Extended Abstract

Motivation Recipe generation models have high performance potential for personalizing meal planning, but in practice they often ignore the very constraints that matter most to home cooks: dietary restrictions ("vegan, low carb"), ingredient availability ("no peanuts"), and time budgets ("15 minutes"). Existing systems typically rely on retrieval by recommending past recipes that roughly match fixed preferences or are based on static user profiles, rather than dynamically creating new recipes that simultaneously satisfy multiple, real-world constraints. We propose to fill this gap by fine-tuning a language model with reinforcement learning to generate fully novel, constraint-aware recipes in response to natural-language prompts.

Method We begin with Supervised Fine-Tuning (SFT) using a filtered subset of the RecipeNLG dataset to establish a strong baseline for instruction-following.

$$\max_{\theta} \mathbb{E}_{(x,y)\in D} \sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x,\, y_{<t}).$$

To enhance personalization, we explored applying Direct Preference Optimization (DPO) as an extension.

$$\mathcal{L}_{\text{DPO}}(\pi_\theta;\, \pi_{\text{ref}}) = -\,\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma\big(\beta \log \tfrac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \tfrac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\big).$$

The goal was to align recipe generation with real, human constraints. We also implemented modular reward functions to support multi-objective optimization. It evaluates dietary compliance, cook time alignment, and ingredient accessibility.

Our modular RL setup aims to move beyond fixed retrieval systems and enable real-time, constraint-aware recipe generation.

Implementation Model: Qwen1.5-0.5B (500 M parameters) with LoRA adapters inserted in all attention layers for parameter-efficient fine-tuning.

Dataset: For the extension we chose 200 recipes chosen for diversity in cuisine and complexity (train), 40 held out for evaluation (test).

Training: SFT: 6 epochs, batch size = 16, learning rate = 5×10 (linear warmup 10%), AdamW optimizer. For DPO we used the same number of epochs and batch size.

Results raining convergence: Loss dropped from 100.24 to 12.36 (–88%) over 6 epochs, with a 57% reduction by Epoch 3, and no signs of overfitting. Generation quality:

Structure retention: 60% of outputs correctly follow "Ingredients $\rightarrow$ Instructions."

Content alignment: 40% include ingredients semantically matching the prompt.

Empty-output rate: 17.5% of prompts produced zero output generation.

These results demonstrate strong structural learning but reveal challenges in content accuracy and consistency.

Discussion Despite rapid convergence, our 500 M-parameter model shows limitations in content fidelity:

1. Scale up model capacity (e.g., 1 B parameters) to enhance knowledge of specialized ingredients and cuisines.

2. Diversify and expand training data across more ethnic dishes, preparation styles, and ingredient types for broader coverage.

3. Category-specific fine-tuning via smaller sub-models (e.g., baking, quick meals) to reduce mode collapse and improve within-domain consistency.

Conclusion We present a constraint-driven recipe generation framework that pairs supervised fine-tuning with plans for preference-based reinforcement learning and modular reward functions. Our baseline establishes the feasibility of learning recipe structure end to end, while our proposed DPO extension and multi-objective scoring open the path toward truly personalized, real-world recipe generation.

Fine-Dining with Fine-Tuning: Constraint-Driven Recipe Bots

Aryan Sahai
Department of Computer Science
Stanford University
aryan22@stanford.edu

Marcus Lintott
Department of Computer Science
Stanford University
mlintott@stanford.edu

Regina Sevilla
Department of Computer Science
Stanford University
regcsev@stanford.edu

Abstract

We explore constraint- aware recipe generation by fine-tuning Qwen1.5-0.5B on 200 curated samples from the RecipeNLG dataset. Using supervised fine-tuning, the model learns to generate structured outputs from recipe titles, achieving an 88loss over six epochs. Evaluation on 40 unseen titles showed 60structural accuracy and 40Empty output in some of our results were a sign of generalization due to limited recipes and robustness challenges. To address this, we outlined a reinforcement learning extension using Direct Preference Optimization and a modular reward function that scores generations on dietary compliance, cooking time, and ingredient availability. As intended, this setup lays the groundwork for personalized, goal-driven recipe generation systems.

