Development of Microcontroller-Based AI Robot Tour Guide Utilizing Custom Language Models

Ian S. Jackson
West Virginia University
Morgantown, United States
isj0001@mix.wvu.edu

Aiden G. Ballard
West Virginia University
Morgantown, United States
agb00033@mix.wvu.edu

Dr. Mohamed Hefeida
West Virginia University
Morgantown, United States
mohamed.hefeida@mail.wvu.edu

Dr. Anurag Srivastava
West Virginia University
Morgantown, United States
anurag.srivastava@mail.wvu.edu

Dr. Prashnna Gyawali
West Virginia University
Morgantown, United States
prashnna.gyawali@mail.wvu.edu

***Abstract*—This paper presents the design and implementation of an AI-powered robotic tour guide system for the West Virginia University Lane Department of Computer Science and Electrical Engineering (LCSEE). The system enables prospective students to engage in real-time, natural language conversations with a robot equipped with department-specific knowledge. Two methodologies were explored: (1) a DirectLLM approach that fine-tunes a large language model on a custom dataset, and (2) a Classify-Retrieve-Generate (CRG) pipeline that modularizes classification, answer retrieval, and natural response generation. A custom SQuAD-style dataset was developed using LCSEE data, supporting both pipelines. The system was deployed on a Raspberry Pi 4 integrated with a MangDang Mini Pupper robot. Evaluation results show that while DirectLLM excels in fluency, it is hindered by retraining requirements and limited scalability. In contrast, the CRG method provides modularity and easier maintenance, with strong performance on classification and retrieval but pending refinement in response generation. Future work includes dataset expansion, CRG generator improvements, DirectLLM fine-tuning, and exploration of hybrid fusion models. This research demonstrates the feasibility of deploying compact, domain-aware AI agents for real-time educational engagement.**

***Index Terms*—Robotic Tour Guide Custom Language Models DirectLLM Classify-Retrieve-Generate (CRG) Raspberry Pi Deployment Domain-Specific NLP**

I. Introduction

Advancements in artificial intelligence (AI) and natural language processing (NLP) have enabled new forms of human-computer interaction, particularly in the realm of educational engagement. Universities increasingly seek innovative ways to connect with prospective students, providing them with immersive experiences that showcase academic programs, research opportunities, and campus life. One such approach is the integration of AI-powered robotic tour guides that allow visitors to interact dynamically with departmental resources.

This research focuses on the development of an AI-driven robotic tour guide designed to enhance the experience of prospective students visiting the West Virginia University Lane Department of Computer Science and Electrical Engineering (LCSEE). The goal is to create an interactive medium where students can ask questions about the department, and the robot, powered by language models, generates informative responses in real-time. Unlike static presentations or scripted responses, this approach enables natural, context-aware interactions, providing a more engaging and personalized tour experience.

To achieve this, two distinct methodologies were explored: (1) DirectLLM, which fine-tunes a large language model (LLM) specifically for department-related queries, and (2) a Classify-Retrieve-Generate (CRG) pipeline, which first classifies the user's question, retrieves relevant information, and then generates a response. A custom Q&A dataset was curated using department-specific data, ensuring that the AI model delivers accurate and contextually relevant information.

The system was deployed on a Raspberry Pi 4, paired with a MangDang Mini Pupper [1] robot as the physical embodiment of the tour guide. The robot is equipped with a simple microphone and speaker interface, facilitating natural language communication with users. The choice of hardware necessitated the use of a lightweight AI model optimized for fast inference, ensuring real-time conversational interactions without significant latency.

This paper details the design, development, and deployment of the AI-powered robotic tour guide, evaluating the performance of both DirectLLM and the CRG-based approach. By comparing these methods, we aim to determine the most effective strategy for delivering responsive and informative AI-driven interactions in an embedded robotics context.

II. Related Work

A. Fine-Tuning Large Language Models for Domain-Specific Tasks

Fine-tuning pre-trained LLMs has become a standard approach to adapting general-purpose models for domain-specific applications. Studies show that fine-tuned LLMs outperform generic models in specialized domains, such as health-

This research was partially supported by the Defense Advanced Research Projects Agency (DARPA) Award No. 000842.
Computational resources were provided by the WVU Research Computing Thorny Flat HPC cluster, partly funded by NSF OAC-1726534.

