# Development of Microcontroller-Based AI Robot Using Custom Language Models

## AI Ambassador - AIIoT 2025

**Ian Jackson - 05/30/2025**

# Personal Introduction - Ian Jackson

- Education
  - M.S. Electrical Engineering (2025)
    West Virginia University
  - B.S. Computer Engineering (2024)
    West Virginia University
- Work Experience
  - SoC PD Intern (Summer 2023 & 2024)
    Apple
  - Application Developer Intern (Summer 2022)
    ManTech Int.
- Fun Fact: I’m colorblind

# Introduction

- University recruitment offices seek innovative ways to connect with prospective students
- Develop an interactive AI-powered tour guide for WVU's Lane Department of Computer Science and Electrical Engineering (LCSEE)
  - Robotic interface equipped with microphone and speaker
  - Achieve real-time natural conversation to aid in any questions during a tour
- Showcases department offerings, student life, research opportunities, and general information

**Lane Department Of Computer Science and Electrical Engineering**

# Related Work

- Fine-Tuning LLMs for Domain-Specific Tasks
  - Fine-tuning LLMs outperform generic models in specialized domains such as legal services and healthcare [1]
  - However, LLMs tend to struggle on edge devices [2]
- Deploying AI Models on Resource-Constrained Devices
  - Low memory and power availability on edge-devices poses a challenge when running AI models
  - Prior work explores optimization techniques [3] and advancements in SLMs [4] show promising results for edge-devices

