TRANSLITERATING URDU TO ROMAN URDU USING NEURAL NETWORK

WAJAHATULLAH KHAN

IŞIK UNIVERSITY
JULY, 2023

TRANSLITERATING URDU TO ROMAN URDU USING NEURAL NETWORK

WAJAHATULLAH KHAN
Işık University, School of Graduate Studies, I.T Master Program,
2023

This thesis has been submitted to Işık University School of Graduate Studies for a Master's Degree. (MA)

IŞIK UNIVERSITY
JULY, 2023

IŞIK UNIVERSITY
SCHOOL OF GRADUATE STUDIES
INFORMATION TECHNOLOGY MASTER'S PROGRAM

TRANSLITERATING URDU TO ROMAN URDU USING NEURAL NETWORK

WAJAHATULLAH KHAN

APPROVED BY:

Asst. Prof. Gülsüm Çiğdem Çavdaroğlu (Thesis Supervisor)	Işık University
Asst. Prof. Şahin Aydın	Işık University
Assoc. Prof. Mehmet Nafiz Aydın	Kadır Has University

APPROVAL DATE: 12/07/2023

TRANSLITERATING URDU TO ROMAN URDU USING NEURAL NETWORK

ABSTRACT

Transliteration is a process of converting a word from the alphabet of one language to another language. Previously used techniques are related to statistical, phrase level and rule-based approaches but Neural Machine Translation (NMT) has replaced it because of its heterogeneous, scalable and dynamic structure. NMT is used for rich resource languages e.g. German, Vietnam, Chinese and also performing well for poor resource languages like Myanmar, Hindi, and Roman-Urdu. Urdu is a low resource language and there is no significant work done to transliterate using NMT models. In this paper, we are working on Urdu to Roman-Urdu transliteration using sequence-to-sequence and attention-based models. This model uses the Encoder-Decoder architecture that takes one language as input (source) and decoder transforms it to desire output (target). Its results are phenomenal for rich resource languages, providing context aware and scalable solutions. In the field of language transliteration, Long Short-Term Memory (LSTM) and Bi-directional models are commonly employed to effectively deal with long-term dependencies. To handle unseen data, a combination of Byte Pair Encoding (BPE) and subword techniques is utilized, employing a hybrid approach that incorporates both word and character level embeddings. In order to evaluate the performance of the transliteration system, experiments are conducted on a parallel corpus consisting of 60k samples, which were generated from scratch. The system undergoes extensive testing to fine-tune the hyperparameters, ultimately achieving state-of-the-art results measured by the BLEU score on both the training and testing datasets. Additionally, the NMT model provides scalable, robust, context aware structure and can handle out-of-vocabulary (OOV) words.

Keywords: Neural Network, Urdu Transliteration, Translation

NEURAL AĞ KULLANARAK URDU'DAN RÖMENCE URDU'YA DÖNÜŞTÜRME

ÖZET

Harf çevirisi, bir kelimeyi bir dilin alfabesinden başka bir dile dönüştürme işlemidir. Daha önce kullanılan teknikler istatistiksel, kelime öbeği düzeyinde ve kural tabanlı yaklaşımlarla ilgiliyken, heterojen, ölçeklenebilir ve dinamik yapısı nedeniyle Nöral Makine Çevirisi (Neural Machine Translation) onun yerini almıştır. NMT, örneğin zengin kaynak dilleri için kullanılır. Almanca, Vietnam, Çince ve ayrıca Myanmar, Hintçe ve Roman-Urduca gibi zayıf kaynak dilleri için iyi performans gösteriyor. Urduca düşük kaynaklı bir dildir ve NMT modellerini kullanarak transliterasyon yapmak için yapılmış önemli bir çalışma yoktur. Bu yazıda, diziden diziye ve dikkat tabanlı modeller kullanarak Urduca'dan Roman-Urduca harf çevirisi üzerinde çalışıyoruz. Bu model, bir dili girdi (kaynak) olarak alan Kodlayıcı-Kod Çözücü mimarisini kullanır ve kod çözücü bunu istenen çıktıya (hedef) dönüştürür. Bağlama duyarlı ve ölçeklenebilir çözümler sağlayan sonuçları, zengin kaynak dilleri için olağanüstü. Dil harf çevirisi alanında, Uzun Kısa Süreli Bellek (Long Short-Term Memory) ve Çift yönlü modeller, uzun vadeli bağımlılıklarla etkili bir şekilde başa çıkmak için yaygın olarak kullanılır. Görünmeyen verileri işlemek için, Bayt Çifti Kodlama (Byte Pair Encoding) ve alt sözcük tekniklerinin bir kombinasyonu kullanılır ve hem sözcük hem de karakter düzeyi katıştırmalarını içeren hibrit bir yaklaşım kullanılır. Harf çevirisi sisteminin performansını değerlendirmek için sıfırdan oluşturulmuş 60 bin örnekten oluşan paralel bir derlem üzerinde deneyler yapılmıştır. Sistem, hiperparametrelerde ince ayar yapmak için kapsamlı testlere tabi tutulur ve sonuçta hem eğitim hem de test veri kümelerinde BLEU (İki Dilli Değerlendirme Öğrencisi) puanıyla ölçülen son teknoloji ürünü sonuçlara ulaşır. Ek olarak, NMT modeli ölçeklenebilir, sağlam, bağlama duyarlı bir yapı sağlar ve sözcük dağarcığı dışındaki (kelime dağarcığı dışında) sözcükleri işleyebilir.

Anahtar Kelimeler: Sinir Ağı, Urdu Transliterasyonu, Çeviri

ACKNOWLEDGEMENTS

I am deeply grateful to the numerous individuals who have played a pivotal role in making my years at graduate school immensely valuable. Foremost, I would like to express my sincere gratitude to Gülsüm Çiğdem Çavdaroğlu, my major professor and dissertation supervisor. Working alongside her throughout the years has been intellectually rewarding and fulfilling. I am also indebted to Şahin Aydin, whose contributions were invaluable from the early stages of my dissertation research. Gülsüm Çiğdem Çavdaroğlu made significant contributions to the development of the econometric model, and I am thankful for her insightful suggestions and expertise. I extend my heartfelt appreciation to the computer staff of the department, who patiently addressed my queries and assisted me with word processing issues. I would also like to acknowledge my fellow graduate student colleagues who provided unwavering support throughout the challenging years of coursework and exams. Lastly, I would like to express my deepest gratitude to my family. To my parents, Muhammad Nadir Khan and Zille Huma, I am immensely thankful for their patience and unwavering encouragement. Finally, I am eternally grateful to my wife, Rabail Sajid, whose unwavering support has been a constant source of strength throughout this arduous journey.

Wajahatullah KHAN

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS LIST

SMT: Statistical Machine Translation

NMT: Neural Machine Translation

RNN: Recurrent Neural Network

WP: Word Penalty

CSLM: Continuous Space Language Model

GNMT: Google Neural Machine Translation Model

OOV: Out of Vocabulary

LSTM: Long Short-Term Memory

BPE: Byte Pair Encoding

OCR: Optical Character Recognition

POS: Part of Speech

ITRANS: Indian Language Transliteration

BJM: Bi-directional Joint Model

SGD: Stochastic Gradient Descent

BLEU: Bilingual Evaluation Understudy

CHAPTER 1

1. INTRODUCTION

Language transliteration is a type of conversion from one script to another script using letters. Transliteration has high importance and it's being used in Greek, Latin Urdu, Hindi and Arabic languages (Ahmed, 2009). The major challenges of transliteration are to distinguish the syntax, semantics and morphology of the languages (Kyunghyun, et al., 2014) (Durrani, Sajjad, Fraser, & Schmid, 2010) (Gupta, Joshi, & Mathur, 2013). In the past, Statistical Machine Translation (Lagarda, Alabau, Casacuberta, Silva, & Diaz-de-Liano, 2009) and rule based techniques were commonly employed for language transliteration. However, these techniques have been suspended as they yielded low-performance results (Luong & Manning, Achieving open vocabulary neural machine translation with hybrid word-character models, 2016). Vocabulary size and unknown words handling are the major issues of these models. Similarly, for transliteration, rule based (Ahmed, 2009) (Gupta, Joshi, & Mathur, 2013) sound based, script based, and diacritics-based handlings were performed but these approaches are not able to handle long-term dependencies and semantics (Bahdanau, Cho, & Bengio, Neural machine translation by jointly learning to align and translate., 2015). With this evidence and to the best of our knowledge, at present, no significant work has been done for transliteration of Urdu to Roman Urdu using Neural Networks.

