A FRAMEWORK FOR PROMPT OPTIMIZATION AND TRANSLATION ACROSS FOUNDATION MODELS

Abhinav Shankaranarayanan Venkataraman, Athanasios N. Nikolakopoulos
Amazon Special Projects
428 Westlake Ave N, Seattle, WA 98109
{vabhina,athanani}@amazon.com

Vishwanath Kumaraswamy, Tao Zhang, Sarath Chander, Rohit Saboo & Suleiman A. Khan
Amazon Catalog AI
550 Terry Ave N, Seattle, WA 98109
{kvishwan,taozhg,smangudi,rohsaboo,suleimkh}@amazon.com

ABSTRACT

Foundation-model upgrades frequently break deployed prompt-based systems: target models differ in chat-template conventions, multimodal interfaces, context limits, and structured-output reliability. We study *cross-model prompt adaptation*: given a prompt program validated on a source model, produce a target-model prompt that preserves a semantic contract and an interface contract under bounded regression risk. We propose a governed, hierarchical adaptation framework that decomposes prompts into transferable semantic components and model-dependent structure and interface components, and optimizes only the non-transferable parts via budgeted search over system-level (L0) and template-level (L1) factors. Our optimization objective combines task utility with hard feasibility constraints (schema validity, parseability, policy compliance) and a risk penalty capturing output instability under stochastic decoding. On a large-scale structured prediction workload (128K labeled instances across text and multimodal settings), automated prompt translation matches expert human prompts while reducing manual iteration by 97%. Across varied model families, we observe consistent transfer patterns: semantic directives transfer reliably, whereas schema enforcement and provider-specific formatting require targeted adaptation; multimodal grounding improves recall but shifts the cost–performance frontier. These results frame prompts as portable programs and provide an auditable recipe for reliable pre-deployment prompt adaptation before upgrading foundation models in real-world deployments.

1 INTRODUCTION

Foundation models are being upgraded at a cadence that outpaces the maintenance cycle of production ML systems. In practice, a model upgrade is rarely a drop-in replacement: providers expose different chat templates, system-message semantics, multi-modal interfaces, output formatting behavior, and decoding defaults. For prompt-based applications that produce structured outputs, these differences routinely cause regressions in accuracy, schema validity, and operational reliability.

We formalize this setting as *cross-model prompt adaptation*. Given a source model M_s and a validated prompt program P_s, the goal is to construct a prompt P_t for a target model M_t that preserves two contracts: (i) a *semantic contract* specifying task intent and constraints; and (ii) an *interface contract* specifying admissible outputs and operational policies. Unlike single-model prompt optimization, cross-model adaptation must transfer validated behaviors across heterogeneous interfaces while bounding regression risk.

Current practice relies heavily on manual prompt re-engineering, which is slow, expensive, and difficult to scale as model ecosystems diversify. We instead treat prompt adaptation as a structured

Table 1: **Improvement-Operator Card** for prompt adaptation.

Field	Specification
Change target	Prompt program components $(P_{\text{struct}}, P_{\text{iface}})$; semantic core P_{sem} is frozen.
Objective	Maximize $J(M_t, P) - \lambda \cdot \text{Risk}(M_t, P)$ where J is dev-set micro-F_1.
Feedback signal	Structured labels on $\mathcal{D}_{\text{dev}}$ / $\mathcal{D}_{\text{val}}$ and diagnostics from schema/parse/policy checks.
Feasibility guardrails	Hard reject if JSON schema invalid, unparsable, or policy-noncompliant under the target interface.
Risk model	Penalize (i) schema violations, (ii) policy violations, and (iii) instability across stochastic decoding samples.
Acceptance rule	Select the best configuration on $\mathcal{D}_{\text{dev}}$ and confirm generalization on $\mathcal{D}_{\text{val}}$ / held-out tests.
Rollback / halt triggers	Halt and retain the last stable prompt if validation utility degrades beyond a tolerance or feasibility failures exceed a threshold.
Compute / budget	Bounded trials (N) with small inner-loop dev slices for fast iteration; full evaluation on held-out splits.
Expected failure modes	Overfitting to dev slice; brittle schema adherence under temperature; modality contradictions (image vs text).

optimization problem over a hierarchical prompt space, with explicit governance and feasibility constraints suitable for production deployment.

