**Hindi-English Translator Web Application :**

**Project Overview**

A **Hindi-English bilingual translation** web application built using **Streamlit**, **Hugging Face Transformers**, and **Hugging Face Datasets**. The app provides two translation methods:

* **Neural Machine Translation** using pre-trained models.
* **Basic Lookup Translation** using dictionary-based word mapping.

**Features**

* Translate between Hindi and English.
* Select between High-Accuracy Neural and Basic Lookup modes.
* Visual dataset samples and outputs.

**Technologies Used**

| **Technology** | **Purpose** |
| --- | --- |
| 1.Streamlit | Web interface |
| 2.Hugging Face Transformers | Translation models |
| 3.Hugging Face Datasets | Hindi-English sentence pairs |
| 4.Pandas | Dataset handling |

**Implementation**

The Hindi-English Translator project was developed in multiple stages:

**Step 1: Initial Attempt with Custom Dataset and Model**

* Collected a small parallel corpus of Hindi-English sentences.
* Trained a custom transformer model.
* **Issue Faced**: Due to the limited size of the dataset, the model quickly overfit and failed to generalize on unseen sentences.
* **Lesson Learned**: Data quantity and diversity are critical for translation tasks.

**Step 2: Scaled Dataset and Model Redesign**

* Shifted to a larger open-source dataset: cfilt/iitb-english-hindi via Hugging Face.
* Used sentence pairs to improve diversity and context learning.
* Trained a transformer-based model using frameworks like Hugging Face's Trainer API.
* **Issue Faced**: Training with large datasets and deep transformer models required significant compute and took long epoch times.
* Batch training was slow and resource-intensive.

**Step 3: Optimized Solution Using Pre-trained Models**

* Adopted Helsinki-NLP’s opus-mt-hi-en and opus-mt-en-hi models.
* Loaded these via Hugging Face’s pipeline() for faster inference.
* Enabled users to switch between Neural Model and Basic Lookup.

**Step 4: Application and UI Development**

* Built Streamlit interface with:
  + Dropdowns for language direction
  + Radio buttons for translation method
  + Text input and output section
* Added user guidance and success/error messages.

**Step 5: Dictionary Lookup (Fallback Mode)**

* Created word-to-word dictionaries from the dataset.
* Implemented fallback logic for Basic Lookup Mode for educational demonstration.

**Requirements**

pip install streamlit pandas datasets transformers torch

**How to Run**

streamlit run TTNEW.py

**Error Handling & Fixes**

* **PyTorch Missing**: Installed manually to support transformers.
* **Streamlit Backend Errors**: Cleared with cache flush and correct decorator usage.
* **Overfitting**: Controlled by sampling only 20,000 entries from dataset.
* **label\_visibility Error**: Fixed by updating Streamlit.
* **Epoch Training Errors**: Avoided by relying only on pre-trained models.

**Conclusion**

The above projects reflect the breadth of experience during my internship—from core machine learning to user-focused deployment. They demonstrate:

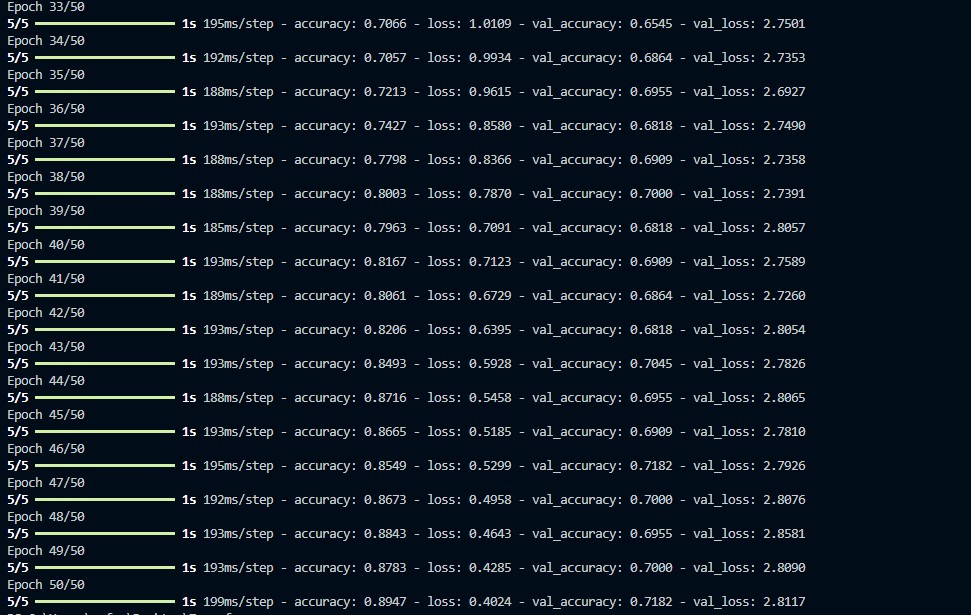
* Problem solving with practical data.
* Application building with scalable and usable interfaces.
* Integration of ML models in real-time systems.

