A ROBUST THREE-STAGE HYBRID FRAMEWORK FOR ENGLISH TO BANGLA TRANSLITERATION

Redwan Ahmed Rizvee, Asif Mahmood,
Shakur Shams Mullick and Sajjadul Hakim

Tiger IT Bangladesh Limited, Dhaka, Bangladesh

ABSTRACT

Phonetic typing using the English alphabet has become widely popular nowadays for social media and chat services. As a result, a text containing various English and Bangla words and phrases has become increasingly common. Existing transliteration tools display poor performance for such texts. This paper proposes a robust Three-stage Hybrid Transliteration (THT) framework that can transliterate both English words and phonetic typed Bangla words satisfactorily. This is achieved by adopting a hybrid approach of dictionary-based and rule-based techniques. Experimental results confirm superiority of THT as it significantly outperforms the benchmark transliteration tool.

KEYWORDS

Transliteration framework, phonetic typing, English to Bangla, hybrid framework, THT

1. INTRODUCTION

In this era of globalization, people are unprecedentedly exposed to information from global sources. Especially with the advent of the internet and smartphones, access to information in a foreign language has become increasingly common. Machine Translation (MT) can play a crucial role in this aspect [1] as it assists information exchange by converting foreign language texts intoa person's native language. One particular challenge of MT is to *named entities* (NE), where transliteration is often preferable.

Transliteration refers to the phonetic conversion of words across different pairs of languages [2]–[4]. However, it is a challenging task since pronunciation rules vary across languages and there are times when exact/similar sounding phonemes are not available in the target language. An example of this would be to transliterate the proper noun Parvez/Parves (a Persian name) in Arabic which has no letter in the alphabet that sounds similar to both 'P' and 'V' sounds.

Another challenge is to transliterate phonetic typed text where native language words are written using primarily English alphabet. Due to the widespread use of social media and internet-based chat applications, encountering various mixture of English and phonetic typed text has become frequent. Translating such texts adds to the challenge since such phonetic typed words need to be reverse-transliterated.

This paper focuses on the transliteration of words from English to Bangla. Bangla (also known as Bengali) is the 6th largest language with over 268 million users[1]. Despite that there is no advanced MT or transliteration tools available that addresses the aforementioned challenges satisfactorily. The most known literature and system which have addressed Bangla transliteration problem are [5]–[7]. Among them, Avro [7] is an open source implementation which also is one of the most

widely used Bangla phonetic typing tool [8], [9]. However, it fails miserably to perform transliteration for most common and simple words. For instance, the English words ‘what’, ‘nation’, and ‘fight’, Avro respectively produces ‘ওহাত’, ‘নাত্তন’ and ‘ািঘত’ which are far from being acceptable transliterations.

In this paper, we propose a novel robust Three-stage Hybrid Transliteration (THT) framework that converts English words into Bangla. Additionally, it can also transliterate phonetic typed words satisfactorily. The proposed framework consists of a three-tier approach where an English word is converted into an intermediate phonetic form before converting it into Bangla. It also revises and improves the spelling of the converted word to produce a more desirable outcome. The THT framework is developed as a part of a broader MT system, hence, it produces a list of candidate transliterations from which the MT system can choose the best one according to its context. This is also suitable for spell-checker applications.

The rest of the paper will critically review the related literature in Section 2 describe the scope and challenges in Section 3, a detailed description of the THT framework in Section 4, evaluate its performance in Section 5 and provide some concluding remarks in Section 6.

2. RELATED WORKS

A foreign language text can be converted to a person’s native language by MT to enable efficient and affordable information exchange. Recent MT solutions are often based on *Neural Machine Translation* (NMT) which utilises advanced machine learning and artificial intelligence techniques [1], [10]–[12]. These NMT solutions require large bilingual corpus of original and translated texts for training and generally produces better output than classic rule-based solutions. However, they are difficult to implement for low-resource languages where a good corpus is yetto be made.

Transliteration is often employed in MT solutions for conversion of NE from the original language to the target language. In the case of English to Bangla transliteration being considered in this paper, original language is English and target language is Bangla. Another application of transliteration is to convert a phonetic typed text. Since both original and target language in this case is Bangla, it often does not follow standard English pronunciation rules and requires careful processing.

While transliteration problem has been addressed by many literature for other language pairs [13]–[16], these solutions cannot be used readily for English to Bangla transliteration. Firstly, each language has its own pronunciation rules and Bangla pronunciation rules and transliteration conventions are different from others. Therefore, rule-based solutions require new sets of transliteration rules for Bangla. Secondly, there are no large dataset available for English to Bangla transliterations, therefore neural network architecture-based solutions will be difficult to implement as well. Lastly, existing solutions mostly address transliteration of NE and are not effective for phonetic typed texts. Therefore, a novel solution is necessary for English to Bangla transliteration problem.

3. SCOPE AND CHALLENGES

The scope of this paper has been set to the problem of transliterating English words to Bangla for this paper. Particularly, English names and terms as well as phonetic typed Bangla words are considered. A possible extension to other language pairs is briefly discussed in Section 4.4.

3.1. Input Types

The input word, E, can belong to either English or Bangla language. In both cases, the THT framework expects E to be written using the 26 letters form the English alphabet.

3.1.1. English Words (E_{En})

This category consists of English words including names, which are generally found in English literature. The expected output of the framework is a transliterated Bangla word, , written using Bangla alphabet whose pronunciation matches exactly or closely to the original English word, E_{En}. For instance, if E_{En} = consciousness then output, B, is expected to be 'কনশাসনসস' or 'কনসাসনসস' based on its pronunciation and Bangla accent [17].

