Report: Bhashathon_ASR Output Correction

0. Team Name and Details of Team Members

Team name: Global Optima

No	Name	Email
1	Disha Timbadiya	dishatimbadiya055@gmail.com
2	Om Soni	oms279400@gmail.com
3	Vrund Dobariya	vrund3626@gmail.com

1. Name of the Problem Statement and Languages Participated

Problem Statement: IV ASR Output Correction

Language: English, Hindi, Malayalam.

2. Dataset Description

For training (T5, Flan-T5, smolLM-v2):

1. **Dataset-1 Common Voice Dataset** (Delta 20,19,18, and 17 releases)

   Source: Mozilla
   License: CC BY 4.0 (Creative Commons Attribution 4.0)
   Usage: This dataset can be used for commercial and non-commercial purposes with proper attribution.
   Citation: Mozilla, "Common Voice Dataset", https://commonvoice.mozilla.org
   Modifications: None

2. The **datasets** provided with the problem statement.

Above mentioned datasets were **used to generate transcripts** for audio recordings with the help of given ASR models:

1. `OpenAI-whisper-tiny`
2. `Facebook s2t small libre-speech ASR`

Dataset links:

Generated with S2T-small [1935 samples]
Generated with whisper-tiny [1446 samples]

For evaluation:
Used dataset provided by the Hackathon team.

3. Preprocessing Details

- Sentences are **converted to lowercase(for English)** and **trimmed** to remove unnecessary spaces.
- Sentences with **more than 2 unmatched words** with their generated transcripts were **omitted**.
- Removed transcripts generated in **other languages**.

