

**PHONETIC ENCODING FOR BANGLA AND ITS
APPLICATION TO SPELLING CHECKER,
TRANSLITERATION, CROSS LANGUAGE INFORMATION
RETRIEVAL AND NAME SEARCHING**

A Thesis

Submitted to the Department of Computer Science of
BRAC University

by

Naushad UzZaman

Student ID: 01201019

In Partial Fulfillment of the
Requirements for the Degree of
Bachelor of Science in Computer Science
May 2005



BRAC University, Dhaka, Bangladesh

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of
Supervisor

Dr. Mumit Khan

Signature of
Author

Naushad UzZaman

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor, Dr. Mumit Khan. He gave me not only full freedom to choose my research topic, but also extended a lot of guidance throughout its development. I was lucky enough to work under him at the *Center for Research on Bangla Language Processing, BRAC University* for last one year. Although being extremely preoccupied with his busy schedule, he often showed much enthusiasm and took time and lot of pain to review drafts of my paper that enabled me to improve the contents as well as my presentation. I learned plenty of useful things from his comments, revisions and discussions during this period which taught me to write better research papers. Being a undergraduate student, I got two of my papers with him accepted in International conferences and also submitted two more with him in another International conference, which I feel a great achievement of my under-graduation study at BRAC University. And it was possible for me mainly because of my supervisor's support.

I want to give my heartiest gratitude to all the faculty members of BRAC University for their helping hands. Many thanks to all of my friends for being with me and always encouraging me and special thanks to my colleagues working in Dr. Mumit Khan's research team, Sajib Dasgupta, Dewan Shahriar Hossain Pavel, Asif Iqbal Sarkar, Sheemam Monjel and Rowshon Jahan Nupur, who have seen the Double Metaphone a thousand times and being patient listeners helped me with their valuable suggestions.

I am also grateful to the reviewer of conferences, where I submitted my papers. Their valuable reviews were very helpful for me to understand and

improve my paper a lot.

Finally, special thanks and love to my father for his constant guidance and encouragement in my study and research work, my mother for her prayers and my one and only brother for his brotherly support and love.

Last but not the least, thanks to the Almighty for helping me in every step of this work.

To my family, friends & well-wishers

ABSTRACT

We present a phonetic encoding for Bangla that can be used by spelling checkers, transliteration, name searching application and cross-lingual information retrieval to drastically improve the quality. The complex, and often inconsistent, rules of Bangla word present a significant challenge in producing a proper phonetic code. We propose a phonetic encoding for Bangla, taking into account the various context-sensitive rules, including those involving the large repertoire of conjuncts in Bangla.

TABLE OF CONTENT

DECLARATION	ii
ACKNOWLEDGMENTS	iii
ABSTRACT	vi
TABLE OF CONTENT	vii
LIST OF TABLES	x
LIST OF FIGURES	xi
Chapter I: INTRODUCTION	1
CHAPTER II: PHONETIC ENCODING.....	3
2.1. Definition	3
2.2. Phonetic Encoding for English	3
2.2.1. Soundex.....	4
2.2.2. Metaphone	6
2.2.3. Phonix.....	8
2.2.4. Double metaphone.....	8
2.3. Existing Phonetic Encoding for Bangla.....	10
2.3.1. Hoque and Kaykobad’s soundex type encoding [10, 11]	10
2.3.1.1. Producing simplified format / open format.....	12
2.3.1.2. Producing sound-code or Bangla soundex or Bsoundex	12
2.3.2. Zaman and Khan’s soundex type encoding	13
Case 1	13
Case 2	14
Case 3	14
2.3.2.1. Phonetic matching technique for Bangla.....	15
2.3.2.2. Summary of soundex for Bangla	17
2.3.2.3. Encoding reasoning for 0 (not coded) characters	18
2.3.2.4. Example of error correction using phonetic matching	19
CHAPTER III: RESEARCH QUESTION.....	20
3.1. Scope Of Our Proposed Encoding	20
3.2. Limitation Of Previous Encoding	21
1. Hoque and Kaykobad, 2002, BSoundex [10, 11]	22

2.	Zaman and Khan, 2004 [9], soundex type phonetic encoding.....	23
3.3.	Why It Is Worthwhile To Answer	24
CHAPTER IV: PROPOSED ENCODING		26
4.1.	Proposed phonetic encoding for words	26
4.1.1.	Phonetic encoding table.....	27
4.1.2.	Encoding reasoning	31
CHAPTER V: APPLICATIONS OF PHONETIC ENCODING		48
5.1.	Spelling Checker	48
5.1.1.	Spelling error patterns	48
5.1.2.	Previous spelling checking techniques	50
5.1.2.1.	Approximate string matching algorithm.....	50
	Levenshtein Edit Distance[23, 24, 25].....	50
	Longest common substring (LCS) [26].....	51
5.1.2.2.	BB Choudhury's Reverse dictionary method.....	53
	Phonetically similar character error correction	53
	Reversed word dictionary.....	53
	Error detection & position finding and Error correction	54
	Insertion and transposition:	56
	Deletion and substitution:	56
5.1.2.3.	Abdullah and Rahman's Recursive simulation method [19] ...	57
	Algorithm: RecursiveSimulation	61
	Sorting the suggestion list:	63
5.1.2.4.	Hoque and Kaykobad's soundex type encoding	66
5.1.2.5.	Zaman and Khan's soundex type encoding	66
5.1.3.	Performance of previous techniques.....	66
5.1.4.	How to rank.....	69
5.1.5.	Performance of our proposed encoding	70
5.2.	Transliteration	73
5.2.1.	What is transliteration.....	73
5.2.2.	Previous transliterations.....	74
5.2.3.	Proposed new technique for transliteration	75

5.2.3.1.	Direct mapping.....	75
5.2.3.2.	Phonetic mapping	78
	Algorithm of phonetic mapping.....	78
5.2.4.	Example of transliteration.....	82
5.2.4.1.	Direct mapping.....	82
5.2.4.2.	Phonetic mapping	83
5.3.	Cross Language Information Retrieval	85
5.3.1.	What does it handle	85
5.3.2.	Previous work.....	85
5.3.3.	How does it work.....	86
5.3.4.	Example	87
	Bangla Text:.....	87
	Encoding of Bangla Text:.....	87
5.4.	Name Searching and Matching	89
5.4.1.	Proposed name encoding for Bangla.....	90
5.4.2.	Rationale for Name encoding.....	96
5.4.1.	Algorithm and performance of name searching using proposed phonetic encoding	99
	Algorithm for Name searching	100
CHAPTER VI: CONCLUSION		102
6.1.	Summary of contributors	102
6.2.	Future research	103
REFERENCES		- 1 -
APPENDICES		i
A.	Bangla Alphabet	i
B.	Bangla Unicode Chart.....	ii
C.	IPA (International Phonetic Alphabet).....	vi
D.	Bangla IPA Chart	vii

LIST OF TABLES

Table 1: Soundex encoding table	4
Table 2: PHONIX encoding table.....	8
Table 3: Hoque and Kaykobad's phonetic encoding for Bangla	10
Table 4: Bangla phonetic encoding table.....	15
Table 5: Suggestions for misspelled words	19
Table 6: Double Metaphone Phonetic Encoding table for words	27
Table 7: Edit distance example.....	51
Table 8: LCS example	52
Table 9: Challenges for spelling checker and performance of previous techniques	68
Table 10: Encoding performance.....	70
Table 11: Error distribution	70
Table 12: Performance of proposed phonetic encoding	71
Table 13: Table for direct mapping	75
Table 14: Modification in proposed encoding	79
Table 15: Proposed encoding for phonetic mapping	80
Table 16: Few examples from above paragraph to make the process clear	83
Table 17: English word, encoding of English word, Bangla word with the same encoding from the text.....	87
Table 18: Proposed Name Encoding for Bangla.....	90
Table 19. Example of vowels encoding	96
Table 20. Example of স and ছ.....	97
Table 21. Example of হ.....	98
Table 22. One to one transformation of ঃ.....	98
Table 23: Generating suggestions for names using name encoding and other trivial methods	100

LIST OF FIGURES

Figure 1: The Soundex algorithm	5
Figure 2: Flowchart of producing sound code from a given word	11
Figure 3: The Soundex algorithm for Bangla	17
Figure 4: Error localization by conventional and reverse dictionary.....	55
Figure 5: List of phonetically similar letters	58
Figure 6: List of vowel-symbols (known as kaar)	58
Figure 7: List of consonant symbols (known as folaa & reff)	58
Figure 8: Circular lists of the grouped letters	58
Figure 9: Some compound letters and their formation.....	59
Figure 10: Some common Bangla words with their miss-spelt forms	59
Figure 11: Simulated suggestion list for the word misspelled word, using Recursive Simulation algorithm.	65

Chapter I: INTRODUCTION

Bangla, also known as Bengali, is the language of approximately 210 million people, the majority of whom live in Bangladesh and in the Indian state of West Bengal, making it the 4th most widely spoken language in the world. It belongs to the leftmost branch, called the Aryan or Indo-Iranian, of the Indo-European family of languages, and is written in the Brahmi-derived Bangla script. Bangla underwent a period of vigorous Sanskritization that started in the 12th century and continued throughout the middle ages, resulting in the vast gap between the script and the pronunciation [2]. Bangla lexicon today consists of *tatsama* (Sanskrit words that have changed pronunciation, but retaining the original spelling), *tadbhava* (Sanskrit words that have changed at least twice in the process of becoming Bangla), and a fairly large number of “loan-words” from Persian, Arabic, Portuguese, English, and other languages. There are also a large number of words of unknown etymology, which may have originated from Dravidian, Austric or Sino-Tibetan languages. All of these contribute to the complexity of the Bangla spelling rules, with the Sanskritization process as the largest contributor. An additional factor is the large number of consonant clusters or *juktakhors* (typically represented as conjuncts in the written form) in Bangla, where each consonant in the cluster except for the last one loses its inherent vowel. One impact of this complexity can be seen in the observation that two of the most common reasons for misspelling are (i) phonetic similarity of Bangla characters, and (ii) the difference between the grapheme representation and the phonetic utterances [3].

Phonetic encoding has a wide variety of applications. It was first proposed for name searching application in census [4] but later its application extended to spelling checker. The performance of these applications depends on how better

the code represent the pronunciation. Phonetic encoding is always a great challenge in a language. English and other Western languages have well established phonetic encodings [4, 5, 7, 8], but similar work for Bangla has barely begun [9, 10]. These efforts are based mostly on Soundex [4] or other *ad-hoc* methods, which cannot handle the complexity of Bangla spelling rules. This is the primary motivation for creating a phonetic encoding that can handle such complexity.

After Introduction, In Chapter II, we will describe about phonetic encoding in detail, which will include well established encoding for English and also existing encoding for Bangla as well. In Chapter III, we will describe the Research question, which includes the scope of our encoding, the limitations of other encoding and the importance of our encoding. In Chapter IV, we proposed our encoding with reasoning. In Chapter V, we have shown how currently spelling checker, transliteration, name searching work and how we drastically improve its performance using phonetic encoding, we also introduced a new application for Bangla, which is cross-lingual information retrieval for Bangla in this chapter. And finally in conclusion, we summarized how this encoding helps in applications like spelling checker, transliteration, cross-lingual information retrieval and name searching.

CHAPTER II: PHONETIC ENCODING

2.1. Definition

Code a string based on how it is pronounced. [1]

The input of a phonetic encoding or “sound-alike” algorithm is a word, and the result is an encoded key, which should be same for all words that are pronounced similarly, allowing for a reasonable amount of fuzziness.

For example, metaphone-encoding [5] gives the code *RLS* for the word *realise* in English.

We know that *realize* and *realise* has the same pronunciation. Hence a good encoding in English should be able to give the same code *RLS* to *realize* as well.

2.2. Phonetic Encoding for English

In English, a major class of approximate string matching algorithms is the various phonetic methods, from the eighty-year old Soundex [4], to the more recent Metaphone [5], Double metaphone [7] and PHONIX [8]. The basic principle behind these phonetic matching schemes is to partition the consonants by phonetic similarity, and then use a single key to encode each of these sets. For these particular algorithms, only the first few consonant sounds are encoded, unless the first letter is a vowel. Metaphone for example encodes “Stephan”, “Steven”, and “Stefan” as STFN, so all three names compare equal when encoded.

A brief description on each of the phonetic encoding for English is given below.

2.2.1. Soundex

Among all the phonetic methods, Soundex method is by far the oldest, first patented by Odell and Russel in 1918. Soundex partitions the set of letters into seven disjoint sets, assuming that the letters in the same set have similar sound. Each of these sets is given a unique key, except for the set containing the vowels and the letters *h*, *w*, and *y*, which is considered to be silent and is not considered during encoding. The Soundex codes are shown in Table 1: Soundex encoding table. The Soundex algorithm itself, shown in Figure 1: The Soundex algorithm, transforms all but the first letter of each string into the code, and then truncates the result to be at most four characters long. Zeros are added at the end if necessary to produce a four-character code. For example, Washington is coded W-252 (W, 2 for the S, 5 for the N, 2 for the G, remaining letters disregarded), and Lee is coded L-000 (L, 000 added).

A limitation of Soundex is that it does not know the intricacies of complex spelling rules for English, and because it works on a letter-by-letter basis, it often does not produce the expected result. Another limitation is that truncating the words to four-character code ignores differences in long strings, which may not be appropriate when finding alternatives for misspelled words.

An advantage of Soundex is the small table size and simplicity of the letter-by-letter algorithm, which can provide significant speedup over the other phonetic methods.

Table 1: Soundex encoding table

<i>Code</i>	<i>Letters</i>
0 (not coded)	A, E, I, O, U, H, W, Y
1	B, F, P, V

<i>Code</i>	<i>Letters</i>
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

1. Capitalize all letters in the word and drop all punctuation marks. Pad the word with rightmost blanks as needed during each procedure step.
2. Retain the first letter of the word.
3. Change all occurrence of the following letters to '0' (zero):
'A', 'E', 'I', 'O', 'U', 'H', 'W', 'Y'.
4. Change letters from the following sets into the digit given:
 - 1 = 'B', 'F', 'P', 'V'
 - 2 = 'C', 'G', 'J', 'K', 'Q', 'S', 'X', 'Z'
 - 3 = 'D', 'T'
 - 4 = 'L'
 - 5 = 'M', 'N'
 - 6 = 'R'
5. Remove all pairs of digits which occur beside each other from the string that resulted after step (4).
6. Remove all zeros from the string that results from step 5.0 (placed there in step 3)
7. Pad the string that resulted from step (6) with trailing zeros and return only the first four positions, which will be of the form <uppercase letter> <digit> <digit> <digit>.

Figure 1: The Soundex algorithm

2.2.2. Metaphone

The Metaphone algorithm, proposed by Lawrence Phillips, 1990, is also a system for transforming words into codes based on phonetic properties. However, unlike Soundex, which operates on a letter-by-letter scheme, Metaphone analyzes both single consonants and groups of letters called diphthongs, according to a set of rules for grouping consonants, and then mapping groups to Metaphone codes.

Details on metaphone encoding can be found in [5, 6].

The Metaphone Rules

Metaphone reduces the alphabet to 16 consonant sounds:

B X S K J T F H L M N P R 0 W Y

That isn't an O but a zero - representing the 'th' sound.

Transformations

Metaphone uses the following transformation rules:

Doubled letters except "c" -> drop 2nd letter.

Vowels are only kept when they are the first letter.

B -> B unless at the end of a word after "m" as in "dumb"

C -> X (sh) if -cia- or -ch-

S if -ci-, -ce- or -cy-

K otherwise, including -sch-

D -> J if in -dge-, -dgy- or -dgi-

T otherwise

F -> F

G -> silent if in -gh- and not at end or before a vowel
in -gn- or -gned- (also see dge etc. above)

J if before i or e or y if not double gg

K otherwise

H -> silent if after vowel and no vowel follows

H otherwise

J -> J
 K -> silent if after "c"
 K otherwise
 L -> L
 M -> M
 N -> N
 P -> F if before "h"
 P otherwise
 Q -> K
 R -> R
 S -> X (sh) if before "h" or in -sio- or -sia-
 S otherwise
 T -> X (sh) if -tia- or -tio-
 O (th) if before "h"
 silent if in -tch-
 T otherwise
 V -> F
 W -> silent if not followed by a vowel
 W if followed by a vowel
 X -> KS
 Y -> silent if not followed by a vowel
 Y if followed by a vowel
 Z -> S

Initial Letter Exceptions

Initial kn-, gn- pn, ac- or wr-	-> drop first letter
Initial x-	-> change to "s"
Initial wh-	-> change to "w"

2.2.3. Phonix

PHONIX is similar to Soundex in that letters are mapped to a set of codes. Prior to this mapping however, PHONIX applies preliminary transformations to letter groups in order to reduce strings to a canonical form. For example, *gn*, *ghn*, and *gne* are mapped to *n*, the sequence *tjV* (where *V* is any vowel) is mapped to *chV* if it occurs at the start of a string, and *x* is transformed to *ecs*. PHONIX applies altogether about 160 of these transformations. These transformations provide a certain degree of context for the phonetic coding and allow, for example, *c* and *s* to be distinguished, which is not possible under Soundex. The Phonix codes are shown in Table 2: PHONIX encoding table.

