Project Report

Abstract

This report details the development of a transformer-based model for transliterating Urdu poetry into Devanagari script. The model achieved 97% training accuracy and 96% validation accuracy on a dataset of 30,000 ghazals (~400,000 lines). Key challenges included diacritic preservation and computational resource limitations, resolved through NFC normalization and migration to Kaggle Notebooks.

1. Introduction

1.1 Objective

The objective of this project was to develop and train a transformer-based model for the transliteration of Urdu poetry into the Devanagari script, with a particular focus on preserving diacritical marks essential for poetic meter and pronunciation.

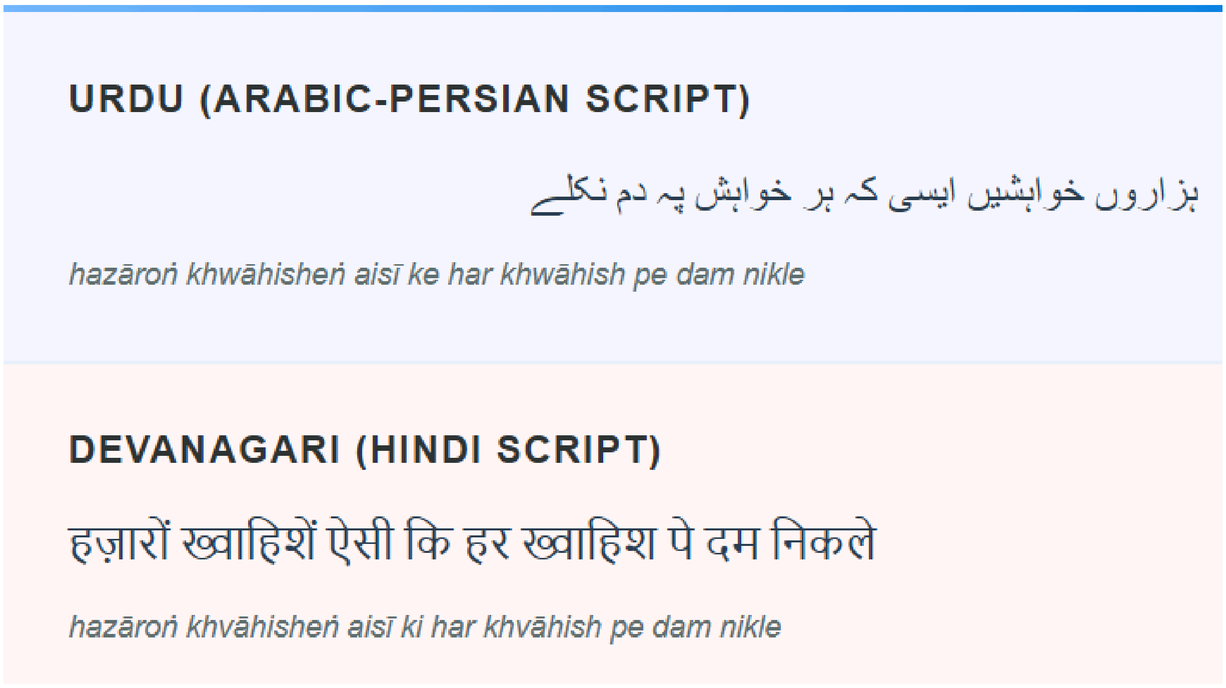
1.2 Motivation

With the growing appreciation for Urdu poetry, there is an increasing need to make these literary works accessible to a wider audience, including those who are unable to read the Urdu script. Transliteration into Devanagari script not only broadens the reach of Urdu poetry but also fosters cross-cultural understanding and literary exchange. However, this task presents significant challenges, as poetry relies heavily on subtle phonetic and orthographic features that must be preserved for the work’s aesthetic and rhythmic qualities to remain intact. Accurate transliteration is therefore essential—not only for accessibility but also for the faithful preservation of poetic form, meter, and meaning. This project aims to address these challenges by utilizing advanced deep learning techniques to enable high-quality, script-accurate transliteration of Urdu poetry into Devanagari.

2. Methodology

2.1 Dataset

The training dataset consisted of 3,000 ghazals, amounting to approximately 400,000 lines of Urdu poetry. The dataset was formatted as parallel text pairs (Urdu and Devanagari).



