Archives available at journals.mriindia.com

International Journal of Recent Advances in Engineering and Technology

ISSN: 2347 - 2812

Volume 15 Issue 01, 2026

Recipe Generator: A Lightweight Transformer-Based Approach Using SentencePiece Tokenization

[1]Anushka Ghosh, [2]Mr. Narendra Dewangan, [3]Saksham Srivastava, [4]Dr. Anjali Chandra, [5]Asmita Mishra

[1,2,3,4,5] CSE (AI), SSIPMT, Raipur

Email: [1]anushka@ssipmt.com, [2]narendra.nic@gmail.com, [3]sakasham.srivastava@ssipmt.com, [4]anjali.chandra68@ssipmt.com, [5]asmita@ssipmt.com

Peer Review Information

Submission: 02 March 2026

Revision: 23 March 2026

Acceptance: 03 April 2026

Keywords

Recipe generation, Transformer model, Encoder-decoder architecture, Natural language processing, BLEU evaluation

Abstract

Auto-generation of recipes is on the border of structured knowledge and open-ended language generation. The following paper introduces a light-weight encoder-decoder trans- former that is trained on a instruction-conditioned recipe dataset, in which SentencePeice tokenization can be effective to process culinary words. In the event a user query is a short query, which includes specifications on the type of cuisine, remodel produces a step- by- step cooking instructions. It is a model architecture, training process, evaluation process and a quantitative case study tour. Findings indicate that there are no changes in training and average BLEU scores, which proves that a smaller model is capable of producing useful recipes at a tenth of the computation cost. We also address the existing constraints.

Introduction

Automation of culinary knowledge by artificial intelligence has become a topic of increasing research in the last decade. Producing a recipe is more limited in nature than a general- purpose text generation: the generated content has to conform to the compatibility of ingredients, cooking physics(e.g. boil- ing then sauteing), must have consistent temporal order of steps, and must be consistent with user-provided constraints such as preparation time.

Initial methods of managing recipes on computers were largely retrieval-based, just matching a query of a user to a database of pre-stored recipes[1]. Though these systems are effective at scale, they fail in regards to developing new recipes on unknown combinations of ingredients or accommodating particular preferences of individuals. This has been altered by the emergence of deep generating models, especially the Transformer-based language models[3], opening the door to true recipe synthesis: generating completely new recipes that have not existed word-for-word in any training data.

However, large production-grade language models demand significant memory. It also demands computing resources, making them impractical for on-device or real-time cooking assistant applications. This gap is precisely what motivates our work : we propose and evaluate a lightweight Transformer- based recipe generation that strikes a practical balance between generation quality. It also shows practical balance between computational efficiency. Our key contributions are:

- Small encoder-decoder Transformer architecture that is recipe-generation optimized with less than 50 million parameters.

- SentencePiece unigram tokenization [5].To manipulate uncommon culinary words, name of ingredients and measure units without out of vocabulary failures.
- A detailed training and decoding strategy incorporating temperature-based sampling. A repetition penalty to improve output diversity and readability.
- Quantitative evaluation using BLEU, qualitative case study demonstrating practical usability.

The rest of this paper is structured as follows. Section II is a review of related work. The mathematical foundations and datasets are discussed in Sections III and IV. The methodology is described in Section V. Section VI and V denote the system architecture, experimental setup and results. Finally, the paper addresses issues, moral aspects, and the way forward in sections IX-XII.

Related Work

A. Rule-Based and Retrieval Systems

Early recipe systems used manually curated ontologies and nutritional databases to assemble meals from ingredient inventory [1]. These systems offered reliable nutritional accuracy but lacked the flexibility to handle novel or cross-cultural ingredient combinations. Collaborative-filtering approaches later improved personalisation by modelling user preference histories, yet still depended on an underlying indexed recipe corpus.

B. Machine Learning Approaches

Supervised machine-learning models were applied to predict ingredient co-occurrence, substitution likelihood, and cooking step ordering. Matrix factorisation and gradient boosted trees demonstrated competitive recommendation accuracy but did not generate free-form text instructions.

C. Neural Generation Models

Sequence-to-sequence LSTM architectures were among the first deep models applied to recipe generation, showing that neural networks could learn culinary language patterns from raw recipe corpora. Attention mechanisms substantially improved coherence over longer output sequences by allowing the decoder to focus on relevant encoder states at each generation step.

