Dynamic Semantic Tags Reduce Hallucinations in Small-LLM Post-Training

Shreyas S. Vidyarthi[1*] Shreyas Pimpalgaonkar[2*]
Iraklis Koutrouvelis[1] Alexandros G. Dimakis[2]
Maddie Daianu[1] Jatin Ghia[1] Wei Huo[1]

[1]Intuit Inc [2]Bespoke Labs
[*] Leading Authors

Abstract

We describe how we built SEEWHY, a compliance-critical fine-tuning pipeline that uses synthetic data with engineered distributions, GEPA prompt optimization (Agrawal et al., 2025), and a novel way of curating supervised fine-tuning (SFT) datasets that we call *Dynamic Semantic Tags* (DSTs). The goal is to train a small, specialized model for product-recommendation explanations: given a credit card and a user profile, explain why this card is a good fit for this particular user, while never hallucinating compliance-critical details such as interest rates or card names. Our specialized model is more accurate compared to frontier models and can served at 15–20× lower cost.

1. Introduction

We describe a fine-tuning pipeline that uses a novel data curation idea, *Dynamic Semantic Tags*, to reduce hallucinations. The goal is to train a small language model called SEEWHY. This is a small specialized model for product recommendation explanations: given a credit card and a user profile, explain why this card is a good fit for this particular user (Figure 1).

There are several reasons for fine-tuning a small specialized model for a narrow task: Operational cost savings, reduced latency, and reduced dependencies on external model version deprecations.

The model we train, SEEWHY, must generate customized product-fit explanations while adhering to strict compliance constraints. For example, it cannot hallucinate interest rates and the stated credit card names must be an exact match. The explanation responses also need to follow all the complex rules for legal and partner compliance.

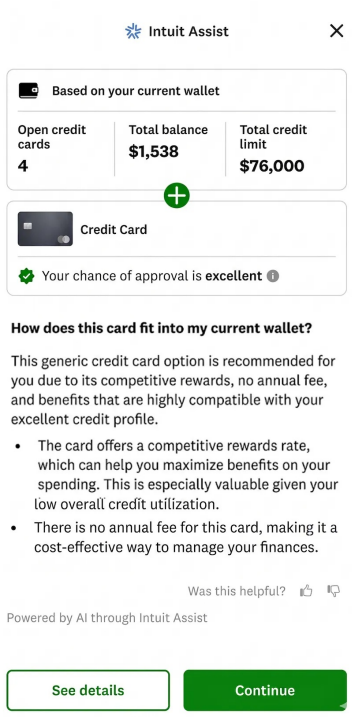

Figure 1. In-app product experience: SEEWHY generates a personalized explanation of how a recommended card fits the user's current wallet.

2. The Problem: When Fine-Tuning Makes Things Worse

We found that using randomly sampled data from users' interaction logs introduces unintentional biases and potential spurious correlations across word phrases. For example, credit card fee data is heavily imbalanced toward $0 annual fees, since most cards do not charge them. Most users who see explanations generated for such cards would see word-sequences like "this $0 fee card…" or "has an annual fee

Figure 2. Instructions and data schema example. The fixed workflow combines system instructions and constraints (left) with curated user-credit and product-fact context (right) that is fed to the specialized language model.

of \$0…" or "has no fees…". During fine-tuning, models can exploit this imbalance by over-predicting \$0 annual fees because it is statistically correct, most of the time. Training directly on raw interaction logs therefore risks reinforcing dataset biases and introduces compliance violations because the model learns to use these spurious correlations.

2.1. The Naive Fine-Tuning Trap

After SFT on imbalanced production data, offline evaluation performance degrades substantially. This is specifically because the model is learning to use spurious correlations from our biased training sample. The model doesn't just become less accurate; it learns to make incorrect disclosures with greater confidence. *Naive fine-tuning makes things worse!*

When benchmarking performance in an offline simulation, the base Meta Llama 3.1 8B Instruct model (Grattafiori et al., 2024), without any task-specific fine-tuning, achieves 96.7% fee accuracy. This may seem high, but it is not sufficient since inaccurate fee or rate disclosures are not merely quality issues—they can constitute potential TILA (Truth in Lending Act) and UDAP/UDAAP (Unfair, Deceptive, or Abusive Acts or Practices) compliance violations.

To overcome this problem, we propose *Dynamic Semantic Tags* (DSTs).

