Syllabified Sequence-to-sequence attention for transliteration

G.Vyshnavi
vgutta7@gatech.edu

G.Sridevi
drlsridevi@vrsiddhartha.ac.in

M.Ritesh
ritesh.s.m@ril.com

P. Krishna Reddy
pkreddy@iiit.ac.in

Abstract

The problem of transliteration deals with the phonetic transcription of text from a source writing system into a target writing system. With the inception of neural net models like Sequence-to-sequence networks, transliteration has seen significant progress in the last decade. However, the accuracy of such systems is still far from ideal. This is made more appar- ent when the source text to be transliterated itself is entered incorrectly by intermediary users, further degrading the performance. In this paper, we propose Syllabified Sequence- to-Sequence network (Syll-S2S) towards im- proving the transliteration quality from Roman script to low-resource Indic scripts like Devana- gari. As part of this, we present the rules of Sonority sequencing principle to Devanagari and other Indic scripts. In addition, the proposed framework incorporates Elastic Search stack which maps incorrect transliterations to their existing reference transliterations for handling erroneous entries of source texts. Experi- ments demonstrate significant performance im- provement of the proposed framework with re- spect to the existing schemes.

1 Introduction

English is one of the most widespread foreign languages in India, a home to 22 official languages and more than 1000 dialects written in more than 14 different scripts. With the rapid advancements in worldwideweb and mobile devices, people in India create, share, tag and search multifaceted data multi- lingually but mostly using the Roman or Latin script (Chanda et al., 2010) across different mediums. Thetext written in a native language, but using a non- native script like Roman, mostly does not follow any standard spelling rule, but uses the orthography of the script based on pronunciation of the words. This process of phonetically transcribing a word or text from one writing system into the another writing system such that the pronunciation of the word remains same is called Transliteration.

Transliteration is a part of Natural Language Processing (NLP) and has several useful applications; Cross language information retrieval, Machine translation etc. It has wide ramifications in low-resource languages in general, where web presence is limited, specifically for Indian languages. A substantial por- tion of textual data being generated and queried upon the web belongs to the transliteration domain, thus containing a good amount of information and therefore needs to be studied.

In this paper, we focus on transliteration from the most commonly used Roman script to native Indian scripts. Most of the major Indian language scripts are derived from the ancient Brahmi script and consequently are highly phonetic in nature. Hence, we primarily focus on the task of transliteration from Roman to Hindi. Hindi, an Indo-Aryan language, written in Devanagari, is the lingua-franca of India. We therefore consider Hindi as the primary link for Roman to native Indian script transliteration, as the quantity of Hindi literature (especially online) is more than twice as in any other Indian language.

For the Roman-Hindi transliteration task, the earliest methods are rule based approaches (Goyal and Lehal, 2009; Kang and Kim, 2000; Jia et al., 2009) which involved character, phoneme and grapheme matching between the parallel transliteration corpora. But the rule based approaches fall short due to the several exceptions possible. Another popular method is Anoop et al.'s Indic-nlp (Kunchukuttan et al., 2020), which is a statistical machine translation based approach relying on language model training. Recently, Sequence-to-sequence attention

based networks (seq-to-seq) proposed in (Sutskever et al., 2014), have garnered wide attention for machine translation (Bahdanau et al., 2014; Luong et al., 2015; Vaswani et al., 2017) and works (ud Din, 2019; Ameur et al., 2017; Mandal and Nanmaran, 2018) extended the same for the transliteration task. However, the existing seq-to-seq models too have failed in producing desirable results.

In this paper, we develop a Syllabified Sequence-to-sequence net (Syll-S2S) for improving the quality of Roman to Devanagari transliteration w.r.t. the state-of-the-art works. The proposed approach uses Sonority Sequencing Principle (SSP) to get the syl- lables of the source (and target data during training) thus enabling the enforcement of syllable-syllable at- tention. This way, the model would be able to learn the target data with a greater precision and would also enable faster knowledge acquisition than con- ventional seq-to-seq models.

