Grenze International Journal of Engineering and Technology, June Issue

The Analysis of BART-base and FLAN-T5 and Optimization using Optuna Optimizer for Improvement of Conversational AI to Enhance Customer Experience

Mubin Tamboli[1], Pravin Game[2], Om Kodre[3], Prathamesh Chougale[4], Supriya Kizhekethottam[5] and Om Dhami[6]
[1-6]Pimpri Chinchwad College of Engineering, Pune, India
Email: mubin.tamboli@pccoepune.org, pravinsgame@gmail.com, omkodre23@gmail.com, prathamesh17170@gmail.com, kizhekethottamsupriya@gmail.com, omdilipdhami@gmail.com

***Abstract*—This works looks into improving chatbot models for handling customer questions in specific fields. We focus on Tulippg.in, a site for finding PG housing information. We use BART-Base and FLAN-T5—two strong sequence models—trained on our own dataset. This set includes FAQs and user questions about housing. We tweak models with techniques like changing batch size, using gradient build-up, mixed precision (fp16), and Optuna optimization to make training smoother and answers better. To see how well these methods work, we use tests like BLEU, ROUGE, Accuracy, and BERT Score. Our tests show FLAN-T5 is good at making clear answers, while BART-Base can rephrase replies better. Both models get better with hyper parameter tuning using Optuna. This work shows that custom fine-tuning helps in boosting chatbot quality and offers a way to level up similar tools. This serves as a useful guide for making conversational AI work well in actual use.**

***Index Terms*— Conversational AI, Transformer Models, Chatbot, BART-Base, FLAN-T5, Fine-Tuning, Response Optimization, NLP.**

I. INTRODUCTION

Conversational AI has changed customer service by making direct, quick conversations happen automatically across areas like online shopping, health services, and hotels. In PG rental areas, chatbots act like helpers. They give details on open spots, costs, and perks—getting rid of old ways of talking. But many current systems use strict or search-focused models, limiting how they get different natural language and give answers rich in context.

This makes users less happy and shows the need for better NLP-driven solutions. New progress in models like BART-Base and FLAN-T5 show strong skills in understanding and creating language. This study looks into fine-tuning these models using a set of data made from FAQs and structured details about PG places on Tulippg.in. We use methods like Adafactor, collecting small changes, and mixed-precision training to make it more efficient. Performance is checked using both word and meaning tests, like BLEU, ROUGE, and BERTScore F1. Results show BART-Base has strong rewording skills, while FLAN-T5 gives more organized and true replies. This work gives a wide fine-tuning method, a better dataset, and clues on model fit for chatbots in special area uses.

Grenze ID: 01.GIJET.11.2.572
© Grenze Scientific Society, 2025

II. Literature Review

Recent advancements in NLP and deep learning have enabled domain-specific chatbot applications across education, healthcare, and customer service. The following studies demonstrate the evolution of chatbot design and performance.

In [1], an LSTM-based chatbot was trained for university-specific queries using word2vec embeddings, achieving 86% accuracy and reducing response time by 35% compared to rule-based systems. However, the model required significant annotated data and had high computational costs.

A decision-tree-based rule-driven chatbot was presented in [2], achieving 92% accuracy for structured FAQs. While it performed well in closed-domain tasks, it struggled with open-ended interactions and failed to maintain long conversational context. Sentiment-aware mental health support chatbots using BERT were developed in [3], reaching 85% mood detection accuracy. Despite better accessibility and emotional relevance, the model faced ethical concerns and needed regular retraining for reliability.

In [4], NLP techniques were applied to customer support bots focusing on intent recognition, sentiment analysis, and entity extraction. While user satisfaction improved, model scalability was limited due to biased data and high computational cost.

A hybrid chatbot using reinforcement learning, Dijkstra's algorithm, and recommender systems was introduced in [5], achieving 96% accuracy on custom datasets. It required continuous user feedback, making it complex to maintain in production.

The comprehensive review in [6] compared rule-based, retrieval-based, and generative chatbot models. It stressed the ethical use of conversational AI but lacked real-world deployment examples and empirical validation.