Table 2: PHONIX encoding table

Code	Letters
0 (not coded)	A, E, H, I, O, U, W, Y
1	B, P
2	C, G, J, K, Q
3	D T
4	L
5	M, N
6	R
7	F, V
8	S, X

2.2.4. Double metaphone

Lawrence Philips, 2000, also proposed double metaphone. Metaphone worked fine in most of the cases but there were a few cases

that metaphone cannot handle [7]. Such as,

- Bryan (BRYN) was not matched to Brian (BRN).
- MacCafferey is encoded to MKKF, an out-and-out bug.
- Retaining Soundex's choice of preserving the first letter (in Metaphone, only for words that started with vowels), "Otto" for example, cannot be matched to "Auto."
- More difficult to deal with, and contributing considerably to inelegance, are the consonants that are pronounced differently in different words. For example, "gh" in light and rough.
- English has a tendency to accumulate a large number of words from non-English sources, notably French, Latin, and Greek. When transliterating from the Greek alphabet, the letter that is pronounced "kh" is Greek (a sound that does not exist in English – think "chutzpah"), is spelled "ch" and pronounced "k": "orchestra", "chorus", etc.
- Most importantly, some familiar names can just as plausibly be pronounced more than one way. Henry Kissinger and Kim Basinger are example of that type. Basinger is pronounced in both way as "Basin-gger" or "Basin-jer".

These problems led Philips to propose another phonetic encoding, Double metaphone, which perform better but not perfect. Main improvement of this encoding is it will give two keys for words and names that can be plausibly pronounced more than one way.

For example, in case of Kuczewski, there are two ambiguous sounds, so "Kuczewski" now comes back as KSSK for the American version, "Kuhzooski," as well as KXFS for "Kutchefski." ('X' is used to represent the "sh" sound, and '0', zero, to represent "th," as in original

Metaphone.)

2.3. Existing Phonetic Encoding for Bangla

Eighty years old technique of phonetic encoding is new in Bangla. Hoque and Kaykobad, 2002 [10], first proposed it. After that Zaman and Khan, 2004 [9], proposed their version of soundex type Bangla phonetic encoding. Both of the encoding use “soundex” in their encoding name. Reason behind it is they follow the general principal of soundex encoding, to partition the letters in to disjoint sets.

2.3.1. Hoque and Kaykobad’s soundex type encoding [10, 11]

Using the major concepts from the soundex encoding [4], Hoque and kaykobad’s proposed encoding maintain following rules. [10]

- Same sounding letters would have same value, e.g. NA (ন) and NNA (ণ)
- Same composite consonants would have single value, as if ক (KA HASANT KA)→ ক (KA)

Their proposed encoding is given in following Table 3: Hoque and Kaykobad’s phonetic encoding for Bangla.

Table 3: Hoque and Kaykobad’s phonetic encoding for Bangla

Group Name	Group Member
1	ক, খ, গ, ঘ, ঙ
2	চ, ছ, জ, ঝ, য
3	ট, ঠ, ড, ঢ
4	ত, থ, দ, ধ, ত্
5	প, ফ, ব, ভ
6	ঙ, ঞ, ং
7	শ, স, ষ
8	র, ড়, ঢ়, ঝ়

৭	ন, ণ
α	ম
β	ল

Like Soundex [4], they only give codes to consonants. But they consider some special cases to handle the exceptions.

Consonant য় sounds like অ, which sounds vowel like. Also, while we pronouncing হ, the airflow do not face any barrier, that implies that হ sound has vowel like impact. So, য় and হ are not involved in grouping. A few vowels are involved in the grouping because of their consonant like sound, e.g. ঞ.

To suggest similar sounding words for the miss-spelled word, each word stored in the dictionary would store sound code. The generation of the sound-code is outlined via flow chart below.

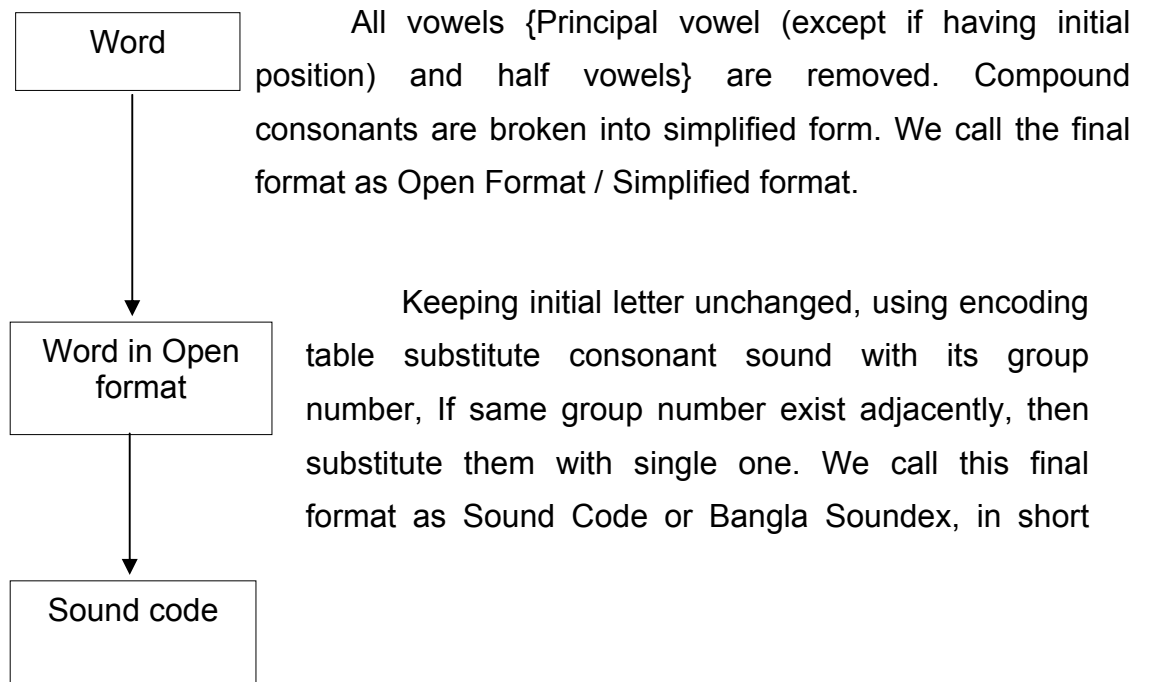



Figure 2: Flowchart of producing sound code from a given word

They did not state specifically but their following description shows that they used the ASCII encoding rather than generalized Unicode encoding for encoding the Bangla text.

2.3.1.1. Producing simplified format / open format

The word need to be simplified before generating Sound-code or Bsoundex. According to them there exists number of character(s) or symbol(s) in Bangla that are placed after the consonants but pronounced before the consonants. For example,  Ref sounds before the sound of the consonant after which it is placed.

At first step, above-mentioned words are rearranged according to their sounding sequences. For example,

Word, 'কর্ম' (ক + ম + ') would be converted to 'র্কম' (ক + ' + ম)

Secondly, all vowels {Principal vowel (except if having initial position) & half vowels} are removed. Compound consonants are broken into principal form that is member of Table 3: Hoque and Kaykobad's phonetic encoding for Bangla.

For example,

র্কম would be converted to করম

তাদের would be converted to তদর

অদ্ভূত would be converted to অদভত

ক্লেশ would be converted to কলশ

Thus, Simplified format or open format is produced.

2.3.1.2. Producing sound-code or Bangla soundex or Bsoundex

Keeping initial letter unchanged, using Table 3: Hoque and

Kaykobad's phonetic encoding for Bangla, substitution of the decomposed or simplified component (alphabet) of word is done with the respective group number, if the adjacent group has the same number then only one such group is used. The final format is the sound code or Bangla Soundex, in short Bsoundex.

For example, কৱম will be converted to a 4 lengthen sound code as “ক8α0”, with zero padding.

This is how, BSoundex encodes a word.

2.3.2. Zaman and Khan's soundex type encoding

Zaman and Khan, 2004 [9], proposed a Soundex type encoding for Bangla. Unlike [11], this Soundex type encoding does not follow each and every rule of English soundex, rather than that it customizes the soundex and based on the Bangla's nature it proposes a encoding. This can better handle the complex part of Bangla word, conjuncts or jukhtakkhor, where each consonant in the cluster except for the last one loses its inherent vowel.

There are some rules for English Soundex, but we can not use those in Bangla. Following are the reasons for that.

Case 1: Soundex keeps the first letter of the string in the encoding.

Problem: This is in fact a general problem with Soundex. If there is a spelling error in the first character of the word, the correct suggestion cannot be produced using Soundex. For example, if we write গুম instead of ঘুম, Soundex will not be able to suggest the correct alternative, as the incorrectly spelled word গুম will begin with গ independent of the character encoding used, Unicode or otherwise. at the beginning. Since the

phonetically encoded lexicon will have the word ঘুম encoded as something that begins with ঘ, the phonetic method will never produce ঘুম as a suggestion for গুম. Of course, other *edit-distance* algorithms (e.g., Levenshtein [23]) are able to produce the correct suggestion in this particular case, so a spelling checker employing other similarity measures will produce the expected result (See [12] for a summary of the various *edit-distance* algorithms).

Case 2: Soundex excludes vowels when encoding strings.

Problem: The অ vowel is often used as a prefix to negate the meaning of Bangla words, and excluding it will often produce suggestions that are of the opposite meaning than the intended one. This may be appropriate behavior for some applications, but not for a spelling checker. For example, the words সুখ and অসুখ will result in the same Soundex code, even though we do not expect one as the suggested alternative for the other, much like we would not expect *unwell* as the suggested alternative for *well*.

Problem: Another problem of excluding the vowels is that words that are not phonetically similar and have very different meanings also produce the same code. বন and বানি, অকাজ and কাজি. বন (forest) and বানি for example will produce the same code if we exclude vowels, even if these words do not have same meaning, and in addition, are phonetically quite different. Similarly, in the case of অকাজ and কাজি, the অ from অকাজ and the ি from কাজি will be excluded to produce the same code, another undesired result.

Case 3: In soundex, consecutive repetitions of the same coded characters are eliminated.

Problem: Unicode specifies that the consonants that make up Bangla *juktakkhors* are separated by *hasant* character, which is not coded in our algorithm (i.e., eliminated during the phonetic encoding process). The side-effect of this decision to eliminate *hasant* is that, at least for a set of *juktakkhors*, consecutive repetitions of the same

consonants will have the same code as the single instance of that consonant. Using our algorithm, ঞ (ঞ্ ন) for example will have the same code as ন, since we exclude the *hasant* embedded in the Unicode representation of the conjunct. This particular problem is not a general Soundex problem, but rather a consequence of the way our algorithm handles Bangla conjuncts.

2.3.2.1. Phonetic matching technique for Bangla

Table 4: Bangla phonetic encoding table shows the proposed Bangla phonetic codes with Letter, Name (according to unicode found at [13]) and Unicode number (from [14]) & Figure 3: The Soundex algorithm for Bangla shows the modified Soundex algorithm using this encoding, suitable for a Bangla spelling checker.

Table 4: Bangla phonetic encoding table

Code	Letter	Name	Unicode
0 (zero)	্	Virama/Hasant	"09CD"
Not Coded	ো	Sign O	"09CB"
	ঁ	Candrabindu	"0981"
"a"	আ	AA	"0986"
	া	Sign AA	"09BE"
"i"	ই	I	"0987"
	ঈ	II	"0988"
	ি	Sign I	"09BF"
	ী	Sign II	"09C0"
"u"	উ	U	"0989"
	ূ	UU	"098A"
	ু	Sign U	"09C1"

	্	Sign UU	“09C2”
“e”	এ	E	“098F”
	ে	Sign E	“09C7”
	ঐ	AI	“0990”
	ৈ	Sign AI	“09C8”
“o”	অ	A	“0985”
	ও	O	“0993”
	ঔ	AU	“0994”
	ৌ	Sign AU	“09CC”
“k”	ক	KA	“0995”
	খ	KHA	“0996”
“g”	গ	GA	“0997”
	ঘ	GHA	“0998”
“m”	ম	MA	“09AE”
	ঙ	NGA	“0999”
	ং	Anusvara	“0982”
“c”	চ	CA	“099A”
	ছ	CHA	“099B”
“j”	য	YA	“09AF”
	জ	JA	“099C”
	ঝ	JHA	“099D”
“T”	ট	TTA	“099F”
	ঠ	TTHA	“09A0”
“D”	ড	DDA	“09A1”
	ঢ	DDHA	“09A2”
“r”	ঋ	Vocalic R	“098B”
	ৠ	RA	“09B0”
	ৡ	RRA	“09DC”
	ৣ	DDHA	“09A2”
“n”	ন	NA	“09A8”

	ণ	NNA	"09A3"
“t”	ত	TA	"09A4"
	থ	THA	"09A5"
“d”	দ	DA	"09A6"
	ধ	DHA	"09A7"
“p”	প	PA	"09AA"
	ফ	PHA	"09AB"
“b”	ব	BA	"09AC"
	ভ	BHA	"09AD"
“y”	য়	YYA	"09DF"
“l”	ল	LA	"09B2"
“s”	শ	SHA	"09B6"
	স	SA	"09B8"
	ষ	SSA	"09B7"
“h”	হ	HA	"09B9"
	ঃ	Visarga	"0983"

- [1] Replace all of s by its phonetic code.
 [2] Eliminate all occurrences of code 0 (i.e., eliminate *hasant*, *candrabinu*, *sign O*).
 [3] Return the resulting string.

Figure 3: The Soundex algorithm for Bangla

2.3.2.2. Summary of soundex for Bangla

Transformations

0 (Not Coded): 3 (Hasant, Candrabindu, Sign O: ো)

Vowels: 5 codes

Consonants: 17 codes

2.3.2.3. Encoding reasoning for 0 (not coded) characters

1. Name: Virama / Hasant; Unicode: 09CD; Character: ্

The absence of vowels between consonants can be represented by Virama / Hasant. This is used in the Jukhtakhor/Conjuncts.

In our encoding, we will give it 0 (zero) code. Because hasant means it is used to connect two or more consonants and we don't need to keep the information of connectors (hasant) in our encoding. And more importantly this is used to lower the sound of 1st consonant in conjuncts. And individually has No Sound in words.

This will also reduce one extra character error. For example, if someone misses the ্, then it's basically all the same. Mean he was trying to write some Conjuncts but missed the connector ্ so, if we consider it as 0 (zero) code we can reduce this error.

Example: দক্ষ = দ গ ্ ধ

We can see that we can easily reduce the ্ from our encoding.

2. Name: Sign O; Unicode: 09CB; Character: ো

ো (Sign O) is given 0 (zero) code, because in bangla words, O in the middle or end of word is an inherent vowel. For example, ভাল and ভালো. Both sound same and even if we don't have ো in ভাল, it will pronounce as ভালো. Because there is an inherent vowel ো in ভাল. Rather than adding inherent vowels in encoding we give ো 0 (zero) code. So, now ভাল and ভালো will have the same code.

3. Name: Candrabindu; Unicode: 0981; Character: ঁ

We give ঁ 0 (zero) code. ঁ is used for nasal words. Our main target is to encode the similar sounded characters in to the same code. Similar sounded characters means which sounds similar when we read it in our normal conversations not according to actual grammar. In normal conversations, we don't emphasize on nasal sounds and simply

pronounce it without ্ most of the cases. So, we can simply omit ্ from our encoding.

2.3.2.4. Example of error correction using phonetic matching

Table 4 shows a set of misspelled words, their corresponding encoded versions, and the suggested alternatives.

Table 5: Suggestions for misspelled words

Input	Encoded	Suggestion
খুমাড়	kumar	কুমার
পাসান	pasan	পাষণ
দগধ	Dgd	দক্ষ (দগ ্ ধ)

CHAPTER III: RESEARCH QUESTION

3.1. Scope Of Our Proposed Encoding

The peculiar orthographic rules in Bangla pose a challenge when creating a phonetic encoding for it. Some of the common cases illustrating these unscientific spelling rules, ones that a candidate encoding must be able to handle, are shown below:

1. There are groups of phonetically similar characters in Bangla; for example, NA (ন) and NNA (ণ); SA (স), SHA (শ) and SSA (ষ), etc. The contrast between long and short vowels in the script is also in the modern version of the spoken language.
2. Bangla has many consonant clusters or conjuncts with unusual pronunciations (i.e., ক্ষ, ক্স, etc.): let us consider ক্ষ. ক্ষ = ক+্ +ষ; ক্ষত [KA HASANT SSA TA] /kʰɔʈo/ is pronounced as খত [KHA TA] /kʰɔʈo/, where ষ does not have any sound.
3. Bangla has different uses of *Phalaa's*, the cluster final form of the semi-vowels in Bangla (BA, MA, YA, RA and LA) which are represented using a distinct sign-form. BA *phalaa* for example has a distinct pronunciation from a BA in any other position in a cluster or in a standalone configuration.
4. Different pronunciation of letters or conjuncts in different contexts: consider again ক্ষ. At the beginning of word, it is pronounced as খ /kʰi/. (ক্ষত → খত /kʰɔʈo/); in the middle or at the end of a word, it is pronounced as কখ /kkʰi/, (দক্ষ → দকখ /dɔkkʰio/).
5. Multiple pronunciations of some letters in the same context, such as হ

with ব: According to Bangla phonological rules, হ should be pronounced as ও or উ and ব should be pronounced as ভ: আহ্বান → আওভান /aovan/. However, most native speakers pronounce these words the same way as it is written. For example, আহ্বান is usually pronounced as আহভান /ahobhan/. Both pronunciations are considered correct.