Additionally, we consider an important practical aspect with regards to transliteration which is the user input. In several applications, the primary source of the input text in Roman is derived from an intermediary user. In such scenarios, the user given input word may have slight variation from its correct form in the supposed cases of well-established words like dictionary-based-words or named entities. For handling such cases, we incorporate Elastic Search stack (ES stack); a distributed, open source search and analytics engine (Gormley and Tong, 2015). In the case where reference transliterations are available, we use fuzzy search query (Gu et al., 2018) on ES stack with added constraints on consonant match for mapping the model output transliterations and the reference transliteration.

The main contributions of this paper are 3-fold.

1. We develop a novel Syllabified Sequence-to-sequence model (Syll-S2S) for improving the transliteration quality from Roman to Devanagari in comparison to the state-of-the-art.

2. In the case where true transliterations in Devanagari are available, we incorporate fuzzy searching on user-given Roman input for accurate matching to its indexed reference transliteration with Elastic Search stack.

3. We have demonstrated the superiority of the proposed approach with extensive performance evaluation.

The remainder of this paper is organized as follows. In Section 2, we present the related work. We present the background of seq-to-seq net, ES stack and inverse mapping in Section 3 followed by proposed approach in Section 4. Experimental results and conclusions are presented in Section 5 and Section 6 respectively.

2 Related work

In this section, we briefly describe few of the existing works which do transliteration.

Methods in (Goyal and Lehal, 2009; Kang and Kim, 2000; Jia et al., 2009) incorporate rule-based transliteration which can again be divided into 3 categories. (1) Character mapping approach (Goyal and Lehal, 2009) uses character mapping for doing transliteration. Under this approach, the characters of source script are mapped to those of the target script on the basis of pronunciation. Character mapping does not give very good results as the pronunciation of characters and the total number of character varies from script to script. (Kang and Kim, 2000) uses (2) Phoneme Based Approach which defines the relation and correspondence between the phonemes of the source and target script. An alignment of the phoneme for the characters of source script to the phoneme of the target script is done using methods like language modeling (Chelba and Jelinek, 2000). (Jia et al., 2009) uses Phoneme Based Approach by defining the relation and correspondence between the (3) graphemes of the source and target scripts.

The second class of works is based on statisti- cal machine translation (SMT). Anoop kunchkuttan et al. propose Indic-NLP in (Kunchukuttan et al., 2020) and develop a SMT approach with a Moses de- coder for transliteration. Moses (Koehn et al., 2007) allows us to automatically train translation models for any language pair. It uses phrase based transla- tion Models and word alignments.

With the recent advancements of Sequence-to-sequence models (seq-to-seq), transliteration quality has improved greatly. Proposed in (Sutskever et al.,2014), these networks are used for translation of an input text from one language to another. Works (ud Din, 2019; Ameur et al., 2017; Mandal and Nan- maran, 2018) have later extended seq-to-seq mod- els for transliteration as well and various attention- based seq-to-seq models have been subsequently been proposed (Bahdanau et al., 2014; Luong et al., 2015; Vaswani et al., 2017). Attention based seq-to-seq models (Bahdanau (Bahdanau et al., 2014)) work by using an encoder to learn representations of the input sequence and a decoder to produce the out- put sequence from the hidden representations the encoder created. Few attention variants in seq-to- seq architectures include (Luong et al., 2015) by Lu-

ong et al. and self-attention (Vaswani et al., 2017). Luong attention differs from Bahdanau in the alignment calculation and the position at which the attention mechanism is introduced in the decoder. Self-attention introduced in (Vaswani et al., 2017), is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence.

3 Background

In this section, we explain the sequence-to-sequence networks and present the details of our employed Elastic Search stack.

