

# Regularizing the Prompt: Stable Model-Agnostic LLM Responses Through Inversion Theory

Saulo Silva

North Bay Software

Bergen, Norway

northbaysoftware@icloud.com

**Abstract**—Large language models from different vendors, tiers and versions exhibit divergent behavioral characteristics when given identical prompts and input, making model-agnostic solutions a non-trivial engineering problem. This article analyzes the statistical foundations of modern models, identifies stochastic base training and vendor-specific alignment fine-tuning as two sources of divergence, and approaches prompt engineering as an ill-posed inverse problem, using geophysical inversion as the mathematical foundation. The prompt is formalized as a regularization instrument analogous to Tikhonov regularization, where insufficient or excessive constraints respectively produce unstable or oversmoothed responses. A methodology for finding the minimum set of constraints that all target models consistently follow is presented and validated through Summerizer, an iOS application that interfaces with multiple models across vendors without prompt branching, providing a foundation for future platform-independent multi-model implementations.

**Index Terms**—LLM, prompt engineering, model-agnostic, geophysical inversion, Tikhonov regularization, CACE

MODERN AI systems, especially LLMs, approximate conditional distributions for possible continuations of text. Models are deterministic at inference time, with fixed weights, input and dropout layers inactive, while the token sampling process is probabilistic. At every step, the next token is computed from previous tokens as a large-scale, conditional probability distribution, to produce responses with natural reading flow.

As LLMs become tools of growing presence in daily tasks, vendors train and tune their models to suit usage ranging from general to specialized, and each exhibits distinct behavioral differences. After interacting with various models through prompting and observing their responses, users prefer one over others depending on the task at hand and tone needed. For those developing solutions intended to work with different vendors and models, a few questions need to be asked: *where do these differences come from? Is it possible to get uniform responses regardless of the model?*

This article addresses these questions based on extensive empirical testing across multiple models and vendors, in an application developed with model-agnosticism as the first design decision.

## I. THE INVERSE PROBLEM, FROM LINEAR TO NON-LINEAR

In order to understand the statistical and stochastic behavior of modern AI models, the mathematical foundations that support this study, inversion theory, will be drawn from one of the fields that benefits the most from it, geophysics, one of

whose objectives is to image the subsurface at depths generally impossible to physically observe. This is done by inferring unknown parameters from observed data within certain constraints as to avoid extrapolations that could lead to false results and potentially expensive consequences.

Inversion estimates a model's unknowns from its observable outputs, since these parameters are not directly accessible. This is true for both geophysics and AI, where the forward operator maps these unknown parameters to observed data, and the inversion attempts to recover them from the observations. Solving the inverse problem requires solving the forward problem first [1].

In the real world, inverse problems are frequently ill-posed, that is, they do not have unique solutions [2][3][4]. Without extra information such as constraints or observed data, all solutions are statistically plausible, and only by providing additional parameters will some become more plausible than others. While geophysics employs linear and non-linear inversion, depending on the dataset size, time and computational resources available, the LLM problem is a non-linear inversion because the mapping from model parameters to output tokens is not just a matrix multiplication, but a composition of several non-linear transformations: activation functions, attention mechanisms and normalization layers.

Unlike linear inversion, where the solution space admits an analytical form, non-linear inversion must be solved iteratively [4], achieved through training via gradient descent [5]. The ill-posed nature of this problem also means that many different parameter configurations can produce equally plausible outputs, the root cause of the behavioral divergence between models discussed later.

The forward linear operator as seen in classic geophysics literature, notably Menke [3], Tarantola [4] and Aki & Richards [6], is given by (Eq. 1), where  $d$  is the data,  $g$  is the linear forward operator, and  $m$  is the model parameters. Neural networks use the non-linear forward operator (Eq. 2), where  $d$  is the target data,  $f$  is the non-linear operator and  $m$  represents its learnable parameters.

$$d = g(m) \quad (1)$$

$$d \approx f(m) \quad (2)$$

To prevent confusion due to notation in classic inversion theory and computing, like the different meanings of  $m$ , or Adam's first moment  $m_t$  and matrix dimensions in (Eq. 3), where  $W$  represents the learned weight matrices of the model, this article will preserve the notations as seen in their literatures and, when an analogy to geophysics is needed, similar terms will be pointed out. For notational alignment with the analogy to geophysics, the input context is suppressed in (Eq. 2), the full dependency is  $d \approx f(m, x)$ , where  $x$  represents the prompt and preceding tokens, and  $d$  the target output, such as the next token in LLMs.

$$W \in R^{m \times n} \quad (3)$$

## II. STOCHASTIC CONCEPTS DEFINING AI BEHAVIOR

### A. Random variables

A random variable is a quantity whose value is uncertain, defined by a probability distribution over its possible outcomes [7]. When sampling is used, the next token outcome is a random variable drawn from the model's output distribution, so generations can differ even for identical inputs. *Greedy* and *beam search* are deterministic alternatives to sampling, selecting the most probable token or sequence and thus limiting the random variable to a fixed outcome.

