=chatgpt.com

35 36 Transformers learn in-context by gradient descent
https://arxiv.org/abs/2212.07677?utm_source=chatgpt.com

45 [2209.11895] In-context Learning and Induction Heads
https://arxiv.org/abs/2209.11895?utm_source=chatgpt.com