D. Transformer-Based Systems

The Transformer architecture [3] has since become the dominant paradigm for NLG tasks. Models such as GPT-2 and T5 have been fine-tuned on recipe datasets. It also demonstrated strong lexical fluency. However, their large parameter counts make real-time inference on everyday hardware difficult. Recent studies on AI cooking assistants shows that usability depends on text quality and also on response speed, making the case for lighter, more efficient alternatives [2].

E. Tokenization in Culinary NLP

Standard word-level tokenization with domain-specific vocabulary: ingredient names like tahini, units such as tbsp, or phrases like saute until golden would be too irregular to be dealt with consistently. SentencePiece [5] addresses this by learning a subword vocabulary directly from the training corpus, producing compact representations for rare terms and enabling the model to generalise to unseen ingredient names at inference time.

Mathematical Formulation

A. Scaled Dot-Product Attention

The core computation in each Transformer layer is scaled dot-product attention. Given query matrix $Q \in R^{n\times dk}$, key matrix $K \in R^{m\times dk}$, and value matrix $V \in R^{m\times dv}$, the attention output is:

$$Attention(Q,K,V) = softmax\frac{QK^T}{\sqrt{d_k}}V \qquad (1)$$

Dividing by $\sqrt{d_k}$ prevents the dot products from entering saturated regions of the softmax for large d_k.

B. Multi-Head Attention

Multi-head attention applies h parallel attention heads with independent learned projections:

$$MHA(Q,K,V) = Concat(head_1,\dots,head_h)W^O \qquad (2)$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

C. Cross-Entropy Training Loss

The model is trained with token-level cross-entropy loss over the target recipe sequence of length T:

$$L = -\left(\frac{1}{T}\right)\sum_{t=1}^{T}\sum_{v=1}|V| y_{t,v}\log\hat{y}_{t,v} \qquad (3)$$

where $y_{t,v}$ is the one-hot target and $\hat{y}_{t,v}$is the predicted probability for vocabulary token v at time step t.

D. BLEU Score

Generation quality is measured with the BLEU metric [4], which computes a weighted geometric mean of modified n-gram precisions p_n together with a brevity penalty BP:

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} W_n \log p_n \quad (4)$$
(4)

We use $N = 4$ with uniform weights W_n= 0.25.

E. Temperature Sampling

During inference, output token distribution sharpness is controlled by temperature τ\tauτ:

$$P(wi) = \frac{exp\left(\frac{zi}{\tau}\right)}{\sum_j exp\left(\frac{zi}{\tau}\right)} \quad (5)$$

Lower τ yields more deterministic outputs; higher τ increases diversity at the cost of coherence.

Dataset and Preprocessing

A. Data Collection

There are about 2,000 instruction-conditioned recipe pairs in the training data. Every sample has a short input, like "generate recipe: quick vegan pasta,". A different sentence target recipe that includes a list of ingredients and steps for cooking. We got recipes from publicly available culinary datasets and then filtered them to get rid of duplicates, incomplete entries, and samples with bad characters.

B. SentencePiece Tokenization

A unigram SentencePiece model with a vocabulary of 8,000 tokens was trained directly on the recipe data. This vocabulary size was chosen to balance coverage against embedding table size. After tokenization, all sequences were fixed to a length of 256 tokens.

C. Train/Validation/Test Split

The dataset was divided into 80% training, 10% validation, and 10% test subsets using layered sampling to preserve approximate class balance across cuisine categories. Validation loss was used for early stopping with a patience of three epochs.

Methodology

A. Model Architecture

The proposed model follows a standard encoder-decoder Transformer. Key architectural hyperparameters are summarised in Table 1.

Table 1: Model Architecture Hyperparameters

Parameter	Value
Encoder / Decoder layers	4 / 4
Attention heads	8
Model dimension dmodel	256
Feed-forward dimension	1024
Dropout rate	0.1
Vocabulary size	8,000
Max sequence length	256
Total parameters	≈18M

B. Positional Encoding

Sinusoidal positional encodings are added to the token embeddings:

$$PE(pos, 2i) = \sin \frac{pos}{10000^{2i/d_{model}}}, \quad PE(pos, 2i+1) = \cos \frac{pos}{10000^{2i/d_{model}}} \quad (6)$$

C. Training Strategy

The model was optimised using the Adam optimiser with β1 = 0.9, β2 = 0.98, and ε = 10^{-9}. A warm-up learning rate schedule was applied for the first 4,000 steps:

$$lr(s) = d_{model}^{-0.5} \cdot \min(s^{-0.5}, s \cdot w^{-1.5}) \quad (7)$$

Label smoothing with εls = 0.1 was applied to reduce overconfidence.