### B. Probability distributions

Describe how likely different outcomes are. When interacting with modern models via API, parameters such as *top-p* and *top-k*, truncate the distribution, while *temperature* shapes it by scaling *logits* before *softmax*, increasing or decreasing differences between them. From a given set of *logits* (vector of arbitrary real numbers),  $z_1, z_2, \dots, z_n$ , *softmax* computes a probability distribution (Eq. 4), where all outputs are positive and sum to 1, with larger *logits* becoming more probable.

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (4)$$

### C. Stochastic processes

A sequence of random variables evolving over time [8]. The randomness emerges at each step from sampling the next token from the model's output distribution rather than selecting it deterministically, where each draw is a probabilistic event whose outcome depends on the probability distribution over the vocabulary at that step, conditioned on the entire preceding context. AI-generated text is  $X_1, X_2, X_3, \dots, X_t$ , a discrete-time stochastic process where each  $X_t$  represents the token generated at step  $t$ . Each token depends on the entire preceding context within the model's context window, distinguishing it from low-order Markov chains and connecting it to the attention mechanism discussed later.

### D. Entropy, the measurement of uncertainty

Entropy measures the uncertainty of a probability distribution and is defined as (Eq. 5) [8]. In the context of token generation, temperature affects *logits* before *softmax*, reshaping the distribution and, consequently, its entropy. High entropy

indicates that the model is uncertain about the next token, with no statistical dominance of any particular outcome. Low entropy indicates confidence, with probability concentrated on a small number of likely tokens.

$$H(P) = - \sum_x p(x) \log p(x) \quad (5)$$

## III. BEFORE MODERN AI AND DEEP LEARNING

Before the wider adoption of neural networks, probabilistic models and Markov-based approaches were dominant for language modeling. Understanding their structure and limitations is useful for contextualizing what modern architectures achieve and why the stochastic approach described in the previous section was not enough.

### A. Markov chain

Future depends on the immediate past, not deterministically, but probabilistically, so that a first-order representation [7] is given by (Eq. 6), where  $P$  is the transition matrix describing the probability of a state  $i$  transitioning to a state  $j$  ( $P_{ij}$ ). Markov chains don't condition on long histories, conditioning instead on the immediate context, that is,  $w_{t-1}$  is the only known context to compute  $w_t$ . Higher-order versions exist, but are impractical for language because the state space would be too big due to the necessity of keeping track of long dependencies.

$$P(w_t | w_{t-1}) \quad (6)$$

### B. Bayesian network

Variables are represented by nodes and the links between them denote conditional dependency, in the form  $A \rightarrow B$ ,  $E \rightarrow F$ , creating acyclic graphs. It can be used to perform inference by computing posterior distribution of variables given evidence of others, by way of transforming joint probabilities  $P(x, \theta)$ , the probability of both  $x$  and  $\theta$  being known, into conditional probabilities  $P(x | \theta)$ , the probability of  $x$  if  $\theta$  is known [1] and seen in the identity (Eq. 7).

$$P(x, \theta) = P(x | \theta) P(\theta) = P(\theta | x) P(x) \quad (7)$$

This combines prior knowledge with observed data to produce a posterior probability estimate [4]. This is the Bayesian inference widely used in several fields: medicine, genetics, psychology, linguistics, cosmology, image processing, training and inference of modern AI models [7].

## IV. NEURAL NETWORKS AND STOCHASTIC FUNCTION APPROXIMATION

Markov chains and Bayesian networks are limited in keeping track of large, past dependencies and compositional structure present in natural language. The state space of higher-order Markov models grows exponentially with context length, and Bayesian networks require explicit specification of conditional dependencies.

Neural networks overcome these limitations by learning distributed representations of data through hierarchical layers

of non-linear transformations [5] [9], approximating complex input-to-output mappings from the training data itself without manually specifying these features. This is consistent with geophysical inversion: the sensitivity kernel needs no manual adjustment since the adjoint method computes it implicitly [4].