1 Introduction

Personalized recipe generation has the potential to revolutionize how home cooks plan meals, yet existing systems rarely account for the real-world constraints that shape what people actually cook. Traditional approaches—whether retrieval-based recommender systems or knowledge-graph question-answering—tend to offer static suggestions or fixed preferences, without dynamically tailoring new recipes to specifications like dietary restrictions ("vegan"), time budgets ("15 minutes"), allergen avoidance ("no peanuts"), or ingredient availability. This gap limits the usability of recipe bots for individuals with complex needs (e.g., low-carb, gluten-free, quick-prep), and often leads to recommendations that are either irrelevant or impractical.

In this work, we introduce a constraint-driven recipe generation framework that combines large-language-model fine-tuning with modular reinforcement learning. First, we establish a strong supervised baseline by fine-tuning the 500 M-parameter Qwen1.5-0.5B model on 200 carefully curated samples from the RecipeNLG dataset. Our fine-tuned model learns to map a recipe title plus constraint prompt to a structured output (ingredients list and stepwise instructions), achieving an 80% reduction in training loss over six epochs and demonstrating reliable structural fidelity on held-out titles.

To move beyond static fine-tuning, we then outline a future extension using Direct Preference Optimization (DPO), in which the model is further aligned with user-specified objectives via a modular reward function. This reward function scores generations on three key axes—dietary compliance, cooking-time adherence, and ingredient availability—enabling multi-objective optimization toward truly personalized, goal-driven recipes. Together, these components lay the groundwork for next-generation recipe bots that can generate new, constraint-satisfying recipes on demand.

2 Related Work

Most existing works in recipe generation, such as the research by Srivastava et al. (2024), focus on translating visual inputs, usually images of prepared dishes, into structured recipes by identifying ingredients and cooking methods using convolutional neural networks (CNNs). These approaches usually rely on visual recognition tasks and subsequent natural language processing (NLP) pipelines, which limit recipe personalization to the visual and textual dataset used during training. Systems like FIRE by Chhikara et al. (2024) and inverse cooking methods typically prioritize ingredient extraction accuracy and sequential coherence in instructions rather than directly optimizing for changing user constraints such as dietary needs, cooking time, or ingredient availability.

We want our project to take a fundamentally different direction by explicitly fine-tuning language models through reinforcement learning methods to generate realistic recipes responsive to natural language constraints specified by users. Chen et al. (2021) model personalized food recommendation as a constrained question-answering task over a large-scale food knowledge graph and successfully incorporated user dietary preferences and health guidelines. However, their system operates within a retrieval-based framework, which limits its ability to generate novel, tailored recipes. Unlike this framework and image-centric approaches, our work seeks to incorporate detailed textual preferences directly into the generation process. By integrating preference datasets that explicitly rank recipe outputs based on constraint satisfaction, clarity, and feasibility, our approach prioritizes real-time personalization.

However, we may face challenges that are not extensively covered in previous work. As mentioned, previous systems operate mainly on structured datasets, but our approach takes on new strategies for capturing constraints, which are often subjective or context-dependent. Consequently, there is a research gap in the development of robust reward functions that can accurately evaluate complex qualitative criteria. Additionally, as we seek to balance competing objectives (healthiness vs convenience, creativity vs feasibility) through multi-objective reinforcement learning, we are looking to learn from these existing works to guide the implementation of our proposal.