Previous efforts in creating phonetic encoding for Bangla [9, 10] are based on Soundex [4]. Soundex partitions the letters into disjoint sets, assuming the letters within the same set have similar sound. It works on a letter-by-letter basis, and cannot handle context-sensitive rules, such as those illustrated earlier. A recently published encoding [9] based on Soundex is able to handle most of the trivial cases, and those involving some of the conjuncts, but it fall far short of producing suggestions for a large majority of the complex misspelled words. Metaphone encoding [5] does consider the context, so it is able to handle all but the last case above, which requires that the encoding be able to produce multiple encoded forms of the same character sequence. Double Metaphone [7] remedies that problem of Metaphone of not being able to produce multiple encodings from the same string. These limitations in part led us to create a Double Metaphone encoding for Bangla that does not suffer from the problems listed above, and in addition, is able to the full complexity of Bangla spelling rules.

Main achievement of our phonetic encoding is, unlike any other previous phonetic encoding either in English or Bangla, our proposed encoding not only gives a proper phonetic encoding [1] but also this can also work as an intermediate code in multi-lingual applications.

3.2. Limitation Of Previous Encoding

So far we have two encoding for Bangla [9, 11], both are based on soundex. Even though they did some customization for Bangla but still they can not encode the pronunciation of most of the words correctly.

1. Hoque and Kaykobad, 2002, BSoundex [10, 11]

It can handle:

- *Phonetic similarities of some letters in Bangla:*
 - Hoque and Kaykobad's encoding partitions letters into 11 disjoint groups. Table 3: Hoque and Kaykobad's phonetic encoding for Bangla. This partition gives the same code to similar sounding letters.
- *Some trivial case of conjuncts:*
 - Before producing Sound-code they generate a simplified format or open format. This part handles the conjunct problem in trivial cases.

It can not handle:

- *Unusual pronunciation of many clusters or conjuncts:*
 - Described in 3.1.2.
- *Different uses of Phalaa's.*
 - Described in 3.1.3.
- *Different pronunciation of letters or conjuncts in different context.*
 - Described in 3.1.4.
- *Multiple pronunciations of some letters in the same context.*
 - Described in 3.1.5.

More constraints:

This encoding has few extra constraints because of exactly following the English soundex algorithm rather than customizing it for Bangla. These are:

- *Keeps the first letter of the string in encoding:*
 - It is a major problem in soundex encoding. And specifically for Bangla. We miss our desired suggestion because of keeping the first letter in the

encoding. Detail of its problem is described in 2.3.2, case 1.

- *Excluding vowels when encoding string:*
 - Even though English face the same problem for excluding vowels, but it also helps them many cases. But main thing is Soundex algorithm was proposed for name searching in census. In case of name searching excluding vowels is required for Bangla as well. But this encoding was proposed for spelling checker; hence it is not acceptable to exclude vowels. Problems of excluding vowel are described in 2.3.2, case 2.
- *Consecutive repetitions of the same coded characters are eliminated.*
 - This also poses a problem, and sometimes we get extra irrelevant suggestion for it. Detail of its problem is described in 2.3.2, case 3.

2. Zaman and Khan, 2004 [9], soundex type phonetic encoding

It can handle:

- *Phonetic similarities of some letters in Bangla:*
 - Zaman and Khan's encoding partitions letters into 23 disjoint groups. Table 4: Bangla phonetic encoding table. This partition gives the same code to similar sounding letters.
- *Conjuncts with usual pronunciation:*
 - Before producing Sound-code they generate a simplified format or open format. This part handles the conjunct problem in trivial cases.

It cannot handle:

- *Unusual pronunciation of many clusters or conjuncts:*
 - Described in 3.1.2.
- *Different uses of Phalaa's.*
 - Described in 3.1.3.
- *Different pronunciation of letters or conjuncts in different context.*
 - Described in 3.1.4.
- *Multiple pronunciations of some letters in the same context.*
 - Described in 3.1.5.

We have not described the problems in detail in this section but referred to those sections, where it has been described. These may seem few lines here, but reading its description one would understand how complex it can be to handle those.

This Thesis paper answers these unanswered questions, more specifically gives an encoding that can properly encode a word representing its sound considering all these complex cases above.

3.3. Why It Is Worthwhile To Answer

Using this proposed encoding we can develop very efficient and useful applications that we can not otherwise.

1. Bangla does not have a very good spelling checker that can give words of same pronunciation in suggestions considering complex Bangla rules, this encoding helps us to develop that.

2. Bangla has many transliteration applications, but all of those give a one-to-one mapping. It will convert each English letter or letters to fixed Bangla letter or letters. There are no transliterations available where if you write in English it will give dictionary words of same pronunciation. Using phonetic encoding we can develop that.

3. Name searching is a very useful application in census,

hospitals, educational institutes, offices, etc. There is no such name searching application in Bangla that gives names with almost same pronunciation in suggestion. This encoding helps to develop this application too.

4. This encoding can work as an intermediate code in multi-lingual information retrieval, where a user issues a query in one language (such as English) to search a collection in a different language (such as Bangla). More specifically, writing the pronunciation of a word in English one can search words with same pronunciation in a Bangla document.

CHAPTER IV: PROPOSED ENCODING

Proposing our encoding we needed to keep few things in our mind. We need to think about phonetic similarity of letters to give them the same code and also the keep in mind the orthographic or spelling rules, to know how letters spell in different context, so that we can encode the letters with similar sounding letters considering the context.

Another purpose of this encoding is to work as an intermediate code in multi-lingual applications. We will be encoding our Bangla letters to a set of Latin alphabets, so that it can easily work as an intermediate language to work with English.

We assume that the Bangla text is encoded using Unicode Normalization Form C (NFC) [13].

4.1. Proposed phonetic encoding for words

We will have two encoding, mainly one for words and a few variations from it for names as well. This section describes about the words encoding.

Throughout the paper we termed our proposed phonetic encoding by Double metaphone phonetic encoding or proposed phonetic encoding. To encode Bangla words we need to consider context and also need to generate multiple codes for the same string. These constraints can be handled in Double metaphone algorithm, which we did for Bangla here. Hence, we termed it as Double metaphone phonetic encoding.

4.1.1. Phonetic encoding table

Following Table 6: Double Metaphone Phonetic Encoding table for words is the table of proposed Double Metaphone phonetic encoding for words. Followed by the table there will be reasoning of each of the encoding.

Table 6: Double Metaphone Phonetic Encoding table for words

Letter	Name	Unicode	Code	Context	Example
্	VIRAMA (Hasant)	\u09CD	<i>Not Coded</i>		
়ো	SIGN O	\u09CB	<i>Not Coded</i>		
ঁ	CANDRABINDU	\u0981	<i>Not Coded</i>		
অ	A	\u0985	“o”		
ও	O	\u0993	“o”		
আ	AA	\u0986	“a”		
া	SIGN AA	\u09BE	“a”		
ই	I	\u0987	“i”		
ঈ	II	\u0988	“i”		
ি	SIGN I	\u09BF	“i”		
ী	SIGN II	\u09C0	“i”		
উ	U	\u0989	“u”		
ূ	UU	\u098A	“u”		
ু	SIGN U	\u09C1	“u”		
ূ	SIGN UU	\u09C2	“u”		
এ	E	\u098F	“e”		
ে	SIGN E	\u09C7	“e”		
ঐ	AI	\u0990	“oi”		
ৈ	SIGN AI	\u09C8	“oi”		
ও	AU	\u0994	“ou”		
ৌ	SIGN AU	\u09CC	“ou”		
ক	KA	\u0995	“k”		
খ	KHA	\u0996	“k”		
ক্ষ		\u0995 \u09CD \u09B7	“k”	@ the beginning	ক্ষত /kʰɔʈo/

Letter	Name	Unicode	Code	Context	Example
ক্ষ		\u 0995 \u09CD \u09B7	“kk”	@ middle/end	দক্ষ /dɔkkɔ/
গ	GA	\u0997	“g”		
ঘ	GHA	\u0998	“g”		
ঙ	NGA	\u0999	“ng”		বাঙলা /banla/
ং	ANUSVARA	\u0982	“ng”		বাংলা /banla/
চ	CA	\u099A	“c”		
ছ	CHA	\u099B	“c”		
য	YA as <i>phalaa</i>	x\u09CD\u09AF x\u09CD\u09AF \u0986	“e”	@ the beginning as YA <i>phalaa</i>	ব্যক্ত /bæktɔ/
		...xy \u09CD z \u09CD \u09AF	Not Coded	@ middle/end with conjuncts	সন্ধ্যা /ʃondʝa/
		...xy \u09CD \u09AF	Doubles: yy	@ middle/end	অদ্য /oɖɖɔ/
য	YA	\u09AF	“j”		
জ	JA	\u099C	“j”		
ঝ	JHA	\u099D	“j”		
ঞ	NYA	\u099E \u099A	“n”	Before CA	অঞ্চল /ɔncɔl/
		\u099E \u099B	“n”	Before CHA	বাঞ্ছা /banɕa/
		\u099E \u099C	“n”	Before JA	অঞ্জলি /ɔnjɔli/
		\u099E \u099D	“n”	Before JHA	যঞ্ছা /jɕɕɔɕa/
		\u099A \u099E	“n”	After CA	যাচঞা /jɔcɕna/
		\u099E \u0985 \u099E\u0987	Not Coded	Before A I	মিঞা /miã/
		\u099C \u09CD \u099E	“ge”	@ the beginning after JA	জ্ঞাত /g̃æɕɔ/
		... \u099C \u09CD \u099E	“gg”	@ middle/end after JA	বিজ্ঞান /big̃g̃æɕn/
\u099E \u09CD	“n”	With hasant	নঞ /nɔɕn/		
ট	TTA	\u099F	“T”		
ঠ	TTHA	\u09A0	“T”		
ড	DDA	\u09A1	“D”		
ঢ	DDHA	\u09A2	“D”		
ঋ	VOCALIC R	\u098B	“ri”	@ the beginning	ঋতু /ritu/

Letter	Name	Unicode	Code	Context	Example
		x\u098B	“ri” xri	@ middle/end	বিকৃত /bikkriṭò/ বিকৃত /bikriṭò/
র	RA as <i>phalaa</i>	x\u09CD \u09B0	“r̥”	@ the beginning	প্রকাশ /prokaʃ/
		...x\u09CD \u09B0	“r̥”	@ middle/end	রাত্রি /rat̥tri/ রাত্রি /rat̥ri/
র	RA	\u09B0	“r̥”		
ড়	RRA	\u09DC	“r̥”		
ঢ়	DDHA	\u09A2	“r̥”		
ন	NA	\u09A8	“n̥”		
ণ	NNA	\u09A3	“n̥”		
ত	TA	\u09A4	“t̥”		
থ	THA	\u09A5	“t̥”		
দ	DA	\u09A6	“d̥”		
ধ	DHA	\u09A7	“d̥”		
প	PA	\u09AA	“p̥”		
ফ	PHA	\u09AB	“p̥”		
ব	BA as <i>phalaa</i>	x\u09CD \u09AC y...	<i>Not Coded</i>	@ the beginning	স্বদেশ /ʃɔdeʃ/
		...x\u09CD y \u09CD \u09AC	<i>Not Coded</i>	BA <i>phalaa</i> with conjuncts	তত্ত্ব /tɔtt̥t̥t̥o/
		... \u09AC \u09CD \u09AC	“bb”	After BA as conjuncts	তিব্বত /t̥ib̥bɔt̥/
		... \u09AE \u09CD \u09AC	“mb”	After MA as conjuncts	লম্ব /lɔm̥bɔ/
		... \u0997 \u09CD \u09AC	“gb”	After GA as conjuncts	দিগ্বিদিক /ɖig̥b̥iɖik/
		\u0989 \u09A6 \u09CD \u09AC	“udb”	After Ud- (U DA BA...)	উদ্বেগ /ud̥beg/
		...y \u09CD \u09AC	Doubles: yy	@ middle/end	বিশ্ব /biʃʃɔ/
ব	BA	\u09AC	“b̥”		
ভ	BHA	\u09AD	“b̥”		
ম	MA as <i>phalaa</i>	x\u09CD \u09AE...	<i>Not Coded</i>	@ the beginning	স্মরণ /ʃɔron̥/
		...x\u09CD y \u09CD \u09AE	<i>Not Coded</i>	MA <i>phalaa</i> with conjuncts	সূক্ষ্ম /ʃuk̥kh̥o/

Letter	Name	Unicode	Code	Context	Example
		... \u0995 \u09CD \u09AE	“km”	After KA as conjuncts	ৰুক্মিনী /rukmini/
		... \u0997 \u09CD \u09AE	“gm”	After GA as conjuncts	যুগ্ম /jugmo/
		... \u0999 \u09CD \u09AE	“ngm”	After NGA as conjuncts	বাঙময় /baṅmoi/
		... \u099F \u09CD \u09AE	“tm”	After TTA as conjuncts	কুট্মল /kutmol/
		... \u09A3 \u09CD \u09AE	“nm”	After NNA as conjuncts	মৃণায় /mrinmōē/
		... \u09A8 \u09CD \u09AE	“nm”	After NA as conjuncts	জন্ম /jɔnmo/
		... \u09AE \u09CD \u09AE	“mm”	After MA as conjuncts	সম্মান /ʃɔmman/
		... \u09B2 \u09CD \u09AE	“lm”	After LA as conjuncts	গুন্ম /gulmo/
		... \u09B6 \u09CD \u09AE	“sm”	@ middle/end with SHA	কাশ্মীর /kaʃmir/
		... \u09B7 \u09CD \u09AE	“sm”	@ middle/end with SSA	কুন্মন্ড /kuʃmando/
		... \u09B8 \u09CD \u09AE	“sm”	@ middle/end with SA	সুন্মিতা /ʃuʃmita/
		...y \u09CD \u09AE	Doubles: yy	@ middle/end otherwise	পদ্ম /pɔddō/
ম	MA	\u09AE	“m”		
য়	YYA	\u09DF	“y”		
ল	LA	\u09B2	“l”		
শ	SHA	\u09B6	“s”		
স	SA	\u09B8	“s”		
ষ	SSA	\u09B7	“s”		
হ	HA	\u09B9 \u09CD \u098B	“ri”	HA with Vocalic R	হৃদয় /rɦidoē/
		\u09B9 \u09CD \u09B0	“r”	HA with R as <i>phalaa</i>	হৃদ /rɔd/
		\u09B9 \u09CD \u09A8	“nn”	HA with NA	পূৰ্বাহ্ন /purbaṅno/

Letter	Name	Unicode	Code	Context	Example
		\u09B9 \u09CD \u09A3	“nn”	HA with NNA	প্রাঙ্গ /prannfio/
		\u09B9 \u09CD \u09AE	“mm”	HA with MA	ব্রক্ষা /brommfia/
		\u09B9 \u09CD \u09AF	“jj”	HA with YA as <i>phalaa @ middle/end</i>	উহা /ujffio/
		\u09B9 \u09CD \u09B2...	“l”	HA with LA @ beginning	হ্লাদ /lfad/
		... \u09B9 \u09CD \u09B2	“ll”	HA with LA @ middle/end	আহ্লাদ /allfad/
		\u09B9 \u09CD \u09AC	“h” “o”	HA with BA	আহ্বান /aovan/ আহ্বান /afiobfian/
হ	HA	\u09B9	“h”	Otherwise	
ঃ	VISARGA	x\u0983 y...	Doubles: yy	@ the middle	দুঃসময় /duffsomoẽ/
		x\u0983	“h”	@ the end, strlen = 1 2	উঃ /ufi/, বাঃ /bafi/
		x\u0983	<i>Not Coded</i>	Otherwise @ the end	পুনঃ /puno/

4.1.2. Encoding reasoning

Name: Virama / Hasant; Unicode: \u09CD; Letter: ্

The absence of vowels between consonants can be represented by Virama / Hasant. This is used in the Jukhtakhor/Conjuncts.

In our encoding, we will give it 0 (zero) code. Because hasant means it is used to connect two or more consonants and we don't need to keep the information of connectors (hasant) in our encoding. And more importantly this is used to lower the sound of 1st consonant in conjuncts. And individually has No Sound in words.

This will also reduce one extra character error. For example, if someone misses the ্, then it's basically all the same. Mean he

was trying to write some Conjuncts but missed the connector ্ so, if we consider it as 0 (zero) code we can reduce this error.

Example: দক্ষ = দ গ ্ ধ

We can see that we can easily reduce the ্ from our encoding.

Final code: Not Coded

Name: Sign O; Unicode: \u09CB; Letter: ো

ো (Sign O) is given 0 (zero) code, because in Bangla words, O in the middle or end of word is an inherent vowel. For example, ভাল and ভালো. Both sound same and even if we don't have ো in ভাল, it will pronounce as ভালো. Because there is an inherent vowel ো in ভাল. Rather than adding inherent vowels in encoding we give ো 0 (zero) code. So, now ভাল and ভালো will have the same code.

Final Code: Not Coded

Name: Candrabindu; Unicode: \u0981; Letter: ঁ

We give ঁ 0 (zero) code. ঁ is used for nasal words. Our main target is to encode the similar sounded characters in to the same code. Similar sounded characters means which sounds similar when we read it in our normal conversations not according to actual grammar. In normal conversations, we don't emphasize on nasal sounds and simply pronounce it without ঁ most of the cases. So, we can simply omit ঁ from our encoding.

Final Code: Not Coded

Name: A; Unicode: \u0985; Letter: অ; IPA: /ɔ/, /o/

Name: O; Unicode: \u0993; Letter: ও; IPA: /o/

If there is a ই or উ after অ then it sounds as /o/. Otherwise, in

most of the cases it sounds as /ɔ/. /o/ and /o/ are very close in pronunciation. So, we are encoding all the cases of অ and ও to “o”.

