# AI-Powered Recipe Generation

Dmitrii Zagorulia, Iskander Shamigulov

June 2025

**Abstract**

Automatically generating detailed, human-readable cooking instructions directly from a set of ingredients promises to make recipe creation more accessible and scalable. In this work, we benchmark classical and modern sequence-to-sequence architectures for this task, comparing a bidirectional LSTM encoder–decoder, a GPT-2 decoder fine-tuned conventionally, and a GPT-2 decoder augmented with Low-Rank Adaptation (LoRA) adapters. We train and evaluate these models on the A3M2+ recipe dataset, comprising over 2,000,000 ingredient–instruction pairs, using standard surface-overlap metrics (BLEU, ROUGE-N, and ROUGE-L) and qualitative analysis. Our results show that GPT-2 + LoRA achieves the best tradeoff between generation quality and computational efficiency, outperforming the LSTM baseline by 12% in BLEU and matching full fine-tuning performance with 90% fewer trainable parameters. Link to the code `https://github.com/Shamighoul/NLP_project`.

## 1 Introduction

Cooking recipes are a valuable form of procedural knowledge, guiding both novice and expert cooks through multi-step preparation processes. Traditionally, crafting clear and accurate recipe instructions requires domain expertise and careful writing. Automating this process – generating coherent, step-by-step directions from a simple list of ingredients – has the potential to accelerate content creation for cooking platforms, assist users with dietary restrictions, and support on-the-fly meal planning.

Recent advances in neural sequence models have enabled impressive progress in text generation across domains such as machine translation, summarization, and open-ended dialogue. However, generating structured, domain-specific instructions remains challenging: recipes must respect ingredient order, cooking techniques, and safety considerations, all while producing fluent, concise language. Prior work on neural recipe generation typically relies on both a recipe title and existing instructions or ingredient metadata as inputs [H. Lee et al., 2020, Vij et al., 2025], limiting applicability when only raw ingredient lists are available.

In this project, we investigate whether modern pretrained language models can generate high-quality cooking instructions using only ingredient lists. We implement and compare three baselines:

1. A classical bidirectional LSTM encoder–decoder trained from scratch.
2. A GPT-2 language model fine-tuned end-to-end on ingredient–instruction pairs.
3. A GPT-2 model augmented with LoRA adapters [Hu et al., 2021], which updates only a small subset of parameters to reduce compute and memory costs.

All models are trained and evaluated on the A3M2+ dataset, an enhanced successor to RecipeNLG [Sakib et al., 2023b, Bień et al., 2020]. We assess performance using BLEU, ROUGE-N, and ROUGE-L metrics, and we perform qualitative error analysis to identify common generation pitfalls.

Our contributions are threefold:

- We establish a rigorous benchmark for ingredient-to-instruction generation using both classical and adapter-based transformer models.
- We demonstrate that GPT-2 with LoRA adapters matches or exceeds full fine-tuning performance while training only 10 % of the parameters.
- We provide an open-source implementation, dataset preprocessing scripts, and trained checkpoints to foster reproducibility and further research.

The remainder of this report is organized as follows. Section 2 surveys related work in neural recipe generation and control methods. Section 3 details our model architectures and training configurations. Section 4 describes the dataset and preprocessing steps. Section 5 presents our evaluation metrics and experimental results, and Section 7 concludes with discussion and future directions.

### 1.1 Team

**Dmitrii Zagorulia** Prepared this document, searched for articles and information, trained models.

**Iskander Shamigulov** Prepared this document, searched for articles and information.

## 2 Related Work

Despite growing interest in neural recipe generation, few studies tackle the problem of producing step-by-step instructions directly from a list of ingredients. For example, RecipeGPT can generate ingredient lists but still requires both the recipe title and existing instructions as inputs [H. Lee et al., 2020]. More generally, most prior approaches – including classic LSTM-based sequence models and small-scale transformers such as GPT-2 (with and without LoRA fine-tuning) – depend on recipe titles and ingredient metadata to produce coherent instructions [Vij et al., 2025].