3. Dynamic Semantic Tags

Before fine-tuning on production data, we add tags on compliance-critical attributes during the sample generation. Each compliance-critical value is wrapped in a semantic XML tag (`<card1, card_name>`, `<card1, annual_fee>`, `<card1, purchase_apr>`, `<credit_score>`) both in a header block and inline within the JSON text.

Our hypothesis is that a tagged attribute like the `<annual_fee>` token is a high-distinctiveness anchor in the pre-training distribution, and therefore the model learns to assign concentrated attention weight (Vaswani et al., 2017) to content between the tags when generating fee-related spans. During SFT, the tags reinforce the grounding pattern: *tag* → *attend* → *copy value*.

The tags appear in the context the model trains on, so by generation time the model has seen thousands of examples where the tagged span is the value it must copy faithfully. The card name is exact, the fee and APR are verbatim from source data, and the response personalizes to the user's specific credit profile.

DSTs become especially valuable in prompts involving multiple financial products. For example, when an LLM is asked to compare several credit cards, prompt construction can assign product-specific tags such as `<card1,`

```
<credit_score>785</credit_score>
<avg_utilization>18.5%</avg_utilization>
<transactions>
..................................................
..................................................
</transactions>
<card1>
  <card_name>Premium Travel Rewards Card</card_name>
  <annual_fee>$220</annual_fee>
  <purchase_apr>19.99%-26.99% variable</purchase_apr>
</card1>
<card2>
  <card_name>Free Tier Rewards Card</card_name>
  <annual_fee>$0</annual_fee>
  <purchase_apr>25.99%-29.99% variable</purchase_apr>
</card2>
{
overview: Credit Karma is recommending the
<card1,card_name>Premium Travel Rewards Card
</card1,card_name> with <card1,annual_fee>$220
</card1,annual_fee> because your financial profile
aligns well with its premium travel benefits. Although
<card2,card_name>Free Tier Rewards Card
</card2,card_name> has <card2,annual_fee>$0
</card2,annual_fee>, frequent travel expenses and
higher reward of <card1,reward_multiplier>5x</card1,
reward_multiplier> points for travel outweigh the
annual fee cost.
}
```

Figure 3. A Dynamic Semantic Tag example for a two-card comparison. Each compliance-critical value is wrapped both in a header block and inline, and tags are scoped per product (`card1`, `card2`).

`annual_fee>` and `<card2, annual_fee>`. This explicit structure helps the model associate each numerical attribute with the correct product, reducing cross-card confusion and improving the reliability of comparisons and disclosures. The reason we call them *Dynamic* is that the tags are named at prompt creation time, since `card1` will be the first card in the prompt, which is dynamically constructed depending on which cards are recommended to this specific user.

3.1. Adding Deterministic Verification at Inference Time

Adding DSTs in post-training makes the model wrap attribute references in semantic XML tags. This provides an additional advantage at inference time: it enables deterministic validation of generated outputs. A simple regex pass over the response can extract every tagged value and verify it directly against the underlying card data. Rather than relying solely on a judge model to detect hallucinations, the tag structure provides a hard, reproducible correctness check. If the value inside a tag does not match the ground-truth source data, the response automatically fails validation, regardless of how fluent or persuasive it appears.

4. Engineering the Training Distribution: Synthetic Data Generation

Raw user interaction logs inform us about some of the "obvious" high-bias modes—the attribute distributions that cause the most systematic failures in generation. Rather than training directly on this biased data, we use the observed failure modes to design a synthetic training distribution that flattens them.

The Hybrid Synthetic Data Generation Engine (HS-DGE) generates synthetic training data from a schema we design. The schema defines three field types: statistically sampled attributes (with controlled distributions), formula-derived attributes (computed from upstream fields), and LLM-generated attributes (conditioned on upstream values for semantic coherence). Fields are resolved in topological order so that downstream content—like a card description referencing a dining multiplier—is always consistent with the values already sampled. The result is a training corpus where compliance-critical attributes are distributed across their full value range, forcing the model to condition on the card data in context rather than fall back on a learned prior.

4.1. Why Not Stratified Sampling?

A natural question is whether stratified sampling over production data would be sufficient to solve this problem—if the problem is distributional skew, why not just re-weight or resample existing data to balance fee tiers, APR ranges, and issuer types?

