## Architecture Design Document

### FreeText Generation for Content Creators

**Written By:** Dharam Chopda  
**Email:** chopdadharam143@gmail.com  
**Date:** 03-11-2024

### **Document Change Control Record**

| **Version** | **Date** | **Author** | **Comments** |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.0 | 03-11-2024 | Dharam Chopda | Initial version |  |  |

### **Table of Contents**

1. **Introduction**
2. **System Overview**
3. **Architecture Overview**
4. **Module Design**
5. **Data Flow and Sequence Diagrams**
6. **Data Structures**
7. **Technology Stack**
8. **Non-Functional Requirements**
9. **Conclusion**

### **1. Introduction**

The purpose of this Architecture Design Document is to describe the structural and technical details of the **FreeText Generation Project**. This project aims to create a system that generates sentences based on a set of concepts provided, using the meta-llama/Llama-2-7b-chat-hf model.

### **2. System Overview**

The FreeText Generation system is designed to help content creators generate freeform text based on input concepts. The model will use a pre-trained large language model to produce coherent sentences containing specified keywords or concepts in a natural way. This system comprises several modules for data processing, model management, text generation, and output evaluation.

### **3. Architecture Overview**

The architecture follows a **modular, layered structure** designed to handle:

* Input data preprocessing
* Model loading and text generation
* Validation of generated text
* Performance and quality evaluation

This architecture is flexible for future upgrades, such as changing the model or expanding to more complex text generation requirements.

### **4. Module Design**

Each component in the architecture is defined below, with its purpose, input, output, and core functions.

#### **4.1 Data Preprocessing Module**

* **Purpose**: Prepare and clean the input concepts, then format them into prompts suitable for the model.
* **Inputs**: Raw concept data (e.g., ['catch\_V', 'dog\_N'])
* **Outputs**: Cleaned concept lists, formatted prompts (e.g., "Create a sentence using these concepts: catch, dog.")
* **Functions**:
  + ConceptCleaner: Strips tags and cleans input
  + PromptCreator: Generates structured prompts for model input

#### **4.2 Model Management Module**

* **Purpose**: Load and manage the meta-llama/Llama-2-7b-chat-hf model.
* **Inputs**: Formatted prompt from the preprocessing module
* **Outputs**: Generated sentence that incorporates the specified concepts
* **Functions**:
  + LlamaModelLoader: Loads the pre-trained model and tokenizer
  + PromptFormatter: Structures prompts into an optimal format for the model

#### **4.3 Text Generation Module**

* **Purpose**: Generate sentences based on model inputs and validate output quality.
* **Inputs**: Formatted prompt
* **Outputs**: Generated sentence that includes all concepts
* **Functions**:
  + TextGenerator: Passes prompts to the model’s generate function
  + KeywordValidator: Checks that all specified concepts are present in the generated text

#### **4.4 Evaluation Module**

* **Purpose**: Evaluate the quality of generated sentences using metrics like BERTSCORE.
* **Inputs**: Generated sentences, optional reference sentences
* **Outputs**: Evaluation scores (BERTSCORE)
* **Functions**:
  + Evaluator: Compares generated output against targets and computes scores

### **5. Data Flow and Sequence Diagrams**

#### **5.1 Sequence Diagram: Process Flow**

**plaintext**

1. User Input (Concepts) --> Data Preprocessing Module

- Cleans and formats data into prompts.

2. Data Preprocessing Module --> Model Management Module

- Model is loaded and configured.

3. Model Management Module --> Text Generation Module

- Generates sentence based on input.

4. Text Generation Module --> Evaluation Module

- Quality of output is measured.

#### **5.2 Data Flow Diagram**

* **Data Preprocessing** -> **Model Loading** -> **Text Generation** -> **Validation and Evaluation**

### **6. Data Structures**

**Inputs**:

* + **Concepts**: List of concept strings, stripped of tags (e.g., ["catch", "dog"])
  + **Prompts**: Structured input strings for the model (e.g., "Generate a sentence using these words: catch, dog.")

**Outputs**:

* + **Generated Sentence**: Natural language output containing all specified concepts (e.g., "The dog jumps to catch the ball.")

### **7. Technology Stack**

* **Programming Language**: Python
* **Deep Learning Library**: transformers library by Hugging Face for model loading and text generation
* **Evaluation Libraries**: BERTSCORE
* **Model**: meta-llama/Llama-2-7b-chat-hf
* **Data Processing Libraries**: pandas for data management, re for text cleaning

### **8. Non-Functional Requirements**

* **Performance**: Should generate sentences within 1-2 seconds.
* **Scalability**: Able to scale with larger datasets or complex models if needed.
* **Error Handling**:
  + **Out-of-Memory**: Manage GPU memory by batching data.
  + **Concept Validation**: Regenerate output if required concepts are missing.
* **Usability**: Clear input/output interface for ease of use.

### **9. Conclusion**

This Architecture Design Document outlines the structural components, data flow, and key modules required for the **FreeText Generation Project**. By implementing this architecture, the project will produce natural language sentences based on input concepts, achieving high quality and relevance. The modularity and flexibility of this design also allow for easy adaptation to additional features or model upgrades.