3.1 Sequence-to-sequence network

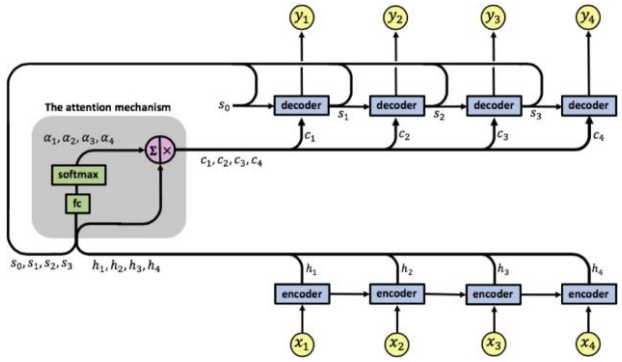

Figure 1: Sequence-to-sequence network

Sequence-to-sequence network (seq-to-seq) (Sutskever et al., 2014) is used for converting one sequence into another, particularly when the input and the output sequence lengths vary. They use encoder-decoder networks (Figure 1). Their details are presented below.

1. Encoder: The encoder has sequential recurrent layers which learn to encode the input data accurately and produce a set of hidden states which are passed to the decoder.

2. Decoder: The decoder takes the states from encoder and uses it to generate context vector at every time-step t. Context vector holds the weighted cumulative information from all of encoder hidden states and varies across the time-steps. At every time-step, the previous decoder hidden state along with the corresponding context vector is passed as input to the recurrent layer at t. The joint output from all time-steps gives us the output text sequences.

The above network employs attention so that the decoder learns to focus on relevant encoder hidden states. It mainly employs two kinds of attention,

- Global attention: considers all the hidden states in creating the context vector.
- Local attention: considers only a subset of the hidden states in creating the context vector.

We train all our seq-to-seq methods using Adam optimiser. The encoder and decoder have bi-directional GRU cells (Deng et al., 2019).

3.2 Elastic Search stack

Consider cross language information retrieval, a well known user-driven application. Such an application consists of transliterating and matching a user-given input in Roman to its already indexed reference transliteration in the target domain (Eg: Devanagari). In such cases, the transliteration framework's output should have to match accurately to the source, regardless of the slight variations in the user-given input in Roman script from its originally intended form. To handle this, we have built an end-to-end framework which incorporates Elastic Search stack into the proposed framework for precise matching to the reference after transliteration. Elastic Search stack (ES stack) (Gormley and Tong, 2015), is a distributed, open source search and an analytics engine. It can also easily map rogue model predic- tions to their references.

3.2.1 Terms

- Index: Adding 'data' to ES stack is known as "indexing." In our case, we can either index thetrue scripts of native words to Devanagari or theknown transliterations of non-native words.
- Mapping: It is the process of defining how a document, and the fields it contains, are stored and indexed.
- Fields: Fields are properties in a mapping. Every mapping contains a list of fields or properties pertinent to the document. In our case we use mappings to define the properties of the words being indexed. There are 2 types of fields: *Key- word* and *text*. Keyword fields are only search- able by their exact value. Text field allows search for individual words within each full textfield.
- Analyzer: The analyzer parameter specifies theanalyzer used for text analysis when indexing or searching a text field. The default analyzer for *Keyword* type is standard and is immutable.

- Match and Multi-match query: A match query returns documents that match a provided text which is further analyzed before matching. The multi-match query builds on the match query to allow multi-field queries. We can influence scoring as needed by prioritizing more importantfields.

- Fuzziness parameter: Adding fuzziness parameter [1] to a multi-match query turns a plain multi-match query into a fuzzy one. It generates matching terms that are within the maximum edit distance specified in the parameter and then checks the term dictionary to find out which of those generated terms actually exist inthe index.

3.2.2 Inverse mapping with fuzzy search

We have used 3 fields in the mapping we have built.Their details are presented below.

1. *Devanagari_script_field*, denotes the script ofa word being indexed in Devanagari. The field type is *keyword*, so the search on this field happens full-text.

2. *Roman_script_field*, denotes the script of the word being indexed in Roman. The field type is*keyword*.

3. *Consonants_field*, denotes the sequence of just the consonants of the word being indexed. The field type is *text*, so the search on this field hap-pens consonant wise. For this, we use an icu an- alyzer [2] with icu tokenizer [3] and a custom char filter which converts all the vowels in the text being analyzed to NULL using the predefined unicode mappings of vowels in Devanagari.