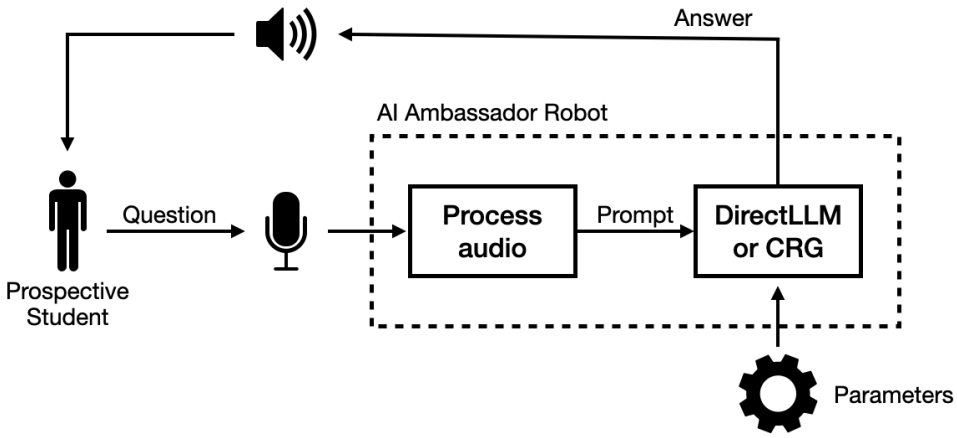

Fig. 1: AI Robot System Diagram

care and legal services [2], by leveraging tailored datasets. The DirectLLM approach in this work applies the same principle, fine-tuning a model specifically on LCSEE information. However, fine-tuning requires significant computational resources, which poses challenges for deployment on edge devices like microcontrollers and single-board computers [3].

B. Deploying AI Models on Resource-Constrained Devices

Deploying AI models on low-power devices such as Raspberry Pi and microcontrollers presents unique challenges, primarily due to hardware limitations in memory and computation. Prior studies have explored optimization techniques such as quantization, pruning, and distillation to reduce model size while preserving accuracy [4]. Recent advancements in small language models (SLMs) offer promising alternatives for efficient, real-time inference [5]. This work integrates such optimizations to deploy a lightweight yet effective AI system on a Raspberry Pi 4.

III. System Overview

This section presents the overall architecture of the AI-powered robotic tour guide system, detailing the hardware platform, software architecture, and end-to-end data flow from voice input to generated response.

The robot features three main parts to facilitate an interactive experience: a microphone, a computer, and a speaker. A simplified system diagram can be seen in Figure 1. The prospective student will interact with the robot via the microphone, where their question will be transcribed from audio to text. This question is then fed into either the DirectLLM pipeline or CRG pipeline. Once the pipeline processes the prompt, it returns a response to the prospective student via a speaker.

The chosen robotic medium is the MangDang Mini Pupper [1]. The decision was made based on the cost, ease of assembly, and the all-inclusive kit. Speech recognition was handled via the Uberi SpeechRecognition library interfaced with Google's API. For text-to-speech, the Flite engine by CMU was used.

IV. Custom Dataset Development

Due to the niche nature of the application, a custom dataset of question and answer pairs about the LCSEE department was developed. The dataset was created by curating information from various sources, including the LCSEE department website, course catalogs, departmental brochures, and talking to tour guides of the department. The goal was to ensure that the dataset covered a wide range of topics relevant to prospective students, including academic programs, faculty research, campus facilities, and student life.

Question and answer pairs were human-generated, ensuring accuracy and eliminating the risk of misinformation. To expand the dataset and improve generalization, data augmentation techniques were employed. This included paraphrasing questions, rephrasing answers, and introducing synonyms to create variations of the original pairs. To efficiently augment the dataset, each question answer pair was fed into ChatGPT to generate multiple variations of the same question and answer.

TABLE I: Custom Dataset Summary

Category	**Number of Pairs**
Degree Programs	102
Research Opportunities	77
Facilities and Resources	115
Clubs and Organizations	130
Career Opportunities	60
Internships	60
Financial Aid and Scholarships	62
Faculty Information	62
Admission Process	60
Location and Contact	9
Generic	13
Total	**750**

The dataset is stored in the SQuAD format [6]. The SQuAD format includes a list of topics as the top level, with each topic having a title. In each topic exist paragraphs, these contain context to the topic and question-answer pairs. To fit the dataset to the task at hand, the topics were chosen based off of types of questions that could be asked about the