3.1.2. Phonetic Typed Bangla Words (E_{Bn})

Phonetic typing has become widely popular especially in social media and internet-based chat applications due to a faster and easier typing experience [18]. Such texts consist of native Bangla words transliterated into English. For example, a Bangla sentence 'আমি ভাত খাই' might be written as 'Ami bhaat
khai' in phonetic typing. In this case, the framework is expected to produce 'আমি', 'ভাত' and 'খাই' for the inputs 'Ami', 'bhaat' and 'khai' respectively.

3.2. Challenges in Transliteration

There are some crucial properties to consider while performing transliteration which make the task more challenging. A good transliteration framework should address these challenges to produce a satisfactory outcome.

3.2.1. Lack of a Comprehensive Dictionary

There is no parallel English to Bangla dictionarythat comprehensively lists transliteration of every words. Building such a dictionary is also impractical since the language's vocabulary is huge and ever expanding.

3.2.2. Pronunciation Variation in E_{En}

Same set of letters make different sounds based on their context across words. For example, in the word 'come' (kʌm) 'o' is pronounced as ʌ according to *International Phonetic Alphabets* (IPA) whereas in the word 'comet', 'o' is pronounced as ɒ [19]. Besides, many words have some *silent* letters which do not contribute to the overall sound, thus compounding the difficulty of transliteration.

3.2.3. Irregularity of E_{Bn}

Since phonetic typing is mostly used in informal communications, there is no hard and fast rule that everyone follows. A Bangla word can have different transliterated forms, for instance, the word 'আমি' (meaning, 'I'), can be written phonetically as 'Ami' or 'Ame'. Moreover, different Bangla words can produce the same transliterated form. A transliterated form 'Amar' can represent both 'আমার' (meaning, 'my') or 'অমর' (meaning, 'immortal'), for example. This irregularity adds to the challenge of transliteration.

3.2.4. Coincidence of E_{En} and E_{Bn}

Another critical issue is that some E_{En} and E_{Bn} can have the exact same spelling, but different pronunciation due to phonetic typing issue discussedabove. This is more obvious in names where native Bangla pronunciation is often different than typical English pronunciation of the transliterated word. For example, 'Bangladesh' is pronounced in English as 'bæŋgləʤ' producing a transliterated form 'ব্যাংলাদেশ'. However, a better transliteration would be 'বাংলাদেশ' since that's how it is spelled and pronounced in Bangla.

Due to the aforementioned issues, a dictionary of English to Bangla transliteration mapping-based solution is not feasible. Moreover, there can be several possible mappings due to the reasons discussed above which will cause the dictionary-based solution to fail. On the other hand, E_{En} and E_{Bn} follows different pronunciation rules, therefore it is challenging to device a set of rules that works on both input types. The situation is further complicated in the cases of non-standard and accented spellings of words. The proposed framework adopts a hybrid of robust rule-based and dictionary-based approach to address these challenges.

4. PROPOSED THT FRAMEWORK

The THT framework primarily consists of three independent modules. The first is a *Phonetic Transliteration* (PT) module which takes an input word, E, and produces an intermediate phonetic representation, P_T. The second is a Bangla Transliteration (BT) module which analyses P_T and produces a list of candidate Bangla transliterations $B_T = \{b_{t1}, b_{t2}, b_{t3}, \dots, b_{tm}\}$ ordered according to their likelihood. Finally, a Spelling Improvement (SI) module performs a heuristic revision of spellings of the elements of B_T and produces a set of possible transliterated forms,

$B = \{b_1, b_2, b_3, \dots, b_n\}$ along with their corresponding likelihood in the language's context. A high level block diagram of the THT framework is presented in Figure 1.

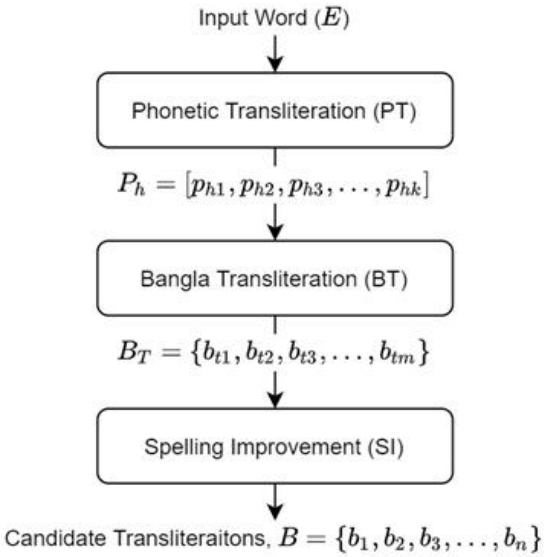

Figure 1. Overview of the THT framework

4.1. Phonetic transliteration

The function of this module is to convert input word, E, to its phonetic equivalent, PT. A third party transliteration tool *Epitran* has been used for this purpose [20], however, other tools can be used as well. To expedite the conversion process and avoid unintended outcomes, a basic phoneme set is chosen to construct the transliteration graph. This may decrease accuracy of the conversion in some cases, which will be compensated in the SI block. The basic phoneme set along with corresponding Bangla transliteration are shown in Figure 2. The Bn_b column holds the base transliteration group whereas possible miscellaneous variations are shown in the column Bn_m.