#### A. Token representation

Discrete tokens must be mapped to continuous vector representations before a neural network can process language. This is achieved through embedding matrices (Eq. 8), which are effectively lookup tables mapping each token in the vocabulary  $V$  to a dense vector of dimension  $d_{\text{model}}$ . Compared to geophysical inversion, this is the step that defines what  $m$  actually is in the neural network context: the continuous parameter vector that the model operates on.

$$E_{\text{token}} \in R^{V \times d_{\text{model}}} \quad (8)$$

Since the attention mechanism is insensitive to the order of tokens in the input, positional information must be injected explicitly. This is done either through learned positional embeddings that adapt to the training data (Eq. 9), or fixed sinusoidal encodings [10], where even and odd dimensions are encoded separately (Eq. 10, Eq. 11), but together they define the positional encoding  $E_{\text{pos}}$ , generalizing to sequence lengths not seen during training. Token embeddings and positional encodings are then summed to produce the input to the first Transformer layer (Eq. 12), carrying both semantic and positional information into the forward pass.

$$E_{\text{pos}} \in R^{T \times d_{\text{model}}} \quad (9)$$

$$PE_{\text{pos},2i} = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \quad (10)$$

$$PE_{\text{pos},2i+1} = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \quad (11)$$

$$X_0 = E[\text{tokens}] + E_{\text{pos}} \quad (12)$$

#### B. Forward pass and training as non-linear inversion

Neural networks can be interpreted as a non-linear inverse problem [4], described by (Eq. 2): each layer applies a linear transformation followed by a non-linear activation. In geophysical inversion, training minimizes a least-squares misfit between predicted and observed data. In LLMs, the analogous objective is the minimization of cross-entropy loss (Eq. 13), which measures the negative log-likelihood of the true next token under the model's predicted distribution, the standard training objective for autoregressive language models [5].

$$\min_m \mathcal{L}(m) = - \sum_t \log p_m(w_t | w_1, \dots, w_{t-1}) \quad (13)$$

The Jacobian (Eq. 14) describes how sensitive the network's output is to its parameters, computed explicitly as the sensitivity kernel in classical inversion. In neural networks the Jacobian is never explicitly formed because computing billions

of parameters would be prohibitive. Instead, the gradient of the loss  $L$  with respect to the weights (Eq. 15) is computed implicitly through backpropagation [9], equivalent to the adjoint-state method used in full waveform inversion [4].

$$J = \frac{\partial f(m)}{\partial m} \quad (14)$$

$$\nabla_m L = J^T (f(m) - d) \quad (15)$$

#### C. Model computation is deterministic

At inference time, a trained model processes an input and produces *logits* in the forward pass. The same input produces the same *logits* (deterministic), but sampling steps add randomness in token selection. API documentations say that lower temperature makes the response more deterministic, while higher temperature makes it more “creative” (stochastic).

As noted by Leite [1], deterministic processes can be regarded as a particular case of probabilistic ones, with the deterministic forward pass representing the limiting case where a given input maps to a single outcome with certainty. In other words, probabilistic processes are more encompassing in the real world than deterministic ones.

#### D. Model training is stochastic

Random initialization, training data distribution and *mini-batch* sampling affect optimization trajectories through Stochastic Gradient Descent, with Adam [11] being the standard adaptive optimization method in modern LLM training. Adam keeps two moment estimates of the gradient at each step: the first moment,  $m_t$  (Eq. 16), is a running mean that tracks the direction of the gradient, and the second,  $v_t$  (Eq. 17), is the exponential moving average of the squared gradients, tracking their magnitude. Since both moments are initialized at zero, they are heavily biased toward zero in early training steps. Bias-corrected estimates  $\hat{m}_t$  and  $\hat{v}_t$  (Eq. 18, Eq. 19) are used instead, with the correction vanishing as training progresses. The parameter update (Eq. 20) produces an adaptive step size for each parameter that is large where gradients are small, and conservative where gradients oscillate. It does so by dividing the corrected first moment by the square root of the corrected second moment.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (16)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (17)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (18)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (19)$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (20)$$

These stochastic elements also shape the model’s “personality”, that is, the style, verbosity and linguistic structure observed in its responses, reflecting artifacts of training data and optimization choices. Two models trained on the same data but with different random initialization or *minibatch* sequences will follow different optimization trajectories and converge to different local minima of the loss landscape [5], producing models with distinct behavioral characteristics even before the alignment fine-tuning step.

### E. Transformers

Input token representation  $X_0$  is produced by (Eq. 12), projected into three distinct spaces through learned weight matrices: the Query matrix  $W_Q$  (Eq. 21), the Key matrix  $W_K$  (Eq. 22), and the Value matrix  $W_V$  (Eq. 23). Multiplying  $X_0$  by  $W_Q$ ,  $W_K$ , and  $W_V$  produces the Query  $Q$ , Key  $K$  and Value  $V$  matrices used in the attention computation. The Query represents what each token is looking for in the context, the Key represents what each token offers, and the Value carries the actual content to be retrieved. An output projection matrix  $W_O$  (Eq. 24) then maps the attention output back to the model dimension. Together, these projections define the attention space in which the model computes contextual similarity between tokens.

$$W_Q \in R^{d_{\text{model}} \times d_k} \quad (21)$$

$$W_K \in R^{d_{\text{model}} \times d_k} \quad (22)$$

$$W_V \in R^{d_{\text{model}} \times d_v} \quad (23)$$

$$W_O \in R^{d_v \times d_{\text{model}}} \quad (24)$$

Transformers condition on the full context window instead of just the immediate past, overcoming the limitation of first-order Markov assumptions [10]. Attention mechanisms learn dynamic, weighted dependencies across the entire context, computing a square  $T \times T$  attention matrix (Eq. 25) whose entries reflect the relevance of each token to preceding tokens. In autoregressive models, such as GPT, Claude and Grok, a causal mask prevents attention to future positions, preserving the generation order from left to right. This mechanism has quadratic complexity in sequence length and explains why Transformers are computationally expensive for long contexts, thus, various approximations exist.