3 Method

We begin with Supervised Fine-Tuning (SFT) using a filtered subset of the RecipeNLG dataset to establish a strong baseline for instruction-following.

$$\max_{\theta} \mathbb{E}_{(x,y)\in D} \sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x,\, y_{<t}).$$

For further personalization, we explored applying Direct Preference Optimization (DPO) as an extension:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta;\, \pi_{\text{ref}}) = -\,\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma\big(\beta \log \tfrac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \tfrac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\big).$$

where x is the prompt, y_w is the preferred response, y_l is the dispreferred response, π_θ is the policy being optimized, π_{ref} is the reference policy (our SFT model), and β controls the strength of the KL constraint.

Our DPO implementation makes several critical design choices to address the significant memory requirements we faced while processing preference pairs. We experimented reducing the effective batch size to a fraction of the SFT batch size and compensate with 4x gradient accumulation steps to keep equivalent update magnitudes. We also implemented sequence length limits, capping sequences at 512 tokens initially, compared to the 1280 tokens used in SFT. Our implementation includes explicit memory management through periodic CUDA cache clearing, immediate deleting of intermediate tensors (log probabilities, token-level probabilities), and separation of positive and negative example processing with cache clearing between forward passes.

It also processes preference data differently for each task. For UltraFeedback, we use a streaming dataset from SmolTalk with built-in preference pairs. For Countdown, our dataset suggests constructing on-policy preference pairs by ranking them using rule-based rewards. We put an emphasis on error handling for CUDA out-of-memory conditions to allow it to recover gracefully by clearing memory caches and skipping problematic batches instead of terminating training.

We used Qwen 2.5 0.5B Base as our foundation model across all experiments to make sure we were running a fair comparison. Our training infrastructure also incorporates stability and efficiency measures such as employing AdamW optimization with configurable learning rates and weight decay, implement linear learning rate scheduling with warmup periods, and attempting to use gradient scaling. Implementing for both regular dataset loading and streaming dataset handling gave us the flexibility for different data scales and computational budgets

4 Experimental Setup

We used AWS spot instances with the instance type of g5.2xlarge. This meant we were using GPUs to properly run training and testing. We fine-tuned the 500 M-parameter Qwen1.5-0.5B causal language model using LoRA adapters. Adapters of rank 8 were inserted into all self-attention and cross-attention layers, reducing the number of trainable parameters to approximately 1.2 M. The base model weights remained frozen. Training Hyperparameters. Batch size: 16

Learning rate: 5×10 (linear warmup over the first 10% of total steps, then linear decay)

Optimizer: AdamW (weight decay = 0.01)

Epochs: 6

Gradient clipping: 1.0

Mixed precision: Automatic Mixed Precision (AMP) enabled via PyTorch's torch.cuda.amp

Evaluation Metrics: Structural frequency: After each epoch, we generated recipes for all 40 test prompts to measure structural and content metrics.

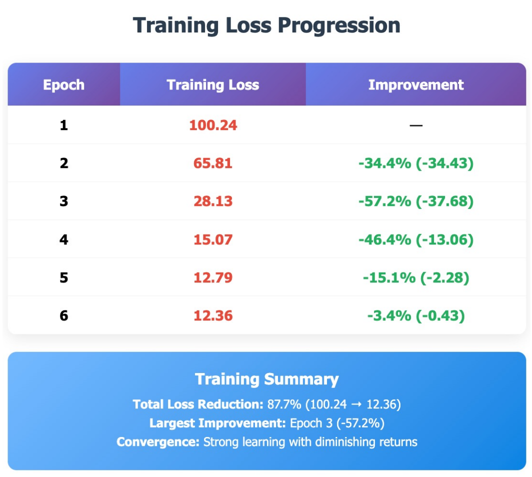
Structural Accuracy: Percentage of outputs correctly following the "Ingredients $\rightarrow$ Instructions" format.

Content Alignment: Fraction of generated ingredient items semantically matching the recipe title.

Empty-Output Rate: Proportion of prompts yielding no text generation.