#	T	Ph	Bn_b	Bn_m
1	c	p	প	পি
2	c	b	ব	বি
3	c	t	ট, ত	টি, তি
4	c	th	ঠ	
5	c	d	ড	ডি
6	c	dh	ঢ, ধ	
7	c	ʧ	চ	
8	c	dʒ	জ	
9	c	k	ক	কে
10	c	kh	খ	
11	c	g	গ	
12	c	gh	ঘ	
13	c	f	ফ	এফ
14	c	v	ভ	ভি

#	T	Ph	Bn_b	Bn_m
15	c	θ	থ	
16	c	ð	দ	
17	c	s	স	এস
18	c	z	জ, স	
19	c	ʃ	শ	
20	c	ʒ	স, জ	
21	c	h	হ	
22	c	m	ম	এম
23	c	n	ন	এন
24	c	ŋ	ং	
25	c	l	ল	এল
26	c	r	র	আর
27	v	w	উ, ও	
28	c	wh	হ	

#	T	Ph	Bn_b	Bn_m
29	c	j	য়, ইয়	
30	c	jh	ঝ	
31	v	æ	্যা, অ্যা	
32	v	a	আ	
33	v	ɑ	অ, আ	
34	v	ɒ	অ, ও, আ	
35	v	ɔ	অ,ও	
36	v	ə	অ, আ, এ, ও, উ, ই	
37	v	ɪ	ই,এ	
38	v	ɛ	এ	
39	v	ʌ	আ,অ	
40	v	ʊ	উ, ও	

#	T	Ph	Bn_b	Bn_m
41	v	jʊ	ইয়	
42	v	U, ju	উ, ইউ	
43	v	o	ও, অ	
44	v	e	এ	
45	v	ej	এ, এই	
46	v	aj	আই, আয়	
47	v	ow	ও	
48	v	wɔ	ওয়া	
49	c	oj	য়, য়ে, অই	
50	v	wʊ	ও, উ	
51	v	əw	উ, এউ, আউ	
52	c	jɪ	য়ি	
53	c	wə	য়া	

Figure 2. Transliteration Graph from IPA phonemes to Bangla Characters for Epitran. Column T denotes the type of phoneme, vowel (v) or consonant (c).

4.2. Bangla Transliteration

This module produces a list of candidate transliterations, BT from the phonetic representation, PT.To appreciate the complete process, knowledge of Bangla alphabet and some spelling rules and word formation process is required.

vowels	অ, আ(া), ই(ি), ঈ(ী), উ(ু), ঊ(ূ), ঋ(ৃ), এ((ে), ঐ(ৈ), ও((ো), ঔ((ৌ)
consonants	ক, খ, গ, ঘ, ঙ, চ, ছ, জ, ঝ, ঞ, ট, ঠ, ড, ঢ, ণ, ত, থ, দ, ধ, ন, প, ফ, ব, ভ, ম, য, র, ল, শ, ষ, স, হ, ড়, ঢ়, য়, ং, ঃ, ঁ

Figure 3. Bangla vowels, consonants and kars

4.2.1. Bangla Alphabet and Spelling Rules

There are 11 vowels and 39 consonants in Bangla alphabet comprising a total of 50 letters. Among the 11 vowels, 10 of them have short forms known as *kar* (কার). The list of Bangla vowels,

consonants and kars are shown in Figure 3 where kars are written beside corresponding vowels within brackets. The first vowel 'অ' does not have a short form and is implied after each consonant.

Bangla words may also contain compound letters (যুক্তবর্ণ or jukto-borno) which are comprised of multiple consonants connected through a *hasant* (U+09CD) [21].

Some of these letters sound very similar to each other and can be represented by a single IPA phoneme. Some other letters and letter combinations do not have an exact equivalent in English. There are some other spelling rules as well dictating which letters can and cannot be put next to each other. The Bangla Transliteration module considers these nuances while producing each candidate transliteration.

4.2.2. Candidate generation

To produce b_{ti}, a list of IPA phonemes, P_h is generated from P_T. This is done by performing a maximum string matching using the Ph column of Figure 2. For example, let $E = \text{Bangladesh}$. Then $P_T = \text{bæŋgləʤʃ}$ and corresponding phoneme list would be $P_h = \{\text{b}, \text{æ}, \text{ŋ}, \text{g}, \text{l}, \text{ə}, \text{d}, \text{ʧ}, \text{ʃ}\}$.
For each $ph \in P_h$, a substitution operation $ph \rightarrow B(ph)$ is performed using the last two columns of Figure 2. Since there are often multiple options for substitution, each option produces a new candidate transliteration b_{ti}.

Some additional processing is done to form kars, and compound letters by sequentially manipulating the list of phonemes P_h . Let us assume that Pre denotes a temporary string containing already substituted Bangla letters, and a list s contains the remaining items from P_h.

1. If the last letter of Pre is consonant and the first item [0] is vowel, then $Pre \leftarrow Pre +$ kar$(s[0])$ and $s[0]$ is removed before the next iteration.
2. Some special substitutions are also considered for vowels for example, kar(আ) → ◌া◌া → য়া and kar(এ) → ◌◌া → ◌য় etc.
3. If both the last letter of Pre and the first item [0] are consonant. Then compound letter substitution is checked for $Pre \leftarrow Pre + hasant +$ [0], and applied if possible. [0] is removed.

After all phonemes in P_h is processed, Pre becomes a candidate transliteration b . Each candidate is then scored based on their likelihood and arranged in a list B_T accordingly.