Final Code: “o”

Name: AA; Unicode: \u0986; Letter: আ ; IPA: /a/

Name: Sign AA; Unicode: \u09BE; Letter: া ; IPA: /a/

Final Code: “a”

Name: I; Unicode: \u0987; Letter: ই; IPA: /i/

Name: Sign I; Unicode: \u09BF; Letter: ি; IPA: /i/

Name: II; Unicode: \u0988; Letter: ঈ; IPA: /i/

Name: Sign II; Unicode: \u09C0; Letter: ী; IPA: /i/

Final Code: “i”

Name: U; Unicode: \u0989; Letter: উ; IPA: /u/

Name: Sign U; Unicode: \u09C1; Letter: ু; IPA: /u/

Name: UU; Unicode: \u098A; Letter: উ; IPA: /u/

Name: Sign UU; Unicode: \u09C2; Letter: ূ; IPA: /u/

Final Code: “u”

Name: E; Unicode: \u098F; Letter: এ; IPA: /æ/, /e/

Name: Sign E; Unicode: \u09C7; Letter: ে; IPA: /æ/, /e/

এ has two sounds. One is /e/, such as in, কে, সে, তেজ

Another one is /æ/, such as in, এক, কেন, দেখা

In our encoding, our main target is to give closely similar sounding word the same code. In this case, /e/ and /æ/ are very close in pronunciation. So, we are encoding all the cases of এ to “e”.

Final Code: “e”

Name: AI; Unicode: \u0990; Letter: ঐ; IPA: /oi/

Name: SIGN AI; Unicode: \u09C8; Letter: ঐ; IPA: /oi/

Final Code: "oi"

Name: AU; Unicode: \u0994; Letter: ঔ; IPA: /ou/

Name: SIGN AU; Unicode: \u09CC; Letter: ঔ; IPA: /ou/

Final Code: "ou"

Name: KA; Unicode: \u0995; Letter: ক; IPA: /k/

Name: KHA; Unicode: \u0996; Letter: খ; IPA: /kʰ/

Both ক and খ are Velar. But ক is Un-aspirated and খ is Aspirated. So, in this case, we are giving the same code to both letters.

Final Code: "k"

Name: Conjunct KHIYO; Unicode: \u0995 \u09CD \u09B7;

Letter: ক্খ = ক্খ; IPA: /k/, /kʰ/

Case 1: At the beginning of a word, it is pronounced as খ /kʰ/. So it is given the same code as খ, which is "k".

ক্ষত /kʰɔt/ → খত → kt

Case 2: In the middle or at the end of words, it is similar to কখ /kkʰ/, so it is encoded as "kk".

দক্ষ /dɔkkʰ/ → দকখ → dkk

Exception: তৎক্ষনাত্ /tɔtʰkʰɔnat/ According to its pronunciation it should be encoded as "ttknat", but it is instead encoded as "ttkknat".

Name: GA; Unicode: \u0997; Letter: গ; IPA: /g/

Name: GHA; Unicode: \u0998; Letter: ঘ; IPA: /gʱ/

Both গ and ঘ are Velar. But গ is Un-aspirated and ঘ is Aspirated. So, in this case, we are giving the same code to both letters.

Final Code: “g”

Name: NGA; Unicode: \u0999; Letter: ঙ; IPA: /ŋ/

Name: ANUSVARA; Unicode: \u0982; Letter: ং; IPA: /ŋ/

ঙ and ং sounds like /ŋ/, so it is encoded as “ng”.

বাঙলা /baŋla/ → bangla

বাংলা /baŋla/ → bangla

Name: CA; Unicode: \u099A; Letter: চ; IPA: /c/

Name: CHA; Unicode: \u099B; Letter: ছ; IPA: /cʰ/

Both চ and ছ are Palatal. But চ is Affricate Un-aspirated and ছ is Affricate Aspirated. So, in this case, we are giving the same code to both letters.

Final Code: “c”

Name: YA; Unicode: \u09AF; Letter: য; IPA: /j/

য as phalaa

Case 1: At the beginning of a word, and if the word is অ-কারান্ত /ɔ/ or আ-কারান্ত /a/, it is pronounced as /æ/.

Example: ব্যক্ত, ধ্যান

At the beginning of a word, and if there is a ই or উ after য phalaa, then it is pronounced as এ /e/

Example: ব্যথিত, ব্যক্তি

We encode both /æ/ and /e/ as “e”.

বাক্ত /bækto/ → bekt

ধ্যান /d̪hæn/ → den

ব্যক্তি /bekti/ → bekti

Final Code: “e”

Case 2: In the middle or at the end of a word with conjuncts, it is usually silent, and so it is Not coded.

সন্ধ্যা /ʃond̪ɦa/ → সন্ধ্যা → snda

স্বাস্থ্য /ʃast̪ho/ → স্বাস্থ্য → sast

Final Code: Not Coded

Case 3: In the middle or at the end of a word it doubles the attached letter, and so the code is doubled as well.

অদ্য /od̪do/ → অদ্য → odd

মধ্য /mod̪d̪ɦo/ → মধ্য → mdd

Final Code: Same as attached letter

Case 4: Otherwise when it is used normally rather than as a phalaa, it is encoded as “j”

Final Code: “j”

Name: JA; Unicode: \u099C; Letter: জ; IPA: /j/

Name: JHA; Unicode: \u099D; Letter: ঞ; IPA: /jɦ/

Both জ and ঞ are Palatal. But জ is Affricate Un-aspirated and ঞ is Affricate Aspirated. So, in this case, we are giving the same code to both letters.

Final Code: “j”

Name: NYA; Unicode: \u099E; Letter: ঞ; IPA: /nã/

Case 1: Usually in conjuncts if a ঞ is added before a চ, ছ, জ,

ব, or after a চ then it is pronounced as ন /n/; in this case it is encoded as “n”.

Before চ: অঞ্চল /ɔ̃ncɔl/ → অনচল → oncl

Before ছ: বাঞ্চা /ban̥ɕa/ → বানছা → banca

Before জ: অঞ্জলি /ɔ̃ɳʝoli/ → অনজলি → onjli

Before ঝ: যঞ্চা /ʝhɔ̃ɳʝha/ → যনঝা → jnja

After চ: যাচ্ঞা /ʝacna/ → যাচনা → jacna

Final Code: “n”

Case 2: If আ-কার and ই-কার is added after a ঞ, then it creates a nasal sound. মিঞা /miã/ → মিআঁ. However, since in our encoding nasal sounds are *Not Coded*, it is also *Not Coded*.

মিঞা /miã/ → মিআ → mia

নাইঞা /naĩ/ → নাই → nai

Final Code: Not Coded

Case 3: In conjuncts after জ, ঞ sounds as “g” at the beginning of the word and as “gg” in the middle or at the end of the word. At the beginning, it is encoded as “g” and in the middle or at the end; it is encoded as “gg”. Again, if at the beginning, আ (া)-কার is added with ঙ then আ (া)-কার is pronounced as “æ”, which is encoded as “e”.

At the beginning:

ঙাত /g̃æɔ̃/ → get

ঙান /g̃æ̃n/ → gen

Final Code: “g”

At the middle/end:

বিজ্ঞান /big̃gæ̃n/ → biggan

বিজ্ঞ /bigg̃o/ → bigg

Final Code: “gg”

Exception: সংজ্ঞা /ʃɔŋga/ should be encoded as “sngga” but it is instead encoded as “snggga”.

Case 4: Otherwise, if there is a VIRAMA/Hasant after it, then it is simply encoded as “n”.

নঞ /nɔn/ → n̄

নঞর্থক /nɔnɔrtthòk/ → n̄rttk

Final Code: “n”

Name: TTA; Unicode: \u099F; Letter: ট; IPA: /t̪/

Name: TTHA; Unicode: \u09A0; Letter: ঠ; IPA: /t̪ʰ/

Both ট and ঠ are Retroflex. But ট is Un-aspirated and ঠ is Aspirated. So, in this case, we are giving the same code to both letters.

Final Code: “T”

Name: DDA; Unicode: \u09A1; Letter: ড; IPA: /d̪/

Name: DDHA; Unicode: \u09A2; Letter: ঢ; IPA: /d̪ʰ/

Both ড and ঢ are Velar. But ড is Un-aspirated and ঢ is Aspirated. So, in this case, we are giving the same code to both letters.

Final Code: “D”

Name: VOCALIC R; Unicode: \u098B; Letter: ঝ; IPA: /ri/

Case 1: At the beginning Vocalic R ঝ and Sign Vocalic R ৠ are encoded as “ri”.

ঋতু /riṭu/ → rītu

Final Code: “ri”

Case 2: According to phonological rules in [19], it doubles the sound of the attached letter if it is in the middle or at the end. However, since people usually pronounce it as “ri” in such cases as well, it is encoded as both codes.

বিকৃত /bikkriṭo/ → bikkrit

বিকৃত /bikriṭo/ → bikrit

Final Code: “ri”, “xri” // x is the code of attached letter

Name: RA; Unicode: \u09B0; Letter: ঋ; IPA: /r/

ঋ as *phalaa*

Case 1: At the beginning of the word ঋ-*phalaa* sounds like ঋ, so it is encoded as “r”.

প্রকাশ /prokaʃ/ → prkas

প্রনাম /pronam/ → prnam

Final Code: “r”

Case 2: According to [19, 20, 21], in the middle or at end, it doubles the attached letter. But if we consider the pronunciation of these words, it is also pronounced as only ঋ. As a solution, we again encode it both the codes using the Double Metaphone approach.

রাত্রি /rat̪tri/ → রাতrি → ratrri

রাত্রি /rat̪ri/ → রাতrি → ratri

ছাত্র /chattr̪o/ → ছাতrত্র → catrtr

ছাত্র /chattr̪o/ → ছাতrত্র → catrtr

Final Code: “r”, “xr” // x is the code of attached letter

Case 3: Otherwise র is encoded as “r”.

Final Code: “r”

Name: RRA; Unicode: \u09DC; Letter: ড়; IPA: /ɽ/

Name: DDHA; Unicode: \u09A2; Letter: ঢ়; IPA: /ɽʱ/

Both ড় and ঢ় are Alveolar and Flapped. So, in this case, we are giving the same code to both letters.

Final Code: “r”

Name: NA; Unicode: \u09A8; Letter: ন; IPA: /n/

Name: NNA; Unicode: \u09A3; Letter: ণ; IPA: /ɳ/

Final Code: “n”

Name: TA; Unicode: \u09A4; Letter: ত; IPA: /t/

Name: DDHA; Unicode: \u09A5; Letter: থ; IPA: /tʰ/

Both ত and থ are Dental. But ত is Un-aspirated and থ is Aspirated. So, in this case, we are giving the same code to both letters.

One debatable topic on ত is Khondo-TA ত্, which is short form of ত. Currently it is not in the Unicode chart and it is written as ত and ত্. In our encoding ত্ is Not coded, so it will automatically get the same code “t” as ত. But if later it is included in the Unicode chart then we can simply give that letter the code “t”.

Final Code: “t”

Name: DA; Unicode: \u09A6; Letter: দ; IPA: /d/

Name: DHA; Unicode: \u09A7; Letter: ধ; IPA: /dʱ/

Both দ and ধ are Dental. But দ is Un-aspirated and ধ is

Aspirated. So, in these cases, we are giving the same code to both letters.

Final Code: “d”

Name: PA; Unicode: \u09AA; Letter: প; IPA: /p/

Name: PHA; Unicode: \u09AB; Letter: ফ; IPA: /ph/

Both প and ফ are Bilabial. But প is Un-aspirated and ফ is Aspirated. So, in these cases, we are giving the same code to both letters.

Final Code: “p”

Name: BA; Unicode: \u09AC; Letter: ব; IPA: /b/

Name: BHA; Unicode: \u09AD; Letter: ভ; IPA: /bh/

Both ব and ভ are Bilabial. But ব is Un-aspirated and ভ is Aspirated. So, in these cases, we are giving the same code to both letters.

Final Code: “b”

ব as phalaa

Case 1: At the beginning ব-phalaa doesn't have any sound.

So it is *Not Coded*.

স্বাধিকার /ʃadʱikar/ → সাধিকার → sadikar

স্বদেশ /ʃodeʃ/ → সদেশ → sdes

জালা /ʃala/ → জালা → jala

Final Code: Not Coded

Case 2: At the middle/end ব-phalaa with ব, ম and গ that is derived from ক keeps its sound. So it is encoded as “b”.

ব: তিব্বত /t̪ib̪b̪ot̪/ → tib̪b̪t̪

সাব্বাশ /ʃab̪baʃ/ → sab̪bas

ম: লম্ব /lombo/ → lmb

সম্বন্ধনা /ʃombordfiona/ → smbrdna

গ: দিগ্বিদিক /digbidik/ → digbidik

Final Code: “b”

Case 3: At the beginning ব–*phalaa* with দ that is derived from উদ্ keeps its sound. So it is encoded as “b”.

দ that is derived from উদ্:

উদ্ব্বেগ /udbeg/ → ud**b**eg

উদ্বোধন /udbodhon/ → ud**b**dn

Final Code: “b”

Case 4: At the middle of the word ব–*phalaa* with conjuncts doesn't have any sound. So it is *Not coded*.

তত্ত্ব /tɔttto/ → ততত → ttt

উজ্জ্বল /ujjal/ → উজজল → ujjl

উচ্ছ্বাস /ucchaʃ/ → উচছাস → uccas

Final Code: Not Coded

Case 5: At the middle/end of the word sound of ব with conjuncts doubles. So it has the same code as the previous code.

দ্বিত্ব /ditto/ → দিতত → ditt

বিশ্ব /biffɔ/ → বিশশ → biss

Final Code: x // x is the code of attached letter

Name: MA; Unicode: \u09AE; Letter: ম; IPA: /m/

ম as phalaa

Case 1: At the beginning of the word ম–*phalaa* doesn't have any sound. So it is *Not Coded*.

স্মরণ /ʃaron/ → সরন → srn

স্মশান /ʃɔʃan/ → সশান → ssan

Final Code: “m”

Case 2: At the middle of the word ম–phalaa with conjuncts doesn't have any sound. So it is Not coded.

সূক্ষ্ম /ʃukkhõ/ → সূকখ → sukk

লক্ষণ /lɔkkhon/ → লকখন → lkkn

Final Code: Not Coded

Case 3: At the middle/end ম phalaa with ক, গ, ঙ, ট, ণ, ন, ম, ল, স, ষ, শ keep its sound. So it is simply coded to “m”.

ক: রুক্মিনী /rukmini/ → rukmini

গ: বাগ্মী /bagmi/ → bagmi

যুগ্ম /ʃugmo/ → jugm

ঙ: বাঙ্গময় /baŋmoẽ/ → bangmy

বাজ্মুখ /baŋmukh/ → bangmuk

ট: কুট্মল /kutmol/ → kuTml

কুট্মলিত /kutmoliṭo/ → kuTmlit

ণ: হিরণ্ময় /hirɔnmɔẽ/ → hirnmy

মৃণ্ময় /mrinmɔẽ/ → mrinmy

ন: উন্মাদ /unmad/ → unmad

জন্ম /ʃɔnmɔ/ → jnm

ম: সম্মান /ʃɔmman/ → smman

সম্মতি /ʃɔmmoti/ → smmti

গ: গুল্ম /gulmo/ → gulm

বল্মীক /bolmik/ → blmik

স: সুস্মিতা /ʃuʃmita/ → susmita

ষ: কুম্ভাণ্ড /kuʃmando/ → kusmand

শ: কাশ্মীর /kaʃmir/ → kasmir

Final Code: “m”

Case 4: Otherwise at the middle/end য with conjuncts doubles the sound of attached letter. So it encoded with the same code of the previous character.

ছদ্বা /chɔddɔ̃/ → ছদদ → cdd

পদ্বা /pɔddɔ̃/ → পদদ → pdd

Final Code: x // x is the code of attached letter

Name: YYA; Unicode: \u09DF; Letter: য়; IPA: /ẽ/

Final Code: “y”

Name: LA; Unicode: \u09B2; Letter: ল; IPA: /l/

Final Code: “l”

Name: SA; Unicode: \u09B8; Letter: স; IPA: /s/, /ʃ/

Name: SHA; Unicode: \u09B6; Letter: শ; IPA: /s/, /ʃ/

Name: SSA; Unicode: \u09B7; Letter: ষ; IPA: /ʃ/

Final Code: “s”

Name: HA; Unicode: \u09B9; Letter: হ; IPA: /h/

হ with ষ: হ doesn't have any sound in conjuncts with ষ. So, it is *Not Coded*.

হৃদয় /rɦiɔẽ/ → রিদয় → ridy

হুতপিন্ড /rɦidpindo/ → রিতপিনড → ritpinD

Final Code: Not Coded

হ with র: হ doesn't have any sound in conjuncts with র. So, it is *Not Coded*.

হুদ /rɦod/ → রদ → rd

হাস /rɦas/ → রাস → ras

Final Code: Not Coded

হ with ণ/ন: হ sounds as নহ /nɦ/ in conjuncts with ণ/ন where ɦ sounds lightly. So, it is encoded as “n”.

পূৰ্বাহ্ন /purbannɦio/ → পূর্বানন → purbann

চিহ্ন /chinnɦio/ → চিনন → cinn

প্রাহ্ন /prannɦio/ → প্রানন → prann

Final Code: “n”

হ with ম: হ sounds as মহ /mɦ/ in conjuncts with ম where ɦ sounds lightly. So, it is encoded as “m”.

ব্রহ্মা /bromma/ → বরামমা → brmma

ব্রাহ্ম /brammo/ → ব্রামম → bramm

Final Code: “m”

হ with য: হ sounds as য in conjuncts with য. So, it is encoded as “j”.

উহ্য /uɦɦio/ → উযয → ujj

ঐতিহ্য /oitihɦio/ → ঐতিযয → oitijj

Final Code: “j”

হ with ল: হ doesn't have any sound in conjuncts with ল at the beginning. So, it is *Not Coded*.