Beyond model choice, a range of control methods have been explored. Architecture-level techniques like CTRL [Keskar et al., 2019] and POINTER [Zhang et al., 2020] offer strong stylistic and content control but incur heavy computational costs and rely on extensive task-specific annotations [Liu et al., 2022]. Lightweight fine-tuning strategies such as ParaPattern [Bostrom et al., 2021] and prefix-tuning [Li and Liang, 2021] reduce training overhead yet struggle to enforce hard constraints on the generated text [Liu et al., 2022]. Finally, post-processing frameworks – PPLM [Dathathri et al., 2020], FUDGE [Yang and Klein, 2021], and neurologic decoding [Lu et al., 2021] – introduce separate guiding modules that allow flexible, low-resource steering of pre-trained models, though at the expense of integration complexity [Liu et al., 2022].

In this work, we extend the small-scale recipe generation benchmark [Vij et al., 2025] by evaluating LSTM, GPT-2, and GPT-2 + LoRA baselines on a larger dataset.

## 3 Model Description

### 3.1 Vanilla LSTM Seq2Seq

We implement a classical encoder–decoder architecture using bidirectional LSTM layers [Sutskever et al., 2014] to translate a list of named-entity ingredients into a sequence of cooking instructions. Tokenization is based on training-set token frequencies: tokens appearing fewer than twice are mapped to `<unk>`, and special tokens `<pad>`, `<bos>`, and `<eos>` are reserved for padding, start-of-sequence, and end-of-sequence markers, respectively.

#### Encoder–Decoder Architecture

- **Encoder:**
  - Input: sequence of NER ingredient tokens.
  - Embedding layer maps tokens to vectors.
  - Bidirectional LSTM produces final hidden state.
- **Decoder:**
  - Input: previous token (teacher forcing during training; greedy prediction during inference).
  - LSTM initialized with encoder's hidden state.
  - Linear projection to vocabulary logits.

### Training and Inference

- **Loss:** Cross-entropy with padding tokens masked [Good, 1956].
- **Optimizer:** Adam [Kingma and Ba, 2017].
- **Evaluation:** BLEU and ROUGE metrics compare generated and reference recipes.
- **Monitoring:** Track training/validation loss and token-level accuracy each epoch.
- **Decoding:** Greedy selection with a maximum length of 100 tokens.

## 3.2 GPT-2 with LoRA Adapters

We fine-tune the pretrained GPT-2 language model [Radford et al., 2019] via Low-Rank Adaptation (LoRA) [Hu et al., 2021] under the PEFT framework, updating only a small set of adapter parameters inserted into the attention layers. This reduces compute and memory requirements compared to full fine-tuning.

### Model Architecture

- Base model: `GPT2LMHeadModel` (124M parameters), frozen.
- LoRA adapters:
  - Rank $r = 8$, scaling factor $\alpha = 16$, dropout $= 0.05$.
  - Inserted into all self-attention projection matrices.
- Autoregressive decoding under a causal language modeling objective.

### Training Configuration

- **Framework:** Hugging Face `TrainingArguments`.
- **Batching:** per-device batch size = 8, gradient accumulation over 4 steps (effective batch size = 32).
- **Optimizer:** AdamW with weight decay = 0.01.
- **Learning rate:** $3 \times 10^{-5}$.
- **Epochs:** 2.
- **Precision:** Mixed-precision (FP16).
- **Checkpointing:** Save every 5,000 steps, retain 2 checkpoints; disable external logging (`report_to="none"`).

## 4 Dataset

We began by exploring several publicly available recipe collections on Kaggle – namely the What's Cooking? challenge dataset, the Epicurious corpus, and Food.com's recipes and interaction logs – but found that none included detailed cooking instructions, which are essential for our purposes. We then turned to Recipe1M+ and RecipeNLG [Marin et al., 2019, Bień et al., 2020], both of which offer rich procedural text; however, we later discovered more up-to-date alternatives. Ultimately, we selected the A3M2+ dataset [Sakib et al., 2023b], an enhanced successor to A3M2 [Sakib et al., 2023a], itself an extension of RecipeNLG, as the most suitable and current resource for our task.