NMT is an advanced architecture that is emerging and now being used in language transliteration (Cho, Merriënboer, Bahdanau, & Bengio, 2014). It

uses encoder-decoder architecture for language transliteration. Its results are phenomenal for rich resource languages, providing context aware and scalable solutions (Luong, Pham, & Manning, 2015). Hence, this technique being used for low resource languages i.e. Arabic, Hindi etc., and it provides promising results (Alam & Hussain, 2017) (Durrani, Sajjad, Fraser, & Schmid, 2010). Urdu is mainly derived from Arabic and Persian in terms of vocabulary and has a close syntactic structure with Hindi, which motivated us to perform this work on Urdu language (Saini & Sahula, 2018). In this paper we used the sequence to sequence (RNN) model for model training. The input (Urdu) sequence of variable length is passed as input, which is converted to fixed length vector and then at decoder level, it converts the Roman-Urdu text into vector form and uses the encoder's last hidden layer output to initialize weights. Both layers results are passed to the projection matrix to generate the results. We used the tensor flow NMT1 model for language transliteration. The NMT model can handle the word alignments with the help of projection matrix (aka 2D alignment matrix) that maps the target word correctly and it provides one-to-Many and Many-to-one words alignment. We used the Long Short-Term Memory (LSTM) based cells that can handle the long term dependency and resolve gradient disappearing problem (Hochreiter & Schmidhuber, 1997) (Luong & Manning, Achieving open vocabulary neural machine translation with hybrid word-character models, 2016). The other used approach is the attention-based model, which adds direct connection between source and target words. Furthermore, this tensor flow NMT model can handle the out of vocabulary (OOV) words by applying sub-word (Sennrich, Haddow, & Birch, 2015) and Byte Pair Encoding (BPE) (Kirchhoff, 2018) technique to it, which is a powerful feature of this model.

Here, it is important to mention that we are using sentence level transliteration to transliterate the sentence based on its context. For example (عام/common can be written as "aam" whereas same word is used for (آم) – mango. Therefore, context awareness is required before transliteration. Length penalty affects the NMT model therefore we used an attention-based model that doesn't attempt to encode the whole input sequence in one go. On the other

hand, it chooses the set of input vectors that are relevant for transliteration with the help of context vector. Therefore, longer length sentences don't affect the model performance (Bahdanau, Cho, & Bengio, Neural machine translation by jointly learning to align and translate., 2015). Here we are using attention-based models with LSTM cells, hence it can handle long term dependencies issues (Hochreiter & Schmidhuber, 1997). Lastly for unknown words, it generates the relevant subwords based on the context of previous words. Another important reason for selecting this sentence level transliteration is that it can handle word alignment issues (Bahdanau, Cho, & Bengio, Neural machine translation by jointly learning to align and translate., 2015). For example, Islamabad is one word in Roman-Urdu whereas in Urdu اسالم آباد are two words. It can handle these words in an efficient way. In absence of an automated process for Urdu to Roman-Urdu transliteration, this research has high importance to generate Roman-Urdu among multilingual sites due to following factors. A medium is required for written communication between Urdu and Hindi speakers, as both the languages are almost same but different scripts. Latin script has more preference and is widely used in social media conversations like Facebook, Twitter, etc., among the speakers of the world. This work includes the development of an open source parallel-corpora for Urdu to Roman-Urdu with extensive data cleaning, which can be seen in Section 3.2. NMT and attention-based models are discussed in Section 3, which is used for the first time to transliterate from Urdu to Roman-Urdu. Out of vocabulary (OOV) words are handled smartly as can be seen in Section 3.6. Extensive hyper-parameter tuning is performed to achieve state of the art results, claimed in Section 4 and it includes the result and conclusion part as well.

[1] Tensor flow official code https://github.com/tensorflow/nmt

CHAPTER 2

2. MOTIVATION AND RELATED WORK

Urdu is a morphologically rich language, derived mainly from Persian and Arabic. Urdu speaking users do not differentiate between multiple letters of Urdu i.e. 'ع' AIn and 'ا' alif in transliteration due to same pronunciation of these letters in words (Akram & Hussain, Improving Urdu Recognition Using Character-Based Artistic Features of Nastalique Calligraphy, 2019), therefore Soundex algorithm can be used to handle these cases (Rajkovic & Jankovic, 2007). Another approach is Nastaliq2 which is used in OCR for features recognition. In this approach, characters are replaced with RASM classes3 that are used for sequence character recognition (Akram, Naseer, & Hussain, Assas-Band, an affix exception-list based Urdu stemmer, 2009). Some researchers used POS tagging and stemming approaches for language transliteration, but POS tagging doesn't differentiate well when we have more than one class (Abbas, 2014). Statistical Machine Translation (SMT) model is another approach used for transliteration, but it doesn't provide a track of missing words and doesn't support word alignment in different languages (Brown, Pietra, Pietra, & Mercer, 1993). However, these issues are partially solved using normalized-attention probabilities over the entire sentence to calculate the SMT reordering score. Brahmi-Net is an online platform that provides transliteration of Indian languages pairs including Urdu (Kunchukuttan, Puduppully, & Bhattacharyya, Brahmi-Net: A transliteration and script conversion system for languages of the Indian subcontinent, 2015). They used language scripts

(Unicode scripts) and ITRANS for language translation/transliteration at word level. SMT based models are passed to 13 languages and results are evaluated using accuracy metric. IJunoon is another site that is using a rule-based approach for language transliteration.

On the contrary, Neural Networks are widely used for language transliteration due to its diverse set of applications. Neural Machine Translation (NMT) uses LSTM based cells to avoid gradient disappearing issue (Kalchbrenner & Blunsom, 2013). The model is plotted to vector illustration using Word2Vec Toolset (Mikolov, Chen, Corrado, & Dean, 2013). This technique is applied to transform Roman-Urdu to Urdu transliteration using Recurrent Neural Network (RNN), which is related to our work (Alam & Hussain, 2017). Some researchers used this technique for Hindi transliteration, which is similar to Urdu Language. A variety of techniques have been used for unknown words handling (Kirchhoff, 2018). Some of those techniques include Word Penalty (WP) and another one is Continuous Space Language Model (CSLM) (Schwenk, 2007). In addition, NMT requires a huge amount of time for training and testing, especially when dealing with rare words. Therefore, Google Neural Machine Translation Model (GNMT) can be used to handle those issues (Wu, 2016). It directly connects the bottom layer of the decoder with the top layer of the encoder network to enable parallelism. They used low precision arithmetic for faster computation during the inference stage. With the help of subword units, they establish a good connection between characters and words during the decoding phase. GNMT uses residual connection which improves the gradient flow and allows training deep encoder and decoder networks. Another paper used an attention-based model for English to Persian translation (Mahsuli & Safabakhsh, 2017). They used the encoder vectors as initial weights. Similarly, the same attention-based model is used to translate English to German (Luong & Manning, Stanford neural machine translation systems for spoken language domains, 2015).

The NMT model can be trained using word level and character level approach. The author uses the NMT character level approach (bi-directional) for different languages during transliteration (Bahdanau, Cho, & Bengio,

Neural machine translation by jointly learning to align and translate, 2014). Some researchers used Bi-LSTM models with word and phrase-based models . In the NMT (Sennrich, Haddow, & Birch, 2015) model, one to many alignments can exist which can be handled by using Bi-directional Joint Model (BJM) (Yao & Huang, 2016). Now a days, multilingual networks are also used for better results. This paper consisted of a multilingual and bilingual comparison of model performance (Kunchukuttan, Khapra, Singh, & Bhattacharyya, 2018). Orthographically similar languages are passed as input and then identified the common features for language transliteration. Moreover, zero shot transliteration (unknown words handling) outperform the bilingual model because the model generalized well on unknown words.