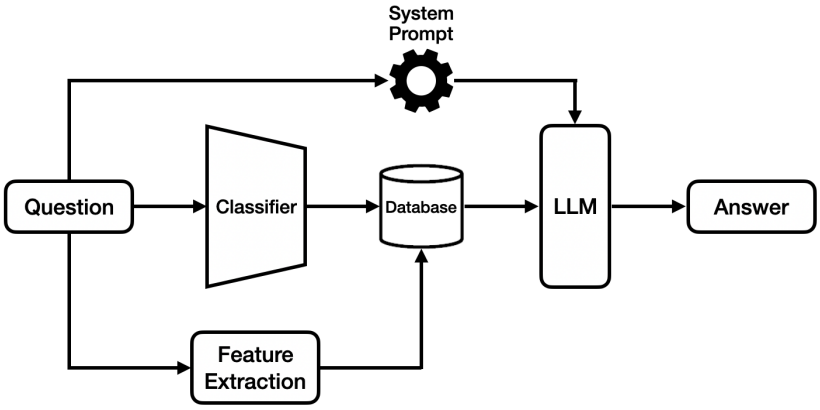

Fig. 2: Classify-Retrieve-Generate (CRG) Pipeline

department: degree programs, research opportunities, facilities and resources, clubs and organizations, career opportunities, internships, financial aid and scholarships, faculty information, admission process, and location and contact.

The dataset consists of 750 question-answer pairs, with 11 categories. A table summarizing the dataset can be seen in Table I.

V. DirectLLM Model

A. Model Selection and Justification

The DirectLLM model consists of a pre-trained LLM that is fine-tuned on the custom dataset. To meet the project's objectives, the language model was developed based on the following criteria: it needed to be open source, capable of effective training on question-and-answer (Q&A) datasets, and optimized for lightweight deployment with quick response times on a microcontroller platform.

An open-source language model was chosen due to its capabilities in lightweight implementations while maintaining high-quality natural language understanding and generation. Open-source models such as FLAN-T5 [7] and BART [8] provide a solid foundation due to their small size and performance on Q&A task with minimal fine-tuning. These models are well-suited for customization and can be fine-tuned with domain-specific data.

Fine-tuned Language-Agnostic Network (FLAN-T5) is a variant of the T5 (Text-to-Text Transfer Transformer) model, which is pretrained on a diverse range of tasks. It is highly adaptable and can be trained on a wide variety of datasets, including Q&A tasks. It also meets the versatile, scalability, and open-source criteria. The model used in this project has 77 million parameters, making it lightweight and suitable for deployment on resource-constrained devices.

Bidirectional and Auto-regressive Transformer (BART) is a sequence-to-sequence model optimized for tasks like text generation, summarization, and translation. It combines bidirectional context encoding with autoregressive decoding. It features capabilities to fine-tune on specific datasets, and effective in both understanding and generating coherent responses. Like FLAN-T5, it meets the criteria concerned with being open-source, scalable, and versatile. The model used in this project has 139 million parameters, making it suitable for deployment on moderately resource-constrained devices.

B. Training and Fine-Tuning Process

Both FLAN-T5 and BART were finetuned on the same custom dataset. The model hyperparameters can be found in Table II. The test dataset included 22 questions pertaining to the LCSEE department, with 2 questions from each of the 11 categories.

TABLE II: Hyperparameters for FLAN-T5 and BART Fine-Tuning

Hyperparameter	**FLAN-T5**	**BART**
Evaluation Strategy	Epoch	Epoch
Weight Decay	0.01	0.01
Learning Rate	5e-5	3e-5
Batch Size	8	8
Number of Training Epochs	10	5

C. Deployment on Microcontroller

The implementation of FLAN-T5 and BART leverages Hugging Face Transformers, a comprehensive framework for working with pre-trained language models. Training was done via the Hugging Face training pipeline, built on the Trainer class and text generation was configured for inference using the Hugging Face pipeline for text generation. Each model utilized their own transformer: The T5 Tokenizer [9] for FLAN-T5 and BART Tokenizer [10] for BART.

VI. Classify-Retrieve-Generate (CRG) Model

To address the issues of how to handle new information in the LCSEE department, a new approach is needed. Instead of a DirectLLM approach, which would need to be retrained with new information, the classify-retrieve-generate (CRG) approach is proposed. This approach uses a neural network to *classify* the user's question into one of the categories from Table I. Once classified, the best-matching answer is then *retrieved* from the custom database based on the question,

category, and extracted features. Finally, the retrieved answer is *generated* into a more natural response using a lightweight generative model. This modular design allows classification and retrieval components to be updated independently of the generative model. New information can be added to the database without needing to retrain the entire model. Lastly, the CRG approach allows for a more efficient use of resources, as the generative model can be smaller and more lightweight than a full LLM. The CRG pipeline can be seen in Figure 2.