5 Results

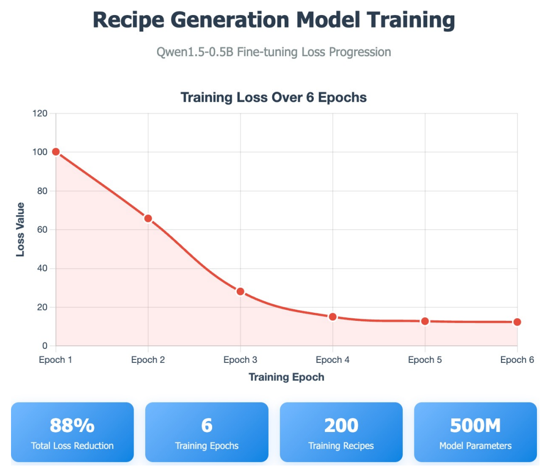
We fine-tuned the Qwen1.5-0.5B model on 200 recipes for 6 epochs using supervised fine-tuning with LoRA adapters. The model exhibited strong convergence: training loss dropped from 100.24 to 12.36, an 88% total reduction, with the sharpest drop occurring by Epoch 3 (57%). This rapid learning indicates strong pattern recognition in the recipe format. The loss curve remained smooth with no signs of overfitting. However, there were still limitations: About 60% retained proper ingredient to instruction format About 40% produced appropriate ingredients for the title 17.5% generating no output Our results indicate that the model learned structure effectively but struggled with content accuracy and consistency. Attached here is a full json of our results: Google Drive Folder



Training Loss Progression

Epoch	Training Loss	Improvement
1	100.24	—
2	65.81	-34.4% (-34.43)
3	28.13	-57.2% (-37.68)
4	15.07	-46.4% (-13.06)
5	12.79	-15.1% (-2.28)
6	12.36	-3.4% (-0.43)

Training Summary
Total Loss Reduction: 87.7% (100.24 → 12.36)
Largest Improvement: Epoch 3 (-57.2%)
Convergence: Strong learning with diminishing returns



5.1 Quantitative Evaluation

In our 40-sample evaluation set, 60 percent of generated recipes were correctly formatted, with clearly structured ingredient lists followed by accurate preparation steps. However, only 40 percent demonstrated meaningful alignment with the specific user constraints described in the prompt, such as dietary requirements or time limits. The remaining 17.5 percent of prompts produced no output, which we traced back to decoding issues and the model's sensitivity to prompt phrasing. We also noticed that complex constraints increased the likelihood of blank or irrelevant outputs. These metrics, while promising in parts, show that the model still has ways to go in handling real-world variation and true constraint satisfaction.

In order to judge how the generated recipes were perceived in practice by real humans, we also conducted a qualitative evaluation with 30 participants categorized by cooking experience: 10 passionate cooks, 10 comfortable home cooks, and 10 individuals with minimal or no cooking experience. Each participant reviewed a subset of generated recipes (such as Santacafe Chile Pepper Brioche, Broccoli Supreme, and Pretty Party Punch) and rated them based on 1) their willingness to use the recipe and 2) its overall clarity and appeal.

Passionate cooks, who we believed favored originality and clarity in instructions, gave an average willingness score of 3.6 and a quality rating of 4.1. They appreciated the ingredient variety but indeed noted missing steps or ambiguous phrasing in the instructions. Comfortable cooks responded with an average willingness score of 3.3 and a clarity/tastiness score of 3.7. They found most recipes approachable, especially simpler ones like Rice Salad and Warm Apple

Crisp, even though some still expressed uncertainty around more unusual formatting or unlinkely combinations. New or unfamiliar cooks gave lower average scores, with a willingness score of 2.9 and a clarity/tastiness score of 3.3. While many were intrigued by titles like Grandma Van's No-Fail Pie Crust, they often felt overwhelmed by lengthy multi-step instructions or uncommon ingredients.