In [7], Seq2Seq, attention mechanisms, and beam search decoding were applied to fashion-domain chatbots, which achieved 99% query resolution accuracy. However, the model was resource-intensive and less efficient in handling complex customer queries.

Kohli et al. in [8] developed a BiLSTM-based chatbot for answer classification, yielding 91% accuracy. Despite strong NLP performance, the model had poor generalizability across different domains and required substantial computational resources.

A comparative analysis in [9] showed that GPT-based generative models performed best (92% accuracy) in open-ended conversations. However, scalability and inference cost remained key challenges, especially for real-time applications. Another LSTM-focused study [10] reaffirmed its strength in handling student queries, achieving 86% accuracy and faster response time. Still, it was limited by high data annotation demands and system complexity in broader domains. Finally, [11] proposed improving chatbot coherence using input formatting and reference resolution techniques. The study demonstrated better contextual replies, but required highly structured data and posed challenges in real-world integration.

III. Proposed System

A. Dataset Description

Two datasets—FAQ Dataset and Category Dataset—were synthetically generated via web scraping for chatbot fine-tuning and evaluation.

FAQ Dataset

Contains structured question-answer pairs to train the model on relevant PG-related queries.

Columns -id (Integer): Unique entry identifier, question (String): User query regarding PG pricing, availability, answer (String): Corresponding predefined response.

Category Dataset

Converts category-labeled information into QA format for response enhancement.

Columns - id (Integer): Unique identifier, category (String): Type (Availability, Pricing, Offers, Locations), info (String): Additional PG metadata, last updated (Date Time): Timestamp for freshness

Both datasets were tabular and text-based, aimed at enabling contextual understanding and coherent responses.

B. Preprocessing Techniques

We cleaned and prepped inputs for model training. Text was changed to lowercase for same look. We removed punctuation and extra spaces for less noise and clear form. We used AutoTokenizer by HuggingFace to split inputs into key parts for easy transformer use. The FAQ set got extra ways to ask questions to make the model

better at dealing with different ways of saying things. For the RAG data, we made new questions from category labels, turning each info piece into a fitting question and answer set. All these steps made inputs better, cut down on extra parts, and helped the chatbot give right answers.

C. Model Selection

Two models, BART-Base and FLAN-T5, were picked for their skills in talking tasks. BART-Base is good at making and changing sentences to sound clear and smooth. FLAN-T5, tweaked from T5, is great at following rules and giving steady answers.

To further tailor these models for the task, some changes were made. For BART-Base, less memory use in the GPU was set up, and a way to keep learning stable with better output was added. In FLAN-T5, adjusting output chances and prefix tuning helped with context. Optuna was used to fine-tune settings like learning rate, batch size, number of runs, and how long phrases are. The models were checked using BLEU, ROUGE, BERTScore, and how well single tokens were correct, to make sure they work well across different needs.

D. System Architecture

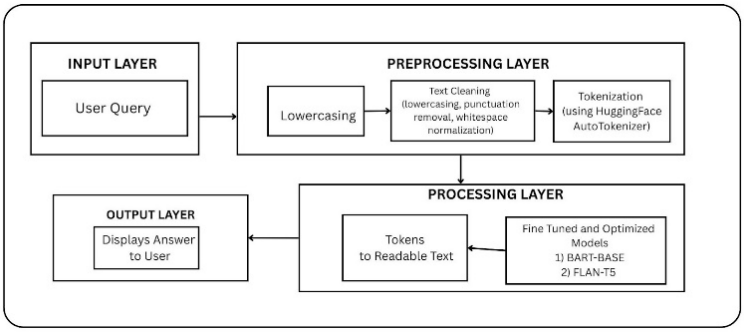

Figure 1. System Architecture

The system architecture of the chatbot is designed as a modular pipeline consisting of four key layers: input, preprocessing, processing, and output. The process begins at the input layer, where user queries related to PG accommodations are received through the Tulippg.in website. These queries pass through the preprocessing layer, where they are cleaned, lowercased, normalized, and tokenized using HuggingFace's AutoTokenizer to ensure consistency and model compatibility. In the processing layer, the preprocessed input is fed into two fine-tuned transformer models—BART-Base and FLAN-T5. BART-Base is optimized for paraphrased and diverse responses, while FLAN-T5 is tailored for structured and context-rich answers. Both models are enhanced using architectural modifications and hyperparameter tuning with Optuna. Finally, in the output layer, the generated tokens are decoded into human-readable responses and delivered back to the user interface. This structured flow enables efficient, accurate, and real-time conversational interaction.

