**Roman Urdu to Urdu and English Transliteration**

****

**Department of Computer Science**

|  |  |
| --- | --- |
| **Tehreem Fatima**  **Kainat Bibi**  **Seemi Shaheen** | **NIM\_BSCS\_2019\_26**  **NIM\_BSCS\_2019\_64**  **NIM\_BSCS\_2019\_65** |
|  |  |

**2019-2023**

Final year project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science

|  |
| --- |
| **Namal Institute,**  **30-KM, Talagang Road, Mianwali, Pakistan.**  [**www.namal.edu.pk**](http://www.namal.edu.pk) |

**DECLARATION**

The project report titled **“Roman Urdu to Urdu and English Transliteration”** is submitted in partial fulfillment of the degree of Bachelor of Science in Computer Science, to the Department of Computer Science at Namal Institute, Mianwali, Pakistan.

It is declared that this is original work done by the team members listed below, under the guidance of our supervisor **“Muhammad Sheraz Anjum”.** No part of this project and its report is plagiarized from anywhere, and any help taken from previous work is cited properly.

No part of the work reported here is submitted in fulfillment of the requirement for any other degree/ qualification in any institute of learning.

**Team Members University ID Signatures**

Tehreem Fatima NIM\_BSCS\_2019\_26 

Kainat Bibi NIM\_BSCS\_2019\_64 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Seemi Shaheen NIM\_BSCS\_2019\_65 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Supervisor**

Dr. Muhammad Sheraz Anjum

**Signatures with date**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Table of Contents**

[Chapter 1 7](#_Toc140763258)

[**Introduction** 7](#_Toc140763259)

[**1.1** **Overview** 7](#_Toc140763260)

[**1.2** **Background Information** 8](#_Toc140763261)

[**1.3** **Problem Statement** 8](#_Toc140763262)

[**1.4** **Problem Solution** 9](#_Toc140763263)

[1.5 Second Methodology: 10](#_Toc140763264)

[1.6 Objectives 11](#_Toc140763265)

[1.7 Scope: 11](#_Toc140763266)

[1.8 Report Structure 12](#_Toc140763267)

[Chapter 2 13](#_Toc140763268)

[**Literature Review** 13](#_Toc140763269)

[**2.1 Brief Overview** 13](#_Toc140763270)

[**2.2 Analysis of Literature Review (in the project’s context)** 13](#_Toc140763271)

[ Development of a novel translator for Roman Urdu to English 13](#_Toc140763272)

[ Deep Learning-based Roman-Urdu to Urdu Transliteration 14](#_Toc140763273)

[Deep Learning-based Roman-Urdu to Urdu Transliteration 15](#_Toc140763274)

[Chapter 3 16](#_Toc140763275)

[**Methodology** 16](#_Toc140763276)

[3.1 Methodology Part 1 (Rule Based Character Substitution) 16](#_Toc140763277)

[3.2 Datasets 17](#_Toc140763278)

[**3.2.1 Dataset Composition** 17](#_Toc140763279)

[**3.2.2 Technologies** 18](#_Toc140763280)

[**3.2.3 Tools** 18](#_Toc140763281)

[3.3 Design Process 18](#_Toc140763282)

[3.4 Created Dictionary Using NLP 20](#_Toc140763283)

[3.5 Our Model*:* 21](#_Toc140763284)

[3.6 Mathematical Model 22](#_Toc140763285)

[3.7 Algorithm 23](#_Toc140763286)

[3.8 Use Case Diagram 24](#_Toc140763287)

[3.9 Sequence Diagram 24](#_Toc140763288)

[3.10 Process Diagram 25](#_Toc140763289)

[3.11 Implementation Constraints 26](#_Toc140763290)

[3.12 Design Process 27](#_Toc140763291)

[3.13 Mathematical Model: 28](#_Toc140763292)

[3.14 Algorithm: 28](#_Toc140763293)

[3.15 Process Diagram: 29](#_Toc140763294)

[3.16 Data Flow Diagram (DFD): 29](#_Toc140763295)

[3.17 Implementation Constraints: 30](#_Toc140763296)

[Chapter 4 31](#_Toc140763297)

[**Testing for Methodology 1:** 31](#_Toc140763298)

[**4.1 Unit Test Cases** 31](#_Toc140763299)

[Chapter 5 32](#_Toc140763300)

[**Results** 32](#_Toc140763301)

[Chapter 6 34](#_Toc140763302)

[Evaluation and Discussion 34](#_Toc140763303)

[**6.1.1 Original Problem** 34](#_Toc140763304)

[**6.1.2 Results** 35](#_Toc140763305)

[**6.1.3 Limitations** 35](#_Toc140763306)

[**6.1.4 Failures and Reasons** 35](#_Toc140763307)

[Chapter 7 37](#_Toc140763308)

[**Conclusion** 37](#_Toc140763309)

[Chapter 8: 43](#_Toc140763310)

[Interfaces: 43](#_Toc140763311)

**List of Tables:**

[**Table 1:** Literature Review 15](#_Toc140763395)

[**Table 2:** Roman & Urdu Sentences 17](#_Toc140763396)

[**Table 3:** Roman Urdu to Urdu Translation Results 33](#_Toc140763397)

**List of Figures:**

**Figure 1 ………………………………………………………….............................................11**

**Figure 2 ………………………………………………………….............................................12**

**Figure 3 ………………………………………………………….............................................21**

**Figure 4 ………………………………………………………….............................................22**

**Figure 5 ………………………………………………………….............................................23**

**Figure 6 ………………………………………………………….............................................24**

**Figure 7…………………………………………………………..............................................26**

**Figure 8 …………………………………………………………..............................................27**

**Figure 9 ………………………………………………………….............................................28**

**Figure 10 …………………………………………………………...........................................31**

**Figure 11 …………………………………………………………...........................................32**

**Figure 12 …………………………………………………………...........................................34**

**Figure 13 …………………………………………………………...........................................34**

**Figure 14 …………………………………………………………...........................................34**

**Figure 15 …………………………………………………………...........................................34**

**Figure 16 …………………………………………………………...........................................36**

**Figure 17 …………………………………………………………...........................................37**

**Figure 18 …………………………………………………………............................................37**

**Figure 19 …………………………………………………………...........................................45**

**Figure 20 …………………………………………………………...........................................46**

**Figure 21 …………………………………………………………...........................................46**

**Figure 22 …………………………………………………………...........................................47**

**Figure 23…………………………………………………………............................................48**

**Figure 24…………………………………………………………............................................49**

**ACKNOWLEDGMENTS**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task. We are greatly indebted to our project Supervisor **“Muhammad Sheraz Anjum”** and our Co-Supervisor **“Abdul Rafay”** for personal supervision, advice, valuable guidance, and completion of this project. We are deeply indebted to them for their encouragement and continual help during this work. And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

**Abstract**

This project aims are to develop a Roman Urdu to Urdu and English transliterator, which can convert text written in Roman Urdu script to Urdu and English scripts. Roman Urdu is a popular way of writing Urdu using the English alphabet, which is widely used on social media and messaging platforms. However, this creates difficulties for Urdu readers who are not familiar with the Roman script. We propose two methodologies for transliterating Roman Urdu text into Urdu and English. The first approach relies on rule-based character substitution, mapping Roman Urdu characters to their Urdu and English equivalents. But there was some limitation in first approach so we shifted our direction to second approach and also utilize RNN sequence-to-sequence model for the purpose in our second approach. We have presented the results of both methodologies, with the RNN model outperforming the rule-based method in accuracy and handling complex cases. The transliterator will be useful for a wide range of Urdu speakers and learners, including those who are not proficient in Roman script but want to communicate effectively in Urdu and English. Overall, this project has the potential to facilitate better communication and understanding among Urdu speakers and learners around the world.

# **Chapter 1**

## **Introduction**

The purpose of project is to explore translation dynamics between Roman Urdu, Urdu, and English languages. Roman Urdu is a composing framework that addresses the Urdu language utilizing the Roman letter set, while Urdu is a distinct language primarily spoken in Pakistan and parts of India. English, on the other hand, is a widely spoken global language. Understanding the translation process between these languages is crucial in bridging the communication gap and facilitating effective multilingual communication. The use of Roman Urdu has gained popularity among Urdu speakers who are more comfortable with the English script or do not have access to an Urdu keyboard. It serves as a convenient means for Urdu speakers to communicate in writing using the English script, especially in digital platforms and social media. However, translating text from Roman Urdu to Urdu or English requires a deep understanding of the language structure and semantics to accurately convey the intended meaning.

 Urdu, written in a modified version known as the Perso-Arabic script, holds significant position as the national language of Pakistan. It has a rich literary tradition and is known for its poetry, prose, and historical writings. The translation from Urdu to English poses unique challenges due to the differences in grammar, vocabulary, and cultural nuances between the two languages. English, as a global language, serves as a lingua franca in various domains, including business, science, technology, and international communication. Translating from Urdu to English requires a careful balance between maintaining the essence of the original text and adapting it to the linguistic and cultural expectations of the target audience.