Another popular approach is segmentation approach for unknown words like out of vocabulary. For each out of vocabulary word, average fact wise evidence (PMI) is conceded with the surrounding material from the sentence. Other used approaches are graph based re-ranking model (Kirchhoff, 2018) and document context level model (DCLM) (Ji, Cohn, Kong, Dyer, & Eisenstein, 2015), which were tested on different with English language. Continuous Bag of Words (CBOW) and Skip Gram models can also be used for unknown words (Mikolov, Chen, Corrado, & Dean, 2013). Hybrid model is another approach for unknown words, and it uses word and character level embeddings to handle rare words (Wang Ling, 2015). Beam search is a commonly used technique to address penalty coverage issues in language generation tasks. It encourages the generation of output sentences that are likely to cover all words in the source sentences.

Hybrid-based approaches play a significant role in handling complex languages, and they have shown promising results compared to single network models. In these approaches, Urdu is first taken as input and converted into a Finite State Model (FSM) to handle diacritics (Malik, Boitet, & Bhattcharyya, 2008). Another technique is the Sequential RNN (SRNN) model, which combines a Feedforward Neural Network (FNN) with an RNN (Oualil & Klakow., 2017) and aspect-based Opinion Target Expression (OTE) that is used to identify the polarity of the embedding words. Neural networks are being used in a

variety of applications like video semantics understanding (Pan, 2016), malware detection (Stokes & W., 2017) and language translation.

Sequence to sequence models is widely used for rich source languages i.e. English, Dutch, Chinese and its results are phenomenal. Therefore, this approach is also used for some low resource languages i.e. Hindi, Arabic (Malik A. , 2009) etc. The rule based/dictionary-based mapping only works well for seen dataset but can't handle the composite and unseen/rare words (Ahmed, 2009). The results of the rule-based approach are discussed in "Results section" where we can see that model performance is not so good for rare and unknown words. To the best of our knowledge till date there is no significant work done to apply sequence to sequence model for Urdu to Roman-Urdu transliteration (Alam & Hussain, 2017). Therefore, sequence to sequence and attention based models applied on Urdu language which addresses the issues of data variations, long term dependencies, gradient vanishing, and OOV words in an efficient way (Luong & Manning, Stanford neural machine translation systems for spoken language domains, 2015). It can also handle unknown/rare words using BPE and sub word approaches. Shortly, NMT model has the dominance over SMT and rule-based methods, which is not only providing scalable, context aware, and dynamic models but also widely being used in language translation due to its features.

[2] Nastaliq is the calligraphy for Arabic scripts which describes the shapes of character and joiners of Urdu language based on position.
[3] RASM classes are sequences of character classes and it contains dimensional features information.

CHAPTER 3

3. METHODOLOGY

In our proposed work, the idea of Neural Machine Translation (NMT) has been adopted for transliteration purpose, hence renamed with Neural Machine Transliteration (NMT) in our work and should be considered like this in

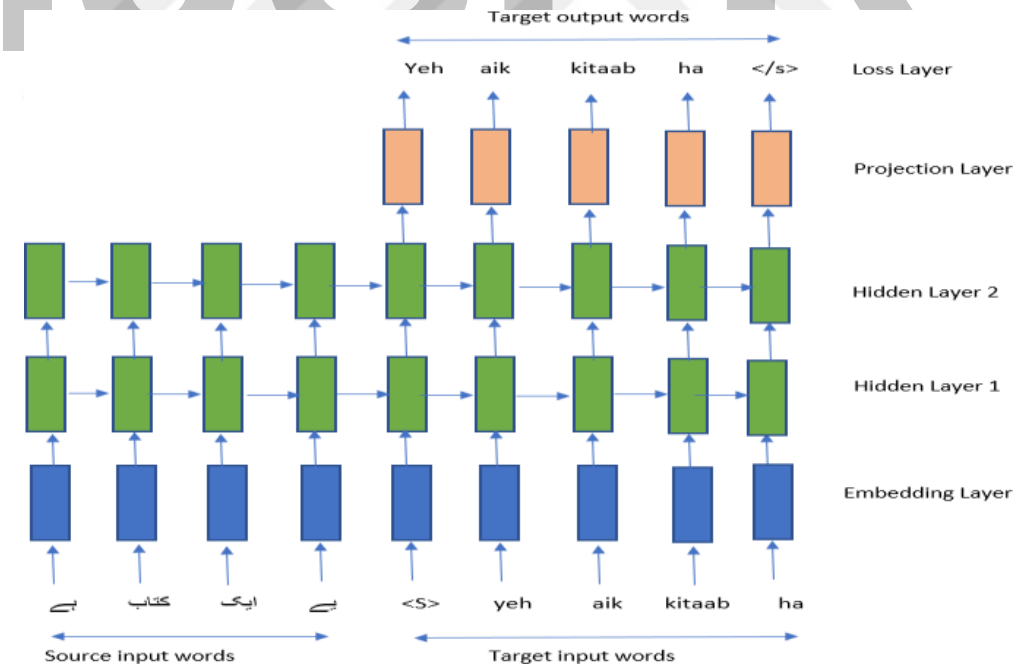

Figure 3.1 Sequence to Sequence NMT architecture

sections ahead. Our NMT is enhanced with an attention-based model discussed in Section 3.1 next. The raw corpora collected for Urdu and Roman-Urdu detailed in Section 3.2, are used for training of this NMT model, which provides

us separate embeddings (vectors) for Urdu and Roman-Urdu corpora. Next, we performed projections for the two embeddings corpora using a transliteration matrix commonly known as transformation or 2D alignment matrix as shown in figure 3.1. This transliteration matrix gives us the estimates of the expected locations of matching transliterations for words in Urdu corpus to Roman-Urdu corpus.

In NMT, we used the sequence-to-sequence model that uses encoder-decoder architecture for model's training as shown in Figure 3.1. The input sequence is passed as input and first it tokenizes it into words and then it generates one hot encoding vector, i.e. [1, 0, 0, 0] vector for the word 'yeh' in Figure 3.2 (Bahdanau, Cho, & Bengio, Neural machine translation by jointly learning to align and translate, 2014). Next in the embedding layer, vocabulary is attached for verification of the input sequence and it acts like a lookup table. Embedding weights are randomly initialized in the range (-1, 1) at start for input as can be seen in 4x4 matrix and the resultant weights are passed to hidden layers, which generate vectors using word2vec models' approach that can be seen in (Liu, 2017). At each timestamp, the model reads the input sequence from the corpus until it finds the EOS (End of Sentence). Now the decoder reads those vector weights of the encoder and passes it as input for transliteration. First it generates the word embedding with the help of target vocabulary and then this information is passed to the next hidden layer with encoder weights. The decoder vector is passed to a 2D alignment matrix which identifies the word alignments of source and target words and generates output. Then the output layer is passed to a softmax layer that uses conditional probability to compute the target sentence using Equation 3.1. The equation shows the computation of target word y, it checks the input word with all previously predicted targets and applies argmax on target vector.

$$P(y|x) = P(y^1|x)P(y^2|x,y^1)P(y^3|x,y^1,y^2) \ldots\ldots P(y^n|y^1,y^2 \ldots y^{n-1}x) \quad (3.1)$$

[4] The NMT model architecture is taken from tensor flow official

In the next step, the predicted word is passed as input and this process keeps going for the decoder sequence. The encoder decoder training example is