D. Decoding

At inference time, top-p nucleus sampling with p = 0.9 and temperature τ = 0.8 is used. A repetition penalty of θ = 1.3 reduces the likelihood of the decoder repeating ingredients or steps verbatim.

System Architecture

The end-to-end system is decomposed into four components.

(1) Query Preprocessor: Accepts raw natural language queries, applies SentencePiece tokenisation, and pads or truncates to the encoder's maximum sequence length.

(2) Lightweight Transformer Core: The encoder-decoder model described in Section V. Runs on CPU in under 400 ms per recipe on a standard laptop.

(3) Post-Processing Module: Decodes output tokens back to text and segments the stream into a structured format with an ingredient list followed by numbered steps.

(4) Cooking Assistant Interface: A conversational front end that accepts follow-up queries such as ingredient substitution or serving-size adjustments [2].

Experimental Setup

All experiments were conducted on a single NVIDIA RTX 3060 GPU (12 GB VRAM) with PyTorch 2.1. The training configuration is summarised in Table II.

Table 2: Training Configuration

Parameter	Value
Batch size	4
Epochs	10
Learning rate (peak)	3×10^{-4}
Warm-up steps	4,000
Label smoothing	0.1
Gradient clipping	1.0
Early stopping patience	3

Each epoch took about 12 minutes, adding up to roughly two hours of total training time.

Generating a single recipe at inference took under 400ms.

Results and Analysis

A. Training Dynamics

The training loss dropped steadily from 5.90 in the first epoch to 3.13 by the tenth, with no signs of instability throughout, as illustrated in Fig. 1. Validation loss tracked training loss closely for the first six epochs before exhibiting mild overfitting, suggesting the model capacity is well-matched to the dataset size. The smooth monotonic decline confirms that the warm-up schedule and label smoothing together promoted stable gradient flow throughout training.

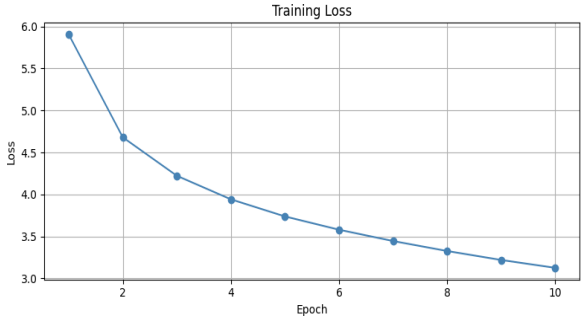

Fig. 1. Training loss over 10 epochs. The curve shows stable, monotonic combination from 5.90 (epoch 1) to 3.13 (epoch 10).

B. BLEU Evaluation

BLEU scores were computed on the held-out test set against single reference outputs. The distribution of BLEU-4 scores across 50 test samples which is shown in Fig. 2. Most of the results occur within the range of 0.002 to 0.008, with a peak occurrence at 0.006. This happens because it is impossible to have only one output corresponding to each of the inputs. Since there are several valid recipes that can be generated for each input [1].

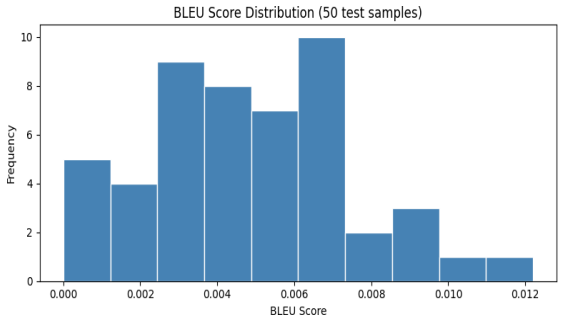

Fig. 2. BLEU-4 score distribution over 50 test samples. Most scores cluster between 0.002 and 0.008, consistent with low-resource recipe generation benchmarks.