 Throughout this project, we will explore the techniques, tools, and considerations involved in translating text between Roman Urdu, Urdu, and English. We will examine the linguistic peculiarities, cultural context, and linguistic resources available to ensure accurate and meaningful translations. By enhancing our understanding of these translation dynamics, we aim to contribute to effective cross-linguistic communication and promote cultural exchange between these languages. By the end of this project, we hope to provide insights and recommendations for translators and language enthusiasts to improve the quality and accuracy of translations between Roman Urdu, Urdu, and English. Through our exploration, we aim to foster greater understanding and appreciation of these languages and their interplay in a multilingual world.

### **Overview**

This report focuses on the translation process between Roman Urdu and Urdu, as well as Urdu to English. Roman Urdu is used to represent the Urdu language in the English writing system. Urdu speakers who are more accustomed to the Roman script find it simpler to write and utilize it frequently. Roman Urdu has grown in popularity over the past few years, especially on social media and in informal conversation. For individuals who are unfamiliar with the Roman script, it can be difficult. To facilitate effective communication and mutual understanding between individuals and communities, there is a need for precise transliteration services that can assist close the gap between Roman Urdu and Urdu or English. Roman Urdu refers to the representation of the Urdu language using the Roman script, while Urdu is a widely spoken language in South Asia.

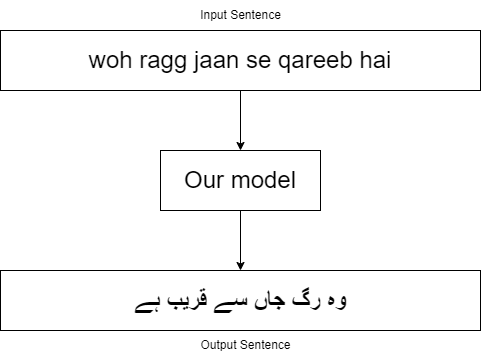
### **Background Information**

Roman Urdu emerged as a means to write and communicate in Urdu using the Roman script, which is commonly used for languages such as English. It provides a convenient way for individuals who are more familiar with the Roman script to express themselves in Urdu. Roman Urdu is often used in informal contexts, such as social media, messaging applications, and casual conversations. It has a rich literary history and is recognized as one of the national languages of Pakistan. Urdu uses a modified version of the Perso-Arabic script and is closely related to Hindi. It has borrowed vocabulary from Arabic, Persian, Turkish, and English, among other languages.

For textual communication, Urdu and English are commonly used throughout South Asia. Roman Urdu users frequently communicate through digital platforms like WhatsApp, Facebook, etc with no set guidelines for spelling. The Pakistani people prefer this language in the form of Roman Urdu to write poetry, literature, and verses. A survey illustrates the fact that 80% of Pakistani people use Roman Urdu. When users feel uncomfortable with their mother language then they try to use the English Alphabet/Language for communication and speaking Urdu using English Alphabet is called Roman Urdu. Roman Urdu is also widely used on social media. So for conversion, different transliteration algorithms have been used but accuracy is reduced because of English words with Roman Urdu.

### **Problem Statement**

In this modern era of technology, we mostly used English Script for communication. The language written using English Alphabet is called Roman Urdu but local users that can't understand roman Urdu (English typing) or even they did not know how to spell English for them we proposed an application that will transliterate roman Urdu into Urdu and then Urdu to English through some basic steps. So there is the need for efficient and accurate translation between Roman Urdu and Urdu, as well as Urdu to English. Although Roman Urdu provides a convenient way for individuals familiar with the Roman script to express themselves in Urdu, the lack of standardized spelling and variations in phonetic representation pose challenges for translation. Additionally, the translation from Urdu to English is crucial for effective communication, accessing a broader range of resources, and participating in global conversations. Therefore, there is a need for reliable translation methods and tools that can bridge the gap between these languages, ensuring accurate and culturally sensitive translations for various contexts and audiences.



**Figure 1** **Example Sentence**

### **Problem Solution**

**1.4.1 Rule-Based Translation for Roman Urdu to Urdu:**

 The rule-based translation approach involves developing a set of linguistic rules and patterns to convert Roman Urdu expressions to their corresponding Urdu equivalents. These rules will consider the phonetic nature of Roman Urdu and account for variations in spelling, taking into account common spelling conventions and linguistic features specific to Roman Urdu. By implementing these rules systematically, the translation process can achieve more accurate and consistent results.

**1.4.2 Dictionary-Based Translation for Roman Urdu to Urdu:**

In addition to rule-based approach, a comprehensive dictionary of Roman Urdu and its corresponding Urdu words will be developed for the translation from Roman Urdu to Urdu. To create this dictionary, a dataset of Roman Urdu and Urdu texts will be collected. The dataset will then be processed by splitting the text into individual characters and tokenizing the words. Each Roman Urdu word will be mapped to its corresponding Urdu translation.

This process involves analyzing the dataset, identifying commonly used expressions, idioms, and phrases in Roman Urdu, and establishing precise translations in Urdu. The dictionary will be curated to include a wide range of vocabulary, taking into account variations in spelling and linguistic nuances specific to Roman Urdu. This comprehensive dictionary will serve as an invaluable resource for translators, providing them with a reliable reference to ensure accurate and contextually appropriate translations from Roman Urdu to Urdu. By utilizing the dictionary-based approach, translators will have access to a vast collection of word mappings, enabling them to accurately translate Roman Urdu expressions into their corresponding Urdu counterparts. This approach enhances the translation process by providing a systematic and consistent framework for converting Roman Urdu text into accurate Urdu translations. It ensures that the translations maintain linguistic integrity and accurately convey the intended meaning, contributing to effective communication and understanding between Roman Urdu and Urdu speakers.

**1.4.3 Utilizing the Google API for Urdu to English Translation:**

For Urdu to English translation, the project proposes leveraging the Google API, which offers a robust and widely-used machine translation system. By utilizing the Google API, the translation process can benefit from advanced algorithms and vast linguistic resources, ensuring accurate translations from Urdu to English. This approach takes advantage of the continuous improvements and updates made to the Google translation system, enhancing the quality and reliability of the translated output

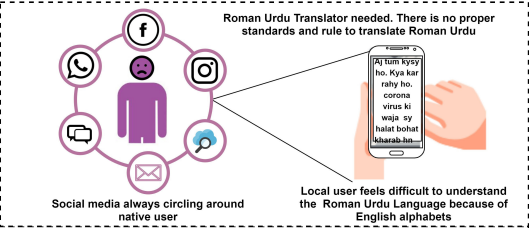


Figure2: Roman Urdu to Urdu Translator

<https://ieeexplore.ieee.org/document/9224843>

In the second methodology, we collected a dataset consisting of pairs of Roman Urdu and corresponding Urdu text. This dataset served as the training data for our RNN sequence-to-sequence model. By training the model on this dataset, it was able to capture the complex mappings and variations between the Roman Urdu input and its corresponding Urdu transliteration, resulting in improved accuracy and performance compared to the rule-based approach.

# **1.5 Second Methodology:**

In the second methodology, we collected a dataset consisting of pairs of Roman Urdu and corresponding Urdu text. This dataset served as the training data for our RNN sequence-to-sequence model. By training the model on this dataset, it was able to capture the complex mappings and variations between the Roman Urdu input and its corresponding Urdu transliteration, resulting in improved accuracy and performance compared to the rule-based approach.

# **1.6 Objectives**

The main purpose of this project is to build up an efficient and accurate translation system for converting text from Roman Urdu to Urdu and from Urdu to English. The primary focus is on utilizing rule-based techniques and a comprehensive dictionary for Roman Urdu to Urdu translation. Additionally, the project aims to leverage the Google API for Urdu to English translation. The overall objective is to facilitate effective communication and understanding between individuals who speak these languages, bridging the language barrier and enabling access to information and resources in different linguistic contexts. Although, this conversion is difficult because roman  
 Urdu has no standard structure, grammar, or vocabulary rules. The following are the objectives of our project**:**

* Develop an application on flutter.
* Users can interact in a user-friendly environment.
* Transliterate Roman Urdu to Urdu and English.
* Transliterate Urdu to English Translation
* Will have possibly highly accurate results (transliteration result).

# **1.7 Scope:**

The scope of this project encompasses the translation process from Roman Urdu to Urdu and from Urdu to English. The project utilizes a rule-based approach for Roman Urdu to Urdu translation, where linguistic rules and patterns are developed to convert Roman Urdu expressions into their corresponding Urdu equivalents. A comprehensive dictionary will also be created, containing commonly used expressions, idioms, and phrases in Roman Urdu along with precise Urdu translations. This dictionary will serve as a valuable resource for translators, ensuring accurate and consistent translations. Furthermore, the project incorporates the use of the Google API for Urdu to English translation. By leveraging the machine translation capabilities of the Google API, the project aims to provide accurate and reliable translations from Urdu to English, taking advantage of the extensive linguistic resources and continuous improvements of the Google translation system.