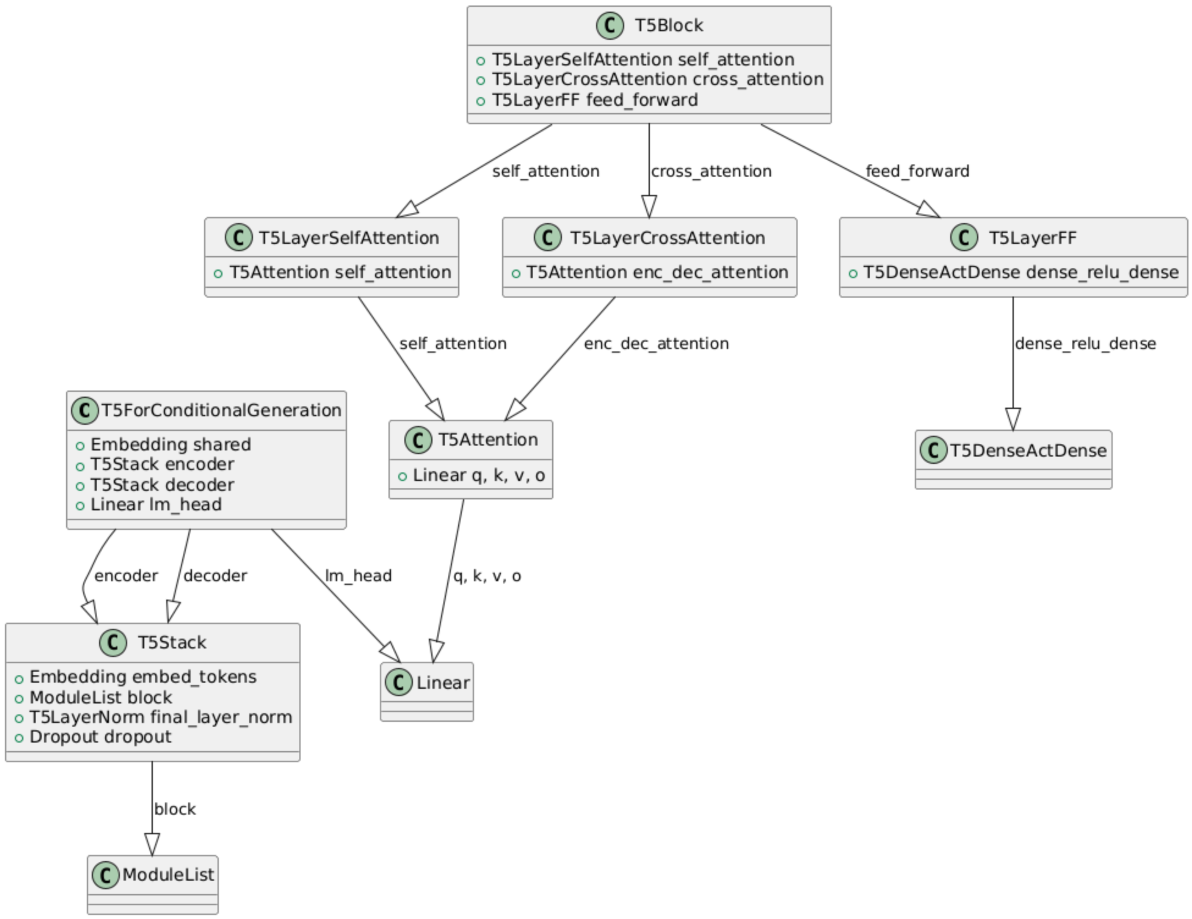
4. Model Architecture

Experiment 1: Fine-Tuning Transformer Model for ASR Correction

1. **Using T5-small Transformer Model:**

- **Description:** The pre-trained `t5-small` model is fine-tuned for ASR output correction. Input sentences are tokenized using the T5 tokenizer, and the tokenized input, including `input_ids` and `attention_mask`, is passed through the model's encoder-decoder transformer architecture. The model learns to correct ASR-generated errors by leveraging self-attention and cross-attention mechanisms. Training is performed using cross-entropy loss, and weights are optimized using the Adam optimizer. The final predictions are generated by the decoder, producing corrected text.

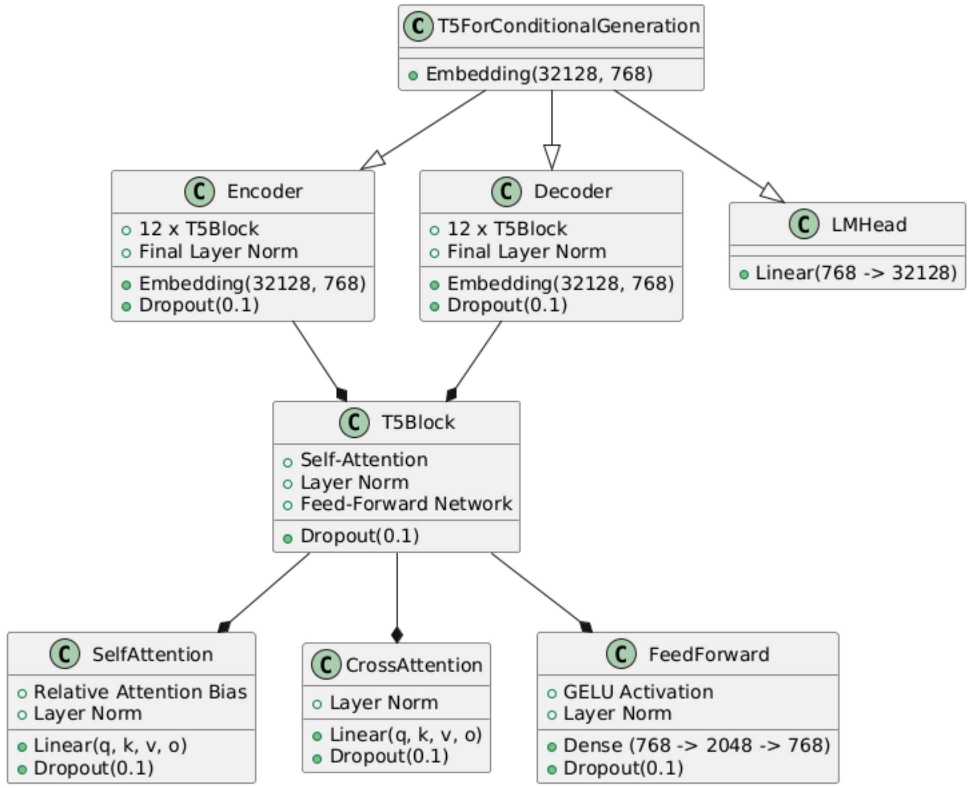
- **Architecture Diagram:**



- **Base Model Variant:** T5-Small is a lightweight version of the Text-to-Text Transfer Transformer (T5). It has 60 million parameters and follows a unified text-to-text framework, making it adaptable to various NLP tasks, including ASR error correction, summarization, and translation. The model was pre-trained on the Colossal Clean Crawled Corpus (C4) and fine-tuned on multiple supervised datasets covering sentiment analysis, paraphrasing, question answering, and more.