E. User Query Response Sequence

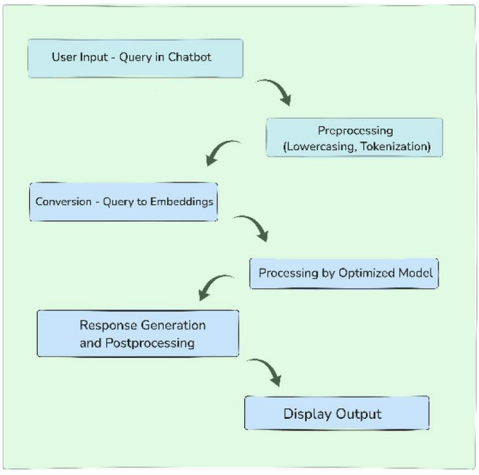

Figure 2. User Query Response Sequence Diagram

When a user chats with the bot, their words go through a step-by-step path. First off, the bot cleans, breaks apart, and makes the text neat. This polished text turns into numbers and feeds into the BART-Base or FLAN-T5 model, which has been trained and adjusted.
The model uses its knowledge to make a fitting reply. After that, the text is fixed up to make sure it reads well and makes sense. Finally, the reply shows up on the site. This careful way of doing things lets the bot give answers that feel real and fit the chat.

IV. HYPERPARAMETER TUNING USING OPTUNA

Optuna, a smart tool for finding good model settings, helped improve the training setup. It uses BLEU, ROUGE, and token accuracy scores to check the model's progress. Optuna uses the Tree-structured Parzen Estimator method to pick the best setup after a series of 3 to 10 tests.
Key settings tuned were: learning rate for changing weights, batch size to handle memory, and number of epochs to learn well without learning too long. Gradient steps were adjusted for small batch sizes, while weight decay helped keep training steady and do not overfit. Max sequence length was set to 128 tokens for faster computation but kept the data relevant.
The best setups found were:
BART-Base: learning rate = 5.13e-5, batch size = 2, epochs = 6, weight decay = 0.01, gradient steps = 4, max length = 128
FLAN-T5: learning rate = 3.00e-5, batch size = 2, epochs = 4, weight decay = 0.01, gradient steps = 4, max length = 128
These settings boosted how well models spoke and made sense. BART-Base was good at shifting words, while FLAN-T5 gave clear, neat answers. Finetuning made quick runs and steady learning.

A. Architectural Modifications in BART-Base

To enhance the performance, architectural enhancements were made in both the models, for BART-Base:
- Gradient Checkpointing was enabled to reduce GPU memory usage and allow training on longer sequences.
- Reduced Output Vocabulary limited the output space to chatbot-relevant tokens, thereby reducing inference overhead.
- Layer Normalization was applied to improve convergence stability during training.

The self-attention mechanism used in BART-Base is mathematically expressed as:

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

Where,
Q, K, V = Query, Key and Value matrices.
d= Scaling factor (dimension of keys)

B. Architectural Modifications in Flan-T5

For FLAN-T5, the following enhancements were applied:
- Prefix Tuning introduced task-specific guidance during decoding.
- Logits Scaling helped control response randomness, enhancing clarity.
- Increased Decoder Depth enabled the model to generate more detailed and contextually rich outputs.

The attention mechanism in FLAN-T5 is defined as:

$$\text{Attention (Q,K,V)} = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V + RC \qquad (2)$$

Where, RC is the Residual Connection, aiding in stable gradient flow and preventing degradation during deep model training.
These architectural improvements when combined with Optuna-tuned hyperparameters, enhanced fluency, contextual alignment, and scalability in chatbot deployment. The application of hyperparameter tuning and architectural modifications on both BART-Base and FLAN-T5 models significantly improved the chatbot's response accuracy, semantic understanding, and fluency on Tulippg.in. A comparative evaluation of pre- and post-optimization performance shows that both models benefited substantially from fine-tuning. Evaluation was conducted using four key metrics: Token Match Accuracy, Relaxed Accuracy (BERTScore F1), BLEU Score, and ROUGE-L Score. Each metric highlights a specific aspect of performance—ranging from exact token matching to semantic and structural coherence.