The project involves the integration of human expertise to review and refine the translations generated by the rule-based techniques and the Google API. Human translators will ensure accuracy, cultural sensitivity, and context preservation in the final translated output. The focus is on maintaining the linguistic integrity of the translations while considering the cultural nuances and idiomatic expressions specific to the respective languages. The project's scope also includes the collection and processing of a dataset of Roman Urdu and Urdu texts to create the comprehensive dictionary. The dataset will be split into characters and tokenized, mapping Roman Urdu words to their corresponding Urdu translations. This dataset will serve as the foundation for the dictionary and contribute to the accuracy and coverage of the translations.

It's important to note that the project primarily addresses translation tasks between Roman Urdu, Urdu, and English. While the project aims to develop an effective translation system, the focus is not on real-time translation or the integration of additional languages beyond the specified language pairs. The project's scope encompasses the development and evaluation of the translation system specifically for Roman Urdu to Urdu and Urdu to English translations.

# **1.8 Report Structure**

In the next chapters, we will be discussing Literature review related to our course, the Methodologies that we have adapted, the Design Process, and testing, and last we will conclude it with a discussion.

# 

# **Chapter 2**

## **Literature Review**

This chapter will provide project background in light of the literature studied.

### **2.1 Brief Overview**

According to some research survey, 300 million people are speaking Urdu, and about 11 million speakers in Pakistan from which maximum users prefer Roman Urdu for communication. In this modern era, technology like computers or mobile phones uses this for communication, and due to native users having to use English typing by hook or by crook. Urdu is mostly spoken in Pakistan, India, the United Kingdom, Canada, and the United States. Arabic, Persian, and most other South Asian languages form the basis of Urdu. Arabic in particular has received extensive study and is a Semitic language. The writing system used in Punjabi, Pashto, Dari, and Farsi (Persian) is also right to left. There are **eleven million peoples** in Asia who speak the Urdu language. According to another research, 1736 distinct words are transliterated into English successfully out of 2000 which shows the accuracy is 86%. A survey study on mobile text data, which collected data from 116 individuals and 346,455 mobile text messages, produced some pretty intriguing findings. These outcomes show that a single user typed the same term using different spellings in many messages. It also demonstrates how the user tried to finish the conversation with the fewest possible words by using abbreviated forms

## **2.2 Analysis of Literature Review (in the project’s context)**

Roman Urdu is a system of writing that uses the Latin alphabet to write the Urdu Language. Roman Urdu is not regarded as the accepted writing system for Urdu, nevertheless. Many efforts have been made to create tools that automatically transliterate Roman Urdu script to Urdu and English. To overcome this problem, we reviewed the literature that was related to our project. The details are given here**:**

# **Development of a novel translator for Roman Urdu to English**

This study was written by Khawar Islam and Muhammad Saeed in May (2019). The author’s main objective was to solve translation issues by developing a novel translator. They discovered that machine learning techniques, especially deep neural networks, performed more accurately and effectively than other techniques.

# **Deep Learning-based Roman-Urdu to Urdu Transliteration**

A hybrid approach that incorporates deep neural networks and rule-based techniques was suggested by Mehreen Alam and Sibt-ul-Hussain to increase the accuracy of Roman Urdu to Urdu script conversion. In comparison to other methods, the authors examined their model on a sizable dataset and reported great accuracy.

* **RUTUT Based on Character Substitution Rules and Unicode Mapping**

This research was done by IEEE Access (members) in 2020. It was all about the conversion from roman Urdu to Urdu using some modules. The author has defined some rules in it that is much helpful for transliteration.

* **Roman Urdu to Urdu Transliteration using the word list**

This essay addresses the transcription of Urdu words into Roman script. They provide a word list-based method that provides a better transliteration for Persio-Arabic letters that have the same or nearly identical sounds but are transcribed as distinct letters in Urdu Script. The suggested rule-based method covers the many ways of writing and provides the best feasible Urdu transliteration based on word lists and roman to Urdu script mapping rules because the roman script for urdu does not adhere to any standards and a single word can be written in a number of different ways.

|  |  |  |  |
| --- | --- | --- | --- |
| **Paper Title** | **Area Focused** | **Author** | **Year** |
| RUTUT Based on Character Substitution Rules and Unicode Mapping | Conversion from Roman Urdu to Urdu by using three basic modules. | Arif Mehmood & Saleem Ullah | October 2020 |
| Roman Urdu to Urdu Transliteration using the word list | This paper followed an encoding scheme for transliteration. | Tafseer Ahmed | September 2017 |
| Development of a novel translator for Roman Urdu to English | Machine Learning Techniques were used for transliteration purpose. | Hafsa Masroor & Muhammad Saeed | May 2019 |
| Deep Learning-based Roman-Urdu to Urdu Transliteration | In this paper, Encoder-Decoder Technique is used in RNN Sequence to Sequence Model. | Mehreen Alam & Sibt ul Hussain | July 2020 |

Table :Literature Review

In conclusion, it’s an expanding field in natural language processing. Although machine learning techniques seem to be the most efficient, hybrid models that use statistical and rule-based approaches might also show promise. Techniques for preprocessing can boost conversion accuracy as well. To improve the handling of irregular spellings and linguistic ambiguity in Roman Urdu text, more study is necessary.

# 

# **Chapter 3**

## **Methodology**

# **3.1 Methodology Part 1 (Rule Based Character Substitution)**

For the transliteration of Roman Urdu to Urdu, we followed some basic steps. In first methodology, we have used the technique of rule based character substitution in which we have imported the library which is regular expressions (re) because it provides a class containing the regular expressions used to perform pattern matching. Then we performed character mapping. It has mapped the characters of the English alphabet with Unicode of the Urdu which is a one-to-one mapping. After that, we created a dictionary using NLP and all the mapped characters are placed into that dictionary. One-to-one mapping means that a single alphabet of English is mapped on a single Unicode of Urdu. After character mapping, we generated some rules that are already defined like replacement and substitution rules. Some rules are generated by ourselves and some are used from literature. For example, in the word **“pehla”** we have replaced eh with h, and in the word **“Jumla”** we substitute a with h. Then, we applied some cheques to these rules for checking that either it is working according to that rules or not. And last, we use **re. find all ()** methods. This method finds all the required characters from the dictionary. So in this way, we have done with transliteration of roman Urdu to Urdu.

In this chapter, we are going to discuss the analysis of the data set collected to deal with this problem and how our model was integrated to get the desired output. We will briefly explain what our model is and in the end we will show the results and how we evaluated our results.

# **3.2 Datasets**

The dataset used in this project focuses on Roman Urdu to Urdu and Urdu to English translation. Acquiring an appropriate dataset is crucial for translation. In this section, we discuss the composition and characteristics of the dataset employed, which comprises two files: one for Roman Urdu and another for Urdu sentences.

## **3.2.1 Dataset Composition**

The dataset consists of a diverse collection of sentences written in Roman Urdu and their corresponding translations in Urdu. Roman Urdu poses a unique challenge due to its absence of specific grammar and structure, making the translation process more complex. The Roman Urdu file includes a range of urdu sentences.  A small glimpse of original and paraphrased sentences data-set is shown in table 3.1:

|  |  |
| --- | --- |
| **Roman Urdu** | **Urdu** |
| pehli jung azeem ke muhasray | پہلی جنگ عظیم کے محاثرے |
| woh darwaish sift ensaan thy | وہ درویش صفت انسان تھے |
| woh urdu ke nasri nazam ke ahem shayar hain | وہ اردو کے نثری نظم کے اہم شاعر ہیں |
| hm khud bhi ghor fikar se kaam letay hain | ہم خود بھی غور فکر سے کام لیتے ہیں |

Table : Roman & Urdu Sentences

### 

## **3.2.2 Technologies**

* Python3
* Visual Studio Code
* Flutter
* Android Studio

## **3.2.3 Tools**

* Matplotlib
* Dataset
* Pandas

# **3.3 Design Process**

The methodology we have proposed is consisting of basically three components like Unicode-based character mapping, rules-based character substitution, and created dictionary using NLP.

#### **Unicode-Based Character Mapping**

In the very first step, we performed character mapping. It has mapped the characters of the English alphabet with Unicode of the Urdu which is a one-to-one mapping. One-to-one mapping means that a single alphabet of English is mapped on a single Unicode of Urdu. If one alphabet of English was mapped on more than one alphabet of Urdu, then we mapped them accordingly.

#### **Rule-Based Character Substitution**

Romanized Urdu is not a recognized language with established grammatical rules, spelling rules, or writing conventions. These guidelines were created with that in mind. We used some rules that are already defined like replacement and substitution rules. Some rules are generated by ourselves and some are used from literature. For example, in the word **“pehla”** we have replaced eh with h, and in the word **“Jumla”** we substitute a with h. Then, we applied some cheques to these rules for checking that either it is working according to that rules or not. Roman Urdu users use different spellings of similar words in different messages.