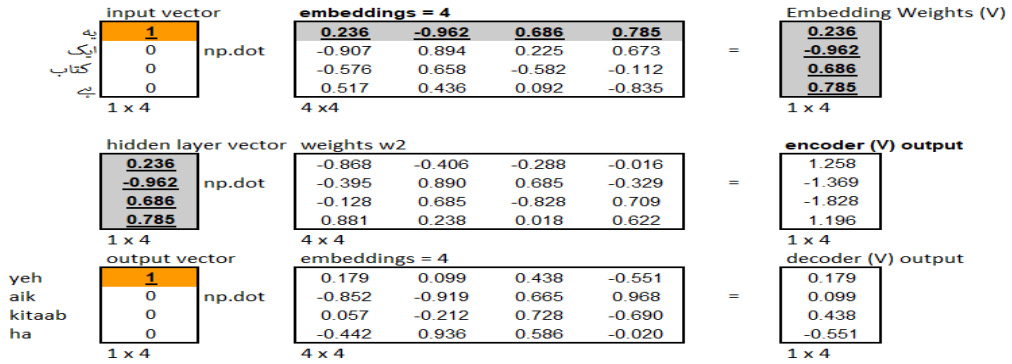

Figure 3.2 Encoder Decoder Example

shown in Figure 3.2 that illustrates the working. The above Figure 3.2 shows that input is converted into one hot encoding vector. Here the input length is 4 therefore 1x4 vector is generated which is multiplied by random weights [-1, 1] at start based on embedding layers size. In this example we used the 4 embedding layers and this 4x4 embedding matrix is multiplied with input vector and embedding weights generated of size 1x4. Here the horizontal vector is used which has 4 dimensions in the above example. Next, these embedding weights passed to a hidden layer where we took the dot product with weight vector W2 of size 4x4 and encoding vector of size 1x4 is generated. At this stage input is completely converted to index form. Now in the next decoder phase, we generated one hot encoding of the target sequence and after taking the dot product with embedding weights of size 4x4, the decoder vector is generated as shown in Figure 3.2. These encoder and decoder vectors are then passed to a 2D alignment matrix where it finds the word alignments between source and target vectors as shown in Figure 3. Finally, the result is passed to the softmax layer, which generates the final output.

Learning rate is an important hyper parameter that controls how much to change in response of estimated error reported during training. The learning rate is updated based on perplexity stats of the model. For each source sequence $f = f^{j}$ there is a target sequence $f = e^{j}$. We want to minimize the

negative log-likelihood or the cross entropy of input sequences $\{< f^{(s)}, e^{(s)} >\}_{s=1}^{s}$ and its objective function is given below

$$Objective\ function = J(\theta) = \sum_{s=1}^{S} \sum_{i=1}^{mi} \sum_{i \quad <i} -\log p\ (e^{s} |\ e^{(s)}; \theta) \tag{3.2}$$

The equation 3.2 states that for each distinct target sentence $y^{(1)} \ldots y^{(n)}$ with the sentence length $m_1 \ldots m_S$ we can compute the loss during training. At the start we used some initial learning rate and during model training, weights are updated in the opposite direction of gradient to minimize the loss. The computation cost is reduced by mini batches of size 128.To find the correspondence between source and target words, a 2D alignment vector is created that identifies the correspondence between encoder and decoder vectors. The correspondence can be one to one, one-to- many, many-to-one and many-to-many. In Figure 3.3, all these word vectors are placed in a dense matrix, which returns a single vector of size V where V is the vocabulary size of target sentence length. For each step, it computes the distance between these vectors based on selected context size. Next, our objective is to figure out the target vector, which has the highest similarity to the source vector. Here in Figure 3.3 we can see the dark cells correspond to the highest similarity between source and target words whereas light grey and while cells show the low similarity among them. The example shows that **آباد اسالم** (2 words) are mapping to a single word in Roman-Urdu so it can handle one-to-many and many-to-many alignments without any issues. Cosine similarity is used to determine the similarity between source and target words. It simply computes the distance

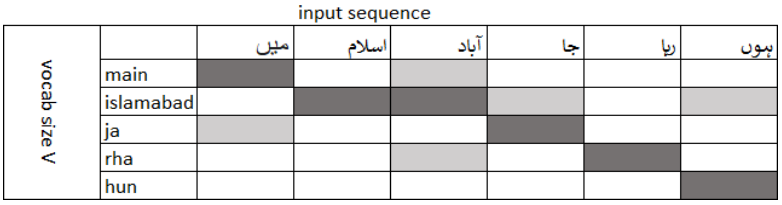

Figure 3.3 Word alignment between sentences from the source and the destination using a two-dimensional alignment matrix.

between source and target vectors and finds the neighbors based on similarity check. The process continues until it reaches the end of the sentence. The cosine similarity of two vectors can be computed with the help of following Equation 3.3. This equation shows that by taking the dot product of two vectors will show the similarity among them and then norm is taken to normalize the results. The main problem of this seq-to-seq model is that we need to track the long-term dependencies to transliterate the language model.

$$sim(x, y) = \frac{x.y}{|x|*|y|} \tag{3.3}$$

The decoder uses the whole last source state as input before starting the decoding process. For each step "t" we need to track the V_t combinations, therefore, the complexity of the model is O (V^T) where V is the vocabulary size of the source input sentence and T is the overall number of steps.

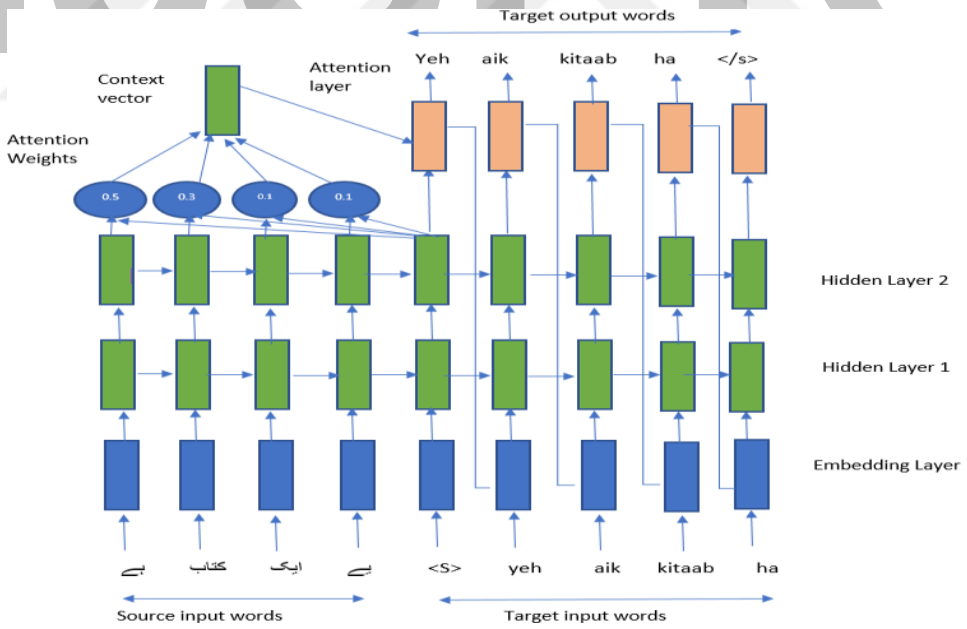

Figure 3.4 Example of Attention based architecture of NMT

3.1 Attention Based Model

As discussed in the above section in the sequence-to-sequence model, we need to track down all the possible combinations for final output, which is computationally expensive for long sentences. The last state of the encoder needs to store all the information of the source before decoding starts. Attention based models resolve these issues of sequence-to-sequence model by adding a straight connection between source and target language. It pays consideration to the relevant context vector, which needs to be transliterated. The attention-based architecture is shown in Figure 3.4. Here, we can see that Input is passed to the embedding layer and then attention weights are computed. In the next step, the weighted average of source states are combined and a context vector is created which is then combined with the current target state to generate the attention vector. Attention vector is then passed to previously discussed alignment matrix and softmax function is applied to get the final output. Luong (Luong, Pham, & Manning, 2015) proposes the multiplicative (where source and target vectors are multiplied with weights) whereas Bahdanau (Bahdanau, Cho, & Bengio, Neural machine translation by jointly learning to align and translate, 2014) proposes the additive style (where source and target vectors are added after multiplying with weights) for score finding of attention weights as shown in Equation 4. Here, h_s and h_t are source and target hidden states and tanh function is applied for normalization purposes. In Equation 3.5, each score of source to target is computed using exponent and then divided (for normalization purpose) with the sum of exponent scores of all the sources. In this way, the attention weights a_{ts} are computed. Next, the product of attention weight a_{ts} and source hidden state h_s are computed for all the source states and then summed up to achieve the context vector c_t as shown in equation 3.6. Finally, the target weights w_t, context vector c_t with decoder hidden state h_s is passed to a function tanh (input), which generates the final target attention vector at according to Equation 3.7. The attention output mostly gets information from such a hidden state, which has the high attention. This process continues until the end of sentence.