হ্লাদ /lɦad/ → লাদ → lad

হ sounds as ল in conjuncts with লহ /lɦ/ in middle/end where ɦ sounds lightly. So, it is encoded as “l”.

আহ্লাদ /allɦad/ → আল্লাদ → ałlad

Final Code: “l”

হ with ব: According to grammatical rules হ should be sounded like ও or উ and ব should be sounded as ভ.

আহ্বান /aovan/ → আওভান

However, most native speakers pronounce these words the same way as it is written. For example, আহ্বান is usually pronounced as আহভান /aɦobɦan/, so we encode it to two different codes for the two different pronunciations.

আহ্বান → আওভান /aovan/ → aoban

আহ্বান → আহভান /aɦobɦan/ → aɦban

Final Code: “o”, “h”

Name: VISARGA; Unicode: \u0983; Letter: ঃ

Case 1: In the middle of the word, ঃ gets the sound of the character next to it.

দুঃসময় /dʊʃɦomoi/ → দুসসময় → duɦsmɪ

দুঃখ /dʊkɦɦio/ → দুকখ → dukɦ

Final Code: x // x is the code of next letter

Case 2: If ঃ is at the end, and the string length is 2 or 3, then ঃ sounds as হ. So it is encoded as “h”.

উঃ /uɦ/ → উহ → uh

বাঃ /baɦ/ → বাহ → baɦ

Final Code: “h”

Case 3: If ঃ is at the end, and the string length greater than

3, then ং sounds as ঙ. However, since ঙ is *Not Coded* in our encoding, ং is *Not Coded* as well.

পুনঃ /puno/ → পুনো → পুন → pun

অধঃ /adhio/ → অধো → অধ → od

Final Code: Not Coded

CHAPTER V: APPLICATIONS OF PHONETIC ENCODING

Proper phonetic encoding is a very good contribution for a language, but it has no significance until it is used properly in applications. Name searching was first such application, where phonetic encoding was used after that spelling checker adopts this phonetic encoding technique.

We have used our phonetic encoding in many applications like spelling checker, transliteration, cross-lingual information retrieval and name searching for Bangla. In each case, we will first show how that application were developed earlier, how they perform and then how phonetic encoding improves its performance.

5.1. Spelling Checker

The problem of detecting error in words and automatically correcting them is a great research challenge. Its solution has enormous potentials in text and code editing, computer aided authoring, optical character recognition (OCR), machine translation (MT), natural language processing (NLP), database retrieval interface, speech recognition, text to speech and speech to text conversion, communication system for the disabled (e.g. blind and deaf), computer aided tutoring and language learning, desktop publication and pen based computer interface. [15]

5.1.1. Spelling error patterns

The word-error can belong to one of the two distinct categories,

namely, *non-word error* and *real-word error*. Let a string of characters separated by spaces or punctuation marks be called a candidate string. A candidate string is a valid word if it has a meaning. Else, it is a non-word. By real word error we mean a valid but not the intended word in the sentence, thus making the sentence syntactically or semantically ill-formed or incorrect. In both cases the problem is to detect the erroneous word and either suggest correct alternatives or automatically replace it by the appropriate word. [15]

In Bangla so far, we do not have any very good technique that deals with non-word errors. Real-word error can be the next step after solving this problem. We will focus only on non-word errors in this chapter, which is the major part in any spelling errors.

In non-word errors, there are mainly two types of errors. One is *typographical error* and another is *phonetic error*. Description of typographical error is as follows.

In an early study, [17] found that 80% of all misspelled words (non-words errors) in a sample of human keypunched text were caused by single error misspellings: a single one of the following errors:

- Substitution error: mistyping *the* as *ther*
- Deletion error: mistyping *the* as *th*
- Insertion error: mistyping *the* as *thw*
- Transposition error: mistyping *the* as *hte* [16]

These are the type of typographical errors, which occurred due to typing mistakes, negligence, and lack of concentrations. But if computer gives a red underline into it, then we can easily correct it without seeing the spelling suggestions.

But scenarios of phonetic errors are not the same. Phonetic errors

occur when the user do not know the spelling of a word but knows the pronunciation of the word. So, using the pronunciation the user may write a word but in suggestion it is impossible to get the desired word in case of Bangla, because of complex Bangla rules described in 3.1.

5.1.2. Previous spelling checking techniques

5.1.2.1. Approximate string matching algorithm

This method uses an approximate string-matching algorithm to check the closeness of dictionary words with the misspelled word. In suggestion it gives the words that are close to it.

Levenshtein Edit Distance[23, 24, 25]

Definition:

The edit distance of two strings, s_1 and s_2 , is defined as the minimum number of point mutations required to change s_1 into s_2 , where a point mutation is one of:

1. Replace a letter,
2. Insert a letter,
3. Delete a letter,
4. Transpose consecutive letters

Example:

$e(\text{"Virginia"}, \text{"Vermont"}) = 5$

Virginia

Verginia

Verminia

Vermonia

Vermont^a

Vermont

Detail on edit distance can be found at [23, 24, 25].

● **How edit distance can be used in spelling checker**

To generate suggestion for a misspelled word we need to generate edit distance with each of the word in the lexicon and the misspelled word. If the edit distance is below a threshold then we can add the word in the suggestion list.

For example, we assume our lexicon consist of following words.

কথা, কাক, কলা, মালা

Our misspelled word is কল. Now when we check the dictionary file we find that there are no such word কল. So, it is a misspelled word according to this dictionary. Now to generate and rank the suggestion, we will generate the edit-distance with all the words of the dictionary.

Table 7: Edit distance example

Dictionary word	Edit Distance with word কল
কথা	2
কাক	2
কলা	1
মালা	3

Hence, our ranked suggestion for কল will be কলা, কাক, কথা, মালা

Longest common substring (LCS) [26]

Longest common substring is the longest substring that is common in two strings.

For example, between “*Naushad UzZaman*” and “*NZ*”, longest common substring is “*NZ*”.

LCS-Len is the length of longest common substring. In this case, *LCS-Len* is 2.

- *How LCS can be used in spelling checker*

To generate suggestion for a misspelled word we need to generate *LCS-Len* with each of the word in the lexicon and the misspelled word. If the *LCS-Len* is above a threshold then we can add the word in the suggestion list.

For example, we assume our lexicon consist of following words.

কথা, কাক, কলা, মালা

Our misspelled word is কল. Now when we check the dictionary file we find that there are no such word কল. So, it is a misspelled word according to this dictionary. Now to generate and rank the suggestion, we will generate the *LCS-Len* with all the words of the dictionary.

Table 8: LCS example

Dictionary word	LCS-Len with word কল
কথা	1
কাক	1
কলা	2
মালা	1

Hence, our ranked suggestion for কল will be কলা, কাক, কথা, মালা

5.1.2.2. BB Choudhury's Reverse dictionary method

[15] describes a unique method for Bangla spelling checker, which does phonetic grouping to handle trivial phonetic errors and uses a reverse dictionary to handle the typographical error.

Phonetically similar character error correction

Bangla letters be partitioned according to phonetic similarity (e.g., I:II, U:UU, NA:NNA, SA:SSA:SHA, etc), with each set represented by a single code. This coding can then be applied to Bangla dictionary to convert it to a non-homophonous one, with each entry pointing to the set of words that correspond to this code.

For example, suppose we encountered string *bAnI* and wish to check if it is a valid word. By phonetic similarity coded notation it can be converted into *bAni*. In D_c , is the dictionary consisting of word list and corresponding code to each of the word, there is a match of *bAni*. Now its corresponding valid words are *bANI* and *bAni*, none of which match with *bAnI*. So our candidate *bAnI* is a wrong word. But the suggested corrected word is either *bANI* or *bAni*.

Reversed word dictionary

For a valid word, its reversed word is a string of characters in reversed sequence. Thus, the reversed version of the words 'read' and 'copy' are the strings 'daer' and 'ypoc', respectively where the first character of the word goes to the last position, the second character occupies the last but one position and so on. In general, the reversed word of a word $W = x_1x_2 \dots x_k$ is $W_r = x_kx_{k-1} \dots x_2x_1$.

In a reversed word dictionary D_r , the reversed version of all dictionary words is maintained. For quick access or retrieval, the words can be alphabetically ordered, partitioned in terms of word length and maintained in indexed flat file or in trie structure. The dictionary structure for our purpose can be indexed or trie depending on the system capability. We have used trie structure for our purpose, because it is computationally faster to access.

The purpose of reversed word dictionary is to look for match of a string S backwards from the last character. We shall show that search in conventional dictionary D_c as well as reversed word dictionary together helps in finding the error position in S as well as in creating a small subset of correction candidate words which indeed contains the intended word.

Note that both forward and reversed word dictionary can be prepared using phonetic alphabet, as discussed in Section 3. This helps us in tackling phonetically similar character substitution error automatically.

Error detection & position finding and Error correction

To start with we have two assumptions.

Assumption 1: There can be only single error in the word, which is one among insertion, deletion, substitution and transposition.

Assumption 2: The correct word is available in both the dictionary (conventional and reversed word) files.

Finding error region:

Consider, an erroneous string S of n characters. Suppose we try

to match the string in conventional dictionary D_c and check the dictionary word that matches at maximum number of character positions, say k_1 in a sequence starting from left. For example, let the erroneous string be 'forvune' where the error has occurred at 4-th position and the correct word is 'fortune'. In the dictionary, there are several words with 'for...!', but no word with 'forv...!'. Thus, here $k_1 = 3$.

k_1 is the length of maximum matched substring of misspelled word with all words of dictionary D_c .

k_2 is the length of maximum matched substring of misspelled reverse word with all the words of reverse dictionary D_r .

S_l is the length of maximum substring of misspelled word when both the dictionaries are used.

S_r is the length of maximum substring of misspelled word when both the dictionaries are used.

Hence, if length of string is n , then error region can be $n - S_l - S_r$.

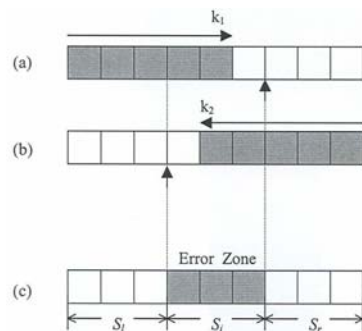


Fig. 1. Error localization by conventional and reverse dictionary.

- (a) Error localization by conventional dictionary.
- (b) Error localization by reverse dictionary.
- (c) Error localization when both dictionaries are used.

Figure 4: Error localization by conventional and reverse dictionary

Note that since the error is a single substitution or deletion the correct word will lie in the dictionary words of n and $n + 1$ characters.

While searching in D_c and D_r , we look only for words of length n and $n + 1$.

Now after finding the error region, we need to correct the misspelled word. Insertion and transposition will be treated in one way and deletion and substitution will be treated another way.

Insertion and transposition:

- If the error is caused due to insertion (transposition), the correct word is deleted (transposed string).
- If there are n characters in a misspelled word, we can make n different strings by deleting one character at a time.
xyz – xy | yz | zx
- Similarly $n-1$ strings can be generated, by transposing one pair of neighboring characters.
xyz – xzy | yxz
- Total, $n + n - 1 = 2n - 1$ strings may be checked in the conventional dictionary and the strings that are valid words are included in the candidate set of correct words.

Deletion and substitution:

- We cannot generate suggestion for deletion and substitution error in the same way as insertion and transposition error.
- If $N =$ alphabet size = 60 for Bangla, $n =$ string length, then $2nN$ strings needed to be generated for checking, hence it is not economical to do it in the same way as insertion and transposition.
- For deletion and substitution reverse dictionary will be used to find the error region.

- If the error region is detected then
 - If the error region is 0 but that word is not in the dictionary then it is a deletion error. To correct this we can just try by inserting each of the letter from alphabet and check if it is a correct dictionary word or not.
 - If the error region is 1 and the word is not in the dictionary then it is a substitution error. To correct this we can delete that letter from the error region and try by inserting each of the letter from alphabet and check if it is a correct dictionary word or not.

This is how Chaudhury's reverse dictionary method works for correcting a misspelled word.

5.1.2.3. Abdullah and Rahman's Recursive simulation method [19]

Recursive simulation method does a grouping for similar sounding letters in Bangla. This considers letters, symbolic form of vowels and consonants. **Figure 5: List of phonetically similar letters, Figure 6: List of vowel-symbols (known as kaar) and Figure 7: List of consonant symbols (known as folaa & reff)** are example of those grouping. Using these grouping they create a circular lists. Each circular list contains similar sounding letters, symbolic form of vowels and consonants. **Figure 8: Circular lists of the grouped letters** are the example of that list.

Set of letters	Spelling
ছ, স, শ, ষ	Between 'S' and 'SH'
জ, য	Between 'JA' and 'JO'
র, ড, ঢ	Between 'RA' and 'RO'
ঙ, ং	Similar to 'ANG'
অ, য়	Between to 'AA' and 'AO'
ই, ঞ	Similar to 'EE'
উ, ঊ	Between 'UO' and 'YO'
ণ, ন	Between to 'NA' and 'NO'

Figure 5: List of phonetically similar letters

Main Vowel	Symbolic Form	Spelling	Position
आ	।	Similar to 'AA'	After the letter
इ	।	Similar to 'EE'	Before the letter
ई	।	Similar to 'EE'	After the letter
ऊ	।	Similar to 'UO'	After the letter
उ	।	Similar to 'UO'	After the letter
ऋ	।	Similar to 'REE'	After the letter
ए	।	Similar to 'E'	Before the letter
ऐ	।	Similar to 'OY'	Before the letter
ॐ	।।	Similar to 'O'	Both sides of the letter
ॐ	।।	Similar to 'OU'	Both sides of the letter

Figure 6: List of vowel-symbols (known as kaar)

Main Consonant	Symbolic Form	Spelling	Position
य	।	Similar to 'JA'	After the letter
र	।	Similar to 'RA'	After the letter
र	।	Similar to 'RA'	Before the letter

Figure 7: List of consonant symbols (known as folaa & reff)

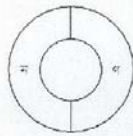


Fig. 4.3. Circular list of the letters spelt as 'NA'.

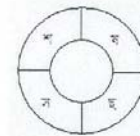


Fig. 4.6. Circular list of the letters spelt as 'SA' and 'SHA'.

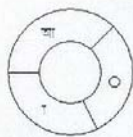


Fig. 4.4. Circular list of the letters spelt as 'AA' (here 'O' means null).

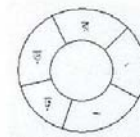


Fig. 4.7. Circular list of the letters spelt as 'RA'.

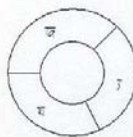


Fig. 4.5. Circular list of the letters spelt as 'JA'.

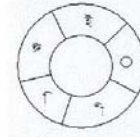


Fig. 4.8. Circular list of the letters spelt as 'EE' (here 'O' means null).

Figure 8: Circular lists of the grouped letters

There are 150 compound letters or conjuncts, which are formed by

joining two or more consonants. Some of them are as following:

Compound Letter	Formation of the Letter	Spelling
ক্ষ	ক + ষ	Similar to 'KHA'
ঙ্গ	ঙ + গ	Similar to 'ONGO'
ক্ত	ক + ত	Similar to 'OKTO'
ন্ত	ন + ত	Similar to 'ONTO'
ষ্ট	ষ + ট	Similar to 'OSHTO'
ষ্ঠ	ষ + ঠ	Similar to 'OSHTHO'

Figure 9: Some compound letters and their formation

Now, in the following Figure 10: Some common Bangla words with their miss-spelt forms some very commonly used Bangla words and their miss-spelt forms are mentioned for farther analysis.

Correct Word	Miss-spelt Form of the Word	Pronunciation of both Correct & Incorrect Forms of the Word
কাগজ	কাগয	KAAGOJ
শাসক	শাশক, সাশক, সাসক	SHAASHOK
জাতীয়	যাতীয়, যাতিয়, যাতীঅ, যাতিঅ, জাতিয়, জাতীঅ, জাতিঅ	JAATIYO
অংশ	অংষ, অংস, অঙশ, অঙষ, অঙস	ANGSHO
ঈষৎ	ইষৎ, ইষত, ইসৎ, ইশৎ, ইসত, ইশত, ঈসৎ, ঈশৎ, ঈসত, ঈশত	EESHOT
শুক	সুক, সুক, সুক্ষ, শুক, শুক, যুক, যুক, যুক	SHOOSHO

Figure 10: Some common Bangla words with their miss-spelt forms

From the data mentioned in Figure 10: Some common Bangla words with their miss-spelt forms, it can be seen that in Bangla language, every single word may have several numbers of analogous phonetic representations and each representation creates its own pronunciation strictly according to its elementary letters and symbols. No word violates the phonetic properties of its constructing letters. But in English words, it is frequently seen that the pronunciation of the word is much different from

the usual phonetic properties of its elementary letters. This is the most prominent difference in the spelling between the Western and South Asian languages.

Since Metaphone and Double Metaphone algorithms manipulate each letter of the alphabet and work mainly on the pronunciation of the syllabi of any word, they are suitable for detecting suggestions in Western languages, which have smaller alphabet of less complexity. But South Asian languages have much bigger alphabet that is also extremely complex. Here, in order to give appropriate suggestions, it is necessary to simulate various representations of the miss-spelt word on the basis of the sets of similarly spelt letters, vowel symbols, consonant symbols and compound letters. But these algorithms are not designed to satisfy all these requirements. Hence, the algorithms like Metaphone and Double Metaphone are not very suitable and efficient for South Asian and many other languages.