The answer lies in the nature of the state space. Production data is text, and the biases a language model absorbs during fine-tuning operate at the level of token sequences, not tabular attributes. Stratifying on annual fee, for example, gives you balanced fee tiers, but the token sequences surrounding those fee values—the phrasing, the context, the co-occurring card attributes, the marketing bullets, etc.—still reflect the production distribution they came from.

More fundamentally, production text encodes modes of bias that are not enumerable in advance. In the high-dimensional embedding space where LLMs operate, it is extremely difficult to identify all the ways a dataset introduces systematic skew, and there is no catalog of what a model will internalize from a given corpus. Stratified sampling can potentially address only the biases we have already measured. Synthetic generation from a controlled schema inverts this: rather than trying to de-bias an existing corpus post-hoc, we define the generative process from scratch, making the distribution an explicit design decision rather than an inherited property of a production dataset.

5. The Data Preparation and Training Pipeline with DST & HS-DGE

Our pipeline (Figure 4) proceeds as follows:

Synthetic Data Generation. The first step is to generate a highly diversified, balanced matrix of baseline profiles

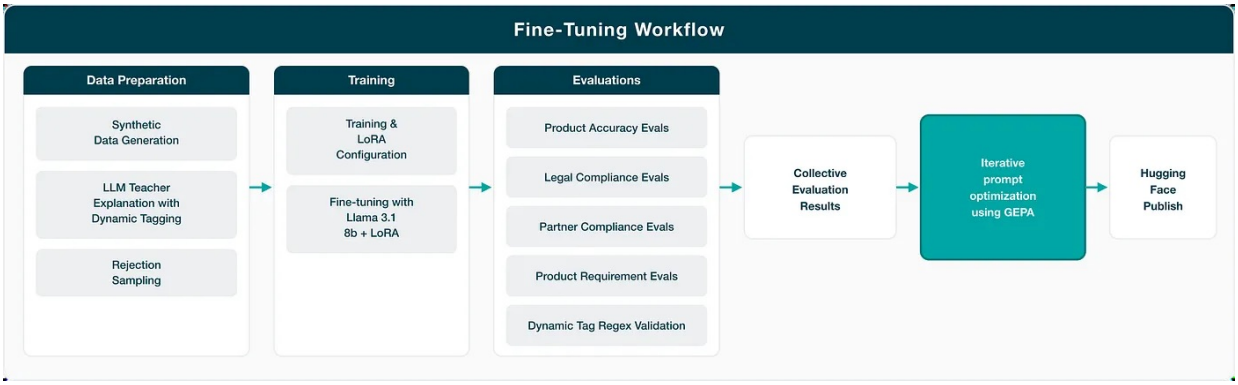

Figure 4. The SEEWHY fine-tuning workflow: data preparation (synthetic data generation, LLM-teacher explanation with dynamic tagging, rejection sampling), training (Llama 3.1 8B + LoRA), a battery of evaluations, and iterative prompt optimization with GEPA before publishing.

and scenarios using controlled parameters. This approach flattens natural distribution skews and forces the model to focus strictly on real-time context rather than relying on embedded historical priors.

LLM Teacher Explanation with Dynamic Tagging. The next step leverages a high-capacity teacher model to synthesize detailed, grounded reasoning structures tailored to each profile. Crucial compliance tokens and identity strings are explicitly wrapped in unique semantic markers to establish clear token-grounding targets during subsequent training.

Rejection Sampling. Our rejection sampling step acts as an automated quality gate by systematically scanning candidate training data against evaluation rubrics and other programmatic constraints. Records failing basic schema validation, length bounds, or fundamental value-consistency checks are permanently filtered out before entering the compute pipeline.

Fine-Tuning with Llama 3.1 8B + LoRA. We used low-rank adapter matrices (Hu et al., 2021) across key linear layers to train structural behavior, data fidelity, and explicit grounding mechanics directly into the model's weights. This process shifts the model away from free-form narrative production toward strict, deterministic token-copying circuits.

Product Accuracy Evals. Our DSTs enabled programmatic cross-reference for generated outputs against the original input source object to ensure absolute factual alignment. This provides verification that core numeric values are reproduced flawlessly without distortion.

Legal & Partner Compliance Evals. Finally, automated regulatory checks identify unauthorized guarantees, structural omissions, or boundary violations. This layer guarantees that outputs remain strictly within permissible conversational guardrails.

Product Requirement Evals. We used various evaluations for conversational tone, user personalization, and layout formatting constraints against predefined production criteria to maintain an positive and uniform user experience.