#	T	Bn	En
1	v	অ	a,o
2	v	আ	a,u,o
3	v	ই	i,e,y
4	v	উ	u,o,w
5	v	এ	e,a,i
6	v	ও	o,w
7	v	ইয়	y,u,j,(ই)(অ)
8	v	অ্যা	a
9	v	্যা	a
10	v	ইউ	u,(ই)(উ)
11	v	এই	(এ)(ই)
12	v	এউ	(এ)(উ)
13	v	আউ	(আ)(উ)
14	v	আই	i,y,(আ)(ই)
15	v	আয়	i,y

#	T	Bn	En
16	v	অই	(অ)(ই)
17	c	এফ	af,ef
18	c	এস	as,es
19	c	এম	am,em
20	c	এন	an,en
21	c	এল	al,el,il
22	c	আর	ar,ur,ir,er,or
23	c	পি	pi,pe
24	c	বি	bi,be
25	c	টি	ti,te
26	c	তি	ti,te
27	c	ডি	di,de
28	c	কে	ka,ke,ca,ce
29	c	ভি	vi,ve,bhi,bhe

Figure 4. Reverse transliteration graph to process vowel sounds

4.2.3. Scoring Mechanism

The score of each candidate b_{ti} indicates how likely its letters are to appear together and is composed of two metrics. The first metric is called *Word Level Language Model Score*. The idea of *Language Model* (LM) is originally developed at the sentence level by calculating the contextual probability of the words occurring together [22]. THT designs a modified version of it at the word level, by considering each letter's contextual occurring tendency. Primary feature of this metric is its ability to calculate score for a new word which did not occur in the training dataset of LM as it works through alphabet context.

For each b_t , LM(b_{ti}) is calculated, negated and normalized between $[0 - 1]$. LM(b_{ta}) > LM(b_{tb}) where $a \neq b$ means b_{ta}'s letter sequence is more likely to occur than b_{tb}'s. For example, it is observed that LM(বাংলা) = 0.7 and LM(ব্াংলা) = 0.3 rightly indicating that, 'বাংলা' is more likely than 'ব্াংলা'.

The second metric considers the vowels particularly since they usually have more substitution options and thus difficult to process as can be seen form the transliteration graph. Besides, variations of some consonants may also produce additional vowel sounds (e.g., row 26 in Figure 2). To properly handle this, a heuristic metric is designed to measure the correspondence between the output Bangla vowel phonemes with the letters of the input word. A reverse transliteration graph is used for this purpose which is presented in Figure 4. It is important to note that the reverse-transliteration graph here contains significantly less phonemes than its counterpart in Figure 2.

Formal definition of *Vowel Mapping* (VM) score is as follows. Let b_t be a candidate Bangla transliteration of original input word E and V be a list of phonemes in b_t that appears in Figure 4. Also let $M_{b_t \leftrightarrow E}$ be a mapping between phonemes in V and phonemes of E. If x is the number of phonemes in V and y is the number of sequential mappings in M, then the VM score is given by

$$\text{VM} = \begin{cases} y/x, & x > 0 \\ 1, & x = 0 \end{cases} \quad (1)$$

The score of a transliterated word, b_t, is then defined as weighted average of LM and VM:

$$\text{SC} = z\text{LM} + (1 - z)\text{VM}, \quad \text{where } 0 \leq z \leq 1 \quad (2)$$

z is the weight factor denoting the weight distribution between the two metrics.

A high SC indicates that the word b_t is more likely to be a better transliteration. Each candidate Bangla transliteration b_{ti} is then sorted according to their score from the highest to the lowest anda candidate list B_T is formed. Hence b_{t1} will be the best transliteration as determined by its score. B_T is sent to the next module for spelling improvement.

4.3. Spelling Improvement

The performance of the BT module depends on the performance of the PT module. The underlying transliteration tool *Epitran* does not handle E_{Bn} well since they follow different pronunciation rules. Consequently, b_{t1} may not be the desirable one. A *heuristic runtime dynamic programming* (HRDP) algorithm has been developed in the SI module to resolve this issue.

4.3.1. Preprocessing

A one-time preprocessing step is required to generate a prefix trie, Tr [23], from a Bangla word database on which the HRDP algorithm operates. Each node of Tr is assigned an id N. The HRDP algorithm scans a candidate transliteration b_{ti} from left to right and traverses Tr accordingly to find valid words.

4.3.2. The HRDP Algorithm

The HRDP algorithm is a modified version of Edit Distance Dynamic Programming algorithm [5], [24] which is often used in spell checkers and string manipulation problems. It takes b_{ti} as input, creates a mapping between phonemes of b_{ti} and , and attempts to manipulate the phonemes in order to create valid words from Tr. Four operations have been defined for string manipulation: *copy*, *replace*, *delete* and *insert*. Each operation incurs a predefined penalty. The HRDP algorithm will now be formally presented.

Let E and b_t denote the original input word and a candidate transliteration respectively. Let $Ph_E = \{ph_{e1}, ph_{e2}, \dots, ph_{ek}\}$ be the phonemes of E and $Ph_{b_t} = \{ph_{b1}, ph_{b2}, \dots, ph_{bl}\}$ be the phonemes of b_t . k and l denotes the total number of phonemes in Ph_E and Ph_{b_t} respectively. Additionally, let $M_{b_t \leftrightarrow E}$ be a sequential mapping between Ph_{b_t} and Ph . Valid mappings between ph_{bj} and ph_{ei} are given in Figure 5.