The search works as follows. The transliterated can- didates are queried against the ES stack consisting of the indexed reference transliterations. This is done using a multi-match query which jointly queries on the fields *Devanagari_script_field* and *Conso- nants_field* with a fuzziness score of 0.7. Those can- didates which are mapped to a reference transliter- ation are updated to be the same as the reference before being returned.

[1] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html

[2] https://www.elastic.co/guide/en/elasticsearch/plugins/current/analysis-icu.html

[3] https://www.elastic.co/guide/en/elasticsearch/plugins/current/analysis-icu-tokenizer.html

4 Proposed framework: Syll-S2S

In this section, we propose a novel Syllabified Sequence-to-sequence model for improving the performance of Roman to Devanagari transliteration to that in the existing approaches. We first present the basic idea of the proposed approach. Next, we explain the proposed approach in detail.

4.1 Basic idea

Local attention based sequence-to-sequence models have been under-explored for the task of transliteration and could dramatically improve the knowledge gain with respect to global attention based sequence-to-sequence models. Leveraging this, the idea is to segment both the input Roman text and its parallel counterpart Devanagari text into syllables before building the attention based seq-to-seq model. Syllable is a unit of spoken language consisting ofa single uninterrupted sound formed generally by a vowel (Eg: a,e,i,o,u) and preceded or followed by one or more consonants (Eg: b,c,d,..). By reducing the sequences into their constituent syllables offline, we can employ a teacher forcing method and force the model to attend to a fixed window length of sylla- bles at each decoding step in the attention part of decoder in seq-to-seq networks. At each decoding time-step, a window centered around the source po- sition based on the syllable alignment is used to com- pute the context vector for the syllable corresponding to the target position. Thus, the attention weights at each time-step are distributed and limited to the corresponding syllables of the positions in the win- dow which allows effective knowledge building by the model.

4.2 Sonority Syllabification

The process of splitting a text into its constituent syllables is referred to as Syllabification (Treiman and Zukowski, 1990). Since it is derived directly from pronunciation, syllables are script-agnostic and are vital information pillars for machine translation and transliteration tasks. For doing syllabification, we utilise Sonority syllabification principles (SSP) available for Roman script (Henke et al., 2012). The Sonority Sequencing Principle (SSP) or Sonority se- quencing constraint is a phonetic principle that aims to outline the structure of a syllable in terms of sonority. Basing on the SSP rules for Roman, we present the SSP rules for Devanagari and other In- dic scripts. We are the first to present SSP rules for Indic scripts.

4.3 Syllabified local attention with seq-to-seq net

Consider the Roman text 'Hajagiree' and its reference transliteration `हजिगरी'. Observe the one-to-one mapping between the source (Roman) and the target (Devanagari) text syllables (Figure 4.3(a)). Moreover, this mapping is both static and sequential. Based on this, we propose Syllabified Sequence-to-sequence model using local attention. Recall from Section 3.1 that in a local attention based seq-to-seq, at a given time-step, the decoder is fed only a part of encoder context information. This information is generally limited to the corresponding input part.

With syllabification, the decoder needs to attend

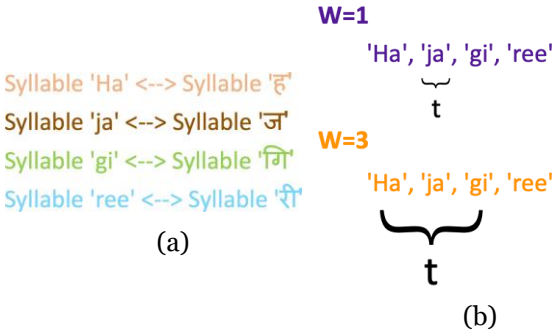

Figure 2: (a) shows the syllable-syllable correspondence between an example's input and output texts. (b) shows the attention windows W={1,3} focused by the decoder when at t^{th} time-step

only on the select parts of the encoded input data specified by us at each time-step. This way, the model can learn to decode more accurately when it focuses on just the needed input syllables rather than the parts which it does not depend on. Thus, the attention scope is limited to just the input syllable/s corresponding to the present decoder time-step. However, considering an x-length window towards left and right could prove beneficial as syllabification is not always perfect (Figure 4.3(b)).