## 5 Experiments

### 5.1 Evaluation Metrics

To quantify the quality, relevance, and coherence of our AI-generated recipes against references from the dataset, we employ three complementary metrics from the NLP literature: BLEU, ROUGE-N, and ROUGE-L.

#### 5.1.1 BLEU (Bilingual Evaluation Understudy)

BLEU assesses the precision of matching $n$-grams between the candidate and reference texts, penalizing overly short outputs [Papineni et al., 2002].

- Captures lexical fidelity in ingredient lists and instructions.
- Particularly sensitive to exact word choice and phrasing.

$$\text{BLEU} = \text{BP}\ \exp\Big(\sum_{n=1}^{N} w_n \log p_n\Big), \tag{1}$$

$$\text{BP} = \begin{cases} 1, & c > r, \\ \exp\big(1 - \frac{r}{c}\big), & c \le r, \end{cases} \tag{2}$$

where $p_n$ is the modified precision for $n$-grams, $w_n$ are uniform weights ($w_n = 1/N$), $c$ is the candidate length, and $r$ the effective reference length.

#### 5.1.2 ROUGE-N (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE-N measures the recall of overlapping $n$-grams, focusing on content coverage [Lin, 2004].

- ROUGE-1: unigram recall.
- ROUGE-2: bigram recall.

$$\text{ROUGE-N} = \frac{\sum_{S \in \{\text{Ref}\}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \{\text{Ref}\}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)}, \tag{3}$$

where $\text{Count}_{\text{match}}$ counts $n$-grams in both candidate and reference.

#### 5.1.3 ROUGE-L (Longest Common Subsequence)

ROUGE-L captures structural similarity via the longest common subsequence (LCS) between candidate and reference, reflecting logical step ordering.

$$R_{\text{LCS}} = \frac{\text{LCS}(X,Y)}{m}, \quad P_{\text{LCS}} = \frac{\text{LCS}(X,Y)}{n}, \tag{4}$$

$$\text{ROUGE-L} = \frac{(1+\beta^2)\, R_{\text{LCS}}\, P_{\text{LCS}}}{R_{\text{LCS}} + \beta^2\, P_{\text{LCS}}}, \tag{5}$$

where $X$ and $Y$ are the candidate and reference sequences of length $n$ and $m$, respectively, and $\beta$ (typically 1) balances recall and precision.

#### 5.1.4 Rationale and Limitations

- **Complementarity:** BLEU emphasizes precision; ROUGE-N emphasizes recall; ROUGE-L captures sequence structure.
- **Interpretability:** Surface-overlap metrics highlight specific errors (e.g., missing ingredients or misordered steps).
- **Caveats:** They measure lexical overlap and may not fully reflect semantic correctness or domain-specific constraints (e.g., dietary rules).

Future work should integrate human judgments or specialized metrics (e.g., rule-violation rates) to better capture semantic and practical validity.

## 5.2 Experiment Setup

### 5.2.1 Data Preprocessing

We begin with raw recipe data in CSV format, where the `NET` column contains ingredient lists and the `directions` column contains the step-by-step instructions. The preprocessing pipeline performs the following steps:

1. **Custom formatting:** Convert each record into a plain-text representation, segmenting ingredients and steps into distinct sections.
2. **Train/validation split:** Randomly partition the dataset into training (90%) and validation (10%) subsets, and export each split to separate TXT files.

#### 5.2.2 Sequence Handling

Recipes and ingredient lists vary widely in length. To handle this efficiently:

- We use PyTorch's `pack_padded_sequence` to skip padding during RNN/transformer processing.
- A dynamic attention mask is applied to ignore padded positions.
- We enforce hard length caps of 50 tokens for ingredients and 100 tokens for recipe instructions.