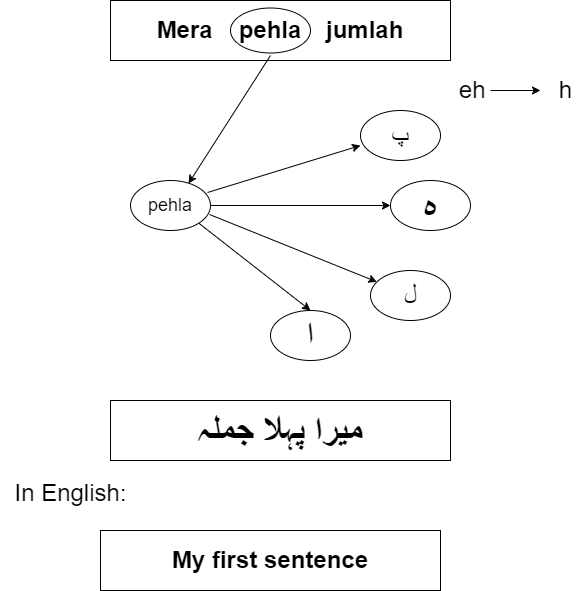


Figure 3: Rule-Based Character Substitution

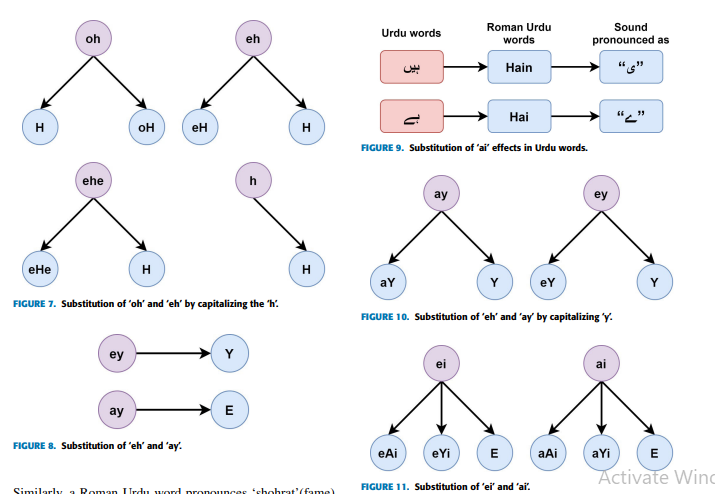


Figure 4: Rule-Based Character Substitution

[2] https://ieeexplore.ieee.org/document/9224843

# **3.4 Created Dictionary Using NLP**

After character mapping and rules generation, we placed all the mapped characters into the dictionary. We have created a dictionary using NLP and then populate our dictionary. So when we start transliterating the sentences, we have entered the roman sentence as input, and this input goes to the dictionary. And then we employ **re. find all ()** method, it checks whether the input words are matching with the dictionary or not. If words match with dictionary characters, then our system translates it as it is otherwise it goes to the rules that we have generated and checks according to that rules. This concludes our use of transliteration.

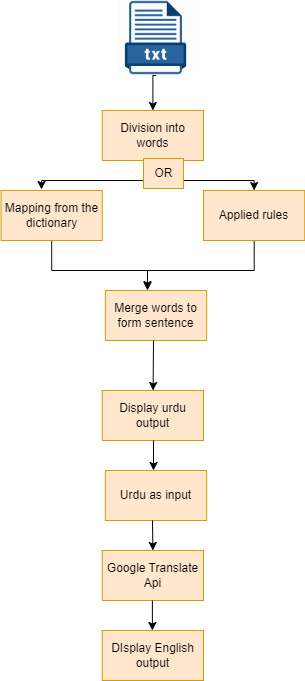


Figure 5: Flowchart of Translation

# **3.5 Our Model:**

This project focuses on the development of a rule-based translation model for converting Roman Urdu text into Urdu, aiming to bridge the gap between English-based Roman Urdu and the native Urdu script. The initial step involved mapping the Roman Urdu characters to their corresponding Urdu characters, forming the foundation for the subsequent translation process. To enhance the translation accuracy and flexibility, a comprehensive dictionary was constructed using a dataset containing Roman Urdu and its corresponding Urdu equivalents. This dictionary served as a valuable resource for the model, enabling the translation of diverse Roman Urdu phrases and expressions into Urdu. Additionally, a set of rules were generated based on linguistic patterns and common grammatical structures found in both Roman Urdu and Urdu. These rules further refined the translation process by accounting for contextual variations and specific syntactical rules unique to the Urdu language. By combining character mapping, a rich dictionary, and rule-based translation, this model aims to provide an efficient and accurate method for converting Roman Urdu into Urdu, facilitating effective communication and understanding for Urdu speakers.

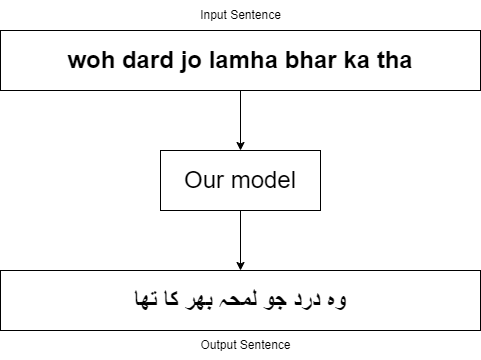


Figure 6: Example of Translation

# **3.6 Mathematical Model**

We are building a flutter app that implements a simple transliterator from Roman Urdu to the Urdu language. It consists of a single screen with a text input field, a translate button, and a text output field. When the user enters some Roman Urdu text into the input field and presses the translate button, the app translates the input text to Urdu and displays it in the output field. The mathematical model of the app can be described as follows:

Let **S** be the set of all possible Roman Urdu strings, and **T** be the set of all possible Urdu strings. Let **R** be a function that maps a Roman Urdu string **s ∈ S** to an Urdu string **t ∈ T**. This function is defined as follows. For each character **c ∈ s**, if c is a valid Roman Urdu character, replace it with its corresponding Urdu character using a predefined set of rules. If c is not a valid Roman Urdu character, leave it unchanged. Replace certain combinations of characters in s with their corresponding Urdu equivalents. The function R can be written as follows:

**R(s) = s. replaceRules(). replaceCombinations ()**

where. replaceRules**()** replaces individual Roman Urdu characters with their corresponding Urdu characters using a predefined set of rules, and .**replaceCombinations()** replaces certain combinations of characters in s with their corresponding Urdu equivalents. The app implements R by defining a Map called rules that maps each Roman Urdu character to its corresponding Urdu character. The app also defines a **transString()** function that takes a Roman Urdu string as input and applies R to it, either forwards or backward depending on a Boolean parameter called reverse. Finally, the app defines **a \_translate()** function that uses **transString()** to translate the input string to Urdu. In the app, the input string is obtained from a Text Field widget, and the output string is displayed in a Text widget. When the user presses the translate button, the app calls the **\_translate()** function to translate the input string and updates the output field using the **setState()** method.

# **3.7 Algorithm**

Below is the pseudocode for the main functions of the app:

**1**. Define the MyApp widget.

**2**. Define the \_translate function that takes the input text and returns the translated output.

**3**. Define the rules map that maps Roman Urdu characters to Urdu characters.

**4**. In the MyApp widget**:**

* Define a TextEditingController for input.
* Define a string variable for output.
* Implement the UI using a Column widget that contains a TextField, ElevatedButton, and Text widget.
* Display the output string in the Text widget.