The pseudo-code of the test process is as follows.

1. Input text in Roman script is cleaned and syllabified using SSP principles.
2. The preprocessed input text is sent to the trained Syll-S2S for transliteration to one of the Indic scripts. Outputs Top-N (N=3) candidate transliterations.
3. The transliterations are queried using inverse mapping described in Section 3.2.2 and the mapped candidate transliterations are updated to their references.
4. Returns the final candidate transliterations.

5 Experimental setup

All the experiments are conducted on an Intel i5 processor with 8GB RAM running Ubuntu Linux oper-ating system.

5.1 Dataset

It is to be noted that the parallel transliteration corpus i.e, the text in Roman and its counter transliteration in Devanagari is very limited and had to be scraped from several existing open sourced works. We have 1.5 million parallel words worth of data in Roman and Devanagari after scraping. The words in the resulting data are diverse ranging from named entities to native words from Roman and Devana- gari. We use a train-val-test split of 4:1:1 and train all the neural net models for 100 epochs.

5.2 Data preprocessing

Data cleaning and preprocessing was done on both the corpus before model building. This includes stripping unwanted characters like tags, HTML entities, UNK tokens, special characters etc and lower-casing the Roman data corpus. We use UTF-8 en- coding on both the source and target corpus.

Once cleaning is done, for implementing the proposed approach, we pass the input text to the Sonority syllabification module (Section 4.2) for splitting into syllables. The syllabified text is then passed as input to Syll-S2S model.

5.3 Methods and metrics

We have considered the following methods for performance evaluation.

- Rule-based (RL): Rule based transliteration system based on predefined character-to-character mapping.
- Google Transliterate (Google): Google's officialopen API for transliteration [4].
- Indic-NLP (Indic) (Kunchukuttan et al., 2020).
- Seq-to-seq (S2S): A sequence-to-sequence model incorporating Luong's attention based scoring (Luong et al., 2015).
- Self attention (SA): A self attention model based on transformers (Vaswani et al., 2017).

[4] https://inputtools.google.com/request?itc=hi-t-i0-und&num=4&cp=0&cs=0&ie=utf-8&oe=utf-8&app=demopage&text=''

Scores	RL	Indic	S2S	SA	Google	Syll-S2S-W{1,3,5}		
						W1	W3	W5
Top-1 Accuracy	0.45	0.57	0.72	0.79	0.83	0.82	0.86	0.85
Top-2 Accuracy	0.52	0.66	0.75	0.81	0.84	0.84	0.87	0.87
Top-3 Accuracy	0.58	0.71	0.77	0.83	0.84	0.84	0.87	0.87
Levenshtein distance	0.63	0.77	0.85	0.90	0.92	0.92	0.94	0.94
BLEU before inverse mapping	0.39	0.48	0.63	0.71	0.74	0.76	0.77	0.75
BLEU after inverse mapping	0.46	0.59	0.75	0.82	0.85	0.85	0.87	0.87

Table 1: Performance comparison of the considered methods in all the metrics. The top scores are highlighted in redand the second best are highlighted in green.

- Syllabified monotonic Sequence-to-Sequence (Syll-S2S): The proposed approach involving monotonic attention from source to target syllables using S2S model. We also vary the syllables' window-size W={1,3,5} and the vari- ants are named as Syll-S2S-W1, Syll-S2S-W3, Syll-S2S-W5 respectively.

A beam size of 3 (means 3 candidate transliterations for every input) is employed for all the neural net methods (Freitag and Al-Onaizan, 2017).

The following performance metrics have been employed.

- Top-N accuracy: The average number of correct transliterations within the Top-N (N={1,2,3}) beam-search candidate transliterations of the source text.
- Levenshtein distance: The number of single-character edits required to change predicted transliteration into the correct reference.
- 3-gram match: The average number of word-level 3-grams with matches to their corresponding 3-grams from the reference text.
- BLEU scores before inverse mapping: The sim-ilarity of transliterated text to its reference be- fore inverse mapping.
- BLEU scores post inverse mapping: Fuzzy-search based inverse mapping (Section 3.2.2) is used to map the candidate transliterations to the existing reference transliterations in the Elastic Search stack (Section 3.2) before computing BLEU scores.