$$score\ (h_t, h_s) = \begin{cases} h_t W h_s & Luong's \\ v_a\ tanh\ (W_1 h_t + W_2 h_s) & Bahdanau's \end{cases} \quad (3.4)$$

$$attention\ weights = a_{ts} = \frac{exp\ (score\ (ht, h_s))}{\sum_{s=1}^{N} exp\ (score\ (h, h))} \quad (3.5)$$

$$context\ vector = c_t = \sum_s a_{ts} h_s \quad (3.6)$$

$$attention\ vector = a_t = f\ (c_t, h_t) = tanh\ (W_c\ [c_t; h_s]) \quad (3.7)$$

We used the LSTM for both encoder and decoder unit cells which can handle the long-term dependencies in an efficient way (Park, Song, & Kim., 2018). Both Luong and Bahdanau methods that are discussed in (Luong, Pham, & Manning, 2015) & (Bahdanau, Cho, & Bengio, Neural machine translation by jointly learning to align and translate, 2014) are applied to validate the model performance. Sequence-to-sequence model uses a forward feed method by default to predict the target but in some cases feedforward information is not enough, therefore, We also used a bi-directional model, which makes superior predictions by taking into account both the subsequent layer and the previously unseen data.

3.2 Data Gathering

To perform language transliteration, data is gathered from multiple resources using scraping packages (scrapy[5]). The important challenge of crawling sites is to find relevant sources, which can provide us with a parallel corpus of Urdu and Roman-Urdu as well[6]. Following are the sources used for collection of corpora.

In Table 3.1, we can see the Roman-Urdu has more vocabulary as compared to Urdu and it is due to the data variation issue.

[5] Scrapy is an open source framework that is used to scrape the data from websites https://scrapy.org/

[6] Parallel corpus https://github.com/wajahatuk/ru_u_corpus

For example, معافی 'forgiveness' can be written as **maafi, mafi, muafi, maafy**, etc., and all these versions can exist in Roman-Urdu but not in Urdu. The dataset is divided into training, holdout and testing sets. We used 75% of sentences as training, 10% for holdout and remaining 15% is used as a test set. The vocabulary is generated with the help of a training set after applying tokenization on it. This whole dataset is passed to the data-cleaning phase that is discussed in Section 4.

Table 3.1 Data Feed Information

Source	**Number of Sentences**	**Vocab Urdu**	**Vocab Roman-Urdu**
https://jang.com.pk/roman	45,000	26,000	43,000
https://ythisnews.com/	17,000	9,800	7,859
Total	**62,000**	**35,800**	**50,859**

CHAPTER 4

4. IMPLEMENTATION AND ISSUES

4.1 Data Cleaning

Data cleaning operations performed to make our corpus able for training on neural machine transliteration (NMT) are as follows:

Case Sensitive issue: NMT doesn't handle case-sensitive issues and mark the same word as <UNK>. Therefore, we need to apply transformation on Roman-Urdu dataset, which is applied by converting uppercase to lowercase. For example Islamabad vs islamabad should belong to the same word embedding in Roman- Urdu.

Deletion of special characters: Through data scraping, some special characters were present which needed to be deleted. These symbols include characters like $, %, #, (), @, & exist which require proper handling. Moreover, there were some encoding issues due to other languages i.e. Arabic or Persian appear in Urdu script.

Data abbreviations handling: Roman-Urdu is a slang language and can have abbreviations in sentences that need proper handling. This issue is handled smartly as acronyms normally appear in capital case; therefore these words are stored in our vocabulary. Some of the acronyms are PTI (Political Party), PMLN (Political Party), ISBP (Army Department), SBP (State Bank of Pakistan) etc.

Splitting sentence issues: In Roman-Urdu, some words and abbreviations normally end with period '.' i.e. Dr., Prof etc. Sentence separation of Roman-Urdu is mostly with period but there are numeric cases like 14.3, which need to be addressed. These cases are handled manually.

4.2 Unknow Words Handling

NMT model performs well as compared to conventional and phrase level models but it takes more computational time for training. Secondly, NMT model performance is not so good for rare/unknown words and it is a challenging task. Therefore, an algorithmic-based approach (sub word) is used in our model. The sub word is a method that performs breakdown on available vocabulary and tries to understand the words. It is the same as using character level embedding for rare words. It is similar as Byte Pair Encoding (BPE) but it creates new subwords by likelihood not by highest frequency pair. For example سربراہ-sarbrah is treated as

سر - sar, and براہ -brah, الزامات ilzamaat as الزام –ilzam and ات –at, etc.

The alternative method in use is called Byte Pair Encoding (Sennrich, Haddow, & Birch, 2015), which improves breakdown uniformity and lessens the issue of adding or removing characters during transliteration. It uses the vocabulary threshold so that the script only uses the character embedding for rare or unknown words. Its algorithm is as follows:

ALGORITHM 1: Byte Pair Encoding Algorithm:

a. Select the desired sub-word vocabulary size.

b. Split the word into sequences of characters and append suffix </w> (end of word) with word frequency.

c. Generate the new sub-word to the high frequency occurrences.

d. Repeat step 3 until reaching the target vocab size defined in step 1 or next highest frequency pair 1.

For example شہری 'Sehri/Citizen', شعبہ 'Shobah/Department', 'متعلق 'Mutalliq/about', جعلی Jaali/fake' have highest frequency pairs as شع، 'Sha', تع 'Ta' and in next iteration it will calculate the target word by adding these vocab as

well.

4.3 Urdu And Roman-Urdu Related Issues

Urdu language has rich morphological structure and due to that same word can appear in different ways. Roman Urdu, lacks any patterns and is a slang language. Data variation is the challenging task because Roman Urdu has multiple versions of the same word that exist i.e. words like raha, rha, raahaa belong to the single Urdu word رہا. These words vectors have high similarity to each other and we need to filter the correct pair among those cases. Secondly, one-to-many and many-to-one word alignment exists for Urdu to Roman-Urdu transliteration, which 2D alignment matrix caters i.e. PTI will map to 3 words (ٹی پی آئی)in Urdu. Realignment of sentences is not a big deal for NMT models. Thirdly, Urdu is one of the low resource languages and its parallel corpus is not available. Another issue we faced during data cleaning was the text delimiter issue for both Urdu and Roman-Urdu. Urdu sentences mostly end with dash (-) but it can be appeared to combine sentences. Similarly, for Roman-Urdu dot (.) can be appeared for abbreviations like P.T.I, Dr. Prof. and also for numeric values 23.5 which requires proper handling. Understanding context is very important for meaningful transliteration. NMT predicts the correct word based on the alignment vector which adds significant improvement in the model accuracy. For example, if the rare words are passed as input then it first maps it to the nearest matched word (using sub word and other techniques) based on word alignment and returns correct results.

4.4 Rule Based Approach For Urdu To Roman-Urdu Transliteration

Urdu is one of the languages that has a rich morphological structure. Therefore, no exact one to one mapping exists between Urdu and Roman-Urdu language vocabulary, especially for vowel cases it's much more complicated. For this rule based approach, a data dictionary (character level) is created with the help of Unicode characters of both languages. The dataset details are shared in table Table 3.1 where we also added special characters to improve the

results. Most common Roman-Urdu mapping issues are diacritic handling, vowels and consonant (i.e. Y can be treated as a vowel as well as consonant based on case) handling, different versions of the same word in Roman-Urdu (i.e. raha, rahha, rha etc) and double roman letter (germination) (Ahmed, 2009). Here we used the simple approach to perform transliteration using that dictionary mapping. Same dataset is passed to a rule based approach without applying any filters. The data also contains abbreviations, complex and rare words as well. Finally, results are evaluated using word and character level BLEU score between predicted vs reference words.