# **3.8 Use Case Diagram**

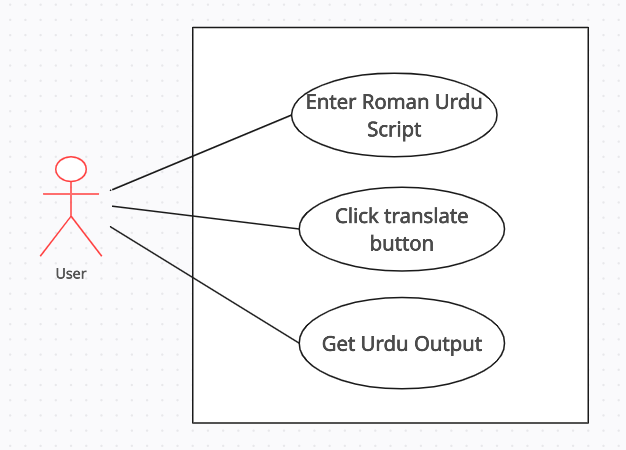


Figure 7: Use Case Diagram for Translator

# **3.9 Sequence Diagram**

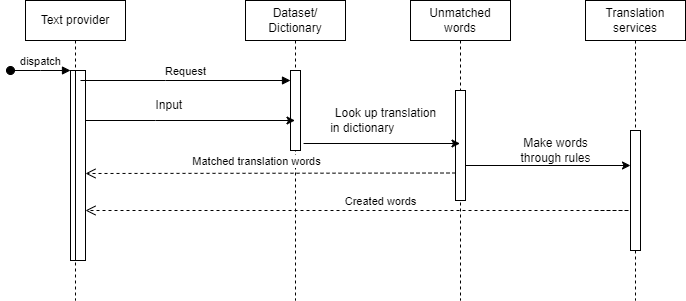


Figure 8: Sequence Diagram for Translator

# **3.10 Process Diagram**

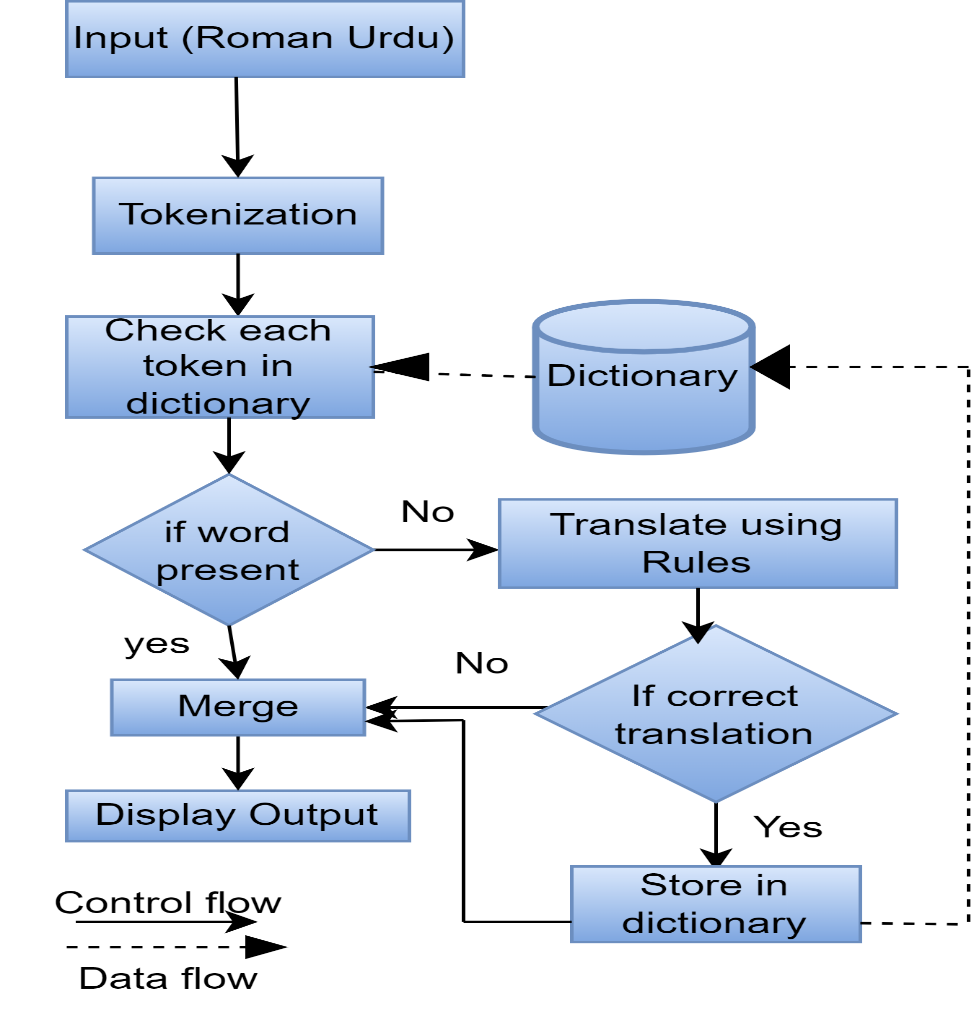


Figure 9: Process Diagram of Translator

# **3.11 Implementation Constraints**

The given code implements a simple text translation app from Roman Urdu to the Urdu language. It uses a map of Roman Urdu characters to their corresponding Urdu characters to perform the translation.

#### **3.11.1 Constraints**

Constraints that could affect the project implementation include**:**

**Accuracy of Translation:** The accuracy of translation from Roman Urdu to the Urdu language depends on the availability and correctness of the map of Roman Urdu characters to their corresponding Urdu characters. If the map is incomplete or contains errors, the translation will be inaccurate.

**Performance:** The performance of the app may be impacted by the length of the input text and the complexity of the translation rules. If the input text is very long or the translation rules are complex, the app may become slow and unresponsive.

To address these constraints, the code uses a simple approach to perform the translation. The translation rules are defined in a map, and the translation is performed by replacing each Roman Urdu character in the input string with its corresponding Urdu character.

#### **3.11.2 Improve Implemented Constraints**

To improve the **accuracy of the translation,** the code also includes rules for handling combined Roman Urdu characters such as 'ai' and 'ei', which are replaced with the appropriate Urdu characters. And **to improve performance,** the translation is performed using a simple loop that iterates through each character in the input string. The code also uses the **setState()** method to update the app's state when the translation ensures that the app's UI is updated with the translated text in real-time.

Overall, the approach used in the code provides a simple and effective solution for performing text translation from Roman Urdu to the Urdu language, while also addressing some of the key constraints that could impact the project implementation.

**Methodology Part 2 (RNN Sequence to Sequence Model)**

In this methodology, we leveraged machine learning techniques to address the challenge of translating between Roman Urdu and Urdu. Specifically, we used an RNN sequence-to-sequence model. Our dataset consisted of Roman Urdu and Urdu text, which underwent preprocessing steps such as cleaning and tokenization. Through a training process, we trained the model on the dataset. Evaluation was conducted using metrics like BLEU to assess translation performance.

# **3.12 Design Process**

This methodology was consisting of some basic steps which are given as:

***3.12.1 Dataset Selection:***

A pre-existing dataset containing pairs of Roman Urdu text and their corresponding Urdu transliterations was utilized for training the RNN sequence-to-sequence model.

***3.12.2 Data Preprocessing:***

The collected dataset underwent preprocessing steps to clean and format the text. This included removing any irrelevant or noisy data, handling special characters or symbols, and normalizing the text to ensure consistency.

***3.12.3 Sequence-to-Sequence Model Architecture:***

The RNN sequence-to-sequence model was chosen due to its ability to handle variable-length input and output sequences. The model consists of an encoder decoder scheme. The encoder performs the Roman Urdu input sequence, while the decoder generates the Urdu transliteration sequence.

***3.12.4 Training:***

Training and test sets were created from the preprocessed dataset. On the practice set, the model was trained. The model's performance was tracked using the validation set in order to avoid overfitting.

***3.12.5 Evaluation:***

Once training was completed, the model's performance was evaluated on a separate test set, which contained Roman Urdu text not seen during training. Evaluation metrics such as accuracy, precision, recall, and F1-score were computed to measure the model's effectiveness in generating accurate Urdu transliterations.

# **3.13 Mathematical Model:**

The model begins with an input consisting of a sequence of Roman Urdu characters, denoted as x = (x₁, x₂, ..., xₙ), and a target Urdu sequence, denoted as y = (y₁, y₂, ..., yₘ), where each xᵢ represents a Roman Urdu character and each yⱼ represents an Urdu character. The model employs an encoder-decoder architecture. The encoder takes the Roman Urdu sequence x and maps each character xᵢ to a dense vector representation through an embedding matrix E, resulting in a sequence of embedded vectors e = (e₁, e₂, ..., eₙ). These embedded vectors are then fed into a recurrent neural network (RNN) encoder. The decoder, also an RNN, takes the previous target character yₜ₋₁ and the previous hidden state hₜ₋₁ as inputs. To train model, a loss function is defined, typically cross-entropy loss, which measures the discrepancy between the predicted probability distribution P(yₜ|y₁, ..., yₜ₋₁, x) and the actual target Urdu sequence y.

# **3.14 Algorithm:**

**Data Preprocessing:**

* Collect a dataset of Roman Urdu and corresponding Urdu transliterations.
* Clean and normalize the text data, removing any irrelevant or noisy information.
* Divide the dataset into training and testing sets.

**Training:**

* Train the model over the training dataset.
* Encode the Roman Urdu sequence using the encoder RNN.
* Generate the target Urdu sequence using the decoder RNN.
* Save the trained model.

**Evaluation:**

* Load the saved trained model.
* Initialize a list to store the predicted Urdu sequences.
* Initialize a list to store the corresponding target Urdu sequences.
* Calculate the BLEU score by comparing the predicted Urdu sequences with the target Urdu sequences.
* Report the BLEU score as the evaluation metric for the transliteration model.