# System Overview

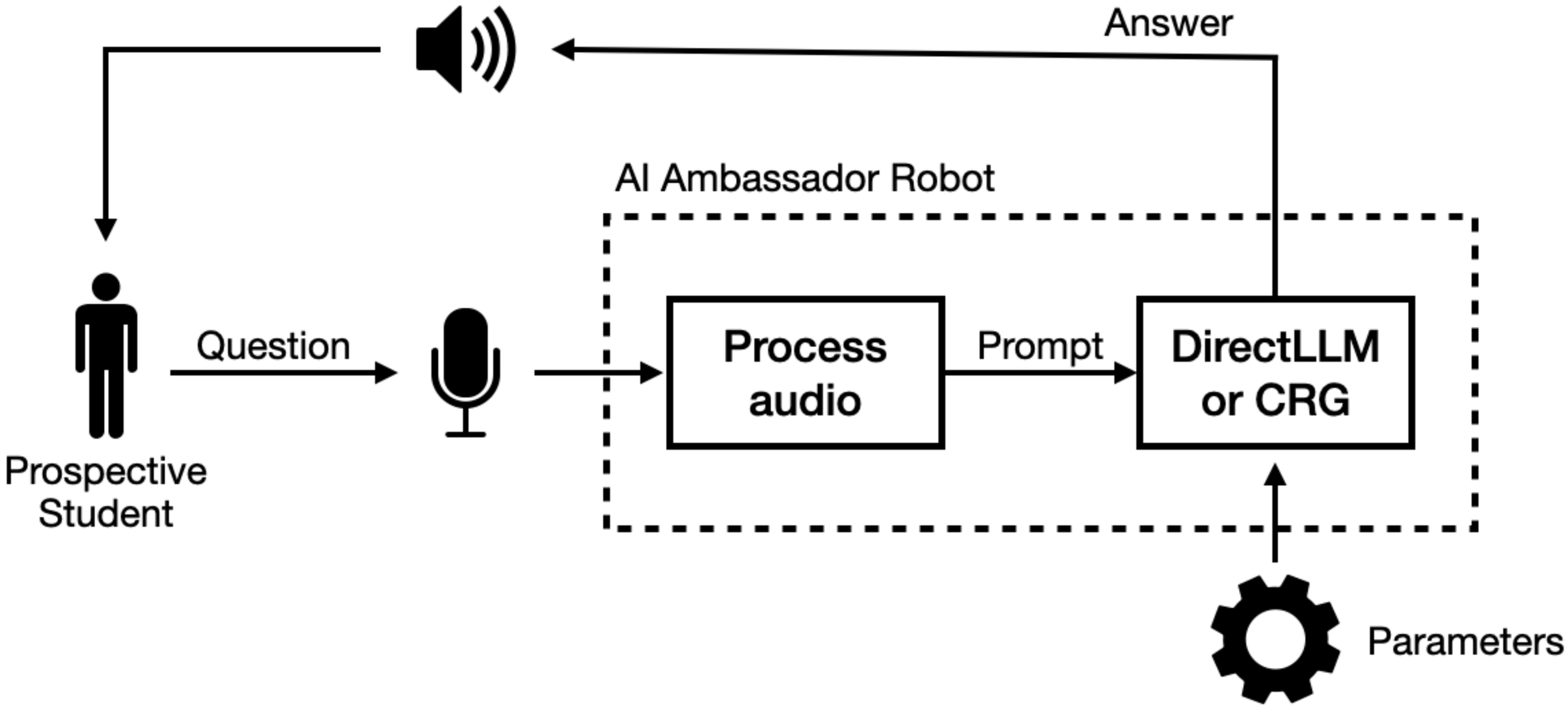


Figure 1. AI Tour Guide System Overview

# System Overview

- Chosen medium: MangDang Mini Pupper [5]
  - Powered by Raspberry Pi 4B (8GB)
  - Speech Recognition: Uberi SpeechRecognition
  - TTS: Flite Engine from CMU
- Other kit-based and custom robotics models considered
- Mini Pupper chosen based on price and ease of use

Figure 2. MangDang Mini Pupper
Adapted from [5]

# Custom Dataset Development

- Due to niche application, custom dataset needed to be developed
- 750 Question-Answer pairs about LCSEE generated in SQuAD format [6]
- QA pair created and verified by human
  - Augmented using ChatGPT
  - Verified for accurate information

Table 1. Custom LCSEE 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** |

# Approach I - DirectLLM

- First approach, fine-tune pre-trained SLMs/ LLMs on custom dataset
- Models considered
  - FLAN-T5 (Google) [7] - 77 million params
  - BART (Facebook) [8] - 139 million params
- Pros: simple, natural conversational abilities
- Cons: computationally expensive, retrain upon new/updated information

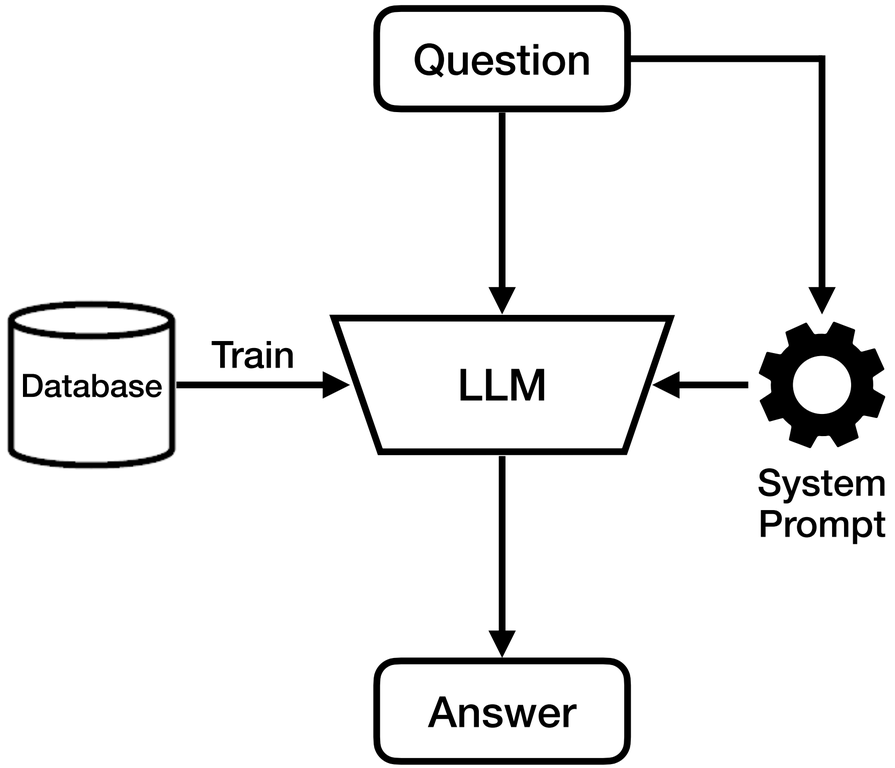


Figure 3. DirectLLM Pipeline

# Approach II - Classify Retrieve Generate (CRG)

- User's question passes through a three-stage pipeline
  - *Classify* the user's question into one of the 11 categories
  - *Retrieve* the best answer from the database using question class and extracted information
  - *Generate* a natural, conversational answer based on retrieved information
- Addresses DirectLLM's retrain issue, information can be updated in database
- Modular design: each stage can have multiple implementations