Our contributions are:

- A formalization of cross-model prompt adaptation with semantic and interface contracts.
- A governed hierarchical optimization framework over L0 (system) and L1 (template) prompt factors.
- Large-scale statistical evaluation with 95% bootstrap confidence intervals, demonstrating parity with expert human translation and a 97% reduction in manual effort.
- An analysis of prompt transferability across model scales and modalities, identifying predictable ceilings and cost–performance trade-offs.

2 Related Work

Prompt-as-code and programmatic prompting. Several systems treat LM interactions as programmable artifacts with explicit constraints and control flow. DSPy (Khattab et al., 2024) compiles declarative LM calls into self-improving pipelines via search and feedback, while LMQL (Beurer-Kellner et al., 2023) introduces a query language for constrained decoding and structured generation. Guidance (Guidance AI and Microsoft Research, 2023) provides a templating and control framework for enforcing output structure. These approaches primarily optimize or compile prompts *within a single model and interface*. In contrast, our work focuses on *cross-model prompt translation*: adapting a validated prompt program to a target model with different chat templates, decoding defaults, and schema reliability, while preserving both semantic intent and interface feasibility constraints. A key distinction is that prior prompt-as-code systems largely assume a fixed model interface and optimize prompt content or constrained decoding within that interface. Our setting treats *interface realization* (role hierarchy, chat-template family, and structured-output conventions) as a first-class optimization variable, with hard feasibility gating (schema/parse/policy) and a stability-aware risk penalty during selection.

Automatic prompt optimization. A large body of work studies automatic discovery of effective prompts for fixed models, including discrete prompt search (Shin et al., 2020; Deng et al., 2022), continuous prompt tuning (Li & Liang, 2021; Lester et al., 2021; Liu et al., 2022), and retrieval-augmented prompting (Khattab et al., 2022; Rubin et al., 2022; Zhang et al., 2022). These methods are complementary to our setting: they optimize prompt content assuming a fixed interface, whereas

we assume a *validated source prompt* and focus on translating it across heterogeneous model interfaces under feasibility and stability constraints.

Prompt transfer and reuse. Prior work observes that some prompt components generalize across models while others are brittle. Soft prompt transfer methods such as SPoT (Vu et al., 2022) show that learned prompt representations can transfer across frozen models, but do not address structured-output guarantees or interface mismatches. More recent work on prompt reuse and transfer focuses primarily on semantic equivalence, whereas our work explicitly decomposes prompts into semantic, structural, and interface components under real deployment constraints. Our work systematizes prompt transfer by explicitly decomposing prompts into semantic, structural, and interface components, and by empirically quantifying which components transfer reliably and which require targeted adaptation under real deployment constraints.

Structured generation and evaluation. Reliable structured outputs have been addressed via constrained decoding and schema enforcement mechanisms (Gao, R. & contributors, 2023; dottxt-ai, 2025; Dong et al., 2024). We treat these not as replacements for prompt translation, but as feasibility constraints that shape the search space and risk function. For evaluation, we follow best practices in holistic and robustness-oriented LM assessment (e.g., HELM (Liang et al., 2023), BIG-Bench (Srivastava et al., 2023), MMLU (Hendrycks et al., 2021)), adapting them to structured prediction with stability and schema-validity diagnostics.

3 METHOD

We develop an automated pipeline for translating structured prompt programs across foundation models. The workflow consists of five stages: (i) information gathering, (ii) guidance extraction, (iii) prompt generation and realization, (iv) batched inference, and (v) evaluation with governance. The design explicitly separates transferable prompt semantics from model-dependent structure and interface components.

3.1 PROBLEM DEFINITION

Let $\mathcal{M}$ denote a space of foundation models. Each model $M \in \mathcal{M}$ is characterized by a tuple $\langle \mathcal{I}, \mathcal{O}, \mathcal{C}, \mathcal{T} \rangle$, where $\mathcal{I}$ denotes supported input modalities (e.g., text, image), $\mathcal{O}$ denotes admissible output schemas and formatting constraints, $\mathcal{C}$ denotes the maximum context window, and $\mathcal{T}$ denotes model-specific prompting and chat-template conventions as specified by the model interface or API documentation.