Under these circumstances, a new algorithm which can satisfy all the mentioned criteria and reduce the complexity of implementation of searching the nearest suggestions for the miss-spelt word in Bangla as well as other South Asian languages is needed. Here, an algorithm, which may solve this problem partially, has been proposed. This algorithm is named *RecursiveSimulation*, which is still under research and development. In this algorithm a set of circular lists has been used. Each of the lists consists of letters that are phonetically similar. For example, some lists are mentioned in Figure 8: Circular lists of the grouped **letters**.

If in the miss-spelt word, any letter exists in any of the lists, then that letter will be replaced by all other letter of the corresponding list one by one and then will be checked for matching in the dictionary. When any null character will be found in the list then a word representation will be

formed where the position of that letter will be simply ignored.

Algorithm: RecursiveSimulation

Input: W, the miss-spelt word
 S, empty array for storing the suggestions
 P, current position of the character in the
 miss-spelt word (default value is the length if W
 and all external procedures always pass this
 value)
Output: S, array containing the found suggestions

1. RecursiveSimulation (W, S, P)

2. C = Pth character of W
3. L = number of letters in alphabet that are phonetically similar to C
4. T = W
5. while L > 0 do
6. M = empty string
7. ReplaceLetter(T, M, P)
8. if M exists in the dictionary and M does not exist in S then
9. append M to S
10. if P > 1 then RecursiveSimulation (M, S, P - 1)
11. T = M
12. L = L - 1
13. end
14. if P = length of W and P > 1 then do
15. W = leftmost (P - 1) characters of W
16. P = P - 1
17. while Pth character of W is any vowel-symbol or


```

any
    consonant-symbol and P>0
18. do
19.     W = leftmost (P - 1) characters of W
20.     P = P - 1
21. end
22. RecursiveSimulation (W, S, P)
23. end

```

1. ReplaceLetter(T, M, P)

```

2. M = leftmost (P - 1) characters of T
3. L = Pth character of T
4. N = next phonetically similar character to L
5. if N is a vowel-symbol then do
6.     if N is grammatically used before the letter then
7.         if L is a vowel-symbol then
8.             if L is grammatically used before the
letter then
9.                 append N to M
10.            else
11.                if P > 1 then insert N at (P - 1)th
position of M
12.            else
13.                if P > 1 then insert N at (P - 1)th
position of M
14.            else
15.                if L is a vowel-symbol then
16.                    if L is grammatically used before the
letter then
17.                        if P < length of T then do
18.                            P = P + 1
19.                            append Pth character of T to M
20.                            append N to M
21.                        end

```

```

22.             else
23.                 append N to M
24.             else
25.                 append N to M
26.         end
27.     else if N is a vowel then do
28.         if L is a vowel-symbol then
29.             if L is grammatically used before the
letter then
30.                 if P < length of T then do
31.                     P = P + 1
32.                     append Pth character of T to M
33.                     append N to M
34.                 end
35.             else
36.                 append N to M
37.         else
38.             append N to M
39.         end
40.     else
41.         append N to M

```

Sorting the suggestion list:

After generating the suggestion it need to be sorted. Edit distance algorithm is used to sort the suggestions.

A simple simulation for the word is given below in Figure 11: Simulated suggestion list for the word misspelled word, using Recursive Simulation algorithm.

Primary simulation tree:

শিমা

```

+---শিম
!
!   +---সিম
!   !   !   !---সীম
!   !   !   !
!   !   !   !---সম (feasible)
!   !   !   !
!   !   !   !---সইম
!   !   !   !
!   !   !   !---সঈম
!   !   !   !
!   !   !   !---সিম
!   !   !   !
!   !   +---ষিম
!   !   !   !---ষীম
!   !   !   !
!   !   !   !---ষম
!   !   !   !
!   !   !   !---ষইম
!   !   !   !
!   !   !   !---ষঈম
!   !   !   !
!   !   !   !---ষিম
!   !   !   !
!   !   +---ছিম
!   !   !   !---ছীম
!   !   !   !
!   !   !   !---ছম
!   !   !   !
!   !   !   !---ছইম
!   !   !   !
!   !   !   !---ছঈম
!   !   !   !
!   !   !   !---ছিম
!   !   !   !
!   !   +---শিম
!   !   !   !---শীম
!   !   !   !
!   !   !   !---শম (feasible)
!   !   !   !
!   !   !   !---শইম
!   !   !   !
!   !   !   !---শঈম
!   !   !   !
!   !   !   !---শিম (feasible)
!   !   !   !
!   !   +---শিমআ
!   !   !   +---সিমআ
!   !   !   !---সীমআ

```

```

!   !   !---সমআ
!   !   !
!   !   !---সইমআ
!   !   !
!   !   !---সঈমআ
!   !   !
!   !   !---সিমআ
!   !   !
!   !   +---ষিমআ
!   !   !---ষীমআ
!   !   !
!   !   !---ষমআ
!   !   !
!   !   !---ষইমআ
!   !   !
!   !   !---ষঈমআ
!   !   !
!   !   !---ষিমআ
!   !   !
!   !   +---ছিম
!   !   !---ছীমআ
!   !   !
!   !   !---ছমআ
!   !   !
!   !   !---ছইমআ
!   !   !
!   !   !---ছঈমআ
!   !   !
!   !   !---ছিমআ
!   !   !
!   !   +---শিম
!   !   !---শীমআ
!   !   !
!   !   !---শমআ
!   !   !
!   !   !---শইমআ
!   !   !
!   !   !---শঈমআ
!   !   !
!   !   !---শিমআ
!   !   !
!   !   +---শিমা
!   !   !   +---সিমা
!   !   !   !---সীমা (feasible)

```

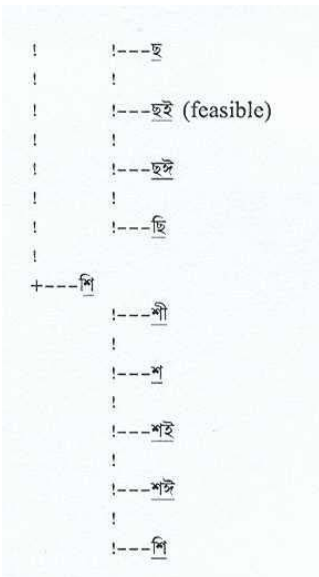
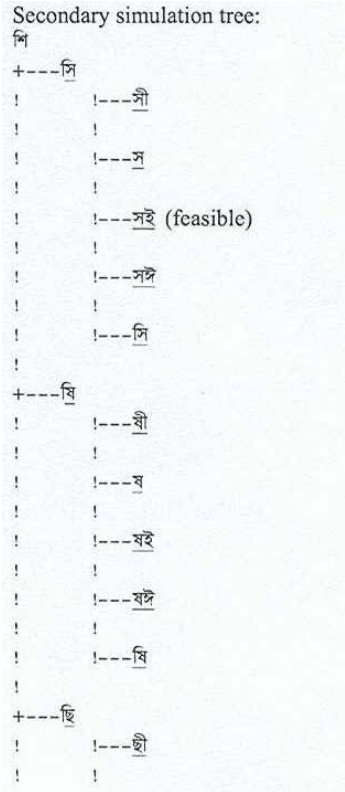
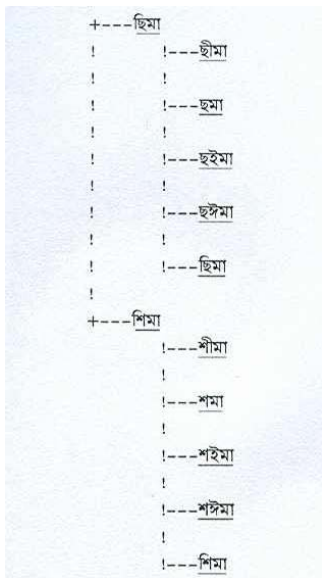
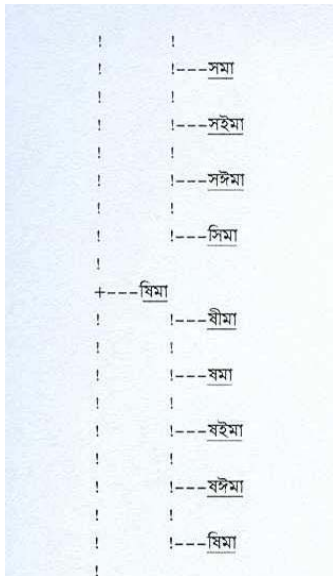


Figure 11: Simulated suggestion list for the word misspelled word, using Recursive Simulation algorithm.

5.1.2.4. Hoque and Kaykobad's soundex type encoding

Description of this method is available in 2.3.1. For spelling checker, we need to use an approximate string matching algorithm after the encoding to get the suggestions.

5.1.2.5. Zaman and Khan's soundex type encoding

Description of this method is available in 2.3.2. For spelling checker, we need to use an approximate string matching algorithm after the encoding to get the suggestions.

5.1.3. Performance of previous techniques

Before considering the performance, we need to find out the challenges for a spelling checker. To find out the challenge we need to understand the peculiar nature of Bangla language, which we need to consider giving similar sounding word in the suggestion. None of the previous technique handles most of the peculiar nature of Bangla and give suggestions accordingly but unfortunately most of them does not even noticed these peculiarities. Following are the challenges to consider when someone generates suggestion in a Bangla spelling checker.

1. There are groups of phonetically similar characters in Bangla; for example, NA (ন) and NNA (ণ); SA (স), SHA (শ) and SSA (ষ), etc. The contrast between long and short vowels in the script is also in the modern version of the spoken language.
2. Bangla has many consonant clusters or conjuncts with unusual pronunciations (i.e., ক্ষ, ক্স, etc.): let us consider ক্ষ. ক্ষ = ক+্+ষ; ক্ষত [KA HASANT SSA TA] /kʰɔʈo/ is pronounced as খত [KHA TA] /kʰɔʈo/, where ষ does not have any sound.
3. Bangla has different uses of *Phalaa's*, the cluster final form of the

semi-vowels in Bangla (BA, MA, YA, RA and LA) which are represented using a distinct sign-form. BA *phalaa* for example has a distinct pronunciation from a BA in any other position in a cluster or in a standalone configuration.

4. Different pronunciation of letters or conjuncts in different contexts: consider again ক্ষ. At the beginning of word, it is pronounced as খ /kʰi/. (ক্ষত → খত /kʰiʈo/); in the middle or at the end of a word, it is pronounced as কখ /kkʰi/, (দক্ষ → দকখ /dɔkkʰi/).
5. Multiple pronunciations of some letters in the same context, such as হ with ব: According to Bangla phonological rules, হ should be pronounced as ও or উ and ব should be pronounced as ভ: আহ্বান → আওভান /aovan/. However, most native speakers pronounce these words the same way as it is written. For example, আহ্বান is usually pronounced as আহভান /aɦɔbhɦan/. Both pronunciations are considered correct.

Typographical error is a trivial challenge in a spelling checker for any language, which can be solved by any string matching algorithms like Edit Distance. But main challenge in a language is a phonetic error. Above, we have described the peculiar phonetic nature of Bangla, which a spelling checker must take in to account generating suggestion.

We have 5 previous techniques, these are:

1. Approximate string matching algorithms: **AS**
2. BB Chaudhury's reverse dictionary method: **BB [15]**
3. Abdullah and Rahman's recursive simulation method: **AR [19]**
4. Hoque and Kaykobad's soundex type encoding: **HK [10, 11]**
5. Zaman and Khan's soundex type encoding: **ZK [9]**

Now, we will see which method can handle the challenges described above.

Table 9: Challenges for spelling checker and performance of previous techniques

Challenge	Mentioned it as problem	Can Handle
There are groups of phonetically similar characters in Bangla; for example, NA (ন) and NNA (ণ); SA (স), SHA (শ) and SSA (ষ), etc. The contrast between long and short vowels in the script is also in the modern version of the spoken language.	<ol style="list-style-type: none"> 1. BB 2. AR 3. HK 4. ZK 	<ol style="list-style-type: none"> 1. BB 2. AR 3. HK 4. ZK
Bangla has many consonant clusters or conjuncts with unusual pronunciations (i.e., ক্ষ, ক্ষ, etc.): let us consider ক্ষ. ক্ষ = ক+্+ষ; ক্ষত [KA HASANT SSA TA] /kʰɔt̪o/ is pronounced as খত [KHA TA] /kʰɔt̪o/, where ষ does not have any sound.	<ol style="list-style-type: none"> 1. ZK 	None
Bangla has different uses of <i>Phalaa's</i> , the cluster final form of the semi-vowels in Bangla (BA, MA, YA, RA and LA) which are represented using a distinct sign-form. BA <i>phalaa</i> for example has a distinct pronunciation from a BA in any other position in a cluster or in a standalone configuration.	<ol style="list-style-type: none"> 1. AR, but describes the trivial one and did not describe the unusual one. 2. HK, also describe only the trivial one and did not notice the unusual one. 3. ZK 	<ol style="list-style-type: none"> 1. AR: can handle the trivial one only. 2. HK: can handle the trivial one only. 3. can handle the trivial one only
Different pronunciation of letters or	<ol style="list-style-type: none"> 1. ZK 	None

<p>conjuncts in different contexts: consider again ক্ষ. At the beginning of word, it is pronounced as খ /kʰ/. (ক্ষত → খত /kʰɔt/); in the middle or at the end of a word, it is pronounced as কখ /kkʰ/, (দক্ষ → দকখ /dɔkkʰ/).</p>		
<p>Multiple pronunciations of some letters in the same context, such as হ with ব: According to Bangla phonological rules, হ should be pronounced as ও or উ and ব should be pronounced as ভ: আহ্বান → আওভান /aovan/. However, most native speakers pronounce these words the same way as it is written. For example, আহ্বান is usually pronounced as আহভান /ahobʰian/. Both pronunciations are considered correct.</p>	None	None

We have seen that none of the previous method can handle the problems, but spelling checker using our proposed phonetic encoding can handle all these problems.

5.1.4. How to rank

To rank the suggestion we used both phonetic edit distance, which is edit distance between phonetic codes, and normal edit distance. We did not use the average of both, but preferred for a weighted average. For example, our

$$\text{score} = a * \text{phonetic_edit_distance} + (1-a) * \text{normal_edit_distance}$$

where, $a > (1-a)$.

We rank the suggestions according to the score achieved for a word.

5.1.5. Performance of our proposed encoding

Table 10: Encoding performance shows the performance of this encoding when it is used on 1607 commonly misspelled words found in [22]. We first apply our encoding to both the correct and misspelled words, and then compute the phonetic edit distance between the two encoded versions. It is considered correct if the edit distance is 0. In our case 134 out of 1607 words do not produce an edit distance of 0 with the correct word, which are termed as error, resulting in an accuracy of 91.37%.

Table 10: Encoding performance

No of words	1607
Correct (Edit Distance 0)	1473
Error	134
Rate of accuracy	91.67%
Rate of error	8.33%

The number of unmatched words fall to 107 and 27 if we consider edit distances of 1 and 2 respectively, as shown in Table 3.

Table 11: Error distribution

Error	134
Edit Distance 1	107
Edit Distance 2	27

When we generate the suggestion list, we can easily add words to the suggestion list when the words have Edit distance ≤ 2 . So, we can always get our expected word in the suggestion list, and more than 91.37% times at the top

of the list.

We can not directly show the performance with each of the previous method. But we have shown in previous section that previous methods can not give us expected suggestions. In Table 12: Performance of proposed phonetic encoding, we will show that how well this can encode the words by comparing with other encoding and approximate string matching methods. Among the Soundex for Bangla by “Hoque and Kaykobad” and “Zaman and Khan”, we have shown only the “Zaman and Khan” one, since they considered Unicode encoding like us. And they are almost similar, just the codes are different.

Table 12: Performance of proposed phonetic encoding contains (i) traditional edit distance algorithm [23], (ii) Soundex encoding described in [zaman khan soundex], and (iii) our proposed encoding. For the Soundex and Double Metaphone methods, the error (denoted by E in the table) is calculated from the phonetic edit distance between the encoded versions. The results clearly show that the proposed encoding performs much better than the other existing methods.

Table 12: Performance of proposed phonetic encoding

Edit Distance			Soundex			Double Metaphone		
Misspelled Word	Correct Word	E	Misspell ed Word	Correct Word	E	Misspelled Word	Correct Word	E
কসট /koʃto/	কষ্ট /koʃto/	2	ksT	ksT	0	ksT	ksT	0
দুকখ /ḍukkʃio/	দুঃখ /ḍukkʃio/	1	dukk	duhk	1	dukk	dukk	0
ষামি /ʃami/	স্বামী /ʃami/	2	sami	sbami	1	sami	sami	0
অতান্ত /ot̪ant̪o/	অতন্ত /ot̪ont̪o/	2	ottant	otjnt	2	ottant	ottnt	1
রিদয় /rʃid̪oi/	হৃদয় /rʃid̪oi/	2	ridy	hrdy	2	ridy	ridy	0

বিসশো /biʃʃo/	বিশ্ব /biʃʃo/	2	biss	bisb	1	biss	biss	0
চাদ /caɖ/	চাঁদ /čãɖ/	1	cad	cad	0	cad	cad	0
অস্তমান /ɔstɔman/	অস্তায়মান /ɔstãẽman/	2	ostman	ostayman	2	ostman	ostayman	2
জ্বরাজীরনো /ʃɔraʃirno/	জ্বরাজীর্ণ /ʃɔraʃirno/	4	jbrajirn	jrajirn	1	jrajirn	jrajirn	0
তরংগ /tɔrɔŋgo/	তরঙ্গ /tɔrɔŋgo/	2	trmg	trmg	0	trngg	trngg	0
কনা /kɔna/	কণা /kɔŋa/	1	kna	kna	0	kna	kna	0
নিন্দ্যনিয় /nindɔniẽ/	নিন্দনীয় /nindɔniẽ/	3	nindjniy	nindniy	1	nindniy	nindniy	0
পদদ /pɔddo/	পদ্য /pɔddõ/	2	pdd	pdm	1	pdd	pdd	0
নিচ /nic/	নীচ /nic/	1	nic	nic	0	nic	nic	0

We have shown the use of proposed encoding in spelling checker. This encoding encapsulates the complex spelling rules for Bangla, and in addition, takes into account some of the dialectic pronunciation differences that are not possible to handle otherwise. The performance results show that it easily outperforms the current state of the art Bangla spelling checkers in producing appropriate suggestions for not only the commonly misspelled words, but also for the large number of “corner” cases which are currently beyond the reach of the other existing methods.