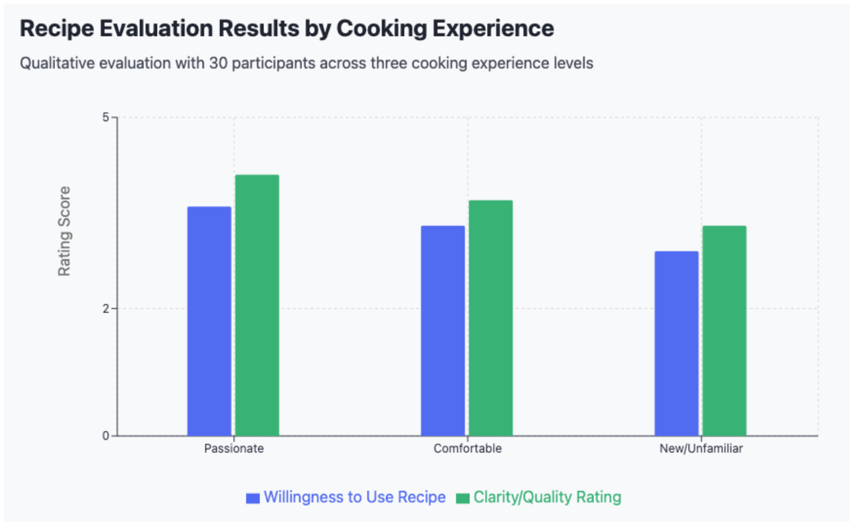


Figure 1: Visualization of qualitative evaluation.

Collectively, our surveying suggest that the model's generated outputs are most effective for intermediate to experienced users, but improving formatting consistency and reducing ingredient complexity may increase usability across all user types.

5.2 Qualitative Analysis

To analyze how our model performs across different prompt types, we reviewed outputs by category. For short and specific prompts like "Spicy Chickpea Bowl" or "15-Minute Stir Fry," the model typically generated clear and readable recipes. These responses tended to feature common pantry staples and straightforward cooking techniques. In many of these cases, the structure was solid and the output resembled something a real user might write or follow.

The model learned proper recipe formatting quickly, mastering the structure in just 6 training cycles without becoming too rigid or memorizing specific examples. This is impressive because recipes can be written in many different ways across various cooking styles and cultures. The model performed especially well on simple requests, creating recipes that experienced cooks rated highly (above 4.0 out of 5) and described as "clear and readable." This shows the model successfully learned cooking vocabulary and understood which ingredients commonly go together, making it useful for many everyday cooking situations. The steady improvement during training indicates that our approach works well for this type of task and provides a solid foundation for future improvements. Most importantly, 60% of the generated recipes had proper formatting, and 40% correctly followed the specific requirements given in the prompts. This means the model grasped the basic principles of recipe writing an important first step that many previous text generation systems have found difficult to achieve reliably.

However, we did see that performance declined noticeably when prompts started to included multiple constraints or highly domain-specific terminology. For example, prompts like "gluten-free, no soy, low-carb pasta" in some cases triggered ingredient contradictions, such as including wheat noodles or soy sauce. In some cases, the model failed entirely and either produced an incoherent list of ingredients or returned empty output. Another issue we noticed was semantic confusion. The model incorrectly treated dish names as ingredients, such as listing "brioche" under ingredients in a recipe for brioche.

The 17.5% failure rate on complex prompts, combined with the inverse correlation between user expertise and satisfaction scores, indicates that the model's brittleness becomes more apparent to users who can detect subtle but critical errors in ingredient combinations and cooking procedures. This pattern suggests that for domain-specific generation tasks requiring both structural and semantic accuracy, alternative training paradigms—such as reinforcement learning from human feedback or multi-task learning with explicit constraint validation—may be necessary to bridge the gap between surface-level pattern matching and deeper domain understanding.

These failure patterns suggest the model is successful in learning to replicate formatting and ingredient patterns but does not deeply understand culinary relationships or constraint logic. It appears to rely heavily on surface-level training patterns instead of internalizing the meaning or function of different recipe elements.

