
**CSC464: Deep Lear.& Nat. Lang Process.**

**Dr. Seifedine Kadry**

**Enhanced Chatbot System with Prompt Refinement and RAG**

**Mohammad Hijazi**
**202207795**

**April 26, 2025**

**Abstract**

This project develops an enhanced chatbot system that combines two techniques: a prompt refinement model using GPT-2 and a Retrieval-Augmented Generation (RAG) system. The prompt refinement component improves poorly written or ambiguous user queries, which are then processed by the RAG system to provide accurate, context-aware responses based on specific knowledge sources. Evaluation metrics show promising results for the prompt refinement model, with a high BERTScore and room for improvement in BLEU, ROUGE, and METEOR scores.

**Keywords:** prompt refinement, retrieval-augmented generation, GPT-2, chatbot, natural language processing

## 1. Project Proposal

### Objective

This project aims to develop an enhanced chatbot system that incorporates prompt refinement capabilities and retrieval-augmented generation to provide more accurate and contextually appropriate responses. The specific goals are:

1. To build a model that improves poorly structured or ambiguous user prompts
2. To develop a knowledge grounded RAG system that can provide factual responses based on specific sources
3. To integrate these components into a unified interface for enhanced user experience

This problem is significant because chatbots often struggle with unclear user queries and may generate hallucinations or incorrect information when not properly grounded in reliable knowledge sources.

### Dataset

For the prompt refinement model, we utilized the "Refined_Prompts" dataset from Hugging Face (https://huggingface.co/datasets/yaswanth-iitkgp/Refined_Prompts), which contains pairs of raw (ambiguous or poorly structured) prompts and their refined corresponding parts. The dataset consists of over 2-thousand examples, which were filtered to include only English language entries. For the RAG system, we used a PDF document containing information (e.g. photosynthesis, medical documents, etc...) as our knowledge source, which was processed into chunks for retrieval purposes.

### Expected Challenges

We anticipated several challenges:

1. Limited data quality in the Refined_Prompts dataset
2. Need for effective preprocessing and filtering
3. Balancing model size with performance for GPT-2 implementation
4. Creating effective retrieval mechanisms for relevant context from documents
5. Integrating the two systems seamlessly

To address these challenges, we planned to implement thorough data cleaning and filtering, utilize pretrained language models to leverage transfer learning, and carefully design the system architecture to enable effective integration.

## 2. Data Preparation and Preprocessing

### Data Cleaning

For the prompt refinement model, we performed several cleaning steps on the Refined_Prompts dataset:

1. Used language detection (langdetect) to filter for English-only examples, as the dataset contained multilingual content
2. Removed examples with missing or corrupted text
3. Filtered out extremely short or long examples that would not be useful for training

The filtered dataset was reduced from the original size to a high-quality subset that contained only well-formed English prompt pairs, ensuring better model training.

### Data Augmentation

No explicit data augmentation techniques were applied for this project, as the refined prompts dataset already provided sufficient variation. For the RAG system, the document was processed using LlamaIndex's SimpleDirectoryReader, which automatically handles document chunking and preparation.

### Normalization/Standardization

For the prompt refinement model, we utilized the GPT-2 tokenizer's built-in normalization procedures. Input prompts were formatted consistently with a specific template:

"Refine the following prompt in English: {original_prompt}\n\nRefined prompt: {refined_prompt}".

This standardization ensured consistent inputs during both training and inference.

### Data Splitting

We split the dataset into training (60%), validation (20%), and test (20%) sets using sklearn's train_test_split with a fixed random seed (42) for reproducibility.

## 3. Model Architecture and Design

### Model Choice

For prompt refinement, we initially selected the T5-base model due to its specialized design for text transformation tasks. However, we had computational efficiency challenges as the model was very large. We then tested T5-small, which proved to be more computationally efficient, but despite filtering our dataset to remove non-English entries, it translated prompts to different languages in some instances (the model is multilingual). These limitations led us to adopt GPT-2, which offered an ideal solution

with its transformer decoder architecture. GPT-2's proven capabilities in language generation, contextual understanding, and coherent text production made it well-suited for our goal of improving ambiguous or poorly written prompts.

For the RAG system, we utilized a combination of:

1. OpenAI Embeddings (text-embedding-3-large) for document vector representation
2. VectorStoreIndex from LlamaIndex for efficient document retrieval
3. OpenAI's GPT-4o as the large language model for response generation

This combined architecture allows for accurate document retrieval followed by contextually appropriate response generation.

**Layer Design**

The prompt refinement model uses the standard GPT-2 architecture:

- Multiple transformer decoder blocks with multi-head self-attention
- Feed-forward neural networks in each block
- Layer normalization and residual connections throughout
- Vocabulary embedding and positional encoding layers at the input
- Language modeling head (linear layer) at the output

The model utilizes the GELU activation function throughout, which has been shown to work well with transformer architectures. For the RAG system, we leveraged pre-built components from LlamaIndex and OpenAI.

**Model Complexity**

For the prompt refinement model, we chose the base GPT-2 model rather than larger variants to balance performance with computational efficiency. This decision was made considering:

1. The relatively specific nature of the prompt refinement task
2. The limited size of our training dataset
3. Computational constraints for training and deployment

The model is complex enough to capture linguistic patterns necessary for prompt improvement but not so large as to require excessive training resources.

**Pretrained Models**

We used pretrained models extensively:

1. For prompt refinement, we used the pretrained GPT-2 model from Hugging Face, fine-tuning it on our specific task
2. For the RAG system, we used OpenAI's text-embedding-3-large for embeddings and GPT-4o for response generation

Using these pretrained models allowed us to leverage transfer learning, where the models' pre-existing knowledge of language structure and semantics provided a strong foundation for our specific tasks.