V. RESULTS

The application of hyperparameter tuning and architectural modifications on both BART-Base and FLAN-T5 models significantly improved the chatbot's response accuracy, semantic understanding, and fluency on Tulippg.in. A comparative evaluation of pre- and post-optimization performance shows that both models benefited substantially from fine-tuning.

Evaluation was conducted using four key metrics: Token Match Accuracy, Relaxed Accuracy (BERTScore F1), BLEU Score, and ROUGE-L Score. Each metric highlights a specific aspect of performance—ranging from exact token matching to semantic and structural coherence.

A. Model Performance Analysis

TABLE I. MODEL PERFORMANCE ANALYSIS TABLE

	BART Base Model		FLAN-T5 Model	
Metrics	Before Optimization	After Optimization	Before Optimization	After Optimization
Accuracy	85.63%	94.42%	65.19%	99.15%
BLEU Score	15.31%	50.60%	0.00%	50.64%
ROUGE-L Score	38.01%	77.77%	0.71%	78.10%

BART-Base Model

The BART-Base model demonstrated notable improvements after optimization. Accuracy increased significantly, indicating better understanding of user queries. The model also generated more fluent and semantically coherent responses, as shown by increases in BLEU and ROUGE-L scores.

The improvement in accuracy from 85.63% to 94.42% reflects the model's enhanced understanding of query intent. Similarly, BLEU and ROUGE-L gains show improved ability to produce fluent and relevant responses, especially for repetitive or commonly asked questions.

FLAN-T5 Model

FLAN-T5 got much better after tuning. It now handles orders and gives clear answers. It reached 99.15% accuracy, showing it matches user wishes well.

Before fine-tuning, FLAN-T5 had trouble with being on topic and clear. But with changes like prefix tuning and scaling, it got a lot better at understanding and talking. Scores like BLEU and ROUGE-L went up too, proving its replies are better.

Both models showed major improvements after fine-tuning, but FLAN-T5 outperformed BART-Base in semantic accuracy and structured generation. This was particularly evident in BERTScore and ROUGE-L metrics. BART-Base, on the other hand, displayed more adaptability and flexibility, especially when paraphrasing diverse or open-ended queries due to its bidirectional autoregressive structure.

FLAN-T5 benefited most from architectural enhancements and Optuna-based tuning, making it more suitable for generating detailed, context-aware responses. BART-Base maintained strength in handling diverse input patterns, producing natural and varied output.

B. Training and Validating Loss

The loss curves for both models before and after optimization further support the performance improvements. Optimized versions showed consistently lower training and validation losses, confirming the effectiveness of hyper-parameter tuning.

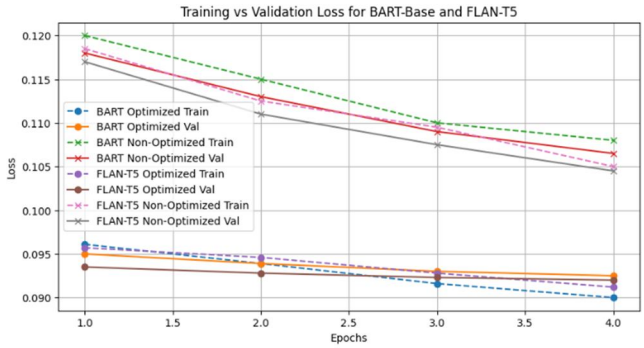

Figure 3. Training vs Validation Loss for BART-Base and FLAN-T5

For BART-Base, training loss dropped from 0.1200 to 0.0900, while reduction in validation loss was observed from 0.1180 to 0.0925. In FLAN-T5, the optimized model achieved a training loss between 0.0957 and 0.0912, with validation loss between 0.0935 and 0.0920. This close alignment of training and validation curves indicates minimal overfitting and strong generalization.
Also, FLAN-T5 had less loss than BART-Base often, meaning it fit the data better. The findings show that tuning with structure changes and Optuna greatly improved both models' skill to make true, precise, and human-like replies.