# **3.15 Process Diagram:**

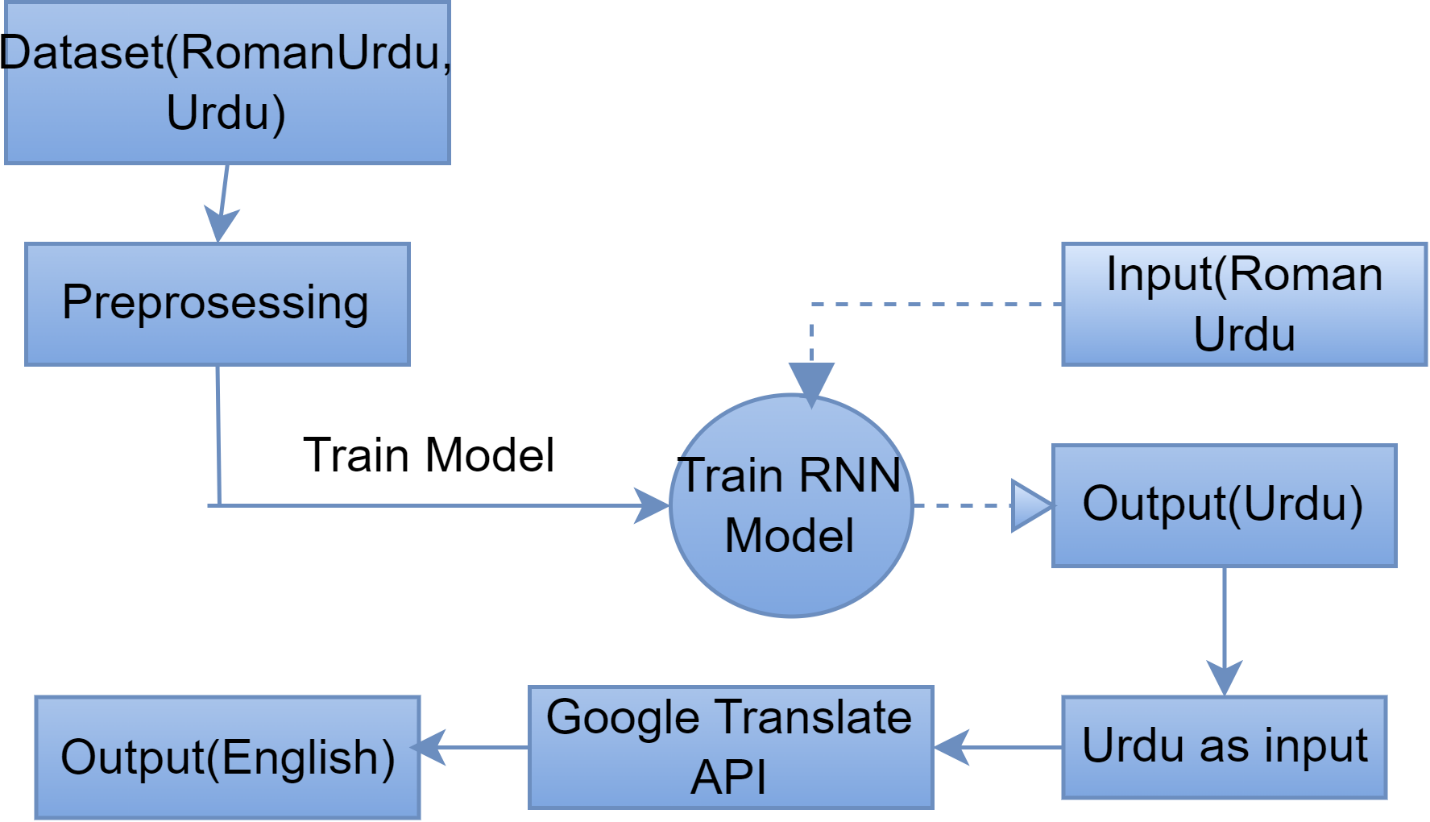


Figure 0: Process Diagram for M2

# **3.16 Data Flow Diagram (DFD):**

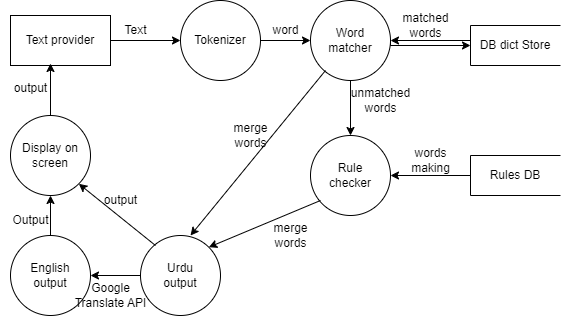


Figure 11: DFD for Rule Based Character Substitution

# **3.17 Implementation Constraints:**

**3.17.1 Computational Resources:**

RNN sequence-to-sequence models can be computationally intensive, especially with large datasets and complex architectures. Ensure that you have access to sufficient computational resources, such as CPUs, GPUs to train and evaluate the models effectively.

**3.17.2 Dataset Size and Quality:**

The performance of the transliteration models heavily depends on the quality and size of the training dataset. It is important to have a diverse and representative dataset that covers a wide range of Roman Urdu input variations and their corresponding Urdu and English transliterations.

**3.17.3 Evaluation Metrics:**

Choosing appropriate evaluation metrics, such as BLEU score or other transliteration-specific metrics, is essential to assess the quality and accuracy of the transliteration models.

# **Chapter 4**

## **Testing for Methodology 1:**

The given code snippet shows the unit test cases for a Flutter app that involves the translation of the text to a different language. Let's review each test case and the testing methodology used for them:

### **4.1 Unit Test Cases**

**4.1.1 Test widget ('Translate button updates output text'...):** This is a widget test case that checks whether the output text gets updated after clicking on the "Translate" button. Here, the Flutter test framework is used to simulate the user input and interaction with the UI elements. The pump Widget method is used to build and render the widget tree, and enter Text and tap methods are used to simulate the user input.

* + 1. **test ('Translation of single character', () ...):** This is a simple unit test case that tests the \_translate method of the app's state class. It verifies that a single character is correctly translated into the target language. Since this test doesn't involve the UI, it is a pure unit test.
    2. **test ('Translation of multiple characters', () ...):** This is another unit test case that tests the \_translate method with a string containing multiple characters. This test case ensures that the app can handle translating longer strings and not just single characters.
    3. **test ('Translation of mixed case input', () ...):** This unit test case checks the \_translate method's ability to handle input strings with mixed cases. It ensures that the app can handle text with different capitalizations.
    4. **test ('Translation of input contain non-mappable characters', ()):** This test case tests the \_translate method's ability to handle input strings containing non-mappable characters. It ensures that the app can handle such characters gracefully and doesn't crash or produce incorrect output.
    5. **test ('Translation of input containing combined characters', () ...):** This unit test case checks the \_translate method's ability to handle

We have used another method in which we used RNN Sequence to Sequence Model. We also performed testing on that, but their results were not encouraging. Our model was translating the sentences from Roman Urdu to Urdu but their accuracy was not good. So, due to this reason, we have not mentioned their test cases here.

# **Chapter 5**

## **Results**

By using different modules and methods, we have done transliteration.

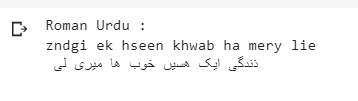


Figure 12: Result 1 of Transliteration

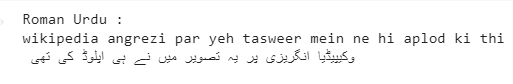


Figure 13: Result 2 of Transliteration

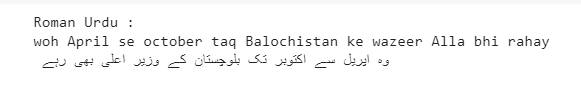


Figure 14: Result 3 of Transliteration

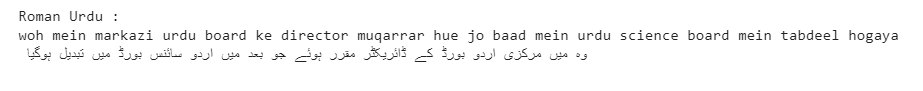


Figure 15: Result 4 of Transliteration

**The reference for above four figures is same and the it’s given below:**

**[3] https://colab.research.google.com/drive/1FdiWHQ\_Jr3bH9bozUQMKUMjT6hx9SrsR?authuser=1**

Above are snaps of result of transliteration that we have done so far. It’s not transliterated words at this time because we are in the doing phase. And in future, we will transliterate more words by adding them into dictionary. For this time being, we have just one type of dataset that’why we get the limited result but when we implement it on different type of datasets then our results will be much better. We will enrich our dictionary so that our system might be able to do work more accurately and efficiently.