The scaled dot-product attention (Eq. 26, Eq. 27) then produces a weighted combination of Value vectors, allowing the model to focus on relevant parts of the history regardless of token distance. Transformers are deterministic functions of their input, where attention weights are computed instead of sampled, but the expressiveness of their context-dependent representations is what motivates stochastic sampling over the resulting token distributions.

$$A \in R^{T \times T} \quad (25)$$

$$\text{Att}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (26)$$

$$A = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) \quad (27)$$

### V. STOCHASTICITY AS PART OF AI MODEL’S SAMPLING

Being statistical machines, LLMs provide possible outcomes to ill-posed problems rather than unique solutions. Large training data allows the learned distribution to support multiple plausible continuations, manifesting as creativity or stylistic variation in the model’s responses. Unlike formal languages, the grammar of natural languages allows for the construction of ambiguous sentences, so that multiple outputs can satisfy the same input, and the outputs themselves may carry ambiguous interpretation.

Ill-posed problems cause this asymmetry not only because they admit different plausible solutions, but the training data also introduces the grammatical elements used to generate responses with possibly multiple interpretations, making the engineering of a model-agnostic prompt that produces stable responses a problem in itself.

### VI. DIVERGENCE BETWEEN MODELS

A modern LLM is defined by its learned weight matrices  $W$  (Eq. 3), whose values are the result of the training process described in the previous sections, where random initialization, *minibatch* sampling and optimization trajectories are the first source of divergence, analogous to two geophysical inversions of the same dataset producing different models depending on the starting point and regularization choices.

Alignment fine-tuning (Eq. 28) is the second source of divergence. After the base training, vendors apply instruction tuning, preference optimization and safety constraints [12], each weighted by vendor-specific  $\lambda$  terms that bias the learned distribution in different directions:  $\lambda_1$  defines how the model parses and prioritizes instructions,  $\lambda_2$  shapes stylistic and tonal preferences, and  $\lambda_3$  handles refusal and constraint behavior.

These terms are neither disclosed nor consistent across vendors, so models from different vendors may share similar base training stochasticity and behave very differently due to the alignment biases introduced by these terms. This was observed empirically: a model may weight instructions placed early in the prompt more heavily than later ones, negative constraints may be treated as strict requirements by one model but as soft stylistic suggestions by another, and verbosity can differ even between models from the same vendor.

$$\begin{aligned} \min_W & \left[ \|f(W) - d\|^2 \right. \\ & + \lambda_1 \text{InstructionLoss}(W) \\ & + \lambda_2 \text{PreferenceLoss}(W) \\ & \left. + \lambda_3 \text{SafetyLoss}(W) \right] \end{aligned} \quad (28)$$

Models don’t have access to their own training metadata, such as the  $\lambda$  terms, optimization parameters and alignment

objectives at inference time, so when queried about their preferences, they produce approximate responses inferred from the learned distribution. In practice, querying provides initial information about a model’s behavioral tendencies as a starting point, to be confirmed during testing. From empirical observation, formulating a model-agnostic prompt becomes a problem of finding the minimum set of constraints that all target models consistently follow, without breaking the behavior in any of them. In other words, the prompt itself becomes the regularization instrument.

## VII. PROMPT ENGINEERING AS AN ILL-POSED INVERSE PROBLEM

Initial contact with an LLM commonly starts with a direct question, “what is this” or “how can this be done”, and the model outputs a direct response out of multiple possible outcomes. Unexpected responses generally means that the model chose the most statistically plausible outcome in the absence of extra information to guide it.

Training data defines what and how a model answers, thus asking the same question to different models can result in widely different responses, all likely to satisfy the prompt. Additional instructions and context help stabilize the output, just like *a priori* conditions constrain an inversion. In an analogy to geophysics, Leite [1] observes that the regularization operators introduced by additional constraints act as smoothing filters, more specifically the low-pass.