A *prompt program* P defines a mapping

$$P : \mathcal{X} \to \mathcal{Y},$$

where $\mathcal{X}$ denotes structured inputs (e.g., text fields, images, metadata) and $\mathcal{Y}$ denotes structured outputs (e.g., JSON records or typed fields). For example, in structured attribute extraction for large e-commerce product data, $\mathcal{X}$ may include a product title, description, image, and some structured attributes (e.g., laptop display size, T-shirt color), while $\mathcal{Y}$ is a JSON object specifying attribute values (e.g., 14 inches, blue respectively) with explicit "`[NO]`/`[NA]`" fallbacks, where `[NO]` indicates there is no information in the input to derive the attribute, and `[NA]` indicates that the attribute is not applicable for the given instance.

Given a source model M_s with a validated prompt program P_s, the goal of cross-model prompt adaptation is to construct a prompt P_t for a target model M_t such that

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\ell(M_t(P_t, x), y)\big] \approx \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\ell(M_s(P_s, x), y)\big],$$

subject to hard feasibility and interface constraints $\mathsf{Cons}(P_t, M_t)$ imposed by the target model. These constraints include schema validity, output parseability, and policy compliance under the target model's interface.

We assume that the dataset $\mathcal{D}$ and labels is invariant across both source and target models, reflecting deployment-time consistency of data. However, we do allow for changes in modalities. Adaptation accounts only for differences in model interfaces and output behavior.

Here, $\ell(\cdot, \cdot)$ denotes a task-specific loss (e.g., $1 - F_1$ for structured prediction), and expectations are estimated empirically using development and validation splits.

3.2 HIERARCHICAL PROMPT DECOMPOSITION

We structure a prompt program as a tuple

$$P = \langle P_{\text{sem}}, P_{\text{struct}}, P_{\text{iface}} \rangle,$$

where P_{sem} encodes task semantics and decision logic (e.g., attribute definitions, labeling rules, and invariants), P_{struct} encodes output structure and constraints (e.g., JSON schemas, field-level requirements, and fallback rules), and P_{iface} captures model- or provider-specific interface conventions (e.g., system-role framing, chat-template format, and decoding directives).

Empirically, we observe that P_{sem} transfers reliably across models, whereas P_{struct} and P_{iface} are sensitive to differences in instruction hierarchy, schema strictness, and decoding behavior. Naively reusing these components frequently leads to schema violations or unstable outputs. Our framework therefore isolates and adapts the non-transferable components, while reusing the semantic core verbatim wherever possible.

3.3 PROMPT GENERATION AND OPTIMIZATION LOOP

Given a provider card describing a target model, an information agent extracts structured attributes (e.g., supported modalities, schema strictness, system sensitivity). A guidance agent distills these attributes into adaptation directives.

We sample prompt configurations from a factored search space $\Phi = \Phi_{\text{uni}} \times \Phi_{\text{fam}} \times \Phi_{\text{cap}}$, covering universal, family-specific, and capability-dependent factors. Φ_{uni} represents prompt edits that are applicable across models such as constraints, instruction phrases, and schema definitions. Φ_{fam} captures the adaptations for each specific model family such as delimiter styles (like XML tags or JSON), response formats, and role framing. Φ_{cap} represents model capabilities such as multimodal inputs, reasoning verbosity, and context window constraints. A realization layer renders each candidate prompt into the target model's native chat-template format.