# CRG Pipeline

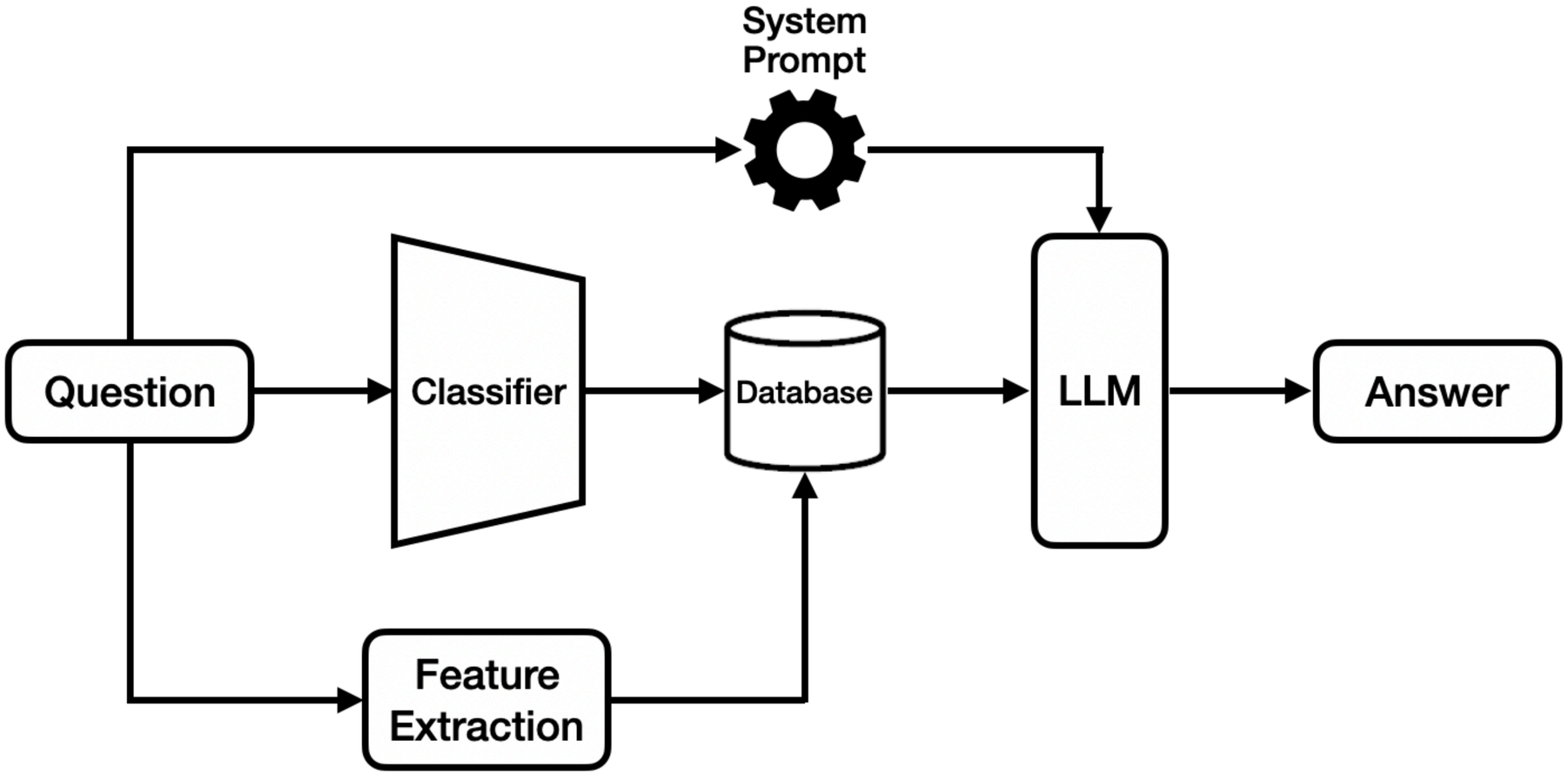


Figure 4. Classify Retrieve Generate (CRG) Pipeline

# Step 1: Classify

- Transform user question into question type using ML classification techniques
- Implementations considered
  - Logistic Regression (LR): input features using TF-IDF, cross-entropy loss
  - Support Vector Machine (SVM): input features using TF-IDF, hinge loss
  - Fine-tuned transformers for classification: Use BERT or DistilBERT fine-tuned on dataset

# Step 1: Classify

Table 2. Summary of classification methods used

| | **LR** | **SVM** | **BERT** | **DistilBERT** |
|---|---|---|---|---|
| **Pros** | - Fast<br>- Good for small datasets | - Higher accuracy<br>- Strong generalization | - Very high accuracy<br>- Context-aware | - Good trade-off of speed and accuracy |
| **Cons** | - Lower accuracy on complex inputs | - Ignores context<br>- No semantic understanding | - Slow inference<br>- Large model size | - Slightly reduced accuracy |
| **Edge Suitability** | High | High | Low | Medium |

# Step 1b: Feature Extraction

- Retrieval step not effective with just the question class
- Need some information from the question for effective retrieval
- Implementations considered:
  - Name entity recognition (NER): utilize spaCy's NER model to extract keywords from question
  - TF-IDF keyword extraction: extract high-valued keywords using TF-IDF
  - Sentence embedding: convert question into 384-dimensional vector using *all-MiniLM-L6-v2*