CHAPTER 5

5. EXPERIMENTS & RESULTS

Rigorous testing performed for hyper parameters tuning. We used Google Colab TPU's for model training. The dataset is separated into three sections, training, holdout and testing dataset. In our experiment, 75% dataset used for training, 10% for holdout and residual 15% used as test dataset. This holdout dataset helped us to estimate the performance of the final model.

Following model settings are selected after doing thorough testing:

- There is no fixed length of sentences used. Input and target sequence length is not the same in most of the cases.
- For each layer 128 LSTM cells used for the model.
- Context window is selected as 5, which means it checks 2 words from left and 2 words from right of the current word while doing transliteration.
- Step sizes are selected in the range of 12k-30k (~12- 30 Epoch) after performing extensive testing. We used the learning decay rate that decreases the learning rate after every 1000 steps. The initial value of learning rate is 1.0 for SGD and 0.001 for Adam optimizer.
- Number of unseen layers is nominated as 2 for encoder and decoder.
- Adam[7] and Stochastic Gradient Descent (SGD)[8] optimizers have been used for the model training.
- Results are evaluated by using Bi-lingual Evaluation Understudy (BLEU). We used the sentence level BLEU score to evaluate the

transliteration results which uses cumulative n-gram models (unigram, bigram, trigram and 4-gram). We assigned equal weights to these n-gram models and it passed as a tuple. i.e. 1-gram (1, 0, 0, 0), 2-gram (0.5, 0.5, 0, 0), 3-gram (0.33, 0.33, 0.33, 0) and 4-gram (0.25, 0.25, 0.25, 0.25) sentence_bleu (reference, candidate, weights = (0.25, 0.25, 0.25, 0.25))

- Initially for the training model, weights are assigned between [-0.1, 0.1] range. Weights are uniformly assigned to both encoder and at decoder level we are using encoder output as weights.

Embedding size of 512 is selected which gives good results. This value was selected after extensive testing. The model takes around 2-3 days for model training with Google GPU's.

5.1 Quantitative

The results are evaluated using BLEU score on training and testing dataset. The configuration details for BLEU are shared in the above paragraph. We used the preexisting rule based approach that is discussed in section 4.4 (Ahmed, 2009). It uses one to one mapping for transliteration. Mapping dictionary is passed as vocabulary and based on it, we achieved the BLEU score of 27.18% at character level transliteration. The performance is low because there are cases where we have vowels, germination, diacritics, Roman-Urdu dataset variation issues and it requires complex rules to handle those cases. From test results we observed that there is a high percentage of cases where diacritics and vowel words caused low match rate. We haven't applied complex rules in our approach. Second rule based approach doesn't generate context aware results that's why this mapping generated wrong results in some cases. Thirdly, it can't handle the abbreviated words i.e. PTI, ISBR, WHO etc.

[7] Adam is an optimizer which updates network weights using an iterative approach. It's also called an adaptive learning approach.
[8] Stochastic Gradient Descent is also an optimizer that keeps the same learning rate during training.

Word alignment (1 to many) is another issue affecting the results. Here BLEU score evaluates the correspondence of predicted and target words by counting the number of matches.

Table 5.1 Rule Based Approach Results

System	**Test BLEU Score %**
Unicode mapping using character level approach	27.3%
IJunoon (Roman-Urdu to Urdu Results)[9]	86%
Brahmi-Net REST API[10]	29.87%

5.1.1 Comparison of Ijunoon Results

There is no doubt that the IJunoon rule based approach handles a lot of cases. Therefore in order to compare and evaluate the model performance, we ran our test dataset on the IJunoon approach and figured out some cases where it's not working well. First case is that it cannot handle complex words and return the same input for those cases. Secondly, in Urdu language, some words can be written in compound format (7th column word) and it didn't generate the transliteration for those words. Thirdly as they are using word level approach, therefore their approach is unable to transliterate those words. Also acronym handling and compound words (i.e. محرم الحرام) is not available in this approach. We also observed that in some cases its handling compound words (i.e. Islamabad) but when we run it on our corpora, we figure out a lot of cases where the model is not performing well. Some of those cases are also shared below Table 5.2.

Table 5.2 IJunoon Results

Input (Ur)	ڈرائنڈنگ	سمبھالے	رہنماؤں	ہرامن	ردعمل	طاہرالقادری	ریکارڈنگ	محرم الحرام
Output (RU)	ڈرانڈنگ	سمبھالے	رہنماؤں	pur aman	ردعمل	طاہرالقادری	ریکارڈنگ	Mehram alhram

5.1.2 Comparison of Brahmi-Net Approach

In the next step, we compared the results of the Brahmi-Net approach discussed in paper (Kunchukuttan, 2015). The results are evaluated on our test dataset. The test sentences are passed to API and results are compared with the original dataset. From the test results, we achieved a BLEU score of 29.87% which is not really good. After further analysis we figure out the reason for the low results. First reason is that this approach is not handling diacritics and vowels, hence, even for common Urdu words, it generates wrong transliteration. The wrong transliteration results are showing below. The vowels handling is mandatory to achieve good results because same letter (Alif ا) can be mapped to different roman-urdu letters (i.e. A, I). As we are using the same test dataset for model evaluation, we figure out compound words are not handled in this approach. Whereas our neural network approach is able to perform transliteration based on context and can handle word alignment without any issues like in Table 5.3.

Table 5.3 Brahmi-Net Results

Input (ur)	بعد	ممکن	مرض	ےس	ہوں	گ	صحاف	ٹرین	اضافہ	وکال
Output(ru)	Bed	mumcon	marji	Tse	Honts	igm	shaphy	train	ajaophy	Okla

5.1.3 Comparison of Nmt Model

On the other hand, Table 5.4 is showing the sequence to sequence model results. Because Vanilla RNN uses greedy approach and takes more time on long sequences, therefore, this model is applied on small chunks. Table statistics show poor performance of the NMT model because we chose a small chunk for training purposes to reduce time. Second, we passed long sentences with a sentence length of 65 words so the model performance remained poor due to issue of long-term dependencies as can be seen through the low scores.

[9] https://www.ijunoon.com/transliteration/urdu-to-roman
[10] https://www.cfilt.iitb.ac.in/brahminet/

Table 5.4 Squence to Sequence RNN Results

Model Name	**Scoring Function**	**Epoch**	**Train BLEU (%)**	**Test BLEU (%)**
Vanilla RNN	Single NMT	18	4.1	0.8
Forward Scaled Luong Model Epoch 18	Scaled Luong	18	36.8	36.8
Forward Scaled Luong Model Epoch 24	Scaled Luong	24	37.8	37.8
Forward Scaled Bahdanau 18 Epoch	Scaled norm Bahdanau	18	31.2	42.8
Bi-direction normed Bah-danau Epoch 20	Normed Bah-danau	20	49.3	72.1
Attention Based Model (Word level)	BPE	30	53.5	31.3
Attention Based Model on Unseen data (Character Level)	Attention based (BPE)	18	40	27.2

The attention based model results are also listed in the above table, which has different scoring functions (Luong and Bhadanau) and Epoch. This attention based model reduces model time and it can handle long sequences with the help of LSTM cells. We tried different scoring functions (Bahdanau, Normed Bahdanau, Luong, and Scaled Luong) for model training. The result shows that after 18 -30 Epoch, model highest score is achieved on Bahdanau bi-directional model with epoch 20. The vocab size is fixed for all these attention-based models.

Due to static vocabulary issues, some of the words are mapped with unknown tokens. Therefore we picked a subset of unknown/rare words and applied sub-word and BPE algorithms to it. It converts rare/unknown words to base words

as stated in Table 5. Results are showing that the NMT model produces a test score of 27.2 on unseen testing data, which is good performance on unseen data. Figure 5.1 shows that only increasing the step size is not enough but a proper hyper-parameter tuning is required to achieve a high performance. At 20 Epoch, we achieved a high BLEU score of 72.1. We can say that the proposed attention based model performs well on Urdu text and it can handle unseen words.