Algorithm 1 Automated Prompt Translation

Require: Provider specification C, development set $\mathcal{D}_{\text{dev}}$, trial budget N

Parse model capabilities and interface constraints from C
Distill adaptation guidance directives
Initialize best utility $U^\star \leftarrow -\infty$
for $k = 1$ to N **or until convergence do**
    Sample prompt configuration ϕ_k
    Generate candidate prompt components $P^{(k)}_{\text{struct}}$ and $P^{(k)}_{\text{iface}}$
    Compose prompt $P_k = \langle P_{\text{sem}}, P^{(k)}_{\text{struct}}, P^{(k)}_{\text{iface}} \rangle$
    Realize P_k into the target model's native format
    if Feasible(P_k, M_t) **then**
        Evaluate task utility J_k on $\mathcal{D}_{\text{dev}}$
        Compute risk-aware utility $U_k = J_k - \lambda \cdot \text{Risk}(M_t, P_k)$
        if $U_k > U^\star$ **then**
            $U^\star \leftarrow U_k$
            $\phi^\star \leftarrow \phi_k$
return best configuration $\phi^\star$

3.4 OPTIMIZATION OBJECTIVE

We optimize a risk-aware utility objective:

$$\max_{P_{\text{struct}}, P_{\text{iface}}} \; J(M_t, P) - \lambda \cdot \text{Risk}(M_t, P),$$

subject to hard feasibility constraints. Here, J denotes micro-averaged F1 computed at the attribute level, aggregating true and false positives and negatives over the test set to reflect end-to-end structured prediction performance under class imbalance. The risk term Risk penalizes schema violations, policy violations, and output instability under stochastic decoding. Candidates violating feasibility constraints are rejected.

Rollback and stopping criteria. To ensure deployment readiness, we treat the adaptation loop as *rollbackable*. If the selected candidate prompt fails feasibility checks on validation (schema/parse/policy) or its risk-aware utility drops below a baseline prompt by more than a small tolerance, we halt adaptation and retain the last known-good prompt. In practice, this prevents unstable self-edits from being promoted even when they appear to improve dev-set utility.

3.5 Translation Dimensions

We characterize model heterogeneity via *translation dimensions* (Table 2), which directly inform the search space and feasibility constraints.

Table 2: Key categories of translation dimensions. Each captures a class of heterogeneity across foundation models and explains why it affects prompt adaptation.

Dimension	Definition / Typical Values	Impact on Translation
Input modality	Supported input types (text; text+image)	Multimodal models require image-handling instructions
Context window	Maximum tokens accepted (128K–1M)	May require prompt compression or truncation
Output format	Preferred serialization (JSON; Markdown+JSON)	Prompts must enforce correct schema
Schema strictness	Tolerance to output deviations (Weak–Strict)	Rigid schemas require tighter constraints
System sensitivity	Dependence on system role framing (Medium–Very High)	High sensitivity demands stronger role instructions
Sampling defaults	Default decoding regime (top-k, top-p, mixtures)	Affects stability of generations
Error handling style	Typical fallback behavior (minimal, graceful, verbose)	Influences how "[NO]/[NA]" cases are instructed
Instruction following	General compliance (High–Very High)	Guides level of scaffolding needed
Chain-of-thought tendency	Likelihood of verbose reasoning (Low–Very High)	Informs verbosity and reasoning emphasis
Multilingual strength	Coverage across languages (Medium–Very High)	Affects whether to add explicit language metadata
Cost / latency	Relative efficiency and token budget (Low–Very High)	Determines cost–performance trade-offs

4 Experimentation Setup

Overview. We evaluate our framework using a systematic pipeline that first optimizes prompt configurations on a development split and then applies the selected templates for large-scale inference and evaluation. The process follows Algorithm 1: configuration sampling, template realization for the target model format, validation execution, risk-aware scoring, and human oversight for deployment readiness.

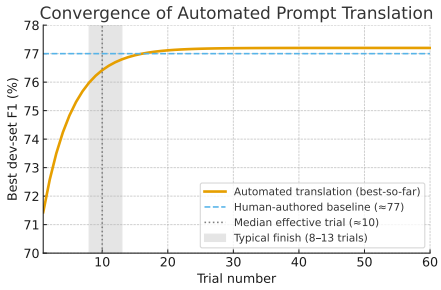

Figure 1: Convergence of automated prompt translation. Most runs converge within 8–13 trials

Data. We use a 128k large e-commerce catalog structured product dataset $\mathcal{D}$ spanning EN-US and EU locales, with a mix of human-audited and machine-labeled examples. The dataset is stratified over product categories and attributes to ensure coverage of both frequent and rare cases. Unless noted otherwise, we use a 60/20/20 split into development $\mathcal{D}_{\text{dev}}$, validation $\mathcal{D}_{\text{val}}$, and held-out $\mathcal{D}_{\text{test}}$ test sets. The development set powers the inner-loop objective, while the held-out test set is reserved for selection confirmation and regression analysis.