Each step of the CRG pipeline has various approaches that can be used to achieve the desired results. The following sections detail the best performing approaches for each step of the pipeline. A summary of the CRG pipeline components and methods can be seen in Table IV.

A. Classification Step

To initiate the CRG pipeline, a classifier is used to determine the category of an incoming question. Each category corresponds to a high-level topic (e.g., degree programs, research opportunities) within the LCSEE department's offerings Several classification approaches were explored, ranging from traditional machine learning to transformer-based fine-tuned models. In each approach, the input question is first transformed into a numerical format using Term Frequency–Inverse Document Frequency (TF-IDF), word embeddings, or contextual token embeddings.

1) Traditional Machine Learning Approaches: **Logistic Regression for Multi-Class Classification:** Logistic regression is a linear model that predicts the probability distribution over a set of classes using a softmax function. It is particularly well-suited for structured, lower-dimensional feature spaces such as TF-IDF vectors.

Given an input feature vector $\mathbf{x} \in \mathbb{R}^n$, the probability of class $i \in 1, 2, \ldots, k$ is given by:

$$P(y_i|\mathbf{x}) = \frac{e^{\beta_i^T \mathbf{x}}}{\sum_{j=1}^{k} e^{\beta_j^T \mathbf{x}}} \tag{1}$$

where β_i is the weight vector for class i, and k is the number of classes. Training involves minimizing the cross-entropy loss between the predicted and actual distributions.

$$\mathcal{L}_{CE} = -\sum_{i=1}^{k} y_i \log(P(y = i|\mathbf{x})) \tag{2}$$

TF-IDF is used to convert questions into vector representations before being fed into the classifier. The term frequency $\text{TF}(t)$ and the inverse document frequency $\text{IDF}(t)$ are defined as:

$$\text{TF}(t) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{3}$$

$$\text{IDF}(t) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right) \tag{4}$$

where $f_{t,d}$ is the frequency of term t in document d, N is the total number of documents, and D is the set of all documents. The TF-IDF vector for a term t in document d is given by:

$$\text{TF-IDF}(t) = \text{TF}(t) \cdot \text{IDF}(t) \tag{5}$$

This method is fast and computationally efficient, making it a strong baseline for classification tasks with small to medium-sized datasets.

Support Vector Machines (SVM): SVMs are another traditional method that finds a hyperplane to separate classes by maximizing the margin. In the multi-class setting, a One-vs-Rest (OvR) strategy is typically employed. The decision function for each class is defined as:

$$f_i(\mathbf{x}) = \sum_{i=1}^{k} \mathbf{w}_i^T \mathbf{x} + b_i \tag{6}$$

where $\mathbf{w}_i$ is the learned weight vector for class i, and b_i is the bias term. The predicted class is:

$$\hat{y} = \arg\max_i f_i(\mathbf{x}) \tag{7}$$

SVMs use the hinge loss function to enforce a margin between decision boundaries:

$$\mathcal{L}_{hinge} = \sum_{i=1}^{k} \max(0, 1 - y_i f_i(\mathbf{x})) \tag{8}$$

These models work well when the data is not deeply contextual, and they offer high speed and generalization performance for TF-IDF inputs.

2) Transformer-Based Classification: **BERT (Bidirectional Encoder Representations from Transformers):** BERT is a transformer-based model pretrained on large corpora using masked language modeling [11]. It captures contextual relationships in text by processing input bidirectionally. For classification, the [CLS] token's embedding is used and passed through a feedforward classifier layer. The model is fine-tuned on the task-specific dataset, with the SQuAD-like format providing labeled classes for supervised training.

Fine-tuning BERT involves updating all transformer weights using a small learning rate and minimizing cross-entropy loss over the predicted class probabilities. While BERT achieves high accuracy, it is computationally heavy and slower during inference, which can be problematic for edge or real-time applications.

DistilBERT: DistilBERT is a lighter and faster version of BERT, trained via knowledge distillation [12]. It retains about 97% of BERT's performance while being 60% faster and using 40% fewer parameters. Like BERT, the classification token is used to predict the question category, and the model is fine-tuned on the custom dataset.

DistilBERT offers a balance between performance and efficiency, making it suitable for mobile or embedded systems where fast inference is critical.

A comparative summary of the classification methods is presented in Table III. These models were all trained and tested on the same labeled dataset of questions and categories. Evaluation metrics such as accuracy, F1 score, and response time are discussed in Section VII.