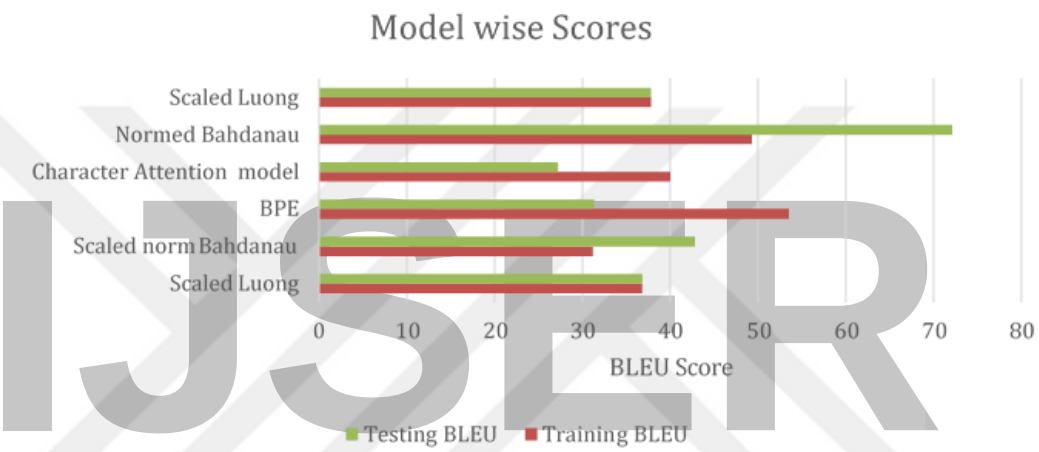

Figure 5.1 Step Size vs BLEU Score

5.2 Qualitative

The results are evaluated using the BLEU score as shown in Table 5.4. In this table we have added different cases of different sentence length, rare words and unknown words. We can conclude these points as follows based on transliteration results.

1. The model is able to transliterate most of the sentences using the NMT and attention-based model. We achieved a BLEU score of 72.1 on test data, which is promising yet.
2. As we are using RNN models, there is no restriction on source and target sequence lengths. Generally, the length is not the similar for the source and the target sentences.
3. Based on results we observed that the model can handle one-to-many

alignments or vice versa cases without any issues. With the help of an attention based model, we can handle these cases whereas a dictionary based/rule-based approach can only handle the one-to-one cases. Some of the abbreviations passed to the models are PTI (abbreviation for the name of political party), JIT (abbreviation of a commission) etc.

4. Roman Urdu is a slang language and abbreviations may exist in dataset i.e. PTI, NA, Dr, Prof, ISBR, etc. The model was trained on these abbreviations and in test results; it not only maps words with correct abbreviation but also provided satisfactory results. Existing techniques transliterate آی words but our model predicts words like this correctly as a single word as P T I three words but our model predicts words like this correctly as a single word.

5. Some researchers have addressed coverage issues by restricting the sequence length. However, in our model, we evaluate the results on both static and dynamic input lengths, including sentences with more than 50 words. While increasing the length may result in longer computation times, our model has no limitation on sentence length. This is possible due to the attention-based model (LSTM), which effectively handles long-term dependencies without any issues. We experimented with a maximum sentence length of 65 words and observed good performance even on such lengthy sentences.

6. In our model, we observed that some erroneous words were predicted, as shown in Table 6. These predicted erroneous words exhibit a close relationship with the correct word, indicating the model's understanding of the context. Additionally, to handle Out of Vocabulary (OOV) words, we applied word segmentation (subword) and Byte Pair Encoding (BPE) techniques. This resulted in a BLEU score of 27 on unseen data.

7. Given the rich morphology of Urdu, we made efforts to cover common multifaceted cases of transliteration from Urdu to Roman Urdu. It was observed that the model predicts the most common words more precisely compared to rare words.. For the rare words, we used subwords and BPE approaches. Some of the less frequent words are سکاٹلینڈ Scotland, تقرری ‘taqarruri/hiring’, احمقانہ ‘ehmaqana/foolish’, الزامات ‘ilzamaat/blame’ etc., are listed in Table 5.5.

Table 5.5 Qualitative result set

Test set Input (Urdu)	Test set Output (Roman-Urdu)
عمران خان آج کراچی میں مارچ کی قیادت کرینگے	Imran Khan aaj Karachi mein March ki qayadat karinge
طارق فاطمی نے الزامات کو بے بنیاد قرار دے دیا	Tariq Fatimy ne ilzamaat ko be bunyaad qarar day diya
پاکستان اور ویسٹ انڈیز کے درمیان ٹیسٹ کا آغاز	Pakistan aur West Indies ke darmiyan test ka aaghaz
افغانستان میں داعش کا سربراہ شیخ عبدالحسیب مارا گیا	Afghanistan mein Daish ka sarbarah Sheikh Abdul hisaab mara gaya
Erroneous Cases	
چمن حملہ، سراج الحق کا افغانستان سے معافی کا مطالبہ	Chaman <**unk**> Siraj ul <**unk**> ka Afghanistan se maffi ka mutalba
جے آئی ٹی پر اثر انداز ہونے کی کوشش احمقانہ ہے، سعد رفیق	JIT par assar andaaz honay ki koshish ehmaqana hai , Saad Rafeeq
سکاٹلینڈ یارڈ نے سرفراز مرچنٹ کو رقم واپس کر دی	Scotland yard ne **Sarfaras** Merchant ko raqam wapas kar di
لاس اینجلس میں مزیدار اس کریم سے بنا ٹھنڈا یخ میوزیم	Los Angeles mein mazedar ice kareem se bana thanda yakh museum
عدت کی کاروائی، لشکر جھنگوی کراچی کا امیر دلدار عرف چچا ہلاک، بھاری مقدار میں اسلحہ و بارود برآمد	**Ikhlaq** ki **nigaran**, <**unk**> **bhikaran** Karachi ka Ameer <**unk**> urf <**unk**> halaak , bhaari miqdaar mein asleha o barood baraamad
پی ٹی آئی میں بعض ایسے لوگ شامل ہوگئے جن کا نام پہلی بار سنا، اسد عمر	PTI mein baaz aisay log shaamil hogaye jin ka naam pehli baar suna , Asad umar
سپریم کورٹ کا تاریخی فیصلہ، وزیراعظم تاحیات نااہل قرار	<**unk**> <**unk**> ka tareekhi <**unk**> <**unk**> e Azam ta hayaat <**unk**> ahal qarar
Longer Dependency Sentence	
بظاہر ایسا معلوم ہوتا ہے کہ پولیس کو نشانہ بنانے کی کوشش کی گئی	Bazahir aisa maloom hota hai ke police ko nishana bananay ki koshish ki gayi
بھارتی فلم انڈسٹری کی سپر اسٹار اور سابقہ ورلڈ ہرکا چوہڑہ نے کل اپنی 36ویں سالگرہ منائی، انہیں اکثر و بیشتر اپنی گہری رنگت کی وجہ سے امتیازی سلوک کا سامنا کرنا پڑتا ہے	Bharti film industry ki super star aur Sabiqa **education** world Priyanka chopra ne kal apni 36 win saalgirah manayi , inhen aksar o beshtar apni gehri rangat ki wajah se imtiazi sulooq ka saamna karna parta hais

CHAPTER 6

6. CONCLUSION AND FUTURE WORK

We transliterated from Urdu to Roman-Urdu using the NMT style, which has been increasingly popular in recent years. The outcomes of the existing models' usage of statistical and rule-based methods for transliterating Urdu are also good. This neural model is primarily used because it has a dynamic design that produces rich, context-aware outputs and can manage long-term dependencies while transliterating languages. A strong aspect of this model is its ability to perform effectively on data that hasn't been seen before. It also runs smoothly on languages with few resources. Finally, using the training and testing datasets, we obtained BLEU scores of 52 and 72, which adds a considerable improvement in the transliteration of Urdu into Roman-Urdu. In future, we will use multi-lingual models (BERT) and transformers (Multilingual and Bi- lingual) models to improve model performance. These models perform training based on generic vocabulary of different languages and this generic model produces phenomenal results.