### A. Regularizing the prompt

Taking Tikhonov regularization (Eq. 29) [2] as a starting point, where  $L$  is the regularization operator, we can map this structure onto the proposed prompt regularization equation (Eq. 30): the forward operator  $f$  maps to the model operator  $M$ , the model parameters  $m$  correspond to the prompt  $p$  and the observed data  $d$  maps to the target output distribution  $y^*$  (the desired model response).

$$\min_m \|f(m) - d\|^2 + \lambda \|Lm\|^2 \quad (29)$$

$$\min_p \|M(p) - y^*\|^2 + \lambda \|p\|_{\text{constraint}} \quad (30)$$

The first term minimizes the distance between the model output and the desired response, while the constraint term stabilizes the solution. A large  $\lambda$  means the constraints are strong, and the prompt is heavily regularized, likely producing an oversmoothed response. A small  $\lambda$  means the constraints are weak, with the output fidelity term dominating the prompt and producing an unstable, divergent response across models.

Prompt constraints are expressed as linguistic instructions, not as quantifiable penalty terms in a vector space, thus, the prompt space is discrete and symbolic. The distance function  $\|M(p) - y^*\|^2$  is a conceptual measure of output divergence instead of a geometric norm, the constraint term  $\|p\|_{\text{constraint}}$  is a qualitative regularization placeholder, and  $\lambda$  is not analytically determined but empirically calibrated through the iterative querying methodology described later. Therefore,

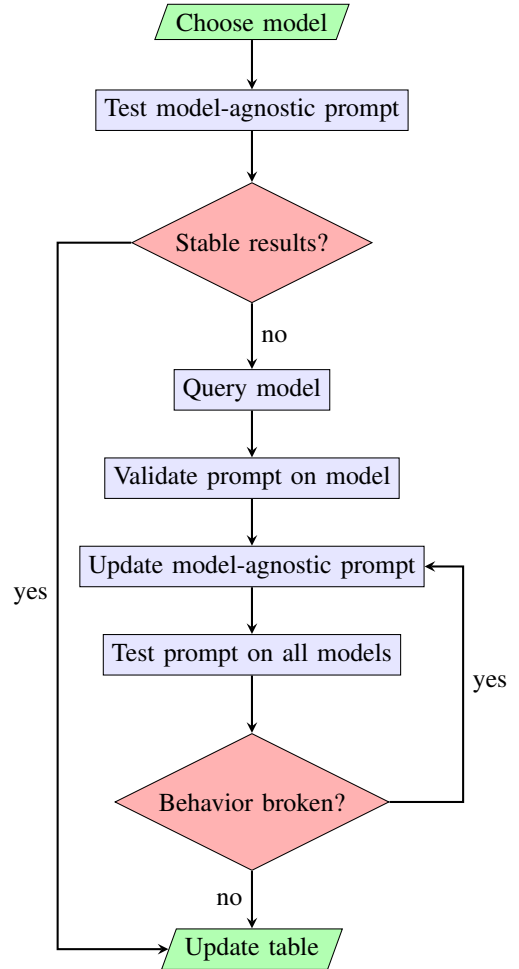
this formalization is analogous to Tikhonov regularization, as it provides a framework for reasoning about prompt constraints, where the analogy is more structural than mathematical.

### B. Beware the “CACE” principle

“Change anything, change everything” [13] is what we aim to prevent when writing prompts targeting different models, that is, altering one single instruction to satisfy a model while breaking the behavior on all others. It is generally not possible to know beforehand which modifications will preserve behavior, so for a model-agnostic implementation such as that presented here, the approach was to build a table with relevant preferences for each target model. To illustrate it, the following sample table groups hypothetical observations:

| Model | Role Assignment | Instruction Order | Negative Constraint |
|-------|-----------------|-------------------|---------------------|
| ABC   | Strong          | Medium            | Strong              |
| XYZ   | Strong          | Strong            | Light               |
| IJK   | Medium          | Medium            | Medium              |

This information comes from querying the models on their preferences and will be approximations. Once enough tests are run, with consistent results observed from varying input data, the table can then be updated from this iterative process:



When testing the model-agnostic prompt via API and the chosen model does not respond as expected, the prompt, the input data and the output are then submitted to the same model through its interactive interface, where it can be queried directly about what may have caused the unexpected behavior. The model will then suggest changes based on its internal preferences, a prompt specific to this model is then formulated, tested, and evaluated against the expected output. Model responses about their own preferences are approximations, inferred from the learned distribution rather than ground truth knowledge of their training, serving as hypotheses to be confirmed through empirical testing, not as explanations of the underlying behavior.

If it is confirmed to work, then the model-agnostic prompt must be carefully updated and tested on all models in incremental steps in order to avoid *CACE*. On many occasions, only a small change is sufficient to restore stable results across all models, and once this minimum common denominator is found, further optimization risks overconstraining the prompt and producing oversmoothed responses, or in the worst case, breaking the behavior on all models.