TABLE III: Comparative Summary of Classification Models

Model	Accuracy	Inference Speed
Logistic Regression	Medium	Fast
Support Vector Machine	High	Fast
BERT	Very High	Slow
DistilBERT	High	Medium

B. Feature Extraction Techniques

In the CRG pipeline, once a question is classified into a specific category (e.g., research opportunities, clubs and organizations), the next step is to extract meaningful features from the question to assist in retrieving the most relevant answer. Since each category may contain numerous similar question-answer pairs, surface-level classification alone is insufficient. This section outlines three primary methods explored for extracting relevant information from user queries: Named Entity Recognition (NER), TF-IDF keyword extraction, and word embeddings for semantic search.

1) Named Entity Recognition (NER): Named Entity Recognition (NER) is a technique used to identify and label key terms in a sentence that refer to proper nouns, such as names of people, organizations, degrees, or locations. For this project, spaCy's pretrained NER model [13] was employed to identify relevant entities within user queries. For example, in the question:

"What research is done in the biometrics field?"
NER extracts: *research*, *biometrics*, *field*

"What dual degrees can I pursue?"
NER extracts: *degree*

This method performs well when questions contain distinct, recognizable entities (e.g., Cybersecurity, Morgantown, Capstone), but may struggle with abstract or compound terms (e.g., dual degree, interdisciplinary program). Additionally, the pretrained models are not optimized for the specific academic domain, leading to inconsistent coverage of niche terminology.

2) TF-IDF Keyword Extraction: TF-IDF can also be applied to extract high-value keywords from a question. It assigns each word a score indicating its importance in the current question relative to a background corpus.

"What research is done in the biometrics field?"
NER extracts: *research*, *biometrics*, *field*

"What dual degrees can I pursue?"
NER extracts: *dual*, *degree*, *pursue*

This technique is computationally lightweight and works well with short inputs like student queries. However, its primary limitation is a lack of semantic awareness; it treats each word independently and cannot recognize synonyms or related phrases. For instance, questions about "job prospects" and "career opportunities" may score very differently even if they refer to the same concept.

3) Word Embeddings for Semantic Search: To overcome the limitations of sparse keyword methods, semantic embeddings are used to convert entire questions into dense vector representations. The *all-MiniLM-L6-v2* model from Sentence Transformers [14] [15] was chosen for this purpose, as it provides high-quality sentence embeddings that capture semantic meaning effectively. The model generates a 384-dimensional vector for each input question, where semantically similar questions are closer in the vector space.

This method yields the best retrieval results in cases where questions are phrased ambiguously or contain rare synonyms. However, it is computationally more expensive than NER or TF-IDF extraction and requires dimensionality reduction or vector indexing to maintain efficiency in larger datasets

C. Retrieval Methods

Once a question has been classified and its key features extracted, the next step in the CRG pipeline is to retrieve the most relevant answer from the custom question-answer (Q&A) dataset. The retrieval component compares the incoming query—represented via keywords or vector embeddings—against existing questions in the dataset filtered by category. Several techniques were evaluated for this purpose, ranging from simple keyword-based matching to semantic similarity scoring. This section outlines the retrieval methods implemented and the mathematical formulations behind them.

1) Exact Keyword Intersection (EKI): In the EKI method, the set of extracted keywords from the user query is compared to the keywords in each candidate question (within the same classified category). Each keyword match is given a score of 1, and the total intersection score is used to rank candidates. Let A be the set of keywords from the user query, and B_i be the set of keywords from the i^{th} candidate question. The intersection score is defined as:

$$\text{EKI}(A, B_i) = |A \cap B_i| \tag{9}$$

The question with the highest EKI score is selected. In case of a tie, the first candidate is returned or one is randomly selected. While EKI is fast and easy to implement, it is sensitive to exact word matches and ignores synonyms or paraphrasing.

2) Jaccard Similarity: The Jaccard index is a set-based similarity measure used to assess the overlap between two sets of keywords. Unlike EKI, it accounts for both shared and distinct elements. The Jaccard score between sets A and B_i is defined as:

$$\text{Jac}(A, B_i) = \frac{|A \cap B_i|}{|A \cup B_i|} \tag{10}$$

This score ranges from 0 (no overlap) to 1 (complete match). The candidate with the highest Jaccard score is chosen. Ties are handled similarly to EKI. Jaccard provides better granularity than raw intersection and is more robust for uneven keyword set.

3) JEKI Score (Jaccard + EKI Hybrid): To combine the strengths of EKI and Jaccard, a weighted score called the JEKI score was introduced:

$$\text{JEKI}(A, B_i) = \lambda \cdot \text{EKI}(A, B_i) + (1 - \lambda) \cdot \text{Jac}(A, B_i) \tag{11}$$

where $\lambda \in [0, 1]$ is a hyperparameter that balances the two components.