5.2. Transliteration

Transliteration from English letters is particularly important for users who are only familiar with the English keyboard layout, and hence could not type quickly in a different alphabet even if their software would actually support a keyboard layout for another language [27]. This is the main reason of a Transliteration. In case of Bangla, also known as Bengali, we have different keyboard layouts. So, it is hard for a beginner to memorize the layout and write smoothly. Even though there are some phonetic keyboard layouts, which is helpful to get started but we still need a very good transliteration process. Bangla is not a very phonetic language; so orthographic rules are not same as phonetic rules. Mean we may pronounce something but when we write it, it is not exactly all the same. Sometime some letters are silent, sometimes some letters get sound of another letter, and sometimes letters sounds differently depending on context, many complexities like these. So, even if we write something in English then it will be hard to get the correct dictionary word with that pronunciation. In this section we described how we could get these complex dictionary words and normal words in a transliteration from the English pronunciation.

5.2.1. What is transliteration

Transliteration in a narrow sense is a mapping from one script into another script. It tries to be lossless, i.e., the informed reader should be able to reconstruct the original spelling of unknown transliterated words [27]. This is opposed to transcription, which maps the sounds of one language to the script of another language. Still, most transliterations map the letters of the source script to letters pronounced similarly in the goal script, for some specific pair of source and goal language [28]. In a more specialized sense, a transcription is (a system of) writing the sounds of a

word in one language using the script of another language. If the relations between letters and sounds are similar in both languages, a transliteration may be (almost) the same as a transcription. In a broader sense, the word transliteration is used to include both transliteration in the narrow sense and transcription [27].

Considering both the challenges we will give the process of a transliteration for Bangla from English. It is clear that there will be two types. One will be direct mapping and another will be phonetic mapping. Direct mapping will do what transliteration means in narrow sense. There will be one to one mapping. Phonetic mapping will do what transliteration means in broader sense; it will also work as transcription.

5.2.2. Previous transliterations

Transliteration for English to other languages is an important research challenge and many researches have been done in this field. For example, English to Japanese [29], English to Arabic [30, 31, 32, 33] and English to Chinese [34]. These transliterations are also used in various applications, like multi-lingual information retrieval and getting the OOV (out of vocabulary) words of same pronunciation using statical analysis.

Main work on Bangla for transliteration was started by ITRANS [35, 36], in early 1991. Now a day this application is becoming popular and useful. There are some word processors that support transliteration on Bangla [38, 39, 40, 41, 42]. But these are phonetically direct mapping, which is mapping from one script to another and will be lossless. No work on phonetic mapping, which will give the word with same pronunciation from dictionary, has been found so far.

5.2.3. Proposed new technique for transliteration

5.2.3.1. Direct mapping

It is a trivial mapping and existing Transliterations use this method. Most popular mapping is the mapping provided by ITRANS [36]. Some software [37, 38] exactly uses this mapping and some use their own. Still we are giving a mapping, which we used for our direct mapping transliteration. Since this direct mapping is still a phonetic mapping but the difference is, it will not look up in the dictionary if it has any word with same pronunciation. We have introduced an intermediate encoding, which will be used to encode before converting. We need it because in some cases it should not be converted directly, like bool pronounce as bul, hence before mapping we convert “oo” to “u”. Another thing is we will not only consider one letter for one to one mapping, we may sometime consider bigrams for mapping. Because, to represent some Bangla letters phonetically in English we use those birgrams. Like for Bangla letter ঋ /kh/ we use kh.

Table 13: Table for direct mapping

English letter or Bigram	Intermediate Encoding	Name	Bangla letter	Unicode
A	A	AA	আ	\u0986
A	A	SIGN AA	া	\u09BE
b	B	BA	ব	\u09AC
bh	Bh	BHA	ভ	\u09AD
c	C	CA	চ	\u099A
ch	Ch	CHA	ছ	\u099B
d	D	DA	দ	\u09A6
dh	Dh	DHA	ধ	\u09A7
D	D	DDA	ড	\u09A1

Dh	Dh	DDHA	ঢ	\u09A2
e	E	E	এ	\u098F
	E	SIGN E	ে	\u09C7
ee	i	SIGN I	ি	\u09BF
f	Ph	PHA	ফ	\u09AB
g	G	GA	গ	\u0997
gh	Gh	GHA	ঘ	\u0998
h	H	HA	হ	\u09B9
H	H	VISARGA	ঃ	\u0983
i	I	I	ঈ	\u0987
	i	SIGN I	ি	\u09BF
l	l	ll	ঐ	\u0988
	l	SIGN II	ী	\u09C0
j	J	YA	য	\u09AF
J	J	JA	জ	\u099C
jh	Jh	JHA	ঝ	\u099D
k	K	KA	ক	\u0995
kh	Kh	KHA	খ	\u0996
l	L	LA	ল	\u09B2
m	M	MA	ম	\u09AE
M	M	CANDRABIN DU	ে	\u0981
n	N	NA	ন	\u09A8
N	N	NNA	ণ	\u09A3
Nh	Nh	NYA	ঞ	\u099E
ng	Ng	ANUSVARA	ং	\u0982
Ng	Ng	NGA	ঙ	\u0999
o	O	A	অ	\u0985
O @ BEGIN	O	O	ও	\u0993

O @ MIDDLE/END	O	SIGN O	ো	\u09CB
oi	oi	AI	ই	\u0990
	oi	SIGN AI	ে	\u09C8
ou	ou	AU	উ	\u0994
	ou	SIGN AU	ৌ	\u09CC
oo	u	SIGN U	উ	\u09C1
p	p	PA	প	\u09AA
ph	ph	PHA	ফ	\u09AB
q	k	KA	ক	\u0995
r	r	RA	র	\u09B0
R	R	RRA	ড়	\u09DC
Rh	Rh	DDHA	ঢ	\u09A2
s	s	SA	স	\u09B8
sh	sh	SHA	শ	\u09B6
S	S	SSA	ষ	\u09B7
t	t	TA	ত	\u09A4
th	th	THA	থ	\u09A5
T	T	TTA	ট	\u099F
Th	Th	TTHA	ঠ	\u09A0
u	u	U	উ	\u0989
	u	SIGN U	ু	\u09C1
U	U	UU	উ	\u098A
	U	SIGN UU	ু	\u09C2
v	bh	BHA	ভ	\u09AD
w	oa	O AA	ওআ	\u0993 \u0986
x @ BEGIN	j	YA	য	\u09AF
x @ MIDDLE/END	ks	KA SA	কস	\u0995 \u09B8

y	y	YYA	য়	\u09DF
Z	j	YA	য	\u09AF
\	\	HASANT	্	\u09CD

5.2.3.2. Phonetic mapping

In phonetic mapping main idea is we will check in the dictionary if we have the word with same pronunciation. Following is the algorithm of phonetic mapping.

Algorithm of phonetic mapping

if there is a word with the same pronunciation in the dictionary

then convert it to that word

else if there are multiple words with the same pronunciation in the dictionary

then give suggestions for that word and the user will select which one to use

else if there are not words with the same pronunciation in the dictionary

then convert it using direct mapping

Now main challenge is how we can get the pronunciation of a Bangla word to check it with an English word and understand it has the same pronunciation. We have used the phonetic encoding for Bangla proposed in section 4.1. That encoding encodes Bangla word in to an English word that represents the pronunciation of a word. So, our only challenge is to convert the English words in the same manner so that both encoding are consistent. For example, কলম is encoded in to *klm*. Our main challenge will be to encode the English word in a way so that when someone writes kolom then it is encoded to *klm*. So, checking

the encoding we can say that it have the same pronunciation as a dictionary word. Problem is we have to modify the proposed encoding in very few cases, so that both of these represent to same code for the same pronunciation. Following Table 14: Modification in proposed encoding is the modification of proposed encoding. After that in Table 15: Proposed encoding for phonetic mapping, we propose our encoding for English word, which will guarantee to have the same code with the same pronunciation, Bangla word. Table 15: Proposed encoding for phonetic mapping is almost same as *Table 13: Table for direct mapping*. Differences are kept bold so that it can be easily distinguished. In direct mapping we had to keep in mind that there should be a one to one mapping to all Bangla letters so that every letter can be written.

Table 14: Modification in proposed encoding

Bangla letter	Name	Unicode	Encoding in our proposed one	Modified encoding
ভ	BHA	\u09AD	“b”	“bh”
ছ	CHA	\u099B	“c”	“ch”
ধ	DHA	\u09A7	“d”	“dh”
ঢ	DDHA	\u09A2	“d”	“dh”
ঘ	GHA	\u0998	“g”	“gh”
ঝ	JHA	\u099D	“j”	“jh”
খ	KHA	\u0996	“k”	“kh”
ফ	PHA	\u09AB	“p”	“ph”
থ	THA	\u09A5	“t”	“th”
ঠ	TTHA	\u09A0	“T”	“Th”

Table 15: Proposed encoding for phonetic mapping

English letter or Bigram	EncodingLikeB angla	Name	Bangla letter	Unicode
a	A	AA	আ	\u0986
a	A	SIGN AA	া	\u09BE
b	B	BA	ব	\u09AC
bh	Bh	BHA	ভ	\u09AD
c	C	CA	চ	\u099A
ch	Ch	CHA	ছ	\u099B
d	D	DA	দ	\u09A6
dh	Dh	DHA	ধ	\u09A7
D	D	DDA	ড	\u09A1
Dh	Dh	DDHA	ঢ	\u09A2
e	E	E	এ	\u098F
	E	SIGN E	ে	\u09C7
ee	I	SIGN I	ি	\u09BF
f	Ph	PHA	ফ	\u09AB
g	G	GA	গ	\u0997
gh	Gh	GHA	ঘ	\u0998
h	H	HA	হ	\u09B9
H	H	VISARGA	ং	\u0983
I	I	I	ই	\u0987
	I	SIGN I	ি	\u09BF
I capital	I	II	ঈ	\u0988
	I	SIGN II	ী	\u09C0
j	J	YA	য	\u09AF
J	J	JA	জ	\u099C
jh	Jh	JHA	ঝ	\u099D

k	K	KA	ক	\u0995
kh	Kh	KHA	খ	\u0996
l	L	LA	ল	\u09B2
m	M	MA	ম	\u09AE
M	Not Coded	CANDRABIN DU	়ে	\u0981
n	N	NA	ন	\u09A8
N	N	NNA	ণ	\u09A3
Nh	N	NYA	ঞ	\u099E
ng	Ng	ANUSVARA	ং	\u0982
Ng	Ng	NGA	ঙ	\u0999
o	Not Coded	A	অ	\u0985
O @ BEGIN	O	O	ও	\u0993
O @ MIDDLE/END	Not Coded	SIGN O	ৌ	\u09CB
oi	Oi	AI	ই	\u0990
	Oi	SIGN AI	়ে	\u09C8
ou	Ou	AU	উ	\u0994
	Ou	SIGN AU	ৌ	\u09CC
oo	U	SIGN U	উ	\u09C1
p	p	PA	প	\u09AA
ph	Ph	PHA	ফ	\u09AB
q	K	KA	ক	\u0995
r	R	RA	র	\u09B0
R	R	RRA	ড়	\u09DC
Rh	R	DDHA	ঢ়	\u09A2
s	S	SA	স	\u09B8
sh	S	SHA	শ	\u09B6
S	S	SSA	ষ	\u09B7

t	T	TA	ত	\u09A4
th	Th	THA	থ	\u09A5
T	T	TTA	ট	\u099F
Th	Th	TTHA	ঠ	\u09A0
u	U	U	উ	\u0989
	U	SIGN U	ঊ	\u09C1
U	U	UU	ঊ	\u098A
	U	SIGN UU	ঊ	\u09C2
v	Bh	BHA	ভ	\u09AD
				\u0993
w	Oa	O AA	ওআ	\u0986
x @ BEGIN	J	YA	য	\u09AF
x @ MIDDLE/END	Ks	KA SA	কস	\u0995 \u09B8
y	Y	YYA	য়	\u09DF
z	J	YA	য	\u09AF
\	Not Coded	HASANT	্	\u09CD

5.2.4. Example of transliteration

We have described two of our Transliteration methods. Now we will show some examples that will make it clear.

Suppose we have written the following text.

ami bhalo achi. tomar khobor ki. ajke shondha bela tumi ki korcho. obak bepar holo, ami ekhon bangla likhte pari iNGLISH diye. aro mojar bepar holo ami dui bhabe likhte pari. ekTa DairekT arekTa phoneTik. tomar desh e koto Taka te ek Dolar. ami ai bhabe abar juk\to bor\no likhte pari.

5.2.4.1. Direct mapping

Output in direct mapping will be following.

আমি ভালো আছি. তোমার খোবোর কি. আয়কে শোনধা বেলা তুমি কি কোরছো. অবাক বেপার হোলো, আমি এখন বাংলা লিখতে পারি ইংলিশ দিয়ে. আরো মোয়ার বেপার হোলো আমি দুই ভাবে লিখতে পারি. একটা ডাইরেকট আরেকটা ফোনেটিক. তোমার দেশ এ কোতো টাকা তে এক ডোলার. আমি আই ভাবে আবার যুক্তো বোঁনো লিখতে পারি.

5.2.4.2. Phonetic mapping

Output in phonetic mapping will be following.

আমি বহাল/ভাল/ভালো আছি. তোমার খবর কই/কি/কী. আজকে সন্ধ্যা বেলা তুমি কই/কি/কী করছ. অবাক বেপার/ব্যাপার হল, আমি এখন/এখনো বাংলা/বাঙলা লিখতে পারি/পাড়ি ইংলিশ দিয়ে. আর/আরো/আড় মজার বেপার/ব্যাপার হল আমি দুই ভাবে লিখতে পারি/পাড়ি. একটা ডাইরেকট আরেকটা ফোনেটিক . তোমার দেশ/দেঘ এ কত/কোঁত টাকা/টাকা তে একো/এক ডলার . আমি আই ভাবে আবার যুক্ত বরণ/র্বণ/ব্রণ লিখতে পারি/পাড়ি

Table 16: Few examples from above paragraph to make the process clear

English word	Output in direct mapping	Output in phonetic mapping	Selected word
shondha	শোনধা	সন্ধ্যা	সন্ধ্যা
bela	বেলা	বেলা	বেলা
bepar	বেপার	বেপার/ব্যাপার	ব্যাপার
mojar	মোয়ার	মজার	মজার
DairekT ¹	ডাইরেকট	ডাইরেকট	ডাইরেকট
ami	আমি	আমি	আমি
ek	এক	একো/এক	এক
jukto	যুক্তো	যুক্ত	যুক্ত
bor\no	বোঁনো	বরণ/র্বণ/ব্রণ	র্বণ

¹ Not found in the dictionary. So, used direct mapping for its suggestion

We have given this Table 4 to show that how we can handle the similar sounding multiple words in suggestion. Basically, we have to just select our expected word among the suggestions.

5.3. Cross Language Information Retrieval

In cross language retrieval, a user issues a query in one language to search a collection in different language. If the two languages use same alphabet then similar sounding word can be written in the same way in two languages and can easily be found as well. However, if two languages use two different alphabets then it is not an easy task to issue a query in one language to search a collection in different language. In this section, we will describe how our proposed encoding in section 4.1 can be used to work as an intermediate code for this cross language information retrieval.

5.3.1. What does it handle

In this cross language information retrieval between English and Bangla, main work is to take a word in English and it will find the similar sounding word in a Bangla document. To give the suggestion it easily handles the complex Bangla orthographical rules, because of using our proposed phonetic encoding.

5.3.2. Previous work

There are no such efforts given in this type of application for Bangla so far. But in many languages it is solved by first transliteration of one language to another language, after that we can easily search that transliterated word in the document.

For example, we want to search Bangla word বানান /banan/ in a Bangla document. We issue a query in English by “banan”. So, in trivial method by transliteration “banan” will be converted to বানান /banan/ first, and then বানান /banan/ will be searched in the Bangla document, which is a trivial task.

This method is used in languages, where we have a good transliteration, such as [29, 30, 31, and 34]. [33] uses this method for Arabic to English cross language information retrieval.

5.3.3. How does it work

In this cross-lingual information retrieval, our proposed phonetic encoding works as an intermediate code. We code Bangla by our proposed code to Roman alphabets, which represent the pronunciation of the word. Now our main challenge is to encode the English word in the same way, so that its pronunciation can be represented by that code. So, rather than the original word we can operate on the code and find the word of similar pronunciation. For example, in our proposed encoding, কলম is encoded in to *klm*. Our main challenge was to encode the English word in a way so that when someone searches with kolom then it is converted to *klm*.

Our limitation is we can not actually handle complex English words, but if someone writes Bangla using English then they write it in simple English. So, we can easily handle these cases.