Models. We target eight representative base models spanning text and multimodal variants (e.g., Nova Lite/Pro/Premier, Claude 3.7/4 Sonnet, Mistral 3.1, QwQ-32B, Nemo). For each model, we parse the provider card to recover context mode, instruction hierarchy, post-training chat-template family (Harmony, HF-ChatML, INST, Claude-XML), and capabilities (image input, tool use, reasoning).

Model heterogeneity. Although the target models share a broadly instruction-following interface, they differ substantially in operational characteristics. Table 3 summarizes representative dimensions that directly affect prompt translation choices, including input modalities, context windows, output formats, schema strictness, sensitivity to system prompts, default sampling regimes, and cost–latency trade-offs. These heterogeneities directly inform the optimization pipeline: schema strictness feeds into feasibility constraints $\mathsf{Cons}(P_t, M_t)$, context window size and default temperatures inform the search space Φ, and variation in instruction-following or chain-of-thought tenden-

Table 3: Representative model-specific dimensions that feed directly into prompt translation and optimization choices.

Dimension	Nemo	Nova Lite	Nova Pro	Nova Premier	Claude 3.7 Sonnet	Claude 4 Opus	Claude 4 Sonnet
Modalities	T	T+I	T+I	T+I	T	T+I	T+I
Ctx window	128K	300K	300K	1M	200K	200K+	200K
Outputs	J	J	J	J	M/J	M/J	M/J
Schema	W	S	S	SS	F	S	S
Sys. sens.	M	H	H	H	H	VH	H
Temp. def.	0.7	0.5	0.5	0.3	1.0	0.8	0.8
Sampling	Tk	Tp	Tp	Tp	Mix	Tp	Tp
Error hand.	Min	GF	GF	SF	VR	CR	CR
Instr. follow	H	H	VH	VH	VH	VH	H
CoT tendency	L	M	H	H	M	VH	H
Multi-ling.	M	H	H	H	VH	VH	H
Cost/latency	L	M	H	VH	M	VH	H

Legend: T = Text only, T+I = Text+Image; J = JSON, M/J = Markdown+JSON; W = Weak, S = Strong, SS = Strict, F = Flexible Schema; M = Medium, H = High, VH = Very High, L = Low; Tk = Top-k, Tp = Top-p, Mix = mixture; Min = Minimal, GF = Graceful, SF = Strong fallback; VR = Verbose rationale, CR = Concise rationale.

cies drives model-specific guidance construction. In this way, the table serves as a bridge between model analysis and optimization.

Evaluation, Regression, and Governance. We parse model generations into label sets and compute micro/macro Precision, Recall, and F_1 using a Synonym Matcher to reduce lexical false negatives. We report 95% bootstrap confidence intervals (1,000 resamples) and subgroup diagnostics by attribute and category. We also run a compatibility suite covering schema validity, JSON parseability, and stability under positive temperature ($T = 0.5$), as well as mixture tests to guard against overfitting. Failures trigger a closed-loop return to optimization with updated (λ, Φ). Baselines include standardized prompts (no translation), expert human-translated prompts, and ablated automated variants (text-only vs. multimodal), all evaluated using identical manifests and code.

Implementation Details and Reproducibility. The optimization runner defaults to `NUM_TRIALS` $= 60$ with a small development split ($|\mathcal{D}_{\text{dev}}| = 10$) for rapid inner-loop evaluation. The template runner executes end-to-end regression with `MAX_RETRIES` $= 5$ and `NUM_WORKERS` $= 5$. Prompts are realized into each model's native post-training format (e.g., Harmony, HF-ChatML, INST, Claude-XML) via a realization layer $\mathcal{C}_{\mathsf{CT}}$. Feasibility is enforced through hard constraints (CONS) on JSON schema validity, safety policy compliance, and parser compatibility, while soft risk scores capture structural conformance and jailbreak heuristics, with λ controlling the quality–risk trade-off during selection.