TABLE IV: Summary of CRG Pipeline Components and Methods

Step	Method	Advantages	Limitations	Edge Suitability
Classification	Logistic Regression (TF-IDF)	Fast, good for small datasets	Lower accuracy on complex inputs	High
	SVM (TF-IDF)	Higher accuracy, strong generalization	Ignores context, no semantic understanding	High
	BERT	Very high accuracy, context-aware	Slow inference, large model size	Low
	DistilBERT	Good trade-off of speed and accuracy	Slightly reduced accuracy	Medium
Feature Extraction	NER (spaCy)	Domain-agnostic, fast	Misses abstract academic terms	High
	TF-IDF Keyword Extraction	Lightweight, interpretable	No synonym/semantic handling	High
	Sentence Embeddings	Captures semantics, handles varied phrasing	Higher compute/memory cost	Medium
Retrieval	Exact Keyword Intersection	Simple, fast	Needs exact match, low recall	High
	Jaccard Similarity	Accounts for partial matches	Still keyword-based, no context	High
	JEKI (Hybrid)	Balances raw and normalized	Requires tuning λ	Medium
	Cosine Similarity	Best semantic matching	Most compute-intensive	Medium
Generation	FLAN-T5-Small	Lightweight, open-source	Prompt adherence issues	High
	TinyLlama (1.1B)	Coherent output, promising	Needs fine-tuning	High
	GPT-Neo (1.3B)	Baseline benchmark	Medium resource use, inconsistent	Medium
	Mistral-7B	High-quality responses	Too large for edge deployment	Low

This hybrid approach improves answer selection, especially in tie cases where EKI or Jaccard alone yields similar scores. The balance between raw match count and normalized similarity allows JEKI to adapt across diverse query styles.

4) Vector Space Retrieval via Cosine Similarity: For a more semantic retrieval method, user queries and candidate questions can be encoded into dense vector embeddings using pretrained models (as described in Section VI-B3). These vectors are compared using cosine similarity, defined as:

$$\cos(\theta) = \frac{\vec{q} \cdot \vec{q_i}}{||\vec{q}|| \cdot ||\vec{q_i}||} \tag{12}$$

where $\vec{q}$ is the vector for the user query and $\vec{q_i}$ is the vector for the i^{th} candidate question.

The question with the highest cosine similarity is retrieved. This method captures contextual closeness even when wording differs significantly (e.g., "student groups" vs. "clubs and organizations"). It requires more memory and computation but yields the highest retrieval accuracy in semantically diverse queries.

D. Generation Step

The final step in the CRG (Classify-Retrieve-Generate) pipeline is the generation stage, which takes the retrieved answer and rephrases it into a more natural, engaging, and conversational response. This layer transforms a static Q&A system into a dynamic tour experience, simulating human-like dialogue. Rather than relying on a large end-to-end model like in the DirectLLM approach, this modular step uses lightweight generation models tailored for refinement, not inference.

The inputs to the generation model include: the original user query, the retrieved answer, and a system prompt that defines the tone and style of the assistant. The generation model is prompted to use the retrieved answer directly, then follow it up with a question or elaboration to encourage ongoing interaction. A sample system prompt used in experimentation was:

> "You are an AI assistant for a university tour guide. Your role is to provide engaging and helpful responses based on the retrieved answer. You must always use the retrieved answer in your response first. Then, ask a relevant follow-up question to encourage further conversation."

Several small-scale generative models were tested for compatibility with edge devices. FLAN-T5: Instruction-tuned, lightweight transformer. Easy to deploy but struggles with prompt adherence at smaller sizes. TinyLlama (1.1B): Small, instruction-following model showing early promise in producing coherent responses with minimal compute. DistilBERT: Evaluated but rejected—lacks generative capabilities. GPT-Neo and Mistral-7B: Tested for baseline comparison; too large for final deployment.

All models were run with variations in system prompts and temperature settings to observe response variability and coherence. This step remains work-in-progress. Preliminary results show inconsistent behavior in prompt adherence—many models fail to include the retrieved answer or ask relevant follow-up questions. Further fine-tuning and prompt engineering are planned.

TABLE V: Summary of Response Generation Models (Preliminary)

Model	Params	Output Quality	Edge Suitability
FLAN-T5-Small	80M	Inconsistent	High
TinyLlama	1.1B	Promising	High
GPT-Neo	1.3B	Mixed	Medium
Mistral-7B	7B	High	Low

VII. Evaluation and Comparison

This section compares the performance of the DirectLLM approaches and the CRG approaches in addressing question about the LCSEE department. The evaluation framework considers both quantitative metrics (accuracy, BLEU, response

time, and memory usage) and qualitative observations (response fluency, robustness, ease of update).