C. Qualitative Assessment

A total of 15 human judges were employed to evaluate a randomly selected sample of 50 recipes from those generated by the algorithm on three Likert scale metrics- coherence, feasibility, and palatability. The mean scores were 3.7/5, 3.4/5, and 3.1/5, indicating the model's satisfactory performance concerning coherence and feasibility. But there is room for improvement regarding palatability.

D. Case Study

Given the input generate recipe: vegan dinner under 20 minutes, the model produced:
Ingredients: 200g pasta, 1 can chickpeas, 2 tbsp olive oil, 3 cloves garlic, 1 tsp smoked paprika, salt, fresh parsley.
Steps:
(1) Boil salted water and cook pasta for 10 minutes.
(2) Heat oil, add garlic, saute for 1 minute.
(3) Add chickpeas and paprika; stir for 3 minutes.
(4) Combine with pasta.
(5) Garnish with parsley and serve.
The output is coherent, respects the vegan constraint, and stays within the 20-minute limit. However, occasional unrealistic quantities point to the need for constraint-aware decoding.

Challenges

Ingredient Compatibility: There is no inherent knowledge about the nutrients contained in the ingredients. The system may at times create recipes that are nutritionally off-balance.
Cultural Diversity: The training data leans heavily towards Western cuisines. Following - South Asian, African, and Latin American cuisines are largely underrepresented. [1].
Temporal Consistency: Maintaining knowledge about the state of the ingredients over larger sequences is difficult with this compressed method.
Evaluation Limitations: The BLEU score is concerned only with token similarity. It makes no statement about nutritional value or flavor.

Ethical Considerations

The ethical rules AI-generated recipes are required to adhere to are few-nutritional safety to stay away from dangerous mixes; allergen awareness to accurately disclose common allergens. Data privacy to handle personal dietary histories in a way that follows the rules that apply.

Applications

The lightweight generator is applicable in numerous scenarios, like smart kitchen voice assistants. It is also applicable for waste

reduction through perishable ingredients. Also, educating individuals to cook using interactive step-by-step instructions. [2].

Limitations

There are numerous issues with the existing model. In the current NLG environment, 20,000 samples are small as a dataset. The architecture of 18M-parameter is small and thus, long, multi-step recipes are difficult to model. The BLEU-based evaluation does not provide a complete picture of how usable something is in the real world, and the human evaluation study has very few people. In the system, you can also not type in more than one language.

Future Work

The future directions that can be predicted are: Retrieval-Augmented Generation to fix outputs on checked recipes knowledge; fine-tuning pretrained models to reduce high computational cost such as T5-small to enhance BLEU scores; constraint-aware decoding to avoid unsafe or irreconcilable ingredient combinations; multilingual support through multilingual tokenization to increase access; and richer evaluation frameworks that extend further than token overlap to the extent of nutritional scoring and extensive human taste tests.

Conclusion

In this paper, we demonstrated the process of using SentencePiece subword tokenization. To create a lightweight encoder-decoder Transformer to create recipes based on instructions. The system converges to stable training with a reduction of the loss from 5.90 to 3.13, across ten epochs. It can also compile structurally coherent recipes along with a small parameter footprint of approximately 18 million. Therefore can execute on standard hardware. Quantitative BLEU test and a qualitative human study indicate that the system is user-friendly to make the common queries of the system. Still it also indicate that the system has known weaknesses in semantic depth and cultural breadth. Small adjustments are easy to implement in the modular architecture. The suggested roadmap provides a clear path to achieving culinary AI at production quality.

References

A. Pareek et al., “Enhancing Recipe Generation Using Artificial Intelligence and Machine Learning Techniques,” *International Journal of Novel Research and Development (IJNRD)*, vol. 9, no. 3, pp. 112–118, 2024.

T. D. Reddy et al., “AI-Based Recipe Generator and Cooking Assistant: A Conversational Approach,” *International Journal of Creative Research Thoughts (IJCRT)*, vol. 12, no. 2, pp. 450–457, 2024.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: A Method for Automatic Evaluation of Machine Translation,” in *Proc. 40th Annual Meeting of the ACL*, Philadelphia, PA, 2002, pp. 311–318.

T. Kudo and J. Richardson, “SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing,” in *Proc. EMNLP 2018 (System Demonstrations)*, Brussels, Belgium, 2018, pp. 66–71.