5.4 Results

5.4.1 Performance comparison

Table 1 shows the performance of all the implemented methods in various score metrics (leftmost column). The best scores highlighted in red are by the proposed approach Syll-S2S. Note the performance improvement from S2S to Syll-S2S. This is because of the monotonic local attention employed in the latter by enforcing syllable-syllable correspondence between the source and the target texts. In particular, Syll-S2S-W3 which is Syll-S2S with window-size 3 is performing the best overall with re- spect to the second best method Syll-S2S-W5 (win- dow size=5) highlighted in green. This is because of the reduced attention window size in Syll-S2S- W3 w.r.t. W=5 as in the former, only the imme- diate left and right syllables apart from the current syllables are focused while computing the attentionscores whereas in W=5, the left and the right window span increases by a step more leading to distributed attention. Between Syll-S2S-W1 and Syll-S2S-W3, the latter does well as vowels might be split between the current and the next/before syllables making the learning process more reliable when the attention window covers them beside just the current syllable. The model closely following the Syll-S2S-W{1,3,5} variants is the Google transliterate. Even the model with the best architecture, SA is still behind the pro- posed approach because of over-fitting due to limited data.

Effect of inverse mapping: It can be observed from the table that the BLEU scores have noticeably improved after incorporating fuzzy-search based in- verse mapping for all the approaches (Section 3.2.2). Recall that the above mapping matches the candi- date transliterations to their existing source translit- erations using fuzzy searching on consonants. The values depict the merit of mapping the candidate transliterations as most often there are only minor differences between the candidate and the reference transliterations, evidenced by higher values in the metric Levenshtein distance.

5.4.2 Effect of syllabified local attention

To understand the advantages of using syllabified monotonic local attention as in the proposed ap-

Sentence	Reference transliter-ation	Syll-S2S-W5 result	Syll-S2S-W3 result
My Birthday Song	माय बथडे सं◌ॉग	माइ ि◌बरटहडाय सोंग	माय बथडे सं◌ॉग
Happy Phirr Bhag Jayegi	हैप्पी ि◌फर भाग जायगी	हापपय ि◌फर भाग जायेगी	हैप्पी ि◌फर भाग जायगी
Wo India Ka Shake-speare	वो इं◌ड्डया का शेक्सपीयर	वो इं◌ड्डया का शक् े◌स्पोरे	वो इं◌ड्डया का शेक्सपीयर
Yamla Pagla Dee-wana Phir Se	यमला पगला दीवाना ि◌फर से	यमल पगला दीवानन ि◌फरसे	यमल पगला दीवाना ि◌फर से
Kaashi in the search of Ganga	काशी इन सचर् ऑफ़ गंगा	काशी इन सचर् ऑफ गंगा	काशी इन सचर् ऑफ़ गंगा
Mausam Ikrar Ke Do Pal Pyar Ke	मौसम इकरार के दो पल प्यार के	मौसम इकरार के दो पाल प्यार के	मौसम इकरार के दो पल प्यार के
Jal bin machhli nritya bin bijli	जल ि◌बन मछली नृत्य ि◌बन ि◌ब-जली	जळ ि◌बन मछली नृत्य ि◌बन ि◌बजली	जाल ि◌बन मछली नृत्य ि◌बन ि◌बजली
Albert pinto ko gussa kyu aata hai	अल्बटर् ि◌ंपटो को गुस्सा क्यों आता है	अल्बटर् ि◌ंपटो को गुस्सा क्यों आता ही	अलबटर् ि◌ंपटो को गुस्सा क्यों आता है

Table 2: Example Roman input sentences, their reference transliterations and the output transliterations from Syll- S2S-W5 and Syll-S2S-W3 methods. Results in red indicate exact match, in green indicate the results with edit distance=1 and in blue indicate the results with edit distance > 1 from the reference.