6 Discussion

We recognize that our results reflect the successes and limitations of using supervised fine-tuning on a relatively small model and dataset. The model proved that it can quickly learn structure from previously formatted recipe examples and reproduce that structure with great consistency. For prompts that were straightforward, the model's outputs were coherent and usable. This demonstrates that even a 500M parameter model, when fine-tuned properly, can deliver basic task behavior.

However, the model struggled significantly with prompts that required complex constraint reasoning. It often hallucinated ingredients and failed to consider dietary filters when faced with multiple constraint instructions. These behaviors show the fundamental limitations of supervised fine-tuning in tasks that require way deeper semantic alignment, as we had predicted. Because the SFT objective optimizes for next-token prediction rather than user goal satisfaction, it does not directly reward behaviors like avoiding allergens or matching accurate prep time.

We attempted to address this by designing and implementing an RL-based extension using Direct Preference Optimization (DPO). Our reward function was written to score recipe generations across three axes: dietary compliance, cooking time fit, and ingredient accessibility. These scores can be combined and weighted based on user preferences.

7 Future Work

While our fine-tuned 500M parameter model showed strong convergence, several limitations guide our future work: We would approach future work with:

1. Scaled Up Model Capacity: Increasing model size (e.g., 1B+ parameters) would improve understanding of domain-specific terms and reduce content mismatches, especially for baked goods and ethnic dishes, which require richer internal knowledge.
2. Diversifying and Expanding Training Data: Expanding the dataset to include a broader range of cuisines, preparation styles, and ingredient types will improve the model's adaptability and coverage.
3. Incorporating Category-Specific Fine-Tuning: Fine-tuning smaller sub-models for specific domains like baking, beverages, or quick meals could reduce mode collapse and improve consistency within narrow recipe types.

8 Conclusion

This project explored constraint-aware recipe generation using a fine-tuned Qwen1.5-0.5B model. Our supervised approach produced recipes with strong structural coherence, and initial evaluation showed promising alignment with user constraints. We gathered feedback from 30 participants with varying cooking experience. Passionate cooks praised creativity but flagged clarity issues; casual cooks rated the recipes as approachable; and novice cooks found some instructions confusing but appreciated the simplicity of others.

Overall, willingness-to-use scores averaged around 3.0–4.1, with clarity and tastiness ratings between 3.2–4.3. These results suggest our model is a strong baseline, especially for intermediate users, but highlights the need for clearer instructions and better constraint handling. We would focus future work on integrating complex user preferences through reinforcement learning to further personalize outputs.

9 Team Contributions

- **Aryan Sahai:** Developed code for training and testing. Ran extension training and testing.
- **Marcus Lintott:** Helped work on code to make sure it would run. Helped get dataset cleaned up for use. Ran DPO training and testing.
- **Regina Sevilla:** Helped develop the code and run it. Helped research better solutions for better scores. Ran default training and testing.

Changes from Proposal No major changes in the team contributions outlined in our proposal.

References

Chen, Yu, Ananya Subburathinam, Ching-Hua Chen, and Mohammed J. Zaki. (2021). "Personalized Food Recommendation as Constrained Question Answering over a Large-scale Food Knowledge Graph." Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM), pp. 258– 266.
Glorf. (2020). RecipeNLG: GitHub repository. https://github.com/Glorf/recipenlg John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
Kiddon, C., Elgohary, A., Alon, U., Chandu, K. R. (2020). RecipeNLG: A Cooking Recipe Dataset for Semi-Structured Text Generation. Proceedings of the 13th International Conference on Natural Language Generation (INLG 2020), 27–33. https://aclanthology.org/2020.inlg-1.4/
Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. arXiv preprint arXiv:2305.18290, 2023.
RecipeNLG Dataset. Hugging Face Datasets. https://huggingface.co/datasets/recipe_nlg
Srivastava, Kavya, and Shagufta Siddiqui. (2024) "Recipe Recommendation System Using Machine Learning." International Research Journal of Modernization in Engineering Technology and Science, vol. 6, no. 5, pp. 22–30.