#### VIII. SUMMERIZER: THE METHOD IN PRACTICE

Summerizer is an AI-powered communication app written for iOS with an integrated workflow comprising multi-language transcription and text processing with optional translation steps before the input is processed by the LLM (a language normalization pass), and after the LLM outputs the result (a final translation pass). The user selects the operation to apply to the input text, such as generating a neutral breaking news ready for publishing, an executive report tailored for decision-makers needing only the core ideas of the text, or a specified number of bullet points. Transcription and optional translation passes are handled by Apple's proprietary models, while the text processing is handled by the user-chosen model (high-tier or cost-effective) from one of the supported vendors.

Aimed at communication professionals, interviewers, reporters, meeting organizers and lawyers, Summerizer serves all those who need a robust and dependable tool that makes handling large amounts of information simple, integrating the functionalities that matter. Its UI is designed around simplicity, with few taps triggering core actions and no hidden features. User-provided API keys eliminate subscription costs. Full documentation is available on Summerizer's support page, including a comprehensive FAQ, privacy policy and API key setup tutorials. The translation steps are tightly coupled to the workflow so that it is not mistaken for a pure translation tool. This highly integrated text processing tool has its origins in the summer of North Bay.

Development started with the application interfacing only with ChatGPT-3, but as soon as 3.5-Turbo and Grok 3 were added, the results were no longer stable, and at this point the investigation into the sources of such behavioral differences began. In order to keep the code base clean, branching was discarded entirely, so the natural solution was to find the minimum set of instructions that all selected models would

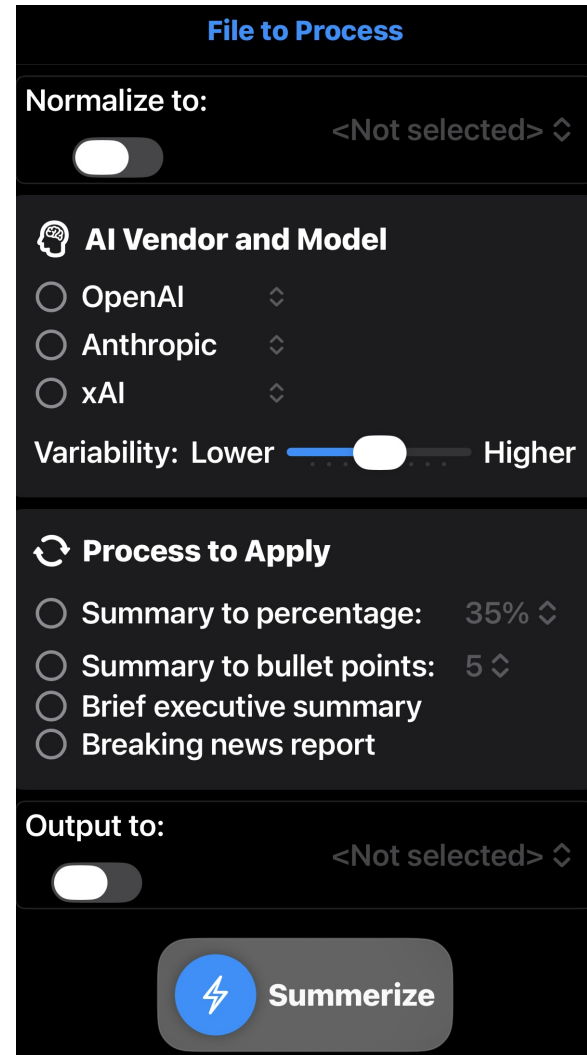


Figure 1. Processing page, where the user selects input text file, selects vendor/model, operation and optional translation steps. The *Variability* slider is a more user-friendly term for temperature. Each process is a model-agnostic prompt tuned and tested using the described methodology, along with other parameters only known at run-time.

consistently follow, the linguistic equivalent of the regularization parameters discussed earlier.

With the decision of offering a high-tier and cost-effective model from each supported vendor, so that the user manages API usage costs by selecting the appropriate model depending on how nuanced the input data is, also came the necessity of searching for the minimum common denominator in a systematic way. As vendors phase out old models and roll out new ones, the methodology stands: test the existing prompt on that model with different input datasets, observe the results, preserve the prompt if the results are as expected, or query the model and slowly rework parts of the prompt until stability is restored. At the time of writing, the supported high-tier models are GPT-5, Grok 4-High and Sonnet 4.5, while GPT-5-mini, Haiku 4.5 and Grok 4-Low are the cost-effective alternatives.