## 4. Training Process

### Training Strategy

For the prompt refinement model, we used fine-tuning of the GPT-2 model on our dataset. The training process involved:

1. Formatting each example as a prompt-response pair
2. Using PyTorch Lightning for structured training
3. Implementing checkpointing to save the best models
4. Early stopping to prevent overfitting

We used a causal language modeling approach where the model learns to predict the next token given previous tokens, effectively teaching it to transform raw prompts into refined versions.

### Loss Function and Optimization

We used the standard language modeling loss (cross-entropy) for training the GPT-2 model, as it's well-suited for next-token prediction tasks. For optimization, we selected AdamW with the following parameters:

- Learning rate: 2e-5
- Weight decay: 0.01

A linear learning rate schedule with warmup was implemented to stabilize the early stages of training.

### Hyperparameter Tuning

We tuned several key hyperparameters:

- Learning rate: Tested values in the range [1e-5, 5e-5]

- Batch size: Tested 4, 8, and 16
- Number of epochs: Tested 3, 5, and 10
- Temperature for generation: Tested 0.6, 0.7, and 0.8

Final values were selected based on validation loss and assessment of generated refinements. The final settings were:

- Learning rate: 3e-5
- Batch size: 8
- Epochs: 5 (with early stopping)
- Temperature: 0.7

**Regularization**

To prevent overfitting, we implemented several regularization techniques:

1. Weight decay (0.01) in the AdamW optimizer
2. Early stopping with a patience of 3 epochs, monitoring validation loss
3. Dropout (inherited from the pretrained GPT-2 model)
4. During generation, we used techniques like top-p sampling (0.9) and no_repeat_ngram_size (2) to improve output diversity and quality

**5. Evaluation and Analysis**

**Evaluation Metrics**

We selected multiple metrics to comprehensively evaluate the prompt refinement model:

1. BLEU score: To measure the lexical overlap between predicted and reference refined prompts
2. ROUGE scores (ROUGE-1, ROUGE-2, ROUGE-L): To assess the overlap of n-grams and longest common subsequences
3. BERTScore: To capture semantic similarity
4. METEOR: To evaluate word-to-word alignments with (synonyms included)

These metrics were chosen to evaluate both lexical accuracy and semantic aspect of the refined prompts, covering different aspects of NLP evaluation.

**Validation and Test Results**

The prompt refinement model showed the following results on both validation and test sets:

Test Set Performance:

- BLEU: ~0.32
- ROUGE-1: ~0.48
- ROUGE-2: ~0.33
- ROUGE-L: ~0.45
- BERTScore F1: ~0.88
- METEOR: ~0.41

This model produces outputs that are semantically similar to reference refinements from the dataset (high BERTScore) but differ significantly in exact wording and phrasing (low BLEU and ROUGE-2). The model appears to understand the meaning of prompts and refines them with different vocabulary and structure while preserving the semantic aspect.

The RAG system was evaluated qualitatively through example queries and responses, showing good contextual understanding and accurate information retrieval from the source document.

**Error Analysis**

Analysis of errors in the prompt refinement model revealed several patterns:

1. Occasionally missing certain keywords that the user might specifically want
2. creating very long or multi-part prompts (sometimes unnecessary)

**Model Comparison**

**We experimented with different configurations for the prompt refinement model:**

1. T5-base vs. T5-small (both encountered issues: T5-base had computational inefficiency while T5-small translated prompts to different languages despite dataset filtering)
2. GPT-2 (base) vs. GPT-2-medium
3. Different generation parameters (temperature, top-p values)

**The base GPT-2 model with carefully tuned generation parameters provided the best balance of performance and efficiency.**

## 6. Interpretability and Insights

### Model Interpretability

To understand how the prompt refinement model operates, we analyzed examples where the model made significant improvements:

1. **Clarification of ambiguity**: The model effectively identified and resolved ambiguous phrasings in original prompts
2. **Grammar correction**: Consistent improvement of grammatical errors and structural issues
3. **Addition of context**: When appropriate, the model added relevant context that made prompts more specific

For the RAG system, we examined the retrieved chunks for several queries, finding that the retriever effectively matched semantic content and not just keywords.

### Insights and Observations

Key insights from the project:

1. The combined system (prompt refinement + RAG) provides better responses than either component alone
2. Prompt refinement is particularly valuable for users who struggle to articulate clear prompts
3. The OpenAI embedding model shows strong performance in capturing the semantic meaning of both documents and queries
4. Generation parameters (temperature, top-p) significantly impact output quality and should be carefully tuned
5. The integrated Gradio interface makes the system accessible to non-technical users

## 7. Conclusion and Future Directions

### Summary of Contributions

This project successfully developed:

1. A prompt refinement model based on GPT-2 that effectively improves unclear or poorly written prompts
2. A RAG system that retrieves relevant information from documents and generates contextually appropriate responses

3. An integrated interface that combines both components, allowing users to either refine their prompts or directly submit them for responses

The project demonstrates how combining these techniques can create a more effective chatbot system that addresses common limitations of existing chatbots.

**Limitations**

Despite the promising results, several limitations were identified:

1. The prompt refinement model was trained on a limited dataset and may not generalize to all domains
2. The RAG system's performance is heavily dependent on the quality and coverage of the documents provided
3. Computational requirements for the full system are high, potentially limiting deployment options
4. The system currently handles only English language inputs and outputs

**8. Documentation and Code**

The code is organized into two main components:

1. The prompt refinement model (GPT-2 based)
2. The RAG system (using LlamaIndex and OpenAI)

Both components are integrated in a Gradio interface that allows users to either refine their prompts before submission or directly submit them for responses.