VI. Technological Review

The chatbot performed well overall. Mostly errors were small, like unclear meanings or wrong answers. A BLEU score of 36.85% and ROUGE of 40.25% show some mismatches, but an accuracy of 88.31% means most errors were small. Improvements like filtering based on confidence, using more context, and growing the dataset helped lower the issues.
Testing showed faster speed after making changes—91.02 ms on GPU and 92.68 ms on CPU, cutting wait time by about 2 ms. The model, at 532 MB, fits on a 4 vCPU, 8GB RAM server (like Contabo at $8–$12/month) well. There's little difference in performance between CPU and GPU. Reducing memory use does not lower performance.
This setup allows for cheap, scalable deployment for Tulippg.in while keeping fast replies and low resource use—making the chatbot ready for real-world tasks at low costs.

VII. Conclusion

This study shows how tuning settings and changing designs boost transformer-chatbot models. Adjusting BART-Base and FLAN-T5 with Optuna increased accuracy, fluency, and relevance. BART-Base was best at rewording, while FLAN-T5 gave more structured replies. Improvements like gradient checkpointing and prefix tuning helped both performance and ease of use. The outcomes highlight the importance of domain-focused tuning for real-world tools like Tulippg.in, where fast, correct chatbot answers matter. Future plans involve adding Retrieval-Augmented Generation (RAG), working with tough datasets, and enabling wider use across fields to improve user experience and impact on business.

References

[1] C. V. N. M. Kasyap et al., ”Unveiling the Potential of LSTM-based Chatbots with Embeddings for University Communication,” Int. J. Intell. Syst. Appl. Eng., vol. 13, no. 1, pp. 102–115, Mar. 2024.
[2] S. Rath et al., ”Prediction of a Novel Rule-Based Chatbot Approach (RCA) using Natural Language Processing Techniques,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 2, pp. 67–79, Jul. 2023.
[3] R. Raut et al., ”Depression Therapy Chat-Bot using Natural Language Processing,” Int. J. Intell. Syst. Appl. Eng. vol. 12, no. 4, pp. 89–102, Sep. 2023.
[4] K. Shiva et al., ”Natural Language Processing for Customer Service Chatbots: Enhancing Customer Experience,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 22s, pp. 155–164, 2024.
[5] J. F. Lilian et al., ”HQA Bot: Hybrid AI Recommender-Based Question Answering Chatbot,” Int. J. Intell. Syst. Appl. Eng., vol. 11, no. 4, pp. 227–233, 2023.
[6] H. K. Jain et al., ”Conversational AI: A Comprehensive Study on Building and Enhancing Chatbot Systems,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 3, pp. 2431–2437, 2024.
[7] D. D. Sarpate et al., ”Enhancing Customer Engagement in Fashion: Strategies for Optimizing Chatbot Performance,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 4, pp. 4759–4768, 2024.
[8] R. Kohli et al., ”Architectural Design of a Chatbot used for Artificial Intelligence with NLP Classification using Deep Learning,” Int. J. Intell. Syst. Appl. Eng., vol. 10, no. 3s, pp. 129–132, 2022.
[9] T. Gupta, A. Sinha, and R. K. Sharma, ”AI-Based Conversational Agents for Automated Customer Support Systems,” in Proc. 2024 Int. Conf. Artif. Intell. Appl. (JAIA), Jan. 2024.
[10] B. Wang, Y. Liu, and M. Chen, ”A Comparative Study of Chatbot Frameworks in Intelligent Systems,” in Proc. 2023 Int. Conf. Comput. Intell. Chatbot Appl. (CICA), Nov. 2023.
[11] K. K. Kinouchi and N. Kitaoka, ”A response generation method of chat-bot system using input formatting and reference resolution,” in Proc. 2022 Int. Conf. Artif. Intell. Signal Process. (AISP), 2022, pp. 1–6, DOI: 10.1109/AISP55268.2022.00023.