A. DirectLLM Evaluation

To evaluate the performance of FLAN-T5 and BART, two metrics are employed: Answer accuracy and BLEU Score. These metrics provide insight to the models' understanding and generation tasks. Another metric used to evaluate the model is average response time, or the mean of the time it takes to process an input and provide an output over all test data. Lastly, the average memory usage is recorded to determine the feasibility of deploying the model on a microcontroller.

BLEU Score: The Bilingual Evaluation Understudy (BLEU) Score measures the quality of the text generated by comparing it with one or more of the reference texts. It assesses how similar the generated text is to human-provided references, in this case the reference is the provided answer to the test question. The BLEU Score formula is seen in Equation 13.

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log(p_n)\right) \tag{13}$$

$$\text{BP} = \begin{cases} 1, & c > r \\ e^{1-r/c}, & c \leq r \end{cases} \tag{14}$$

where BP is the Brevity Penalty (seen in Equation 14), p_n is the precision of n-grams, and w_n is the weight associated to n-grams (typically equal $\forall n$).

B. CRG Evaluation

The various combinations of implementations for each step in the CRG pipeline were evaluated to determine their effectiveness in answering questions about the LCSEE department. In total there are 96 combinations of the CRG pipeline, the best three combinations are shown in this section. Table VI summarizes the three best-performing CRG configurations, including the classification model, retrieval method, and generation model used in each case.

TABLE VI: Best Performing CRG Combinations

CRG Model	Classification	Extraction	Retrieval
CRG-1	LR	Vector	Cosine
CRG-2	SVM	NER	EKI
CRG-3	DistilBERT	Vector	Cosine

The classification is evaluated on accuracy, which is the ratio of correctly classified questions to the total number of questions in the test set. The retrieval is evaluated using a retrieval score (RS), which is a binary score indicating where the correct answer was retrieved from the dataset.

$$\text{RS} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(y_i = \hat{y}_i) \tag{15}$$

Both DirectLLM and CRG are evaluated by their mean response time and average memory usage over the test dataset. The DirectLLM results are seen in Table VII, while the CRG results are seen in Table VIII.

TABLE VII: DirectLLM Evaluation

Model	Acc.	BLEU	Resp. Time (s)	Mem (MB)
FLAN-T5	68.18%	91.17%	38.45	2115.1
BART	27.64%	74.41%	9.961	648.86

TABLE VIII: CRG Evaluation

Model	Class Acc.	RS	Resp. Time (s)	Mem (MB)
CRG-1	78.57%	0.929	12.78	996.7
CRG-2	85.71%	0.750	1.596	613.6
CRG-3	64.29%	0.929	14.21	1279.8

VIII. Discussion

A. Lessons Learned

Through the development and evaluation of both the DirectLLM and Classify-Retrieve-Generate (CRG) approaches, several key insights emerged regarding their suitability for deployment in a robotic tour guide system.

One of the primary challenges with the DirectLLM pipeline was the training bottleneck imposed by dataset requirements. As a fine-tuned small language model (SLM), DirectLLM demands a substantial and diverse dataset to maintain conversational accuracy across a wide range of questions. Furthermore, any updates to department-specific information (e.g., new faculty, facilities, or programs) would necessitate retraining the entire model. This approach is neither scalable nor maintainable for dynamic environments, particularly when operating on constrained hardware.

In contrast, the CRG framework introduced a modular alternative. By decoupling classification, retrieval, and generation into distinct steps, CRG enables individual components to be updated or replaced without the need to retrain an end-to-end model. For example, new answers can be added directly to the retrieval database without affecting the classifier or generator. This modularity makes CRG especially well-suited for educational deployments where content frequently evolves.

B. Trade-Offs

While CRG offers clear advantages in adaptability and maintainability, it comes with certain trade-offs. DirectLLM, once trained, can provide highly fluent and contextually rich responses, particularly for open-ended questions. The model leverages deep contextual embeddings to infer nuanced intent and generate more natural responses.

However, this fluency comes at the cost of scalability and update flexibility. CRG, though more structured and sometimes less fluent in response style, allows for deterministic output based on a well-controlled retrieval mechanism. Additionally, CRG's modular design permits fine-grained improvements to individual components such as classification accuracy or retrieval algorithms without affecting the entire pipeline.