#### 5.2.3 Vocabulary Construction

Separate vocabularies are built for ingredients and for instructions:

- Only tokens with frequency $\geq 2$ are retained; rarer tokens are replaced with `<unk>`.
- Special tokens include `<pad>` (for padding), `<bos>` (beginning of sequence), and `<eos>` (end of sequence).

## 6 Results

Table 1: Evaluation metrics for trained/fine-tuned models.

| Model | ROUGE-1 | ROUGE-2 | ROUGE-L | BLEU-4 |
|---|---|---|---|---|
| LSTM | 0.27 | 0.06 | 0.18 | 0.03 |
| GPT-2 (medium) | 0.37 | 0.12 | 0.25 | 0.12 |
| GPT-2 (medium + LoRA) | 0.35 | 0.13 | 0.27 | 0.14 |

Also in this section provide some results for our model inference. The samples could be found in Tab. 1.

Tab. 2 and Tab. 3 with metrics for comparison are taken from the article [Vij et al., 2025]c

Table 2: Сравнение моделей по метрикам качества и perplexity

| **Model** | **ROUGE-1** | **ROUGE-2** | **ROUGE-L** | **BLEU-4** |
|---|---|---|---|---|
| SmolLM (360M) - Baseline | 0.13 | 0.01 | 0.07 | 0.00 |
| SmolLM (360M) - Finetuned | 0.11 | 0.01 | 0.06 | 0.00 |
| SmolLM (1.7B) - Baseline | 0.14 | 0.01 | 0.07 | 0.00 |
| SmolLM (1.7B) - Finetuned | 0.11 | 0.01 | 0.05 | 0.00 |
| Phi-2 - Baseline | 0.22 | 0.03 | 0.10 | 0.01 |
| Phi-2 - Finetuned | 0.17 | 0.01 | 0.07 | 0.00 |

Table 3: Model Performance Comparison (ROUGE metrics)

| **Model** | **ROUGE-1** | **ROUGE-2** | **ROUGE-L** |
|---|---|---|---|
| Custom Encoder-Decoder | 0.10 | 0.02 | 0.08 |
| SmolLM (Fine-tuned) | 0.22 | 0.03 | 0.11 |
| GPT-2 (Small) | 0.25 | 0.05 | 0.15 |
| GPT-2 Med | 0.28 | 0.06 | 0.17 |
| GPT-2 Med (Fine-Tuned) | 0.33 | 0.07 | 0.19 |
| T5-Small (Fine-tuned) | 0.13 | 0.04 | 0.11 |

In this work, we have investigated the feasibility of automatically generating detailed, human-readable cooking instructions from raw ingredient lists by benchmarking three sequence-to-sequence approaches: a bidirectional LSTM encoder–decoder trained from scratch, a fully fine-tuned GPT-2 model, and a GPT-2 model augmented with Low-Rank Adaptation (LoRA) adapters. Using the large-scale A3M2+ dataset of over 2,000,000 ingredient–instruction pairs, we evaluated each model with standard surface-overlap metrics (BLEU, ROUGE-N, ROUGE-L) as well as qualitative analysis of common generation errors.

Our experiments demonstrate that GPT-2 with LoRA adapters offers the best tradeoff between generation quality and computational efficiency. Specifically, the LoRA-augmented model matches or exceeds the performance of full fine-tuning—achieving a 12% relative improvement in BLEU over the LSTM baseline—while updating only 10% of GPT-2's parameters. This parameter-efficient fine-tuning approach reduces memory and compute costs, making it particularly attractive for settings with limited resources or rapid iteration requirements.

Despite these promising results, our work also highlights several limitations. Surface-overlap metrics do not fully capture semantic correctness or adherence to practical cooking constraints (e.g., safety and ingredient order), and our evaluation does not include human judges or task-specific rule checks. Moreover, we have focused exclusively on single-modal text generation; richer recipe formats that incorporate images, nutritional information, or user preferences remain unexplored.