#	T	ph_{bj}	ph_{ei}
1	c	ক	k,c,q,ch
2	c	খ,ক্ষ	kh,ch
3	c	গ	g
4	c	ঘ	gh
5	c	ং,ঙ,ঞ,ঁ	ng,n
6	c	চ	c,ch,tu,te
7	c	ছ	ch
8	c	জ	j,g,s,z,x
9	c	ঝ	jh
10	c	ট,ত,ৎ	t
11	c	তু	tu
12	c	ঠ,থ	th
13	c	ড	d
14	c	ঢ,ধ	dh
15	c	দ	d,th
16	c	ন,ণ	n
17	c	প	p
18	c	ফ	f,ph,gh
19	c	ব	b
20	c	ভ	bh,v
21	c	ম	m
22	c	য়	y,j,oi,ai
23	c	য	j,g,s,z,x
24	c	র,ৠ,ড,ঢ,ঋ	r
25	c	ল	l
26	c	শ	s,ti,ci
27	c	সি,শি	ti
28	c	ষ	sh,ti,ci
29	c	স	c,s
30	c	হ	h
31	c	হো,হু	wh,ho,hu
32	v	অ	a,o
33	v	আ	a,u,o
34	v	ই,ঈ	i,e,y
35	v	উ,ঊ	u,o,w
36	v	এ	e,a,i
37	v	ঐ	*(অ)(ই)
38	v	ও	o,w
39	v	ঔ	*(অ)(উ)
40	v	ইয়	Y,u,j,*(ই)(অ)
41	v	ইউ	u, *(ই)(উ)
42	v	আই	i,y,*(আ)(ই)
43	v	আয়	i,y
44	v	এই	*(এ)(ই)
45	v	এউ	*(এ)(উ)
46	v	আউ	*(আ)(উ)
47	v	অ্যা	a
48	c	ঃ	y

* Combination of individual letters

Figure 5. Common character maps between Bangla and English alphabets/phonemes, $M_{bt\leftrightarrow E}$.

The HRDP algorithm will scan each phoneme in Ph_{bt} from left to right and construct a string b accordingly. For each phoneme, it tries the *copy*, *replace*, *delete* and *insert* operations in that order using $M_{bt\leftrightarrow E}$ and traverse Tr accordingly, ensuring that only valid words are produced as output.

The main recursive function of HRDP operates on four parameters, namely, current prefix b, current penalty p, index j of the phoneme ph_{bj}, and index i of phoneme ph_{ei}. It also has access to two additional global variables, namely, minimum penalty p_{min} and penalty offset f. These two variables help prune inconsequential recursion branches preemptively. The HRDP function is presented in Algorithm 1.

Algorithm 1: HRDP algorithm to improve spelling

Globals: $p_{min}, f, Tr, M_{b_t\leftrightarrow E}, Flag, B$

procedure HRDP(j, i, p, b)

1: **if** $b \notin Tr$ **then return**
2: **if** $p > p_{min} + f$ **then return**
3: **if** $j = l$ **AND** $i = k$ **then**
4: $p_{min} \leftarrow \min(p_{min}, p)$
5: Add b to B
6: **return**
7: **if** $Flag[j][i][p][b] = 1$ **then return**
8: **else** $Flag[j][i][p][b] \leftarrow 1$
9: $m \leftarrow 0$
10: **if** $ph_{bj} \leftrightarrow ph_{ei} \in M_{b_t\leftrightarrow E}$ **then** $m \leftarrow 1$
11: HRDP$(j + 1, i + m, p, b + ph_{bj})$
12: HRDP$(j + 1, i + m, p + 1, b + ph_r(ph_{bj}))$
13: HRDP$(j + 1, i + m, p + 1, b)$
14: **if** $m = 0$ **then** HRDP$(j, i + 1, p + 1, b + \text{Trans}(ph_{ei}))$

The recursive function first checks pruning conditions to eliminate unnecessary branching. It First checks whether b is a valid prefix in Tr and returns immediately if invalid (Step 1). In the next step, it checks the penalty p and returns immediately if it is too high. It then checks whether end of both E and b_t is reached, updates p_{min} and b accordingly and returns in Steps 3–6.

The HRDP algorithm uses dynamic programming techniques to avoid unnecessary computations. For each function call, it first checks whether the same parameters have appeared before. It achieves this by using a $Flag$ variable which keeps track of each parameter combinations (Steps 7–8).

Finally, the HRDP algorithm manipulates the current string b with the following 4 operations:

1. **Copy:** ph_{bj} is copied as is. Incurs 0 penalty (Step 11).
2. **Replace:** ph_{bj} is replaced with another phoneme ph_r. Possible replacement options are given in Figure 6. Mapping validity is checked afterwards using Figure 5. Incurs a penalty of 1 (Step 12).
3. **Delete:** ph_{bj} is deleted. Incurs a penalty of 1 (Step 13).
4. **Insert:** Transliteration of ph_{ei} is inserted. Incurs a penalty of 1 (Step 14).

For *copy*, *replace* and *insert* operations, an additional check is performed to construct possible *kar* and *jukto-borno*. For a phoneme starting with a vowel sound, two branches are created, one with base form and another with short form. Similarly, for a phoneme starting with a consonant sound, two branches are created, one with the consonant as is, another with a *hasant* prepended to allow formation of combined letters.

Thus each operation can create multiple branches in the recursion tree. Moreover, the *replace* and *insert* operations may create additional branches if more than one ph_r are available for ph_{bj} or multiple transliterations are possible for ph_e . To make the process more efficient, a branch is pruned whenever possible as shown in Steps 2–8 of Algorithm 1. After the HRDP algorithm finishes, the output list of transliterations B is produced.