URDU (ARABIC-PERSIAN SCRIPT)

ہزاروں خواہشیں ایسی کہ ہر خواہش پہ دم نکلے

hazāroṅ khwāhisheṅ aisī ke har khwāhish pe dam nikle

DEVANAGARI (HINDI SCRIPT)

हज़ारों ख्वाहिशें ऐसी कि हर ख्वाहिश पे दम निकले

hazāroṅ khvāhisheṅ aisī ki har khvāhish pe dam nikle

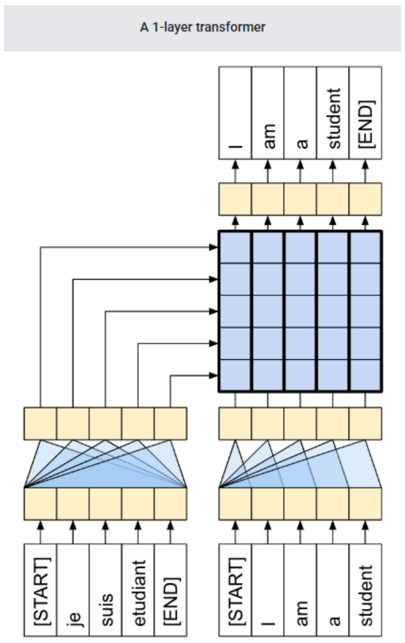
2.2 Data Preprocessing

- Tokenization:
  Standard tokenizers were insufficient for the unique characteristics of Urdu and Devanagari scripts. Custom tokenizers were developed for each script to improve handling of language-specific features.
- Normalization:
  The default NFD normalization in TensorFlow led to the loss of important diacritics (matras). After investigation, NFC normalization was adopted, as it preserved these diacritics in the tokenized text.

2.3 Model Architecture

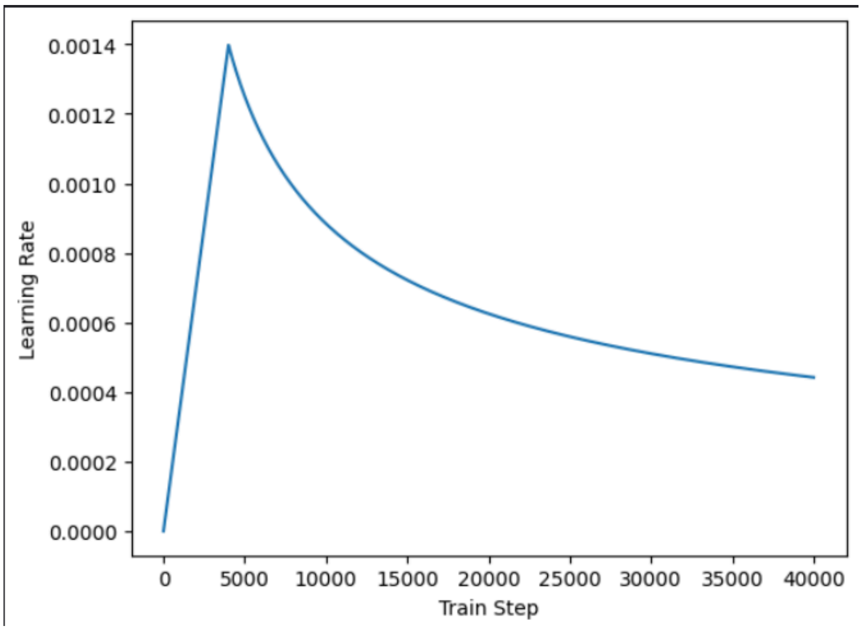
A standard transformer model was implemented using TensorFlow, consisting of three layers. The architecture can be expanded in future retraining scenarios.

- Framework: TensorFlow
- Model Type: Transformer-based NMT
- Number of Layers: 3 (configurable)



A representative 1-layer transformer architecture for Portuguese to English translation

- Optimization: Custom learning rate scheduler as recommended in TensorFlow documentation



Custom Learning Rate Scheduler

2.4 Training Procedure

The model was trained for 10 epochs. The loss curve stabilized after approximately 5 epochs, indicating effective learning. Standard optimization techniques were used.

3. Results

- Training Accuracy: Approximately 97%
- Validation Accuracy: Approximately 96%
- The model demonstrated stable convergence and strong generalization on the validation set.

4. Challenges and Solutions

- Diacritic Preservation:
  Initial preprocessing with NFD normalization resulted in the loss of matras in the output Devanagari script. Switching to NFC normalization resolved this issue.
- Computational Resources:
  Training on Google Colab led to memory leaks and session crashes due to TensorFlow pipeline issues. Migrating to Kaggle Notebooks, which offered 2×T4 GPUs and 28 GB RAM, allowed uninterrupted model training.

5. Conclusion

This study demonstrates that transformer-based models, combined with custom preprocessing, can effectively transliterate Urdu poetry into Devanagari script while preserving critical linguistic features. Future work may involve increasing model depth, expanding the dataset, and optimizing for deployment.

6. References and Acknowledgements

- Dataset by https://www.rekhta.org
- https://www.tensorflow.org/text/guide/subwords_tokenizer
- https://www.tensorflow.org/text/tutorials/transformer