2. Using Flan-T5-base:

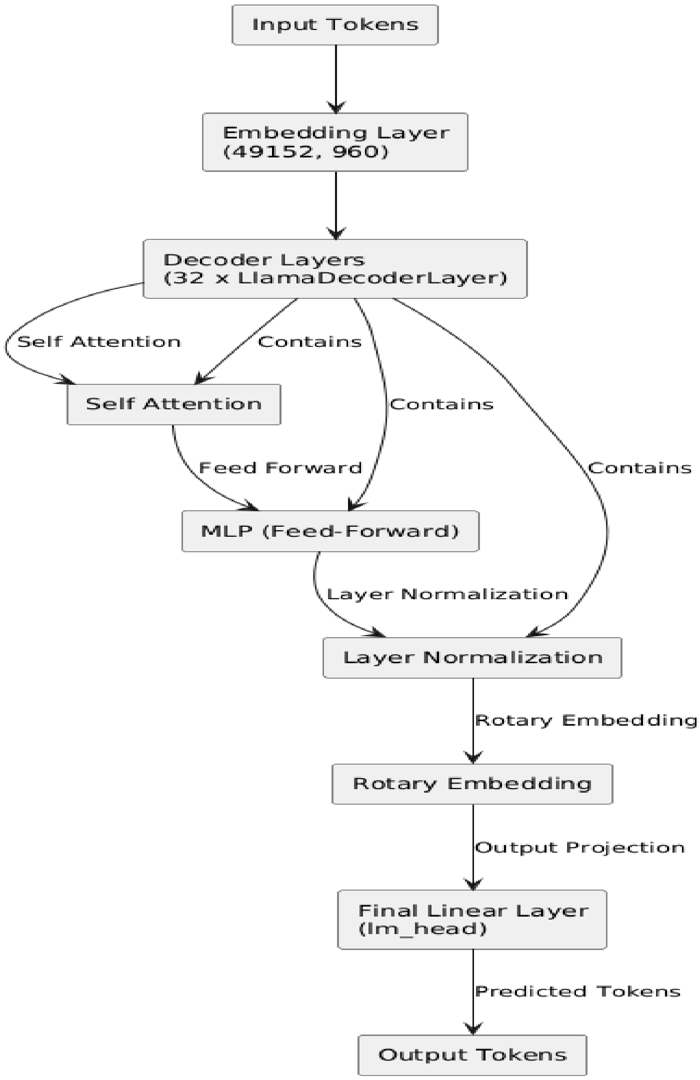
- **Description:** The pre-trained `flan-t5-base` model is fine-tuned for ASR output correction. The input text is tokenized using the T5 tokenizer and processed through the encoder-decoder architecture. The encoder extracts contextual representations, while the decoder generates corrected text. The model is trained with cross-entropy loss and optimized using Adam, refining its ability to correct ASR-generated errors.
- **Architecture Diagram:**



- **Base Model Variant:** Flan-T5-Base is an enhanced variant of the T5 model, fine-tuned using instruction-based learning for better generalization across NLP tasks. It retains the text-to-text framework of the original T5 but is optimized for zero-shot and few-shot learning. Pretrained on diverse instruction-following datasets, Flan-T5-Base improves on standard T5 by incorporating additional reasoning, summarization, and comprehension tasks. It has ~250 million parameters, making it a balanced choice between performance and efficiency for ASR correction tasks.