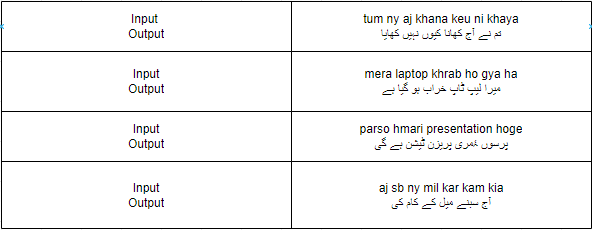


Table : Roman Urdu to Urdu Translation Results

# **Chapter 6**

# **Evaluation and Discussion**

For the evaluation purposes, we used the Bilingual Evaluation Understudy Score (BLEU) metric. It is a metric which is used to evaluate the output text or sentence with respect to the original text. The value BLEU score varies from 0.0 to 1.0 where the perfect mismatch gives the value 0.0 and a perfect match gives 1.0.

Figure 16: BLEU Score Evaluation

**6.1 Discussion**

### **6.1.1 Original Problem**

In this modern era of technology, we mostly used English Script for communication. The language written using English Alphabet is called Roman Urdu but local users that can't understand roman Urdu (English typing) or even they did not know how to spell English for them we proposed an application that transliterated roman Urdu through some basic steps. The system can handle variations and styles of Roman Urdu.

### **6.1.2 Results**

While developing such a system, we faced many difficulties but in the end, we are done with almost translations. Although we are in the doing phase, so in the future, we will be working on it transliterating many words that are possible.

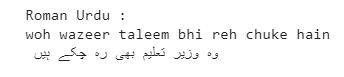


Figure 17: Result 5 of Transliteration

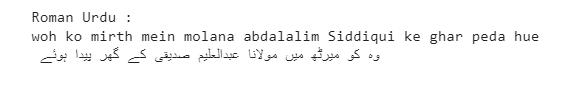


Figure 18: Result 6 of Transliteration

**The reference is also same for above two image the:**

[4] https://colab.research.google.com/drive/1FdiWHQ\_Jr3bH9bozUQMKUMjT6hx9SrsR?authuser=1

### **6.1.3 Limitations**

In our project, we used a dataset and generated rules according to that dataset. So here the limitation was that it was not possible to generate the rules and dictionary for large dataset. For solving this issue, we shifted to another methodology in which we used machine learning model for training it on a large dataset.

### **6.1.4 Failures and Reasons**

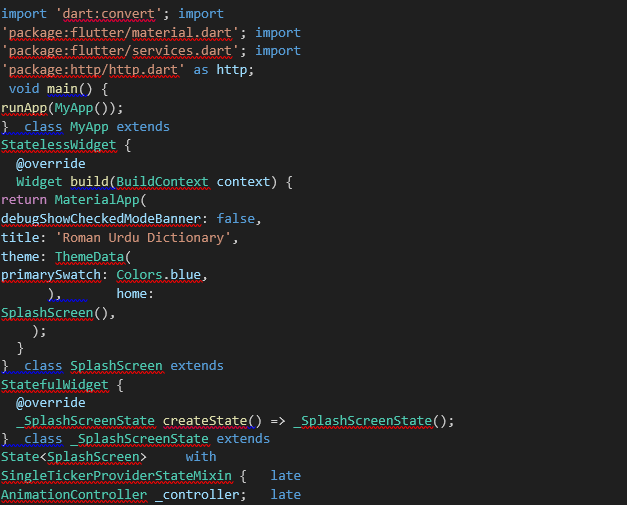
Our system is implemented on just one dataset. So when we try to translate another sentence from a different dataset, then it’s unable to transliterate it. So it’s the failure somehow. We have tried to implement it on different datasets but at the same time, we have changed the track to build an interface to show our work that is done so far. Another failure was that there was a limited rule for limited dataset. We were not able to design many rules in case of large dataset. For resolving this problem, we shifted to another methodology. That’s why, it took much to complete the transliteration part.

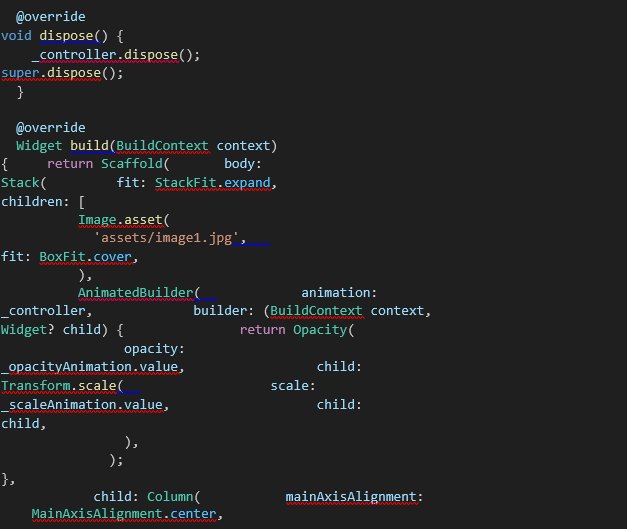
# **Chapter 7**

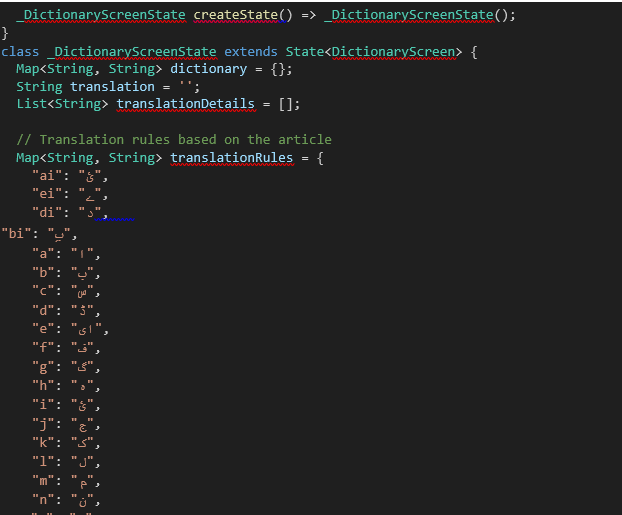
## **Conclusion**

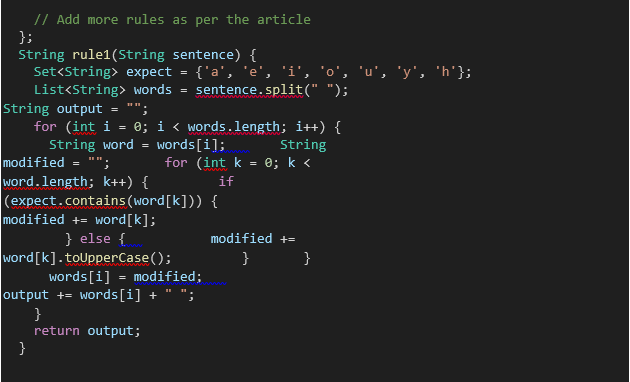
As we know, Roman Urdu is used to represent the Urdu language in the Roman Urdu writing system. Urdu speakers who are more accustomed to the Roman script and find it simpler to write in utilize it frequently. For individuals who are unfamiliar with the Roman script, it can be difficult. To facilitate effective communication and mutual understanding between individuals and communities, there is a need for precise transliteration services that can assist close the gap between Roman Urdu and Urdu or English. For this purpose, we proposed an application that will transliterate roman Urdu into Urdu and then Urdu to English through some basic steps. The system would be able to handle different variations and styles of Roman Urdu. We implemented this system using some steps which included Unicode-based character mapping, and rule-based character substitution, and created a dictionary using NLP. Also we have used machine learning model because there was an issue in first methodology. In this document, we have clearly explained how we address the problem and how we provide different solutions for it. After applying these methods, we are able to translate Roman Urdu to Urdu. For translating Urdu to English, we have used Google Translate API. For displaying the results of methodology 1, we have designed mobile application and for the results of model, we have designed a web interface.