#	ph_b	ph_r
1	ক	খ,চ
2	খ	ক,ক্ষ
3	গ	ঘ
4	ঘ	গ
5	চ	ছ,সি,শি,তু
6	ছ	চ
7	জ	য,ঝ,গ,স,শ
8	ঝ	জ
9	ট	ত,ৎ
10	ঠ	থ
11	ড	দ,ধ,ঢ
12	ঢ	ধ,ড,দ
13	ণ	ন
14	ত	ট,ৎ
15	থ	ঠ

#	ph_b	ph_r
16	দ	ড,ধ,ঢ,থ,ঠ
17	ধ	ঢ,ড,দ
18	ন	ণ,ঙ, ংঁ, ঞ, ঁ
19	প	ফ
20	ফ	প
21	ব	ভ
22	ভ	ব
23	ম	
24	য	জ,গ,স,শ
25	র	ড়,ঢ়,ঋ,ৃ
26	ল	
27	শ	ষ,স,ট
28	স	শ,ষ,জ
29	হ	
30	য়	জ,য, ংা, ই

#	ph_b	ph_r
31	ংঁ,ঙ,ন,ণ,,ঞ	ঙ,ন,ণ,ঞ
32	হু	হো,হ
33	হো	হু,হ
34	ক্স	
35	গজ	
36	্ (*hasnt*)	
37	অ	আ,ও,ঐ,ঔ
38	আ	অ্যা,অ,উ,এ,ঐ,ঔ,ও
39	ই	ঈ,ঐ,আই,য,এ
40	উ	ঊ,ও
41	এ	অ,আ,ই,ঈ,ঐ,অ্যা
42	ঐ	ই
43	ও	অ,ঐ,ঔ
44	ঔ	উ
45	অ্যা	আ,অ
46	ইয়	অ,ইউ

Figure 6. Search prefixes used in Replace operation of the HRDP algorithm

4.3.3. Candidate Ranking

The list of candidate transliterations B is ordered in a certain way to make more probable transliterations appear towards the beginning of the list. The SI module considers three different metrics for this purpose.

1. ***Segment Mapping Score*** **(SMS):** Since Avro is the most popular Bangla phonetic typing tool, encountering phonetic text written using its typing rules and conventions has become commonplace. Therefore, SI module considers which segments of b_t can be obtained from E following Avro's phonetic typing rules [25] and favours the candidate which better conforms the aforesaid rules and typing conventions.
2. ***Word Popularity Score*** **(WPS):** Popularity of b_t is also considered and is defined as the number of times b_t appeared in a large MT dataset [26]. More popular words are favoured against less popular ones. For example, for an input $E =$ vi, $b_t =$ ভাই is ranked before $b_t =$ ভভ despite the latter being a more accurate transliteration phonetically since the former is more common in Bangla text.
3. ***Edit Distance Penalty*** **(EDP):** The penalty incurred during HRDP algorithm by b_t is also considered and candidates with less penalty are advanced in the final ordering.

EDP depends on the performance of the PT module. The underlying phonetic transliteration engine *Epitran* shows poor performance for E_{Bn} because they often do not follow English pronunciation rules. Consequently, SMS is given a higher priority than EDP to favour phonetic typing rules over usual English pronunciation rules. The final ordering is determined by a linear combination of the three metrics whose coefficients have been set empirically as follows:

$$\text{rank}(b_t) = 1.0 \times \text{SMS} + 0.3 \times \text{WPS} - 0.2 \times \text{EDP} \quad (3)$$

where, each metric has been normalized between $[0,1]$ for all candidates in B. This ensures that the most probable candidates are included in the output if number of candidates is restricted by the application. This also makes the framework suitable for independent transliteration tasks.

4.4. Flexibility of the Framework

The PT and BT modules of the THT framework corresponds to the rule-based transliteration approach whose output is refined at the SI module by integrating a dictionary search. The HRDP algorithm enables the framework to smartly manipulate transliterations and produce improved candidate transliterations.

The THT framework is flexible and allows various configurations to suit its application needs. Firstly, the PT module uses a 3rd party transliteration tool named *Epitran*. However, other more sophisticated or simpler ones can be used. Secondly, both BT and SI modules can be configuredto limit the number of outputs. Reducing the number of candidate transliterations will make the conversion faster, but may miss some desirable outcomes. Finally, the dictionaries used for LM score as well as Tr can be modified to include terminologies of a particular discipline or spellings of different accents and dialects.

Additionally, the THT framework outputs a list of words B with corresponding penalties. Though the word with lowest penalty is usually desirable, sometimes the intended transliteration may havea higher penalty. This is especially frequent for cases discussed in Section 3.2. The MT system uses additional tools like language models to choose the most suitable one according to its context.The

penalty itself can be configured to so that each one of the *copy, replace, delete and insert* operations incurs different penalty.

Though the THT framework is designed for English to Bangla transliteration, it can be extended for other language pairs by providing corresponding transliteration graphs and other mappings in Figures 2, 4, 5 and 6.

5. PERFORMANCE ANALYSIS

The proposed THT framework has been implemented and tested to measure its performance. This section provides a critical analysis of performance results.

5.1. Hyperparameters

Python language has been used for implementing the THT framework. It was then tested on a machine having Intel® Core™ i7-8700 CPU @ 3.20GHz × 12, 16 GB ram and ubuntu 18.04 LTS OS.