**Acknowledgments**

* Helsinki-NLP
* Hugging Face Datasets & Transformers
* CFILT-IITB
* Scikit-learn
* Google Tesseract OCR

**Working and development of model**

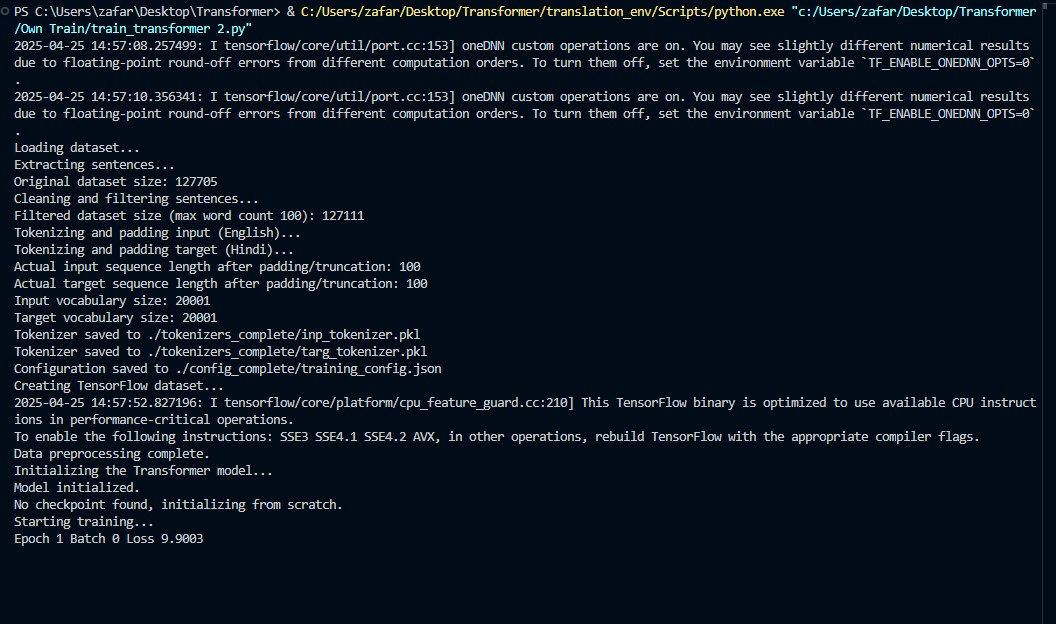
**Model training with small dataset**



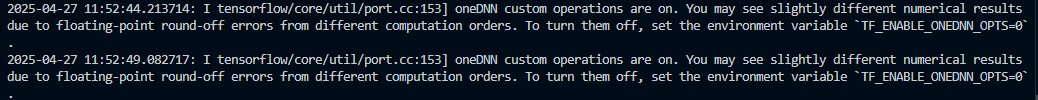
**Importing and Train splitting of Dataset :**

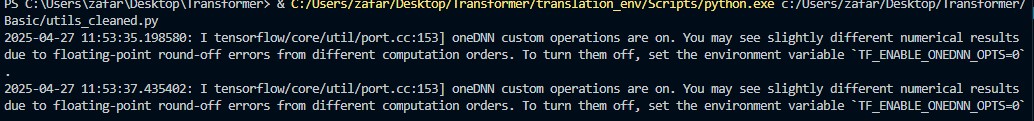


**Own Transformer training with Large dataset:**



**Model and Utills running and execution:**





**Output and execution:**