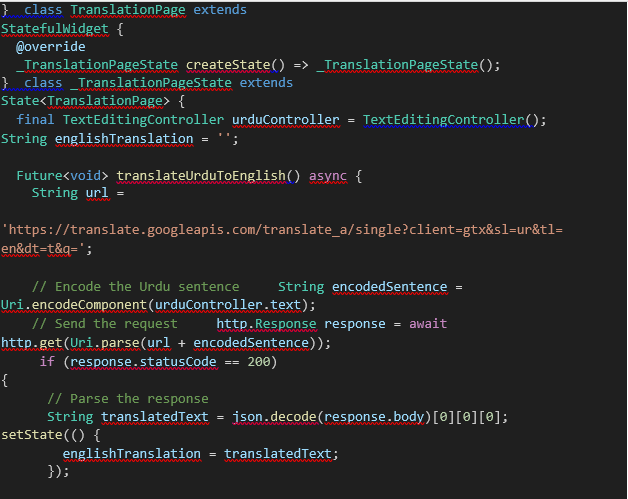
**Code Snippets**











# **Chapter 8:**

# **Interfaces:**

**8.1 Splash Page:**

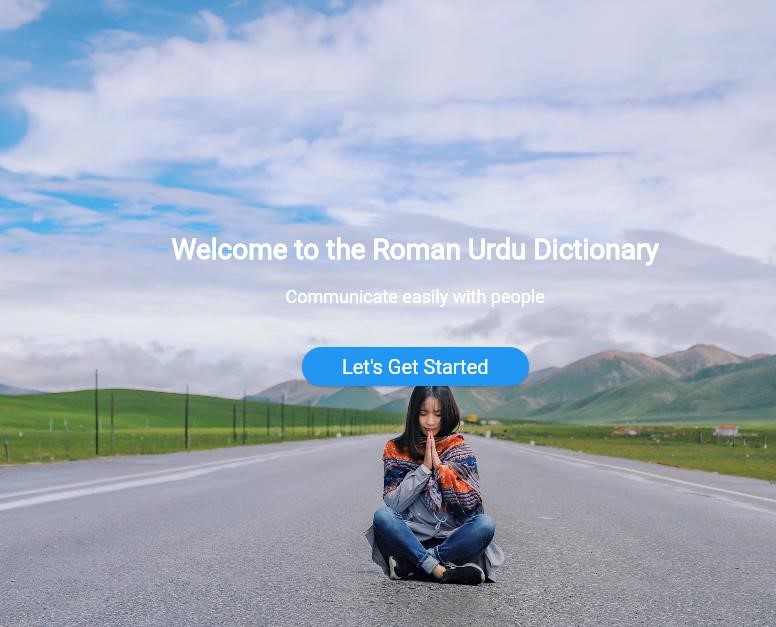


Figure 19: Splash Page

**8.2 Menu:**

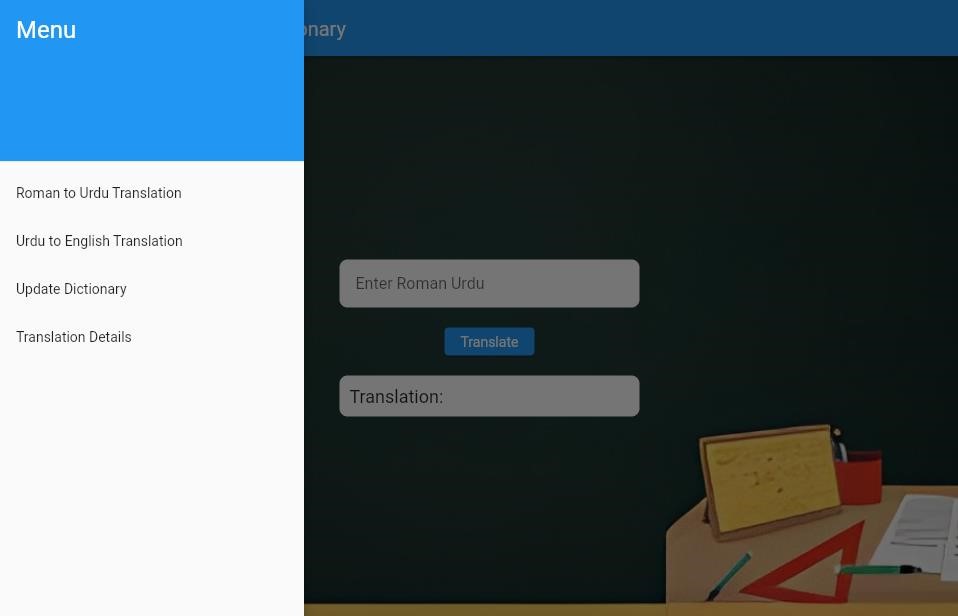


Figure 0: Menu

**8.3 RUTU Interface:**

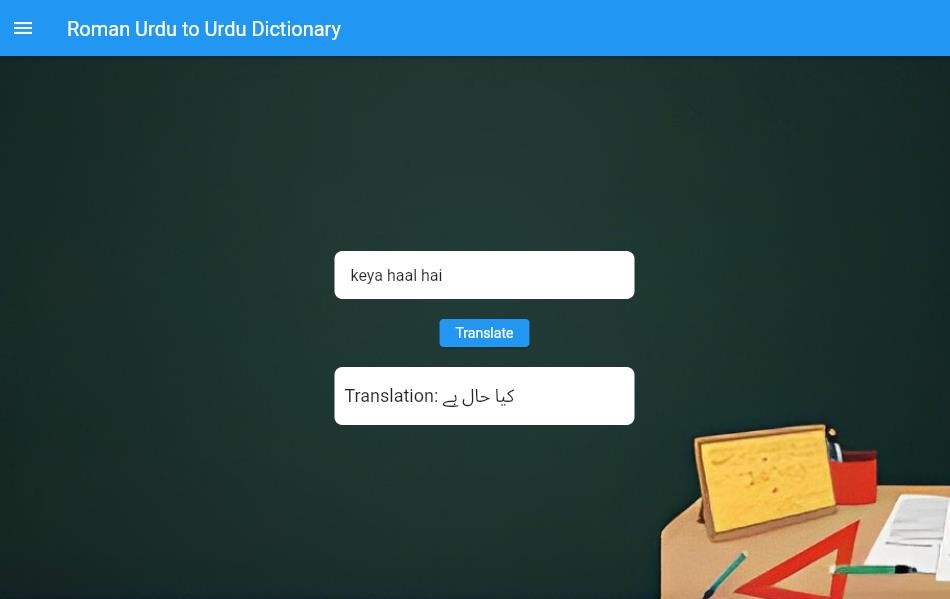


Figure 21: Roman to Urdu Dictionary

**8.4 Updated Dictionary:**

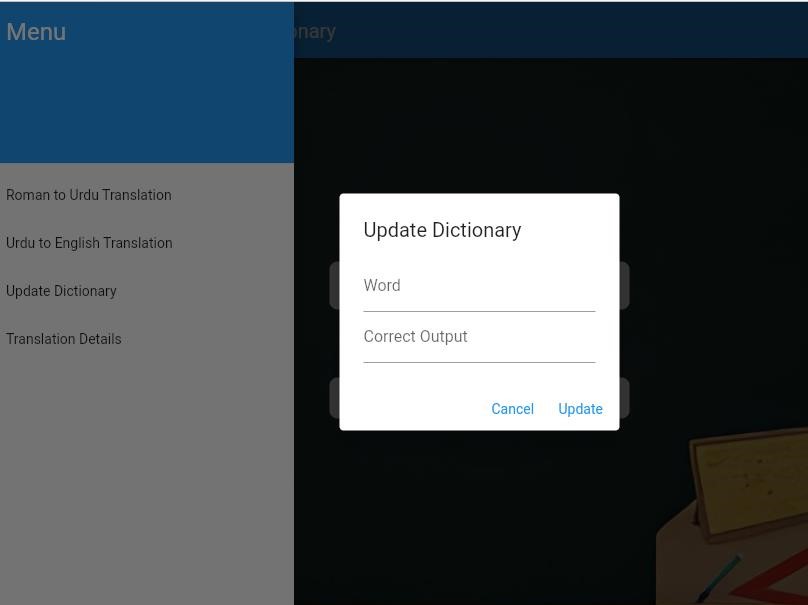


Figure 22: Updated Dictionary

**8.5 Translation Details:**

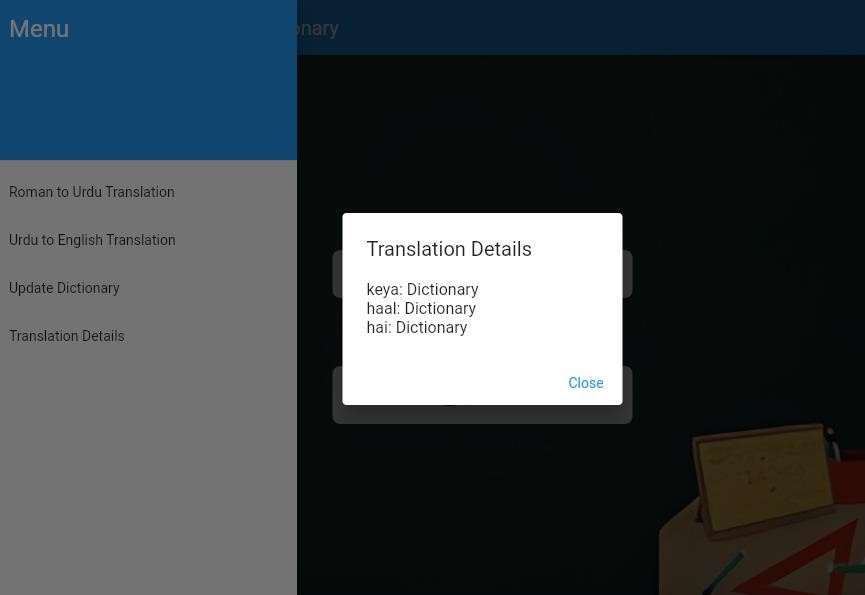


Figure 23: Translation Details

**8.6 Urdu to English Translation:**



Figure 24: Urdu to English Translation