For LM score, a 5-gram Bangla word level corpus over 3703012 unique words was built and *kenlm* was used to calculate the LM probability score [22]. A weight of $z = 0.5$ was used in (2) to assign equal weight to LM and VM. A Bangla word dictionary containing 1654953 unique and clean words was used to construct Tr [27], [28]. A value of $f = 5$ has been used for Spelling Improvement module.

5.2. Experimental design

The THT framework has been tested in two scenarios. Firstly, as a component of a MT system where a suitable transliteration is chosen from the list of candidate transliterations by the MT system. Therefore, performance of THT depends on whether the desired transliteration appears in the candidate list B. Additionally, the index of the desired transliteration in B is also noted.

The second experiment assess the performance of THT as a standalone transliteration tool. Only the first candidate b_1 is considered in this case and checked how closely it matches with the desirable transliteration.

One notable issue here is that there is no well-known Bangla transliteration dataset available in the literature. As a result, a small transliteration dataset has been prepared for these experiments with help from independent volunteers. A total of fifteen volunteers participated in the study. All of them are Bangladeshi individuals who use various social media and chat applications on a daily basis. The participants are aged from 18 to 35 and have at least twelve years of formal education.

5.3. Experimental results analysis

5.3.1. THT as a component of MT

For the first experiment, the participants contributed one or more sentences from their everyday communication over a period of one month. Each sentence is written using English alphabets and includes both English and phonetic typed Bangla words. They also provided desired transliterations of respective sentences. The dataset is then cleaned up to remove punctuations, unnecessary spaces and some obvious spelling mistakes.

A total of 3537 unique words is then chosen from the sentence database for the experiment. These words are then provided to the THT framework as input and index of corresponding transliteration in the candidate list B is noted which has been presented in Table I. It can be seen that THT successfully produced desired transliterations for over 98% of the input words. Moreover, more than 90% of the transliterations appeared within the first 10 candidates.

Table 1. Performance of THT at producing desired transliterations

Index	**Count**	**%**
Top 1	1852	52.36%
Top 5	2915	82.41%
Top 10	3209	90.73%
Top 20	3374	95.39%
Not found	49	1.39%

It is evident form Table I that THT was able to produce the desired transliterations for almost all input words. It should be noted that the input contains both English and Phonetic typed Bangla words and THT has no information regarding a word's context.

THT did fail to produce desired transliteration for a small portion of the input. Upon further inspection, these words can be primarily categorized into two groups:

- **Abbreviation and shorthand:** Some abbreviation and shorthand popular in chat applications were included in input. For example, 'w8' as a shorthand for the word 'wait'.
- **Non-standard spelling:** Some inputs were found to be composed of multiple words where standard practice is to write them separately. For example, an input 'manayna' should be 'manay na'

The primary reason that THT could not produce desired output is because these words have non-standard spellings and do not belong to the dictionary used for LM training and building Tr in the SI module. Including such words in the respective dictionaries will compensate for this and improve THT performance.

5.3.2. THT as a standalone transliteration tool

For the second experiment, the participants were asked to provide some English and phonetic typed Bangla words they would like to see transliterated. The provided words were divided into four categories, namely, English word, English name, Bangla word and Bangla name. From each category, 50 words have been randomly selected to create the test dataset.

Next the 50 chosen words from each category, totalling 200 words, are transliterated using the THT framework as well as Avro for comparison. Afterwards, the participants were asked to evaluate each transliteration. The source of a transliterated word was kept hidden from them for unbiased evaluation.

The survey participants evaluated each transliteration into three categories based on their preferences:

1. *Expected spelling* (ES): If the spelling matches participants' expectation.
2. *Close spelling* (CS) If the spelling is not an exact match, but close to participants'expectation.
3. *Wrong spelling* (WS) If the spelling is wrong in participants' opinion.

Table 2. Performance of THT as a standalone transliteration tool

		ES (%)	CS (%)	WS (%)
English word	THT	78	22	0
	Avro	18	2	80
English Name	THT	96	4	0
	Avro	10	10	80
Bangla Word	THT	88	10	2
	Avro	84	2	14
Bangla Name	THT	96	4	0
	Avro	64	20	16

The survey responses was then aggregated and summarised to obtain critical insights regarding transliteration performance which has been presented in Table II. It is evident that the THT framework significantly outperforms the competition, especially for English words and Names where Avro performed poorly.

Avro is designed as a phonetic typing tool and does not handle English words well, which has been a major motivation of this study. By employing a hybrid of dictionary and rule-based approach, THT was able to produce desired transliteration for most input words. The English words were transliterated correctly by employing the intermediate PT module whereas Bangla words were processed through the SI module to produce better outcomes.

5.3.3. Potential for Dataset Generation

Another potential application of THT can be producing transliteration dataset. The integrated dictionary at the SI module ensures that only valid words are produced as candidate transliterations. Therefore, it can be used to produce large transliteration datasets in a similar manner to [13], [29] where manually creating one would be impractical due to its scale. The produced dataset can then be used for training machine learning models which can eliminate the need for manually creating transliteration graphs and mappings of Figures 2, 4, 5 and 6 [13], [16].

6. CONCLUSION

The proposed *Three-stage Hybrid Transliteration* (THT) framework is a combination of both rule-based and dictionary-based solution which provides robust transliteration of both English and phonetic typed Bangla words. It addresses the limitations of existing tools by adopting a three-layer approach. The input words are first converted into phonetic form before obtaining a Bangla transliteration. Its spelling is later improved using a heuristic runtime dynamic programming algorithm. The design also allows sufficient flexibility to customise various aspects to suit application needs. Experimental results confirm significantly superior performance of the THT framework compared to existing transliteration tool.

