# **Automatic Speech Recognition (ASR) Correction**

## **Overview**

This project focuses on correcting errors in Automatic Speech Recognition (ASR) outputs using three different approaches:

1. **T5-based ASR Correction (ASR\_T5)**: Uses a T5 model to refine ASR-generated text into grammatically correct sentences.
2. **FLAN-T5-based ASR Correction**: Implements the FLAN-T5 model for improving ASR transcription accuracy.
3. **Large Language Model (LLM) for ASR Correction**: Fine-tunes an LLM to correct ASR outputs.
4. Fine Tuning **facebook/mbart-large-50** Model for ASR Correction.

## **Prerequisites**

Ensure you have the following dependencies installed before running the code.

### **Libraries**

* torch
* transformers
* numpy
* pandas
* scikit-learn
* jiwer
* editdistance
* datasets (for FLAN-T5 approach)
* torchvision (for CUDA support)

### **Install the required libraries using:**

| pip install torch transformers numpy pandas scikit-learn jiwer editdistance datasets torchvision |
| --- |

## **Environment**

* **Google Colab** (recommended) or a local machine with a GPU
* **Python 3.7+**
* **T5 Model Variants**: t5-small or google/flan-t5-base
* **Transformer Base Model**: facebook/mbart-large-50
* **LLM Base Model**:[SmolLM2-360M](https://huggingface.co/HuggingFaceTB/SmolLM2-360M)
* **Google Drive** for dataset storage (if running on Colab)
* **CUDA-enabled GPU** (for faster training and inference)

## **Setup & Execution**

#### [**mBART fine-tuning for English**](https://colab.research.google.com/drive/1dVPlIbL8nI2Sg7xfAYYhFzTh1gih9BC7?usp=sharing)

#### [**mBART\_fine-tuning\_for\_hindi**](https://colab.research.google.com/drive/1ci8IRdbNA5JYJFvXkLug1wQL5fleuPzb?usp=sharing)

#### [**mBART\_fine-tuning\_for\_Malayalam**](https://colab.research.google.com/drive/19qdkRHR6mWUt5rfLoho6t3c4JLEQdz_K?authuser=1#scrollTo=ITyVl1Krjr5A)

#### [**T5-Small fine-tuning For English - Experiment**](https://colab.research.google.com/drive/1dVaOIyQsSWztNlQYqBF5psOBLyq3hJK6?usp=sharing)

#### [**FlanT5- base fine tuning for English - Experiment**](https://colab.research.google.com/drive/10atYVbci7_OiS1KhkUllO3pbPQjeQI-w?usp=sharing)

#### [**Using LLM for English - Experiment**](https://colab.research.google.com/drive/1To-Y-VEEPGmG4kHUA3ivvTdtFgYvLD8h?usp=sharing)

### **1. Running Notebooks**

* Open Google Colab or Jupyter Notebook.
* Upload the provided notebooks for each ASR correction method.
* Run the cells sequentially after setting up the paths.

### **2. Mount Google Drive (Colab Users)**

| from google.colab import drive drive.mount('/content/drive') |
| --- |

### **3. Define File Paths**

Modify file paths according to your dataset location:

For each method, the paths are given in respective Colab notebook. Modify them accordingly

**For Example:**

| FILE\_PATH='/content/drive/MyDrive/ASR\_English\_Trainining\_Data.xlsx' CUSTOM\_DS\_FILE\_PATH='/content/drive/MyDrive/asr\_correction\_dataset\_cleaned.csv' MODEL\_SAVE\_PATH='/content/drive/MyDrive/ASR\_T5\_Model/' |
| --- |

## **Model Evaluation**

The model performance is evaluated using:

* **Word Error Rate (WER): Measures the percentage of incorrect words.**
* **Character Error Rate (CER): Measures the percentage of incorrect characters.**
* **Sentence Error Rate (SER): Measures the percentage of incorrect sentences.**

## **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% |

## **Performing Inference**

**Use the trained model to correct ASR-generated text:**

### [**Trained Models**](https://drive.google.com/drive/folders/10_1eMaCQwqMU1ONVixwZ6qHBZwaHdaN5?usp=drive_link)**:**

<https://drive.google.com/drive/folders/10_1eMaCQwqMU1ONVixwZ6qHBZwaHdaN5?usp=drive_link>

### **Infer Model:**

#### **How to Load a Model?**

| import torch import torch.nn.functional as F from transformers import MBartForConditionalGeneration, MBart50TokenizerFast  # Define model path MODEL\_PATH = "/content/drive/MyDrive/ASR\_English\_MBart\_Model.pth" TOKENIZER\_VARIANT = "facebook/mbart-large-50"  # Set device device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")  # Load tokenizer tokenizer = MBart50TokenizerFast.from\_pretrained(TOKENIZER\_VARIANT, src\_lang="en\_XX")  # Load model model = MBartForConditionalGeneration.from\_pretrained(TOKENIZER\_VARIANT) model.load\_state\_dict(torch.load(MODEL\_PATH, map\_location=device)) model.to(device) model.eval()  # Inference function def infer(model, sentence):  # Tokenize input  tokenized\_sentence = tokenizer(sentence, return\_tensors="pt", padding=True, truncation=True, max\_length=512)  input\_ids = tokenized\_sentence["input\_ids"].to(device)  attention\_mask = tokenized\_sentence["attention\_mask"].to(device)   # Generate output  output = model.generate(  input\_ids=input\_ids,  attention\_mask=attention\_mask,  max\_length=512,  num\_beams=5,  early\_stopping=True,  )   # Decode the generated output  decoded\_output = tokenizer.decode(output[0], skip\_special\_tokens=True)  return decoded\_output |
| --- |

| **Example usage** sentence = "OR FOR ELON WHO'S BEN TALKING ABOUT OPTIMISTS THE ROBOT POTENTIALY " corrected\_sentence = infer(model, sentence) print("Corrected Sentence:", corrected\_sentence) |
| --- |

#### 

#### **Notes**

* If using a GPU, ensure PyTorch is installed with CUDA support.
* If memory issues arise, reduce batch size or use a smaller model variant.
* Training duration depends on dataset size and hardware specifications.