Experiment 2: Fine Tuning Small Language Models for ASR Correction

- **Description**: Pre-trained checkpoint of SmolLM2-360M is fine-tuned for ASR output correction. Input sentences are tokenized, and the tokens, along with **`input_ids`** and **`attention_mask`**, are passed through the model's transformer architecture. This includes **self-attention layers** and **feed-forward networks**. The model is trained using **cross-entropy loss** and optimized with the **Adam optimizer** to adjust weights. Through fine-tuning, the model learns to correct ASR output and generate more accurate text. The final predictions are made via a **linear layer**, projecting the internal representation to the vocabulary space.
- **Architecture Diagram:**

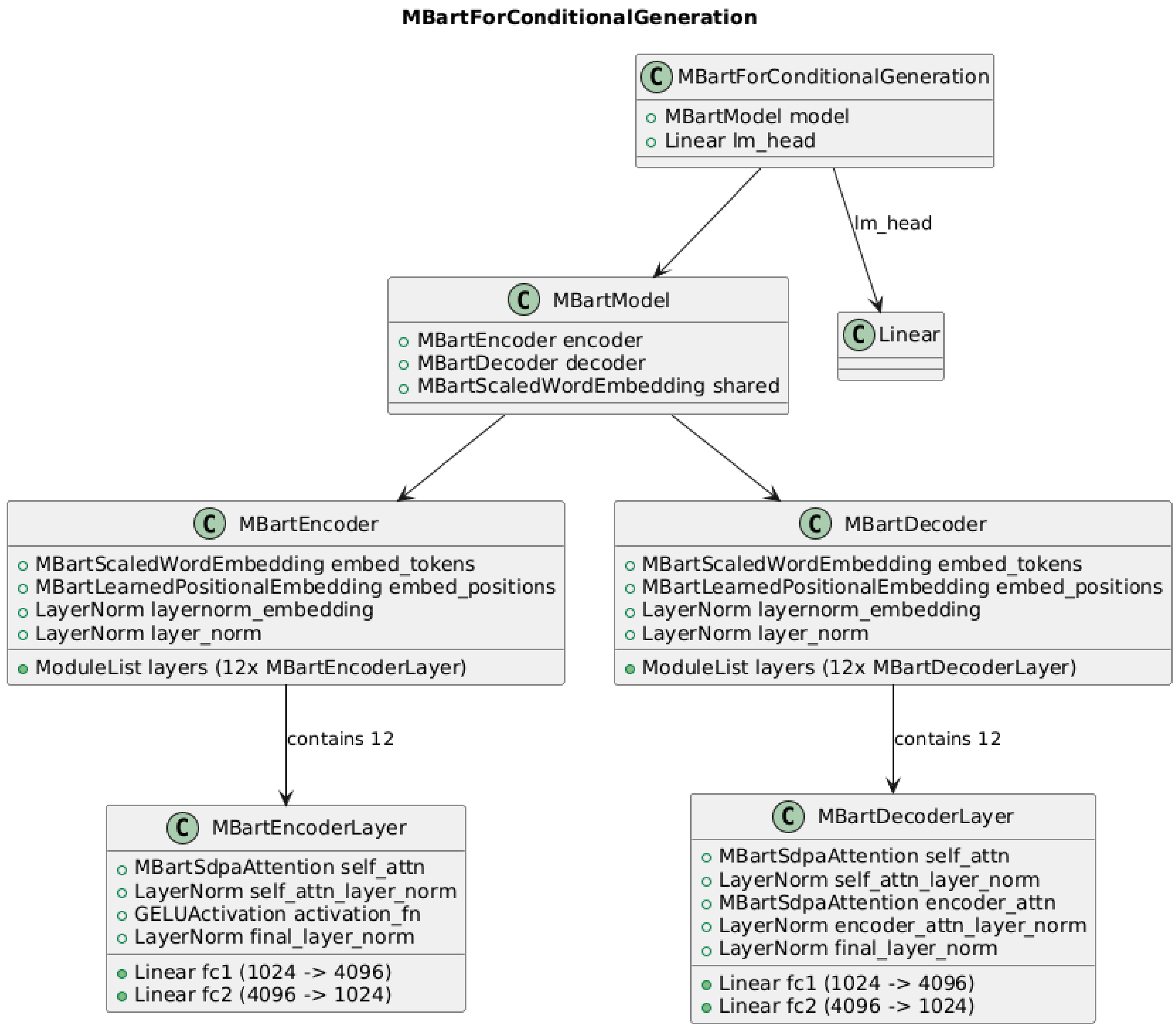


- **Base Model Variant:** [SmolLM2-360M] - SmolLM2 is a family of compact language models, he 360M model was trained on 4 trillion tokens using a diverse dataset combination: FineWeb-Edu, DCLM