Cost, Throughput, and Sojourn. We track cents per inference (proxying tokens under fixed decoding), wall-clock sojourn, and human effort days. Sojourn is defined as the end-to-end wall-clock time from initiating prompt adaptation to a deployment-ready prompt artifact, including optimization runs, retries, and validation, but excluding external governance review and downstream product rollout. Throughput is reported with $W = 5$ workers under rate limits.

Ablations. We ablate individual design choices by selectively disabling components while keeping data splits, evaluation metrics, and optimization budgets fixed. Specifically, we test: (i) no synonym matching, (ii) text-only vs. multimodal prompts, (iii) removal of the risk term λ, (iv) uniform random search, and (v) chat-template mismatch (no realization).

Synonym Matcher for Automated Evaluation. Evaluation uses a multi-layer matcher combining exact match, edit-distance similarity, and embedding-based semantic similarity to detect label equivalence. This reduces false negatives, particularly in multilingual settings, and yields more faithful precision and recall estimates.

Datasets for Prompt Translation Regression Evaluations. We adopt stratified sampling over product categories, balancing head and tail items as well as filled and unfilled slots, and increasing diversity via semantic clustering. Approximately half of labels are human-audited, with the remain-

Table 4: Performance comparison across models and prompt types. Precision, Recall, and F1 are reported with ± margin of error (MOE). Cost is measured per inference.

Model	Prompt	Precision	±MOE	Recall	±MOE	F1	±MOE	Cents/inference
Nemo	Standardized	80.61%	0.23%	71.66%	0.25%	75.87%	0.23%	0.03588
	Human (test)	80.61%	0.23%	73.38%	0.24%	76.69%	0.23%	0.03425
Nova Lite	Standardized	83.16%	0.22%	67.79%	0.26%	75.05%	0.24%	0.03926
	Auto Trans. (text)	82.19%	0.22%	71.58%	0.25%	76.50%	0.23%	0.04657
	Auto Trans. (images)	82.36%	0.22%	73.16%	0.24%	77.25%	0.23%	0.10273
	Human (text)	82.28%	0.22%	71.71%	0.25%	76.61%	0.23%	0.04072
	Human (images)	82.47%	0.22%	73.68%	0.24%	77.35%	0.23%	0.09688
Nova Pro	Standardized	84.95%	0.22%	66.61%	0.26%	74.84%	0.24%	0.5226
	Auto Trans. (text)	84.20%	0.22%	69.94%	0.25%	76.28%	0.23%	0.62088
	Auto Trans. (images)	83.74%	0.22%	73.00%	0.24%	77.73%	0.23%	1.36968
	Human (images)	82.68%	0.22%	72.18%	0.25%	76.94%	0.23%	1.29168
Nova Premier	Standardized	84.64%	0.22%	71.06%	0.25%	77.47%	0.23%	1.7099
	Auto Trans. (text)	85.11%	0.22%	71.67%	0.25%	77.94%	0.23%	1.7794
	Auto Trans. (images)	85.29%	0.22%	73.89%	0.24%	79.00%	0.22%	4.3631
	Human (text)	84.97%	0.22%	70.99%	0.25%	77.38%	0.23%	2.0231
	Human (images)	84.42%	0.22%	71.72%	0.25%	77.54%	0.23%	4.1194
Claude 3.7 Sonnet	Standardized	84.76%	0.22%	71.54%	0.25%	77.70%	0.23%	5.1246
	Auto Trans. (text)	84.62%	0.22%	72.92%	0.24%	78.26%	0.23%	16.2367
	Auto Trans. (images)	84.17%	0.22%	75.12%	0.24%	79.08%	0.22%	20.6096
Claude 4 Sonnet	Standardized	85.87%	0.22%	69.43%	0.25%	76.92%	0.23%	5.1246
	Auto Trans. (text)	84.69%	0.22%	71.76%	0.25%	77.49%	0.23%	16.2367
	Auto Trans. (images)	85.04%	0.22%	72.48%	0.24%	78.04%	0.23%	20.6096

der derived from machine translation. These labels are used uniformly across all methods, enabling statistically reliable relative comparisons (margin of error $\leq 5\%$). For selection, we sample multiple candidates under positive temperature and choose the prompt with best validation generalization, confirmed on a held-out test set to avoid overfitting.