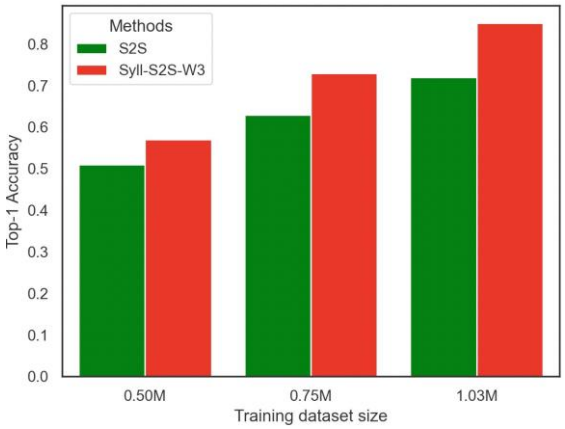

Figure 3: Top-1 Accuracy of S2S and Syll-S2S-W3 meth- ods on varying training data size

proach Syllabified Sequence-to-sequence (Syll-S2S) rather than global-attention as in conventional seq-to-seq (S2S) models, we compare Syll-S2S-W3 model (the best performer, see table 1) with S2S model at varying training data-set sizes. i.e., we train the two models by using the training sizes of 0.5,0.75 and 1 million before testing. Fig 3 shows the Top-1 accuracy vs training dataset-size results for the two approaches. As can be seen, with increasing size of training dataset, the rate of improvement for Syll-S2S-W3 is significantly greater w.r.t. S2S. This means that the proposed approach has a faster knowledge-acquisition rate than S2S model owing to the local attention in Syll-S2S from enforcing monotonic source-target syllable-wise attention. In conclusion, we can deduce that the accuracy improvement for the proposed approach as more training data resources become available would be substantial when compared to conventional seq-to-seq models.

5.4.3 Effect of attention window

To demonstrate the importance of the size of attention window in the proposed approach, we compare Syll-S2S-W3 (window size=3) with Syll-S2S-W5 (window size=5) (refer Figure 4.3(b)). Table 2 shows a qualitative comparison of results from both the methods on few example input sentences of varyinglengths. Results in red indicate exact match to ref- erence transliteration. Results highlighted in green indicate the results with edit distance equal to 1 from reference and ones in blue indicate the results with edit distance greater than 1.

As can be seen, between the two, Syll-S2S-W3 is clearly the best performer as it has the most predic-tions in red and green as compared to Syll-S2S-W5 which has more in blue. This is because of the re- duced attention window size in Syll-S2S-W3 w.r.t. W=3 as in the former, only the immediate left and right syllables apart from the current syllables are fo- cused while computing the attention scores whereas in W=5, the left and the right window span increases by one more step leading to a more scattered atten-

tion. This results in the two left and right syllables on both sides of the current syllable receiving around the same attention as the syllable to be decoded.

5.4.4 Comparison with Google transliterate

- -

Input 1: Jack aur jill pani ki ek balti lene keliye pahadhi par chad gaye
Reference: जैक और िजल पानी कBएक बाल्टी लेने के िलए पहाड़ी पर चढ़ गए

Syll-S2S-W3 result: जैक और िजल पानी कB एकबाल्टी लेने के िलए पहाड़ी पर चढ़ गए
3-gram match score: 1

Google result: जैक और िजल पाि न कBएक बाल्टी लेने के ि लयेे पहाड़ी पर चैड गया
3-gram match score: 0.92

- -

Input 2: Uppu kappurambu nokka polika nundu chooda chooda ruchulu jaada veru purushulandu punya purushulu veraya viswadhaabhiraama, vinura vema
Reference: उप्पु कप्पूरबु नोक्का पोि लका नन्दू चूड चूड रुचुलु जाद वेरय पुरुशूलंदु पुण्य पुरुशुलु वेरय ि वस्ववा"भरामा

Syll-S2S result: उप्पु कप्पूरबु नोक्क पोि लक नन्द चूड चूड रुचुलु जाद वेरया पुरुशूलंदु पुण्य पुरुशुलु वेरया ि वस्वद्य ा"भरामा ि वनुर वेमा
3-gram match score: 0.84

