**Introduction to the Project**

In the modern digital era, the demand for multilingual communication has increased significantly. As language remains a fundamental medium for the exchange of ideas, information, and knowledge, the ability to translate between languages has become a crucial requirement in various sectors including education, governance, media, and technology.

Machine Translation (MT), a subfield of Artificial Intelligence (AI), refers to the use of computer algorithms to translate text or speech from one natural language to another. Over the years, multiple methodologies have emerged to tackle this problem—ranging from Rule-Based and Statistical Machine Translation to the current state-of-the-art Neural Machine Translation (NMT) techniques.

This project aims to develop a Neural Machine Translation system for translating textual data from English to Hindi. The translation model leverages the MBART (Multilingual BART) architecture, a transformer-based encoder-decoder model pre-trained on multilingual data, to achieve accurate and fluent translations. The implementation is carried out using pre-trained models from Hugging Face’s Transformers library.

The core objective of this project is to demonstrate how pre-trained transformer models can be utilized effectively for low-resource language translation tasks such as English to Hindi, thereby contributing to the ongoing research and development in multilingual NLP applications.

**Dataset Used**

The dataset employed in this project is titled **"Hindi-English Truncated Corpus"**, publicly available on Kaggle and contributed by *umasrikakollu72*.  
**Dataset link:** https://www.kaggle.com/datasets/umasrikakollu72/hindi-english-truncated-corpus

This dataset consists of parallel sentence pairs in English and Hindi, structured specifically for machine translation tasks. Each record contains a sentence in English and its corresponding translation in Hindi, making it ideal for supervised training of neural machine translation models.

**Key Features:**

* **Data Format:** CSV (Comma Separated Values)
* **Columns:**
  + english\_sentence: Sentence in English
  + hindi\_sentence: Corresponding sentence translated into Hindi
* **Source:** The dataset is derived from larger multilingual corpora, likely influenced by the **CLARIN parallel corpora** and other publicly available translation datasets.

**Why this Dataset?**

This dataset is particularly suited for this project because:

* It is sentence-aligned, which is critical for training sequence-to-sequence models.
* It focuses on Hindi–English translation, a low-resource pair in global NLP contexts.
* It is clean, well-structured, and compatible with modern transformer architectures like MBART.

By using this dataset, the project aims to demonstrate high-quality translation from English to Hindi using multilingual pre-trained language models.

**Model Used**

The model used for translation in this project is **MBART (Multilingual BART)**, specifically the facebook/mbart-large-50-one-to-many-mmt variant provided by Hugging Face.

MBART is a transformer-based sequence-to-sequence model introduced in the paper titled *“Multilingual Denoising Pre-training for Neural Machine Translation”* by Yinhan Liu et al. Unlike earlier approaches which pre-trained only the encoder or decoder, MBART pre-trains both components jointly using a denoising objective. The model masks spans of input text and learns to reconstruct them, enabling it to capture cross-lingual representations.

Key features of MBART include:

* Support for translation across 50 different languages.
* Joint encoder-decoder pretraining for improved performance.
* Fine-tuning capabilities for specific language pairs (e.g., English to Hindi).
* Strong performance in low-resource translation settings.

MBART is accessed and utilized in this project through the Transformers library by Hugging Face, which provides both the tokenizer and the model checkpoints for implementation.

**Code**

# Install necessary packages (run once)

!pip install fasteda cufflinks plotly\_express scikit-learn transformers sentencepiece kagglehub[pandas-datasets]

import os

import re

import string

from string import digits

from collections import Counter

import pandas as pd

import numpy as np

import cufflinks as cf

import plotly\_express as pe

from fasteda import fast\_eda

from sklearn.model\_selection import train\_test\_split

from sklearn.utils import shuffle

import kagglehub

from kagglehub import KaggleDatasetAdapter

# Load dataset using KaggleHub

file\_path = "Hindi\_English\_Truncated\_Corpus.csv"

df = kagglehub.load\_dataset(

KaggleDatasetAdapter.PANDAS,

"umasrikakollu72/hindi-english-truncated-corpus",

file\_path

)

# Use only first 20,000 rows for training

df = df.head(20000).dropna()

# Set offline mode for cufflinks

cf.set\_config\_file(offline=True)