5 Results

5.1 Experimental Results

Table 4 reports precision, recall, and F1 (with 95% bootstrap MOE) across models and prompt types. **Standardized** denotes verbatim reuse of the source-model prompt without interface adaptation; **Human** denotes expert-authored target-model prompts; and **Auto Trans.** denotes prompts produced by our automated translation pipeline. Costs are measured under fixed decoding settings and averaged across asynchronous runs. Across models, three consistent patterns emerge.

Mid-tier models exhibit the largest gains from translation. Automated prompt translation yields substantial improvements for mid-capacity models where interface mismatches most strongly affect structured-output reliability. For Nova Lite, recall improves from 67.79% under standardized prompts to 71.58% with auto-translated text prompts and 73.16% with auto-translated image prompts, corresponding to a $\sim$540 bps recall increase and a $+2.20$ F1 gain ($75.05\% \rightarrow 77.25\%$). Similarly, Nova Pro recall rises from 66.61% to 73.00% with auto-translated image prompts, improving F1 from 74.84% to 77.73%. These gains bring automated prompts into parity with expert-authored variants (e.g., Nova Lite image: 77.25% vs. 77.35% F1), demonstrating that structured translation can recover expert-level performance without manual re-engineering.

Diminishing returns appear for higher-capacity models. For stronger models such as Nova Premier and the Claude variants, standardized prompts already operate near saturation (e.g., Nova Premier: 77.47% F1; Claude 3.7: 77.70% F1). Automated translation yields smaller but consistent gains: Nova Premier improves by $+0.47$ F1 with auto-translated text prompts and by $+1.53$ F1 with auto-translated image prompts, while Claude models see $+0.56$ to $+1.38$ F1 improvements depending on modality. This pattern is consistent with a ceiling effect in which higher-capacity models are less sensitive to prompt interface mismatches, though translation remains useful for portability and schema enforcement across heterogeneous provider interfaces.

Automated translation achieves parity with human prompt engineering. Within each modality, automated prompts match or slightly exceed expert-authored prompts. For Nova Lite, auto-translated and human-authored text prompts achieve nearly identical F1 (76.50% vs. 76.61%). For Nova Pro, auto-translated image prompts slightly outperform human-authored image prompts (77.73% vs. 76.94%). These results indicate that guardrailed automated translation can substitute for costly manual prompt engineering without sacrificing aggregate quality.

5.2 Effort and Turnaround

Table 5 compares human effort and end-to-end turnaround between manual and automated approaches. Manual design requires $\sim$60 person-days per model across the board. Automation reduces this to 2 days of oversight, with turnaround ranging from 1–6 days depending on model family. The variance reflects model-specific heterogeneities: lightweight models such as Mistral 3.1 and QwQ stabilize quickly (1 day), while Claude 3.7 requires longer governance cycles (6 days) due to stricter schema enforcement and higher system sensitivity. Overall, these results show that automated translation scales across model families while absorbing provider-specific conventions, reducing both human effort and adaptation time by more than an order of magnitude.

5.3 Ablation Studies

We conducted ablations to assess the contribution of each design choice: (i) removing synonym matching reduces recall by 2–3 points, confirming its importance for faithful scoring; (ii) dropping the risk term λ increases unsafe generations and JSON violations; (iii) multimodality-only ablations confirm that visual grounding is the main driver of recall improvements; and (iv) uniform random search underperforms risk-aware search by $\sim$2 F1 points on average. Together, these results validate both the necessity and effectiveness of the structured search loop.

5.4 Qualitative Examples

Error analysis reveals common strengths and failure modes. Auto-translations reliably preserve JSON schemas and adapt to provider chat templates. Multimodal prompts correctly infer visual attributes (e.g., *"red floral dress"* from product imagery). However, failures occur in ambiguous cases, such as distinguishing *navy* vs. *black*, or when images contradict textual metadata. These examples highlight that while automation preserves structure, attribute-level accuracy still depends on input signal quality.