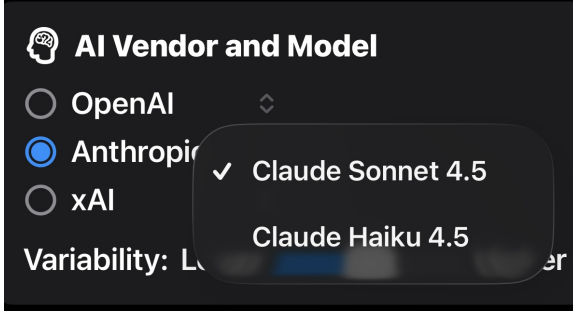


Figure 2. Vendor and model selector, with a high-tier and a cost-effective alternative. The user sets the vendor’s API key in the app’s *Preferences*, and it is then checked before the job is submitted.

## IX. OBSERVED DIVERGENCE AND VALIDATION

### A. Models tested

Table I summarizes the models integrated into Summerizer, the timeline of their addition, and the key behavioral observations that drove prompt adjustments.

Table I  
MODELS INTEGRATED INTO SUMMERIZER

| Model            | Timeline   | Observations                            |
|------------------|------------|-----------------------------------------|
| GPT-3            | May/2025   | Baseline prompts built                  |
| Grok 3           | June/2025  | Role assignment                         |
| GPT-3.5-Turbo    | July/2025  | Hints at imperative phrasing            |
| Grok 4           | Sept/2025  | Language enforcement                    |
| GPT-4o           | Oct/2025   | First instructions have stronger weight |
| GPT-4o-mini      | Dec/2025   | Sizes bullets by sentences, not words   |
| GPT-5            | Jan/2026   | Negative constraints                    |
| Sonnet/Haiku 4.5 | Feb/2026   | Strict imperative phrasing              |
| GPT-5-mini       | Mar/2026   | No changes needed                       |
| Grok 4-High/Low  | April/2026 | No changes needed                       |

GPT-3 is a simple text completion model, against which the first prompt was built. It weighted instructions evenly, as long as they did not conflict. Grok 3, through an online prompt session, stated the importance of role assignment as the very first instruction in an API call. GPT-4o introduced reasoning and shifted the behavior towards earlier instructions being followed more consistently than the later ones, suggesting a weighting system was in place, and then confirmed by the model through querying it. This made the observation about GPT-3’s even weighting clear in hindsight.

Grok 4 required an explicit instruction to output the response in the dominant language of the text, otherwise it would come out in English. Grok 3 and GPT-3.x models did it implicitly. GPT-5 preserved 4o’s behavior and further added weight to negative constraints, such as “do not reason about information not present in the text under any circumstance”, without which some results would contain unwanted extrapolation. Reinforcing with “under any circumstance” ensures this instruction is not treated as stylistic detail.

When tuning the bullet points prompt, GPT-4o-mini provided truncated responses. Inspecting the JSON reply payload confirmed the token budget was being exceeded, and querying the model revealed it preferred bullet length measured in sentences instead of words. This change (fortunately) did not break behavior on other models. xAI does not offer “mini” variants of Grok 4, and instead takes reasoning effort in JSON to adjust run costs, thus “low effort” replaced Grok 3.

Regarding imperative phrasing, GPT-3.5-Turbo sometimes ignored instructions without imperatives, while Sonnet and Haiku would totally ignore them without unambiguous imperative instructions. GPT-5-mini and Grok 4 with High or Low reasoning worked without any changes. As for models’ preferences and styles, vendors tune them according to their expected scope: Grok 3 is more verbose than Grok 4, just like Haiku 4.x, despite being the lighter model, is more verbose than Sonnet. These are behaviors described earlier as the model’s “personality” and attempting to override them with extra prompt constraints can introduce instability across other models, consistent with *CACE*.

### B. Test data and iterations

Various datasets in different languages were employed to confirm the stability in responses from all processes in Figure 1. These included: Eddie Van Halen’s page in Wikipedia, in English and Dutch, Battle of Midway online articles, news around the petroleum industry from February/2026 onwards in the publication’s original language, and medical data including the author’s diagnosis of lateral epicondylitis and sleep apnea.

Existing prompts were tested against any newly added model with all datasets, accounting for at least 6 API calls. If no issues were observed in the responses, the model replaces its previous equivalent in the app. If instability is observed, the methodology is applied: present the prompt, the input data and the response to the model in a chat window and query why it was not strictly followed in the API call; the model replies with its approximated preferences and the prompt is modified accordingly, then tested again with all datasets. If it works, this new prompt has to be tested against all models and datasets to confirm stability, and slowly adjust the prompt in the same way one would calibrate regularization parameters.

While major version upgrades required significant changes to the prompts, minor versions required little to no change. Each prompt update required several runs across all models with all datasets: at least 40 runs if it worked in the first attempt. Since the app is targeted at professionals for use in the field, there was no compromise in test runs to ensure stable and predictable behavior. When Claude Sonnet 4.5 and Haiku 4.5 were added, tuning the prompt required over 120 runs until all models responded correctly. Both were the least sensitive to how instructions were ordered, but were the most sensitive to imperative phrasing: *CACE* at its finest.