# Feature engineering: sentence length and token counts

df["eng\_char\_count"] = df["english\_sentence"].str.len()

df["hindi\_char\_count"] = df["hindi\_sentence"].str.len()

df["eng\_tok\_count"] = df["english\_sentence"].str.split().str.len()

df["hindi\_tok\_count"] = df["hindi\_sentence"].str.split().str.len()

# Filter dataset to only 'ted' source for consistency

df = df[df['source'] == 'ted']

# Preprocessing text: lowercasing, removing quotes, punctuation, digits

df["english\_sentence"] = df["english\_sentence"].str.lower().str.replace("'", "").str.replace(",", " ")

df["hindi\_sentence"] = df["hindi\_sentence"].str.lower().str.replace("'", "").str.replace(",", " ")

exclude = set(string.punctuation)

remove\_digits = str.maketrans('', '', digits)

df["english\_sentence"] = df["english\_sentence"].apply(lambda x: ''.join(ch for ch in x if ch not in exclude)).apply(lambda x: x.translate(remove\_digits)).str.strip().str.replace(" +", " ")

df["hindi\_sentence"] = df["hindi\_sentence"].apply(lambda x: ''.join(ch for ch in x if ch not in exclude)).apply(lambda x: x.translate(remove\_digits)).str.strip().str.replace(" +", " ")

# Remove Hindi digits

df["hindi\_sentence"] = df["hindi\_sentence"].apply(lambda x: re.sub("[२३०८१५७९४६]", "", x))

# Add start and end tokens to Hindi sentences

df["hindi\_sentence"] = df["hindi\_sentence"].apply(lambda x: "START\_ " + x + " \_END")

# Build vocabulary sets

all\_eng\_words = set(word for sent in df["english\_sentence"] for word in sent.split())

all\_hindi\_words = set(word for sent in df["hindi\_sentence"] for word in sent.split())

print("English vocab size:", len(all\_eng\_words))

print("Hindi vocab size:", len(all\_hindi\_words))

# Create token-to-index mappings

input\_words = sorted(all\_eng\_words)

target\_words = sorted(all\_hindi\_words)

input\_token\_index = {word: i + 1 for i, word in enumerate(input\_words)}

target\_token\_index = {word: i + 1 for i, word in enumerate(target\_words)}

# Shuffle the dataframe

df = shuffle(df)

# Find max sentence lengths

max\_input\_length = df["english\_sentence"].apply(lambda x: len(x.split())).max()

max\_output\_length = df["hindi\_sentence"].apply(lambda x: len(x.split())).max()

print("Max English sentence length:", max\_input\_length)

print("Max Hindi sentence length:", max\_output\_length)

# Optional: Filter sentences longer than 20 words

df = df[(df["english\_sentence"].str.split().str.len() <= 20) & (df["hindi\_sentence"].str.split().str.len() <= 20)]

# Split dataset into train and test sets

x = df["english\_sentence"]

y = df["hindi\_sentence"]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.5, random\_state=42)

# Save splits for later use

x\_train.to\_pickle("x\_train.pkl")

x\_test.to\_pickle("x\_test.pkl")

from transformers import MBartForConditionalGeneration, MBart50TokenizerFast

import torch

# Load pre-trained mBART model and tokenizer for English to Hindi translation

model = MBartForConditionalGeneration.from\_pretrained("facebook/mbart-large-50-one-to-many-mmt")

tokenizer = MBart50TokenizerFast.from\_pretrained(

"facebook/mbart-large-50-one-to-many-mmt",

src\_lang="en\_XX"

)

def translate(text):

# Tokenize input text

model\_inputs = tokenizer(text, return\_tensors="pt")

# Generate translation with forced Hindi language token

generated\_tokens = model.generate(

\*\*model\_inputs,

forced\_bos\_token\_id=tokenizer.lang\_code\_to\_id["hi\_IN"]

)

# Decode generated tokens to text

translation = tokenizer.batch\_decode(generated\_tokens, skip\_special\_tokens=True)

return translation

# Translate first 10 English sentences from dataset

sample\_sentences = df["english\_sentence"].head(10)

translations = sample\_sentences.apply(translate)

# Display example translation vs original

print(f"English Sentence: {sample\_sentences.iloc[0]}")

print(f"Translated Hindi: {translations.iloc[0][0]}")

print(f"Original Hindi: {df['hindi\_sentence'].iloc[0].replace('START\_ ', '').replace(' \_END', '')}")

# Clean up to free memory

del model

del tokenizer

**RESULTS**