Google result: उप्पू कप्पूरबु नोक्का पोि लका नन्दू चूडा चूडा रुचुलु जादा वेरु पुरुशूलंदु पुण्य पुरुशुलु वेरय
ि वस्ववा"भरामा
ि वनुरा वेमा
3-gram match score: 0.71

- -

The above inputs depict two example sentences and their reference and Top-1 candidate transliteration from Syll-S2S-W3 (the best performer, see table 1) and Google transliterate. We also show the 3-gram match score which is the the average number of word-level 3-grams which are correctly matched to their corresponding 3-grams from the reference text. 3-gram match scores act as a direct measure of a model's strength when plugged in other crucial downstream tasks like Information retrieval, attribute linking etc and having a high 3-gram scores would be the ideal for all models.

There is a clear distinction in the scores from the two approaches with Syll-S2S-W3 giving higher val- ues in comparison to Google transliterate. Observe how the difference in scores becomes more apparent with increase in the input-length. This shows that in applications involving more input complexity, the proposed approach is expected to produce the best result.

6 Conclusions

In this paper, we have proposed Syllabified sequence-to-sequence attention model for improving the transliteration quality with respect to the current works. For this, we present the Sonority syllabification principles for Devanagari and other Indic scripts. To handle the erroneous entry of user-given text and to further boost the performance, we incorporate fuzzy-search based inverse mapping with consonants by employing Elastic Search stack. This fa- cilitates the mapping of output transliterations from the model to their existing reference transliterations. Experiments demonstrate the superiority of the pro- posed approach in comparison to the state-of-the-art techniques.

As part of future work, we plan to investigate the end-to-end forward and backward transliteration tasks in a unified architecture. We also plan to improve the performance when transliterating words with varying pronunciation to their written forms and are currently working on populating the Elastic Search stack using the verified model predictions.

References

Mohamed Seghir Hadj Ameur, Farid Meziane, and Ahmed Guessoum. 2017. Arabic machine transliteration using an attention-based encoder-decoder model. *Procedia Computer Science*, 117:287–297.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Sukalpa Chanda, Umapada Pal, Katrin Franke, and Fumitaka Kimura. 2010. Script identification–a han and roman script perspective. In *2010 20th international conference on pattern recognition*, pages 2708–2711. IEEE.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech & Language*, 14(4):283–332.

Yaping Deng, Lu Wang, Hao Jia, Xiangqian Tong, and Feng Li. 2019. A sequence-to-sequence deep learning architecture based on bidirectional gru for type recognition and time location of combined power quality disturbance. *IEEE Transactions on Industrial Informatics*, 15(8):4481–4493.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*.

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O'Reilly Media, Inc.".

Vishal Goyal and Gurpreet Singh Lehal. 2009. Hindi-punjabi machine transliteration system (for machine translation system). *George Ronchi Foundation Journal, Italy*, 64(1):2009.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine trans- lation. In *Proceedings of the AAAI Conference on Ar- tificial Intelligence*, volume 32.

Eric Henke, Ellen M Kaisse, and Richard Wright. 2012. Is the sonority sequencing principle an epiphenomenon. *The sonority controversy*, 18:65–100.

Yuxiang Jia, Danqing Zhu, and Shiwen Yu. 2009. A noisy channel model for grapheme-based machine transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 88–91.

In-Ho Kang and Gil Chang Kim. 2000. English-to-korean transliteration using multiple unbounded overlapping phoneme chunks. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computa- tional linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Avik Bhattacharyya, Mitesh M Khapra, Pratyush Kumar, et al. 2020. Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages. *arXiv preprint arXiv:2005.00085*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Soumil Mandal and Karthick Nanmaran. 2018. Normalization of transliterated words in code-mixed data using seq2seq model & levenshtein distance. *arXiv preprint arXiv:1805.08701*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Rebecca Treiman and Andrea Zukowski. 1990. Toward an understanding of english syllabification. *Journal of Memory and Language*, 29(1):66–85.

Usman Mohy ud Din. 2019. Urdu-english machine transliteration using neural networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.