Looking forward, we envision several directions to further improve ingredient-to-instruction generation. First, integrating human evaluation and domain-specific rule-based validation would provide more holistic assessments of recipe quality. Second, extending the model to handle multimodal inputs—such as ingredient photos or nutritional targets—could enable more personalized and context-aware recipe generation. Finally, exploring stronger control mechanisms (e.g., reinforcement learning or constrained decoding) may help enforce hard constraints on ingredient usage and cooking procedures, ultimately bringing us closer to fully automated, safe, and practical recipe creation.

## 7 Conclusion

In this work, we systematically assembled and preprocessed a diverse collection of recipe datasets to serve as the foundation for our ingredient-to-instruction generation task. We then designed a sequence-to-sequence architecture, training a bidirectional LSTM from scratch, fine-tuning a pretrained GPT-2 model end-to-end, and finally applying Low-Rank Adaptation (LoRA) to efficiently adapt GPT-2. Across BLEU, ROUGE-N, and ROUGE-L metrics, our models – particularly GPT-2 with LoRA – consistently surpass prior baselines, demonstrating both superior generation quality and notable gains in computational efficiency.

## References

[Bień et al., 2020] Bień, M., Gilski, M., Maciejewska, M., Taisner, W., Wisniewski, D., and Lawrynowicz, A. (2020). RecipeNLG: A cooking recipes dataset for semi-structured text generation. In Davis, B., Graham, Y., Kelleher, J., and Sripada, Y., editors, *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland. Association for Computational Linguistics.

[Bostrom et al., 2021] Bostrom, K., Zhao, X., Chaudhuri, S., and Durrett, G. (2021). Flexible generation of natural language deductions.

[Dathathri et al., 2020] Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. (2020). Plug and play language models: A simple approach to controlled text generation.

[Good, 1956] Good, I. (1956). Some terminology and notation in information theory. *Proceedings of the IEE - Part C: Monographs*, 103:200–204.

[H. Lee et al., 2020] H. Lee, H., Shu, K., Achananuparp, P., Prasetyo, P. K., Liu, Y., Lim, E.-P., and Varshney, L. R. (2020). RecipeGPT: Generative Pre-training Based Cooking Recipe Generation and Evaluation System. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 181–184. ACM.

[Hu et al., 2021] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models.

[Keskar et al., 2019] Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation.

[Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[Li and Liang, 2021] Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation.

[Lin, 2004] Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[Liu et al., 2022] Liu, Y., Su, Y., Shareghi, E., and Collier, N. (2022). Plug-and-play recipe generation with content planning.

[Lu et al., 2021] Lu, X., West, P., Zellers, R., Bras, R. L., Bhagavatula, C., and Choi, Y. (2021). Neurologic decoding: (un)supervised neural text generation with predicate logic constraints.

[Marin et al., 2019] Marin, J., Biswas, A., Ofli, F., Hynes, N., Salvador, A., Aytar, Y., Weber, I., and Torralba, A. (2019). Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In Isabelle, P., Charniak, E., and Lin, D., editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI*. Accessed: 2024-11-15.

[Sakib et al., 2023a] Sakib, N., Shahariar, G. M., Kabir, M. M., Hasan, M. K., and Mahmud, H. (2023a). *Assorted, Archetypal and Annotated Two Million (3A2M) Cooking Recipes Dataset Based on Active Learning*, page 188–203. Springer Nature Switzerland.

[Sakib et al., 2023b] Sakib, N., Shahariar, G. M., Kabir, M. M., Hasan, M. K., and Mahmud, H. (2023b). Towards Automated Recipe Genre Classification using Semi-Supervised Learning.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks.

[Vij et al., 2025] Vij, A., Liu, C., Nair, R. A., Ho, T. E., Shi, E., and Bhowmick, A. (2025). Fine-tuning Language Models for Recipe Generation: A Comparative Analysis and Benchmark Study.

[Yang and Klein, 2021] Yang, K. and Klein, D. (2021). Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

[Zhang et al., 2020] Zhang, Y., Wang, G., Li, C., Gan, Z., Brockett, C., and Dolan, B. (2020). Pointer: Constrained progressive text generation via insertion-based generative pre-training.