Upon replacing Grok 3 with Grok 4 Low Reasoning, no changes to the prompts were needed because Grok 4 was already satisfied from before, so only a set of 40 runs was necessary to confirm the replacement was safe. The number

of required iterations correlates with the degree of divergence between a new model and the existing prompt, consistent with the divergence framework presented earlier.

The *temperature* (*variability* in the app) began having less influence on the response as prompt constraints were added. Parameters that adjust token sampling, *top-k* and *top-p*, were left at the default (not passed in the JSON payload), but this may change in future versions. The one instruction that appears to have negated the temperature's effect was "do not reason about information not present in the text under any circumstance", which is critical for a communication app targeted at users needing exclusively factual analysis. It also confirmed the strong filtering that imperative sentences cause, with every single model following it even at maximum temperature. In this case, the only observed changes with temperature were stylistic choices of words, with all models "sounding like themselves". This suggests that prompt regularization operates at the logit instead of the sampling level.

#### X. EXISTING APPROACHES TO PROMPT OPTIMIZATION

Prompt optimization has been approached through gradient-guided discrete search [14] and through soft prompt tuning, where learnable embeddings are optimized through backpropagation against a fixed objective [15]. These methods optimize prompts for a single model and require either gradient access or a defined evaluation metric. The approach presented here differs in two fundamental ways: it operates across multiple models simultaneously without gradient access, and targets behavioral stability and not performance on a fixed benchmark. This methodology uses the model's self-reported preferences as hypotheses, confirmed through iterative empirical testing, to find the minimum set of constraints that all target models consistently follow, instead of optimizing a prompt analytically.

#### XI. CONCLUSIONS

Investigation into multi-model behavioral differences was motivated by the need to stabilize responses without branching code, that is, one prompt for each model, because it would create two sources of complexity: generating a stable prompt for each model, and its respective code path. Accepting that LLMs can produce varied outputs despite all of them functioning with the same statistical foundation, the research aimed to systematically estimate a common set of instructions that the selected models would consistently follow, characterizing the problem as ill-posed since multiple prompts can produce equally stable results, and the solution is not unique.

The analogies to geophysics, where inversion has been an integral tool for several decades, allow for identifying where the stochasticity comes from, and for acting on what is under the developer's control to interact with the LLM, that is, the prompt. AI vendors provide several parameters to shape the distribution where tokens are drawn from, for example, *top-k*, *top-p*, or reasoning effort, but none of these parameters can "reconcile" behavioral divergences introduced by different training trajectories and alignment choices. Prompt engineering with a regularization objective fills this gap.

With this methodology in place, it was possible to find the minimum common denominator for various models with minimal changes when some needed to be replaced by newer iterations. This approach saved time and resources during testing, since each API call incurs a cost on the user's API key balance. The identified sources of divergence provide the ground for extending the regularization framework and iterative methodology to future multi-model implementations, preserving response stability regardless of platform or vendor.

#### DISCLAIMER: ON GENAI USAGE

Anthropic's Claude Sonnet 4.6 was employed in the following steps during the writing of this article: laying out the basic sequence of the statistical concepts in the logical order they appear in an LLM; general corrections and clarifications of AI-specific equations, especially when their notation conflicted with those used in geophysics; complete English grammar review of the text.

All geophysical background, empirical observations, cross-disciplinary framing, and editorial decisions are the author's own. All results and observed behavioral patterns across models from multiple vendors came from extensive testing conducted as part of the development of Summerizer.

#### REFERENCES

- [1] L. W. B. Leite, *Conceitos da Análise Espectral de Sinais em Geofísica*, 1st ed. UFPA-IG, 2015.
- [2] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*. Winston & Sons, 1977.
- [3] W. Menke, *Geophysical Data Analysis: Discrete Inverse Theory*, 2nd ed. Academic Press, 1989.
- [4] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005.
- [5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [6] K. Aki and P. G. Richards, *Quantitative Seismology*, 2nd ed. University Science Books, 2002.
- [7] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [8] R. M. Gray, *Entropy and Information Theory*. Berlin/Heidelberg, Germany: Springer Science & Business Media, 2011.
- [9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. doi: 10.1038/323533a0
- [10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, arXiv:1706.03762.
- [11] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015, arXiv:1412.6980.
- [12] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, arXiv:2203.02155.
- [13] D. Sculley, G. Holt, D. Golovin, E. Davydov, J.-F. Crespo, D. Dennison, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.
- [14] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "AutoPrompt: Eliciting knowledge from language models with automatically generated prompts," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020. doi: 10.18653/v1/2020.emnlp-main.346 pp. 4222–4235.
- [15] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021. doi: 10.18653/v1/2021.emnlp-main.243 pp. 3045–3059.