Table 5: Comparison of effort and turnaround time between manual and automated prompt approaches.

Model	Manual Approach	Automated Approach	
	Human Effort (days)	Human Effort (days)	Turnaround (days)
Nova Lite	60	2	2
Nova Pro	60	2	4
Claude 3.7	60	2	6
Mistral 3.1	60	2	1
QwQ-32B	60	2	1

Artifact and reproducibility statement. We will provide a minimal reproducer sufficient to re-execute the optimization loop and evaluation: configuration schemas, prompt templates, the realization layer interface, evaluation scripts, and fixed seeds for all reported experiments. When data or provider cards cannot be shared, we will include sanitized equivalents and detailed manifests that preserve the reported comparisons.

6 Conclusion

We introduced *cross-model prompt adaptation* as a principled method for maintaining prompted systems across heterogeneous foundation models. Our hierarchical optimization framework demon-

strates that automated adaptation can match expert human prompt translations while exposing predictable transferability patterns across model families and interfaces. Our findings highlight three key insights: (1) prompt components exhibit asymmetric transferability, with semantic cores transferring reliably while structural and interface specifications require targeted adaptation; (2) mid-tier models benefit most from systematic translation, closing large gaps induced by interface mismatches; and (3) multimodal adaptations consistently improve recall, albeit at increased computational cost. While our study focuses on structured prediction workloads and a fixed set of model families, the framework generalizes naturally to other prompt-driven applications. Future work includes automated discovery of model-specific interface conventions, tighter integration with constrained decoding methods, and extending adaptation to continuously evolving model APIs. As foundation models continue to diversify, systematic prompt adaptation will be essential for translating model advances into reliable, production-ready systems.

REFERENCES

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7(PLDI): 1946–1969, 2023. doi: 10.1145/3591300. URL https://dl.acm.org/doi/10.1145/3591300.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *EMNLP*, 2022. URL https://aclanthology.org/2022.emnlp-main.222/.

Yucheng Dong, Siyuan Lai, Junru Zhao, Bohan Shao, Haoran Shen, Lianmin Zheng, Tianqi Chen, et al. XGrammar: Flexible and efficient structured generation for large language models. *arXiv preprint arXiv:2411.15100*, 2024. doi: 10.48550/arXiv.2411.15100. URL https://doi.org/10.48550/arXiv.2411.15100.

dottxt-ai. Outlines: Structured outputs for LLMs. https://github.com/dottxt-ai/outlines, 2025.

Gao, R. and contributors. JSONformer: A bulletproof way to generate structured JSON from LLMs. https://github.com/1rgs/jsonformer, 2023.

Guidance AI and Microsoft Research. Guidance: A guidance language for controlling large language models. https://github.com/guidance-ai/guidance, 2023.

Dan Hendrycks et al. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive NLP. *arXiv preprint arXiv:2212.14024*, 2022. doi: 10.48550/arXiv.2212.14024. URL https://doi.org/10.48550/arXiv.2212.14024.

Omar Khattab, Arnav Singhvi, et al. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021. URL https://aclanthology.org/2021.emnlp-main.243/.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021. URL https://aclanthology.org/2021.acl-long.353/.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research (TMLR)*, 2023. URL https://openreview.net/forum?id=iO4LZibEqW. Featured Certification.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022. URL https://aclanthology.org/2022.acl-short.8/.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *NAACL*, 2022. URL https://aclanthology.org/2022.naacl-main.191/.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, 2020. URL https://aclanthology.org/2020.emnlp-main.346/.

Aarohi Srivastava et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research (TMLR)*, 2023. URL https://openreview.net/forum?id=uyTL5Bvosj.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. SPoT: Better frozen model adaptation through soft prompt transfer. In *ACL*, 2022. URL https://aclanthology.org/2022.acl-long.346/.

Yiming Zhang, Shi Feng, and Chenhao Tan. Active example selection for in-context learning. In *EMNLP*, 2022. URL https://aclanthology.org/2022.emnlp-main.622/.