# Step 1b: Feature Extraction

Table 3. Summary of extraction methods used

| | NER | TF-IDF | Sentence Embeddings |
|---|---|---|---|
| **Pros** | - Domain-agnostic<br>- Fast | - Lightweight<br>- Interpretable | - Captures semantics<br>- Handles varied phrasing |
| **Cons** | - Misses abstract academic terms | - No synonym/semantic handling | - Higher compute/memory cost |
| **Edge Suitability** | High | High | Medium |

# Step 2: Retrieve

- Best matching answer from database retrieved using question class and extracted information
- Methods considered:
  - Exact keyword intersection (EKI): score based on size of intersection of user's question keywords and each question's keywords in database (1)
  - Jaccard similarity: simular to EKI, but accounts for shared and distinct elements (2)
  - Jaccard + EKI (JEKI): weighted sum of Jaccard and EKI scores (3)
  - Cosine similarity: compare embedded user question and database question in their 384-dimensional vector space (4)

$$\text{EKI}(A, B_i) = |A \cap B_i| \quad (1)$$

$$\text{Jac}(A, B_i) = \frac{|A \cap B_i|}{|A \cup B_i|} \quad (2)$$

$$\text{JEKI}(A, B_i) = \alpha\text{EKI}(A, B_i) + \beta\text{Jac}(A, B_i) \quad (3)$$

$$\cos(\theta) = \frac{\vec{q} \cdot \vec{q}_i}{||\vec{q}|| \cdot ||\vec{q}_i||} \quad (4)$$

# Step 2: Retrieve

Table 4. Summary of retrieval methods used

| | **EKI** | **Jaccard** | **JEKI** | **Cosine Similarity** |
|---|---|---|---|---|
| **Pros** | - Simple<br>- Fast | - Accounts for partial matches | - Balances raw and normalized | - Best semantic matching |
| **Cons** | - Needs exact match<br>- Low recall | - Still keyword-based<br>- No context | - Requires tuning $\lambda$ | - Most compute-intensive |
| **Edge Suitability** | High | High | Medium | Medium |

# Step 3: Generate

- Takes retrieved answer and rephrases into more natural and engaging conversation by the use of language models
- Inputs: retrieved answer and system prompt (includes user question)
- Models considered: Flan-T5-Small, TinyLlama, GPT-Neo, Mistral-7B
- *Work in progress*
  - Inconsistent behavior in prompt adherence
  - Failure to include retrieved answer

# Step 3: Generate

Table 5. Summary of generation methods explored

| | **FLAN-T5-Small** | **TinyLlama** | **GPT-Neo** | **Mistral-7B** |
|---|---|---|---|---|
| **Pros** | - Lightweight<br>- Open-source | - Coherent output<br>- Promising | - Baseline benchmark | - High-quality responses |
| **Cons** | - Prompt adherence issues | - Needs fine-tuning | - Medium resource use<br>- Inconsistent | - Too large for edge deployment |
| **Edge Suitability** | High | High | Medium | Low |

# Evaluation - DirectLLM

- To evaluate performance of Flan-T5 and BART, four metrics used.
  - Answer accuracy: measured by manual inspection of answer
  - The Bilingual Evaluation Understudy (BLEU) score: measures quality of text generated compared to human-provided reference texts

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

  - Average response time throughout test set
  - Average memory usage throughout test set

# Evaluation - CRG

- Multiple implementations at each stage yields 100+ CRG combinations, chose best three
- Evaluated using four metrics:
  - Classification accuracy
  - Retrieval score (RS): accuracy measure for retrieval step
  - Average response time
  - Average memory usage

Table 6. Best performing CRG models used for evaluation

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

# Results

Table 7. 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 8. 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 |

# Discussion

- Lessons learned
  - Primary DirectLLM challenge is dataset bottleneck, need for large and diverse dataset
  - Any department updates require retraining, CRG solves by segmentation
- Trade offs
  - Given DirectLLM challenges, offers fluent responses and contextual understanding
  - CRG showed better response time and memory usage, lacks fluency and contextual awareness
- Practical considerations
  - For microcontroller based implementations, response time is critical constraint
  - CRG’s deterministic logic enhance reliability

# Conclusion

- Developed framework for a LCSE robotic tour guide using two pipelines; DirectLLM and CRG
  - DirectLLM showed natural and fluent responses, faces data bottleneck and retraining upon new information
  - CRG provided modular architecture with fast response times but lacked the natural conversational output
- Future work
  - Expansion of dataset (in LCSEE domain or university wide domain)
  - Finish generation step of CRG
  - Improvement of each stage of CRG pipeline, ensuring accurate and efficient classification/ retrieval
  - Explore hybrid DirectLLM and CRG model

# References

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

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

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

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

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

[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.

# Questions?