C. Practical Considerations

For deployment on microcontroller-based robots, several practical considerations must be addressed. Inference time and

memory usage are critical constraints. DirectLLM models, such as BART and FLAN-T5, exhibited slower average response times and higher memory footprints. In contrast, the CRG system, particularly when using lightweight models like Logistic Regression or DistilBERT for classification, offered significantly faster and more deterministic performance, making it a better fit for real-time conversational interactions on low-power hardware.

Furthermore, CRG's deterministic logic and rule-based fallback mechanisms enhance its reliability, which is crucial in educational contexts where incorrect or hallucinated responses could erode user trust.

IX. Conclusion

This research presented the development of an AI-powered robotic tour guide for the Lane Department of Computer Science and Electrical Engineering, leveraging two natural language processing (NLP) pipelines: DirectLLM and Classify-Retrieve-Generate (CRG). Each approach was designed, implemented, and evaluated with the goal of enabling natural, accurate, and efficient interactions between prospective students and the department's offerings.

The DirectLLM approach demonstrated strong performance in generating fluent and context-rich responses, benefiting from end-to-end fine-tuning on a curated dataset. However, this pipeline faces a critical training bottleneck: any update to the departmental information requires full model retraining, making it inefficient and unsuitable for dynamic content environments.

In contrast, the CRG pipeline offers a modular architecture that separates classification, retrieval, and generation into discrete components. This separation enables more efficient updates—new content can be added directly to the retrieval database without retraining. CRG also supports targeted tuning of individual components such as the classifier or generator, improving maintainability and adaptability for evolving informational needs.

Despite its flexibility, CRG faces its own challenges, particularly in ensuring the fluency and coherence of generated responses. The system still requires improvements in response generation fidelity and optimization for real-time usage.

Future Work

To enhance the robustness and usability of the tour guide system, several directions are proposed for future research: Expand the custom dataset with more diverse and detailed question-answer pairs to improve both DirectLLM and CRG coverage. Finish development of the CRG generation step, ensuring it reliably incorporates retrieved answers and poses engaging follow-up questions. Further fine-tune the DirectLLM models, particularly BART and FLAN-T5, using the expanded dataset to improve their fluency and domain-specific relevance. Improve classification accuracy in the CRG pipeline, which involves further fine-tuning. Optimize code efficiency in each CRG stage, especially in retrieval and generation, to meet real-time requirements on embedded hardware. Explore hybrid fusion models that combine the strengths of both pipelines—such as using CRG for structured queries and DirectLLM for free-form conversational fallback—to balance performance and flexibility. Together, these enhancements aim to advance the system toward a fully autonomous, scalable, and engaging AI tour guide platform deployable across academic institutions.

References

[1] Mangdang, "Mangdang Store," Available: https://mangdang.store

[2] M. Rajkomar et al., "Scalable and accurate deep learning with electronic health records," npj Digital Medicine, vol. 1, no. 18, 2018.

[3] T. Dettmers et al., "Sparse fine-tuning for efficient deployment of large-scale language models," NeurIPS Conference Proceedings, 2022.

[4] Z. Jiang et al., "Efficient deep learning inference on edge devices: Challenges and techniques," ACM Computing Surveys, vol. 55, no. 6, 2023.

[5] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv:2307.09288, 2023.

[6] R. Rajpurkar, "SQuAD: The Stanford Question Answering Dataset," SQuAD Explorer.

[7] Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, et al. "Scaling Instruction-Finetuned Language Models." arXiv (2022). https://doi.org/10.48550/arXiv.2210.11416.

[8] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," CoRR, vol. abs/1910.13461, 2019. [Online]. Available: http://arxiv.org/abs/1910.13461.

[9] H. Xu, M. Scales, A. Nguyen, R. Ritter, C. Raffel, K. Shankar, S. A. Saurous, and Q. V. Le, "Deep Evolutionary Reinforcement Learning," arXiv preprint arXiv:1910.10683, 2019. [Online]. Available: https://arxiv.org/pdf/1910.10683.

[10] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," arXiv preprint arXiv:1910.13461, 2019. [Online]. Available: https://arxiv.org/pdf/1910.13461. [Accessed: Dec. 17, 2024].

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[12] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2020. [Online]. Available: https://arxiv.org/abs/1910.01108

[13] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017.

[14] N. Reimers and I. Gurevych, "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020. [Online]. Available: https://arxiv.org/abs/2004.09813

[15] Hugging Face, "all-MiniLM-L6-v2," [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. [Accessed: 01-Apr-2025].