REFERANCES

Abbas, Q. (2014). Semi-Semantic Part of Speech Annotation and Evaluation. Proceedings of LAW VIII - The 8th Linguistic Annotation Workshop, August 2014, pp. 75-81. Association for Computa-tional Linguistics and Dublin City University, Dublin, Ireland.

Ahmed, T. (2009). Roman to Urdu transliteration using word list. Proceedings of the Conference on Language and Technology, vol. 305, p. 309.

Akram, Q. U. A. and Hussain, S. (2019). Improving Urdu Recognition Using Character-Based Artistic Features of Nastalique Calligraphy. *IEEE Access*, *7*, 8495-8507.

Ain, Naseer, A. and Hussain, S. (2009). Assas-Band, an affix-exception-list based Urdu stemmer. Proceedings of the 7th Workshop on Asian Language Resources (ALR7), pp. 40-47, Suntec, Singapore.

Alam, M. and Sibt ul Hussain (2017). Sequence to sequence networks for Roman-Urdu to Urdu transliteration. Proceedings of International Multi-topic Conference (INMIC), pp. 1-7.

Auli, M., Galley, M., Quirk, C. and Zweig, G. (2013). Joint Language and Translation Modeling with Recurrent Neural Networks. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1044-1054, Seattle, Washington, USA.

Bahdanau, D., Cho, K. and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint, arXiv*:1409.0473

Bengio, Y., Ducharme, R., Vincent, P. and Jauvin, C. (2003). A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13.

Brown, P. F., Pietra, V. J., Pietra, S. A. and Mercer, R. L. (1993). The mathematics of statistical machine translation. Parameter estimation. Computational Linguistics, Susan Armstrong (Ed.) In Using Large Corpora,19, (pp. 223-224), London, England: MIT.

Cho, K., van Merrienboer, B., Bahdanau, D. and Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103-111, Doha, Qatar.

Durrani, N., Sajjad, H., Fraser, A. and Schmid, H. (2010). Hindi-to-Urdu Machine Translation through Transliteration. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 465-474, Uppsala, Sweden.

Gupta, V., Joshi, N. and Mathur, I. (2013). Rule based stemmer in Urdu. Proceedings of 2013 4th international conference on computer and communication technology (ICCCT), pp. 129-132, IEEE.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), pp. 1735-1780.

Ji, Y., Cohn, T., Kong, L., Dyer, C. and Eisenstein, J. (2015). Document Context Language Models. *arXiv preprint, arXiv*:1511.03962

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent Continuous Translation Models. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, October, 2013, pp. 1700-1709, Seatle, USA.

Khan, W., Daud, A., Khan, K., Nasir, J. A., Basheri, M., Aljohani, N. and Alotaibi, F. S. (2019). Part of Speech Tagging in Urdu: Comparison of Machine and Deep Learning Approaches. *IEEE Acces*s, *7*, 38918-38936. https://doi.org/10.1109/access.2019.2897327.

Liu, A. and Kirchhoff, K. (2018, January 25). Context Models for OOV Word Translation in Low-Resource Languages. *arXiv preprint, arXiv*:1801.08660

Kunchukuttan, A., Khapra, M., Singh, G. and Bhattacharyya, P. (2018). Leveraging Orthographic Similarity for Multilingual Neural Transliteration. *Transactions of the Association for Computational Linguistics*, *6*, 303-316.

Kunchukuttan, A., Puduppully, R. and Bhattacharyya, P. (2015). Brahmi-Net: A transliteration and script conversion system for languages of the Indian subcontinent. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp. 81-85, Denver, Colorado.

Kyunghyun, C., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H. and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724-1734, Doha, Qatar.

Lagarda, A. L., Vicent Alabau, Casacuberta, F., Roberto Nascimento Silva, and D'iaz-de-Liaño, E. (2009). Statistical post-editing of a rule-based machine translation system. Proceedings of the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers, pp. 217-220, Seattle, USA.

Lemao, L., Finch, A., Utiyama, M. and Sumita, E. (2016). Agreement on target bidirectional lstms for sequence-to-sequence learning. In Proceedings of the AAAI Conference on Artificial Intelligence, 30(1). https://doi.org/10.1609/aaai.v30i1.10327.

Liu, H. (2017). Sentiment analysis of citations using word2vec. *arXiv preprint, arXiv*:1704.00177.

Luong, M. T., Pham, H. and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412-1421, Lisbon, Portugal.

Luong, M.-T. and Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domains. In Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign, pp. 76-79, Da Nang, Vietnam.

Luong, M.-T. and Manning, C. D. (2016). Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, vol 1, pp. 1054-1063, Berlin, Germany.

Mahsuli, M. M. and Safabakhsh, R. (2017). English to Persian transliteration using attention-based approach in deep learning. Proceedings of 2017 Iranian Conference on Electrical Engineering (ICEE), pp. 174-178, Yazd, Iran.

Malik, A. (2009). A hybrid model for Urdu Hindi transliteration. Proceedings of 47th Annual Meeting of the Association of Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of NLP ACL/IJCNLP Workshop on Named Entities (NEWS-09), pp. 177-185, Singapore.

Malik, M. G., Boitet, C. and Bhattcharyya, P. (2008). Hindi Urdu Machine Transliteration using Finite-state Transducers. In Proceedings of 22nd International Conference on Computational Linguistics (COLING), pp .537-544, Manchester, United Kingdom.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint, arXiv*:1301.3781

Oualil, Y. and Klakow., D. (2017). A Neural Network approach for mixing language models. Proceedings of 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5710-5714, New Orleans, USA.

Pan, e. a. (2016). Jointly modeling embedding and translation to bridge video and language. Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4594-4602, Las Vegas, USA.

Park, S., Song, J.-H. and Kim., Y. (2018). A Neural Language Model for Multi-Dimensional Textual Data based on CNN-LSTM Network. Proceedings of 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 212-217, Busan, South Korea.

Rajkovic, P. and Jankovic, D. (2007). Adaptation and application of daitch-mokotoff soundex algorithmon Serbian names. Proceedings of XVII Conference on Applied Mathematics.

Saini, S. and Sahula, V. (2018, March). Neural Machine Translation for English to Hindi. Proceedings of 2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP), pp. 1-6, Kota Kinabalu, Malaysia.

Schwenk, H. (2007). Continuous space language models. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pp. 723-730, Sydney, Australia.

Sennrich, R., Haddow, B. and Birch, A. (2015). Neural Machine Translation of Rare Words with Subword Units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics vol. 1, pp. 1715-1725, Berlin, Germany.

Stokes, B. A. and W., J. (2017, March). Malware classification with LSTM and GRU language models and a character-level CNN. Proceedings of 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp. 2482-2486, New Orleans, USA.

Wang Ling, e. a. (2015). Character-based neural machine translation. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, vol.2, pp. 357-361, Berlin, Germany.

Wu, Y. e. , Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W. and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint, arXiv:*1609.08144.

Yao, Y., & Huang, Z. (2016). Bi-directional LSTM recurrent neural network for Chinese word segmentation. Proceedings of Neural Information Processing: 23rd International Conference, ICONIP 2016, pp. 345-353, Kyoto, Japan.

CURRICULUM VITAE

Wajahatullah Khan received his BS degree in Information and Communication Systems. He has a lot of experience in various data roles, including Data Engineer, Database Developer, Data Analyst, and BI Engineer, spanning industries such as telecom, banking, R&D, e-commerce, digital marketing, and online classifieds. Demonstrated experience in managing, architecting, and analyzing big data to develop data-driven insights and high-impact data models that drive business growth and innovation. Proficient in multiple cloud services (AWS, GCP, Azure), with a primary focus on GCP, and hands-on experience with cloud-native distributed systems and MPP databases. Designed and implemented enterprise-level, petabyte-scalable data stacks using cloud-native technologies such as AWS Redshift and Greenplum. Expert in SQL, Python, Shell, and Java programming, specializing in data handling libraries. Familiar with software development best practices, including CI/CD and Agile methodologies.