Experiment 3: Fine Tuning facebook/mbart-large-50 Model for ASR Correction

- **Description:** The pre-trained mBART model is fine-tuned for ASR output correction. The input text is tokenized using the mBART tokenizer and processed through the encoder-decoder architecture. The encoder captures contextual representations, while the decoder generates corrected text. The model is trained using cross-entropy loss and optimized with Adam, enhancing its ability to correct ASR-generated errors across multiple languages.

- Used for multiple languages: English, Hindi, Malayalam.
- **Architecture Diagram:**



- **Base Model Variant:** mBART-50 is a multilingual sequence-to-sequence model with 610M parameters, pre-trained using the Multilingual Denoising Pretraining objective. It supports 50 languages and is primarily used for machine translation and text generation tasks.

5. Training Approach (If applicable)

- Training setup:
  - Environment: Google Colab
  - Hardware: T4-GPU
  - Number of GPUs: 1
- Hyper Parameters:
  - Learning rates: 2e-5, 5e-5

- Optimizer: Adam optimizer
- Data Augmentation Techniques:
  - Lower casing
  - Removing leading and trailing whitespaces
- Loss function and evaluation metrics:
  - Loss function: Cross Entropy Loss
  - Evaluation metrics: WER (Word Error Rate), CER (Character Error Rate), SER (Sentence Error Rate).
- Transfer Learning
  - Base models experimented : t5-small, flan-t5-base, SmolLM2-360M, facebook/mbart-large-50
  - Taken pre-trained checkpoints of the above model and retrained them for 3 to 4 epochs for each model on a custom dataset for ASR correction.

6. Inference, Performance, and Error Analysis

Colab NoteBooks:

1. **mBART fine-tuning for English**
2. **mBART_fine-tuning_for_hindi**
3. **mBART_fine-tuning_for_Malayalam**
4. **T5-Small fine-tuning For English - Experiment**
5. **FlanT5- base fine tuning for English - Experiment**
6. **Using LLM for English - Experiment**

- Report key evaluation metrics: **WER , CER, SER.**

Performance Metrics:

For English Language:

Metrics	Fine Tuned LLM Smollm-v2	Fine Tuned Transformer t5-small	Fine Tuned Transformer google/flan-t5-base	Fine Tuned facebook/mbart-large-50
WER	21%	14.94%	11.21%	5.80%
CER	15%	64.7%	55.7%	3.21%
SER	90%	70.84%	58.26%	36.02%

Based on the evaluation metrics, the Fine-Tuned **facebook/mbart-large-50 model is finalized** as it achieves the lowest WER (5.80%), CER (3.21%), and SER (36.02%) **for English**, making it the best for ASR error correction.

For Hindi Language:

Metrics	Fine Tuned facebook/mbart-large-50
WER	13.11%
CER	9.81%
SER	40.44%

For Malayalam Language:

Metrics	Fine Tuned facebook/mbart-large-50
WER	5.85%
CER	1.27%
SER	25.31%

● Common Failure cases:

1. Errors in proper names and technical terms
   Example: "`MICROSO`" instead of "`MICROSOFT`".
2. Struggling with long or complex sentences
3. Missing punctuation
   Example: “wont” instead of “won’t”

● Post-Processing Steps:

- Convert text to lowercase for consistency.
- Remove unnecessary spaces.

7. Limitations of the Solution

- Might struggle with domain-specific contexts. for example, medical, technical, etc.
- When there is a drastic ambiguity in transcribed vs actual word, the model struggles to predict correctly.
- Larger parameter models like 1B, 1.5B, and 3B can perform better.

Challenges encountered during development:

- Computational cost for fine-tuning the LLM
- Colab has limits on GPU usage and RAM, leading to memory crashes and GPU limit exceed issues