REFERENCES

[1] Y. Wu et al., ‘Google’s neural machine translation system: Bridging the gap between human and machine translation’, arXiv preprint arXiv:1609.08144, 2016.

[2] K. Knight and J. Graehl, ‘Machine transliteration’, arXiv preprint cmp-lg/9704003, 1997.

[3] N. Habash, A. Soudi, and T. Buckwalter, ‘On arabic transliteration’, in Arabic computational morphology, Springer, 2007, pp. 15–22.

[4] K. Regmi, J. Naidoo, and P. Pilkington, ‘Understanding the processes of translation and transliteration in qualitative research’, International Journal of Qualitative Methods, vol. 9, no. 1,

pp. 16–26, 2010, doi: 10/gcdvcf.
[5] Naushad UzZaman and Mumit Khan, 'A comprehensive Bangla spelling checker', in Proceeding of the International Conference on Computer Processing on Bengali (ICCPB), Dhaka, Bangladesh, 2006, p. 8.
[6] M. F. Akan, 'Transliteration and Translation from Bangla into English: A Problem Solving Approach', British Journal of English Linguistics, vol. 6, no. 6, pp. 1–21, 2018.
[7] AvroPad. OmicronLab, 2014. Accessed: Dec. 20, 2021. [Online]. Available: https://github.com/omicronlab/avro-pad
[8] T. Dasgupta, A. Basu, A. Das, and P. Mandal, 'Design and evaluation of Bangla keyboard layouts', in 2010 IEEE Students Technology Symposium (TechSym), 2010, pp. 248–254. doi: 10/cv76qd.
[9] M. Z. Rahman and F. Rahman, 'On the context of popular input methods and analysis of phonetic schemes for Bangla users', in 2007 10th international conference on computer and information technology, 2007, pp. 1–5.
[10] B. Ahmadnia and B. J. Dorr, 'Augmenting neural machine translation through round-trip training approach', Open Computer Science, vol. 9, no. 1, pp. 268–278, 2019, doi: 10/gnxz23.
[11] B. Ahmadnia, J. Serrano, and G. Haffari, 'Persian-Spanish low-resource statistical machine translation through English as pivot language.', in RANLP 2017 - Recent Advances in Natural Language Processing Meet Deep Learning, 2017, pp. 24–30. doi: 10/gnx2j9.
[12] D. He et al., 'Dual learning for machine translation', Advances in neural information processing systems, vol. 29, pp. 820–828, 2016.
[13] Y. Merhav and S. Ash, 'Design challenges in named entity transliteration', in Proceedings of the 27th international conference on computational linguistics, 2018, pp. 630–640.
[14] S. Ahmadi, 'A rule-based Kurdish text transliteration system', ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), vol. 18, no. 2, pp. 1–8, 2019, doi: 10/gntdnx.
[15] D. Bhalla, N. Joshi, and I. Mathur, 'Rule based transliteration scheme for English to Punjabi', arXiv preprint arXiv:1307.4300, 2013.
[16] S. Kundu, S. Paul, and S. Pal, 'A deep learning based approach to transliteration', in Proceedings of the seventh named entities workshop, 2018, pp. 79–83. doi: 10/gntdnb.
[17] B. Hayes and A. Lahiri, 'Bengali intonational phonology', Natural language & linguistic theory, vol. 9, no. 1, pp. 47–96, 1991, doi: 10/c778gk.
[18] D. C. Uthus and D. W. Aha, 'Multiparticipant chat analysis: A survey', Artificial Intelligence, vol. 199, pp. 106–121, 2013, doi: 10/gbdb5z.
[19] I. P. Association, I. P. A. Staff, and others, Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet. Cambridge University Press, 1999.
[20] D. R. Mortensen, S. Dalmia, and P. Littell, 'Epitran: Precision G2P for many languages', presented at the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 2018.
[21] 'Bangla Juktoborner Talika O Gothon Pronali-বাংলা যুক্তবর্ণের তালিকা ও গঠন প্রনালী'. https://writebangla.com/juktoborno.html (accessed Dec. 20, 2021).
[22] K. Heafield, 'KenLM: Faster and smaller language model queries', in Proceedings of the sixth workshop on statistical machine translation, 2011, pp. 187–197.
[23] D. E. Willard, 'New trie data structures which support very fast search operations', Journal of Computer and System Sciences, vol. 28, no. 3, pp. 379–394, 1984, doi: 10/cvq9sr.
[24] P. Mandal and B. M. Hossain, 'Clustering-based Bangla spell checker', in 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), 2017, pp. 1–6. doi: 10/gnq7r5.
[25] omicronlab, 'Bangla Computing Documentations, articles, help files'. https://www.omicronlab.com/docs.html (accessed Dec. 19, 2021).
[26] T. Hasan et al., 'Not Low-Resource Anymore: Aligner Ensembling, Batch Filtering, and New Datasets for Bengali-English Machine Translation', in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, Nov. 2020, pp. 2612–2623. doi: 10/gm3nqt.
[27] M. Kamal, 'Bengali Dictionary'. https://github.com/MinhasKamal/BengaliDictionary
[28] A. El-Kishky, A. Renduchintala, J. Cross, F. Guzmán, and P. Koehn, 'XLEnt: Mining Cross- lingual Entities with Lexical-Semantic-Phonetic Word Alignment', Online, 2021.
[29] B.-J. Kang and K.-S. Choi, 'Automatic Transliteration and Back-transliteration by Decision Tree Learning.', 2000.