Dynamic Tag Regex Validation. This step extracts structural semantic markers from the output stream to confirm that the boundaries of critical data assertions are syntactically intact and fully populated.

Iterative Prompt Optimization using GEPA. We executed an automated evolutionary search over the system prompt configuration using GEPA (Agrawal et al., 2025). It systematically alters instructional phrasing to close performance gaps on secondary rubrics while holding primary compliance vectors completely stable.

6. Offline Simulation Results

The success of the SEEWHY fine-tuning experiment in the offline setting demonstrates that a small, fine-tuned model can achieve frontier-level accuracy and compliance through targeted architectural and data engineering. The dynamic semantic XML tagging serves as a structural anchor, forcing the Llama 3.1 8B model to ground its outputs precisely and enabling a "defense in depth" validation layer. By treating the data distribution as an explicit design decision, and by using structural tags to ensure verbatim grounding, we are able to achieve a compliance-critical model that delivers high-trust explanations at an unprecedented scale and cost efficiency.

On compliance rubrics (Table 1), the fine-tuned Llama 3.1 8B model reaches 100% card-name accuracy and matches or exceeds the frontier `gpt-4.1-mini` baseline across fees, rates, rewards, and performance—

Table 1. Performance on compliance rubrics. The fine-tuned Llama 3.1 8B model (DST + GEPA) matches or exceeds the `gpt-4.1-mini` baseline and substantially improves over the Llama 3.1 8B base model, reaching 100% card-name accuracy.

Compliance Rubric	gpt-4.1-mini (Baseline)	Llama 3.1 8B Base	Llama 3.1 8B FT + GEPA
Card Name	50.7%	76.2%	**100%**
Fees	98.8%	95.1%	**99.2%**
Rates	99.3%	99.3%	**99.9%**
Rewards	**98.9%**	97.6%	98.3%
Performance	98.4%	85.3%	**99.8%**

Table 2. Model operations. The fine-tuned Llama 3.1 8B model improves average and p95 latency and token throughput relative to the gpt-4.1-mini baseline. (↓ lower is better, ↑ higher is better.)

Performance Metric	gpt-4.1-mini	Llama 3.1 8B fine-tuned	Improvement
Avg. Latency ↓	3.41s	3.06s	**−10.2%**
P95 Latency ↓	3.80s	3.21s	**−15.5%**
Throughput ↑	310 tok/s	440 tok/s	**+42%**

while the base Llama model alone (*e.g.*, 76.2% card-name accuracy, 85.3% performance) falls well short. On model operations (Table 2), the fine-tuned model reduces average latency by 10.2% and p95 latency by 15.5%, and improves token throughput by 42% relative to the API baseline—in addition to the 15–20× cost reduction from self-hosting a small model.

7. Conclusions

We introduced the idea of Dynamic Semantic Tags (DSTs), which add structure into post-training data. Most post-training pipelines treat the prompt and response as largely unstructured text, forcing the model to implicitly infer which numerical attributes belong to which entity. This becomes especially brittle in domains like finance, where multiple products may share overlapping attributes such as APRs, annual fees, rewards rates, or credit score ranges.

Dynamic tags inject lightweight relational structure directly into the training data. By assigning identifiers such as `<card1, apr>` or `<card2, annual_fee>`, the dataset explicitly binds each value to a specific entity and attribute. This reduces ambiguity during training and teaches the model a cleaner latent representation of entity-attribute relationships.

We hypothesize that this representation improves grounding by reducing ambiguity over which attributes belong to which entities, especially in prompts involving multiple products with overlapping numerical features. This reduces value-swapping errors and hallucinations while improving the model's ability to perform comparisons across entities. Because the generated outputs preserve semantic structure, DSTs also enable deterministic verification at inference time. Importantly, these tags can be generated automatically during prompt construction, allowing scalable creation of structured supervision without manual annotation.

Dynamic Semantic Tags sit between natural-language supervision and fully structured tool use. They preserve the flexibility of language modeling while introducing some symbolic structure to improve fidelity during post-training. The same principle may extend to domains such as healthcare, legal reasoning, enterprise databases, and retail catalogs, where models frequently struggle with entity-attribute binding.

References

Agrawal, L. A., Tan, S., Soylu, D., Ziems, N., Khare, R., Opsahl-Ong, K., Singhvi, A., Shandilya, H., Ryan, M. J., Jiang, M., et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Grattafiori, A., Dubey, A., et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.