It operates almost similarly to our proposed transliteration in phonetic mapping, described in 5.2.3.2. In this section we will refer to some of the tables of that section.

Challenge is we have to modify the proposed encoding in very few cases, so that both of these represent to same code for the same pronunciation. Table 14: Modification in proposed encoding is the modification of proposed encoding. After that in Table 15: Proposed encoding for phonetic mapping, we propose our encoding for English word, which will guarantee to have the same code with the same pronunciation, Bangla word.

5.3.4. Example

Following is an example of a Bangla text. We will try to retrieve few words from this text by issuing queries in English.

Bangle Text:

আমি ভালো আছি. তোমার খবর কি. আজকে সন্ধ্যা বেলা তুমি কি করছ. অবাক ব্যাপার হল, আমি এখন বাংলা লিখতে পারি ইংলিশ দিয়ে. আরো মজার ব্যাপার হল আমি দুই ভাবে লিখতে পারি. একটা ডাইরেকট আরেকটা ফোনেটিক . তোমার দেশ এ কত টাকা তে এক ডলার . আমি আবার যুক্ত বর্ণ লিখতে পারি

Encoding of Bangla Text:

ami bhal achi. tmar khbr ki. ajke shndha bela tumi ki krch. obak bepar hl, ami ekhn bangla likhte pari english diye. ar mjar bepar hl ami dui bhabe likhte pari. ekta DairekT arekta phnetik. tmar desh e kt Taka te ek Dlar. ami abar jukt brn likhte pari.

Following Table 17: English word, encoding of English word, Bangla word with the same encoding from the text contains the English word, which we used to search in a Bangla text; encoding of English word from Table 15: Proposed encoding for phonetic mapping; and Bangla word with same encoding generated by proposed encoding with modification from Table 14: Modification in proposed encoding.

Table 17: English word, encoding of English word, Bangla word with the same encoding from the text

Queries in English word	Encoding of English word	Bangla word with same encoding in the text
Shondha	shndha	সন্ধ্যা
Bela	bela	বেলা
Bepar	bepar	ব্যাপার

Mojar	mjar	মজার
DairekT ²	DairekT	ডাইরেকট
Ami	ami	আমি
Ek	ek	এক
juk\to	jukt	যুক্ত
bor\no	brn	বর্ণ

Hence, we get our desired Bangla word from Bangla text by issuing a query in English.

This is how our encoding work as an intermediate code in multi-lingual information retrieval, where a user issues a query in one language (such as English) to search a collection in a different language (such as Bangla). More specifically, writing the pronunciation of a word in English one can search words with same pronunciation in a Bangla document.

² Not found in the dictionary. So, used direct mapping for its suggestion

5.4. Name Searching and Matching

Names are quite often spelled in a variety of different ways, with all variants considered equivalent. This creates a challenge when searching for and matching names in databases, and linking records among different data sources. The situation is quite complex in Bangla because of its archaic and complex orthographic rules, arising in part from the large gap between the script and pronunciation in Bangla. The Bangla language had gone through a vigorous process of Sanskritization during the 12th century, continuing throughout the middle ages, and this process in large part contributed to this gap. In addition, non-indigenous Bangla names are often derived from a variety of different origins – from Sanskrit, Perso-Arabic languages, Portuguese, and other Western languages. Most of the imported names have gone through at least one significant change in both spelling and pronunciation from the original, and have evolved as names with multiple equivalent spellings in both Bangla and English. However, the spelling variants of most of these names have one thing in common – phonetic similarity – a feature that can be used to match these names with each other. For example, the মুরতোজা /murtoja/ and মরতুজা /mortuja/ are common spelling variants of the same name. The similarity of the two names will be obvious to any native Bangla speaker because of the phonetic similarity along with some knowledge of Bangla name-spelling rules, but may be difficult for an algorithm because of the two character mismatches in two different positions. One solution is to encode the names using a phonetic encoding that encapsulates Bangla orthographic rules along with the peculiarities of the name-spelling rules, and then match the resulting encoded versions. With a variation to the encoding of 4.1 we can propose a phonetic encoding for names that is capable of matching most of the common names in all spelling variants, and in addition, providing the correct suggestion in case of a misspelled name, where

the spelling error is a phonetic one.

While there are well-established phonetic similarity encodings and algorithms available for English and other Western languages, similar work for Bangla, despite it being the 4th largest language by population, is still in its infancy. Most of the recent efforts in Bangla phonetic similarity algorithm are based on Soundex [9, 10], which cannot encode the sound of complex Bangla words; the Double Metaphone encoding in described in section 4.1, tailored for spelling checking application, encapsulates the entire range of orthographic rules, including those involving the large repertoire of conjuncts in Bangla. We base our proposed name encoding on section 4.1, and extend it to support the name-spelling peculiarities in Bangla. We can use this encoding to match similar sounding names in a database, and then use other metrics to rank the match (or the suggestion in the case of a spelling checker).

5.4.1. Proposed name encoding for Bangla

Table 18: Proposed Name Encoding for Bangla details the proposed name encoding for Bangla, followed by the rationale for the various mapping rules. Since any word in Bangla can be name, a fair number of the rules are inherited from the spelling encoding described in section 4.1, and so we describe the rationale for only those that are specifically for names. We assume that the Bangla text is encoded using Unicode Normalization Form C (NFC) [13].

Table 18: Proposed Name Encoding for Bangla

No	Letter	Name	Unicode	Code	Context	Example
1	্	SIGN VIRAMA / Hasant	\u09CD	<i>Not Coded</i>		আব্দুল /abdul/
2	ঁ	CANDRABIN DU	\u0981	<i>Not Coded</i>		চাঁদনী /cḥadni/
3	অ	A	\u0985	<i>Not Coded</i>		
4	আ	AA	\u0986	<i>Not Coded</i>		
5	া	SIGN AA	\u09BE	<i>Not Coded</i>		

No	Letter	Name	Unicode	Code	Context	Example
6	ই	I	\u0987	<i>Not Coded</i>		
7	ঈ	II	\u0988	<i>Not Coded</i>		
8	ি	SIGN I	\u09BF	<i>Not Coded</i>		
9	ী	SIGN II	\u09C0	<i>Not Coded</i>		
10	উ	U	\u0989	<i>Not Coded</i>		
11	ঊ	UU	\u098A	<i>Not Coded</i>		
12	্	SIGN U	\u09C1	<i>Not Coded</i>		
13	্	SIGN UU	\u09C2	<i>Not Coded</i>		
14	ও	O	\u0993	<i>Not Coded</i>		
15	়ে	SIGN O	\u09CB	<i>Not Coded</i>		
16	এ	E	\u098F	<i>Not Coded</i>		
17	়ে	SIGN E	\u09C7	<i>Not Coded</i>		
18	ঐ	AI	\u0990	<i>Not Coded</i>		
19	়ে	SIGN AI	\u09C8	<i>Not Coded</i>		
20	ঔ	AU	\u0994	<i>Not Coded</i>		
21	ৌ	SIGN AU	\u09CC	<i>Not Coded</i>		
22	ক	KA	\u0995	“k”		
23	খ	KHA	\u0996	“k”		
24	ক্ষ		\u0995 \u09CD \u09B7	“k”	@ the beginning	ক্ষত /kʰɔtɔ/
25			\u0995 \u09CD \u09B7	“kk”	@ middle/end	দক্ষ /dɔkkʰo/
26	গ	GA	\u0997	“g”		
27	ঘ	GHA	\u0998	“g”		
28	ঙ	NGA	\u0999	“ng”		বাঙলা /baŋla/
29	ং	ANUSVARA	\u0982	“ng”		বাংলা /baŋla/
30	চ	CA	\u099A	“s”		
31	ছ	CHA	\u099B	“s”		
32	শ	SHA	\u09B6	“s”		শাদমান /ʃaɖman/
33	স	SA	\u09B8	“s”		সামীন /ʃamin/
34	ষ	SSA	\u09B7	“s”		

No	Letter	Name	Unicode	Code	Context	Example
35	য	YA as <i>phalaa</i>	x\u09CD\u09AF	<i>Not Coded</i>	@ the beginning as YA <i>phalaa</i>	শ্যামা /ʃæma/ as YA <i>phalaa</i>
36			...xy \u09CD z \u09CD \u 09AF	<i>Not Coded</i>	@ middle/end with conjuncts	সন্ধ্যা /ʃondʃa/
37			...xy \u09CD \u09AF	Doubles: yy	@ middle/end	সত্যজিত /ʃat̪tojit/
38	য	YA	\u09AF	“j”		
39	জ	JA	\u099C	“j”		
40	ঝ	JHA	\u099D	“j”		
41	ঞ	NYA	\u099E \u099A	“n”	Before CA	অঞ্চল /ɔ̃ncɔl/
42			\u099E \u099B	“n”	Before CHA	বাঞ্ছা /ban̪ʃa/
43			\u099E \u099C	“n”	Before JA	মঞ্জু /mɔ̃ɲʃu/
44			\u099E \u099D	“n”	Before JHA	যঞ্ছা /ʃɔ̃ɲʃa/
45			\u099A \u099E	“n”	After CA	যাচ্ছগ্ন /ʃacna/
46			\u099E \u0985 \u099E\u0987	<i>Not Coded</i>	Before A I	মিঞা /miã/
47			\u099C \u09CD \u099E	“ge”	@ the beginning after JA	জ্ঞাত /gæ̃tɔ/
48			... \u099C \u09CD \u099E	“gg”	@ middle/end after JA	বিজ্ঞান /biggæn/
49			\u099E \u09CD	“n”	With hasant	নঞ /nɔ̃n/
50	ট	TTA	\u099F	“T”		
51	ঠ	TTHA	\u09A0	“T”		
52	ড	DDA	\u09A1	“D”		
53	ঢ	DDHA	\u09A2	“D”		
54	ঋ	VOCALIC R	\u098B	“ri”	@ the beginning	ঋতু /rĩtu/
55			x\u098B	“ri” xri	@ middle/end	বিকৃত /bikkrĩto/ বিকৃত /bikrĩto/
56	র	RA as <i>phalaa</i>	x\u09CD \u09B0	“r”	@ the beginning	প্রকাশ /prokaʃ/

No	Letter	Name	Unicode	Code	Context	Example
57			...x\u09CD \u09B0	“r”	@ middle/end	রাত্রি /rattri/ রাত্রি /ratri/
58	র	RA	\u09B0	“r”		
59	ড়	RRA	\u09DC	“r”		
60	ঢ়	DDHA	\u09A2	“r”		
61	ন	NA	\u09A8	“n”		
62	ণ	NNA	\u09A3	“n”		
63	ত	TA	\u09A4	“t”		
64	থ	THA	\u09A5	“t”		
65	দ	DA	\u09A6	“d”		
66	ধ	DHA	\u09A7	“d”		
67	প	PA	\u09AA	“p”		
68	ফ	PHA	\u09AB	“p”		
69	ব	BA as <i>phalaa</i>	x\u09CD \u09AC y...	Not Coded	@ the beginning	স্বপ্না /ʃɔpna/
70			...x\u09CD y \u09CD \u09AC	Not Coded	BA <i>phalaa</i> with conjuncts	তত্ত্ব /tɔttɔ/
71			... \u09AC \u09CD \u09AC	“bb”	After BA as conjuncts	তিব্বত /tʲibbotʲ/
72			... \u09AE \u09CD \u09AC	“mb”	After MA as conjuncts	লম্ব /lombɔ/
73			... \u0997 \u09CD \u09AC	“gb”	After GA as conjuncts	দিগ্বিদিক /dʲigbidʲik/
74			\u0989 \u09A6 \u09CD \u09AC	“udb”	After Ud- (U DA BA...)	উদ্ব্বেগ /udʒeg/
75			...y \u09CD \u09AC	Doubles: yy	@ middle/end	বিশ্বজিত্ /biʃʃɔjitʃ/
76	ব	BA	\u09AC	“b”		
77	ভ	BHA	\u09AD	“b”		
78	ম	MA <i>phalaa</i>	asx\u09CD \u09AE...	Not Coded	@ the beginning	স্মরণ /ʃɔron/
79			...x\u09CD y \u09CD \u09AE	Not Coded	MA <i>phalaa</i> with conjuncts	সূক্ষ্ম /ʃukʃho/

No	Letter	Name	Unicode	Code	Context	Example
80			... \u0995 \u09CD \u09AE	“km”	After KA as conjuncts	রুক্মিনী /rukmini/
81			... \u0997 \u09CD \u09AE	“gm”	After GA as conjuncts	যুগ্ম /jugma/
82			... \u0999 \u09CD \u09AE	“ngm”	After NGA as conjuncts	বাজ্ময় /baṅmoi/
83			... \u099F \u09CD \u09AE	“tm”	After TTA as conjuncts	কুট্মল /kutmol/
84			... \u09A3 \u09CD \u09AE	“nm”	After NNA as conjuncts	মৃণায় /mrinmœ/
85			... \u09A8 \u09CD \u09AE	“nm”	After NA as conjuncts	জন্ম /jɔnmo/
86			... \u09AE \u09CD \u09AE	“mm”	After MA as conjuncts	রুম্মান /rumman/
87			... \u09B2 \u09CD \u09AE	“lm”	After LA as conjuncts	গুল্ম /gulmo/
88			... \u09B6 \u09CD \u09AE	“sm”	@ middle/end with SHA	কাশ্মীর /kaʃmir/
89			... \u09B7 \u09CD \u09AE	“sm”	@ middle/end with SSA	কুম্মাড /kuʃmandɔ/
90			... \u09B8 \u09CD \u09AE	“sm”	@ middle/end with SA	সুম্মিতা /ʃuʃmiṭa/
91			...y \u09CD \u09AE	Doubles: yy	@ middle/end otherwise	রশ্মি /rɔʃʃe/
92	ম	MA	\u09AE	“m”		
93	য়	YYA	\u09DF	<i>Not Coded</i>		মিয়া /mia/ সায়ম /saiem/
94	ল	LA	\u09B2	“l”		
95	হ	HA	\u09B9 \u09CD \u098B	“ri”	HA with Vocalic R	হৃদয় /rɦidoi/
96			\u09B9 \u09CD \u09B0	“r”	HA with R as <i>phalaa</i>	হৃদ /rɔd/
97			\u09B9 \u09CD \u09A8	“nn”	HA with NA	পূর্বাহ্ন /purbannɔ/

No	Letter	Name	Unicode	Code	Context	Example
98			\u09B9 \u09CD \u09A3	“nn”	HA with NNA	প্রাণ্ন /prannfio/
99			\u09B9 \u09CD \u09AE	“mm”	HA with MA	ব্রম্মা /brommfia/
100			\u09B9 \u09CD \u09AF	“jj”	HA with YA as <i>phalaa</i>	উহ্য /ujffio/
101			\u09B9 \u09CD \u09B2...	“l”	HA with LA @ beginning	ফ্লাদ /lfia/
102			... \u09B9 \u09CD \u09B2	“ll”	HA with LA @ middle/end	আফ্লাদ /allfiad/
103			\u09B9 \u09CD \u09AC	“h” “o”	HA with BA	আফ্বান /aovan/ আফ্বান /afiofian/
104	হ	HA	\u09B9	<i>Not Coded</i>	Otherwise	
105	ং	Visarga	One to one Transformations	Encode using rest of the rules after transformati on		মোং → মোহাম্মদ /mohammad/
106			x\u0983 y...	Doubles: yy	@ the middle	দুঃসময় /dujffomoẽ/
107			x\u0983	“h”	@ the end strlen == 1 2	উঃ /ufi/, বাঃ /bafi/
108			x\u0983	<i>Not Coded</i>	Otherwise @ the end	পুনঃ /puno/

The transformation or rules described in Table 18: Proposed Name Encoding for Bangla were derived from a large set of names in the literature [43, 44, 45], which include both common and uncommon names, and of different origins. We describe the rationale for the name-encoding transformations below.

5.4.2. Rationale for Name encoding

Transformations 1, 2: Reason why SIGN VIRAMA (Hasant) and CANDRABINDU are to be *Not Coded* can be found in section 4.1.

Transformations 3 – 21: In our encoding, vowels are *Not Coded*. This is to account for pronunciation differences from person to person, or region-to-region, where the differences are due to vowels.

The following is an example of a name, which is spelled (and pronounced) differently by native speakers:

মরতুজা /mɔrtuʒa/, মুরতোজা /murtoʒa/, মরতোজা /mɔrtuʒa/, মোরতুজা /mortuʒa/

In our encoding, all of these variants are encoded as “mrtj”, and can be matched against each other regardless of spelling variation. Table 2 shows a few more such examples justifying the decision to mark vowels as *Not Coded*.

Table 19. Example of vowels encoding

Similarly pronounced names	Encoding
নাইম /naim/, নঈম /noim/	“nm”
নাহলীন /nahleen/, নেহলীন /nehleen/	“nlh” ³
নওশাদ /nɔʃad/, নাওসাদ /naʃad/	“nsd”
সুমিন /ʃumin/, সোমেন /ʃomen/	“smn”
রাশেদ /raʃed/, রশিদ /rɔʃid/	“rsd”
মুস্তোফা /mustofa/, মোস্তফা /mostɔfa/	“mstp”

Transformations 22-29: Names are just words, so the rationale is the same as for words, see section 4.1 for its reasoning.

Transformations 30-34: In encodings designed for spelling checkers [5, 7], শ (/s/, /ʃ/), স (/s/, /ʃ/), ষ /ʃ/ are encoded the same as they are very close in

³Rationale for হ to be Not Coded is according to Transformation 104

pronunciation; similarly for চ /c/ and ছ /ch/. However, in case of name encoding, we encode all 5 of these letters to the same code. The reason is that in Bangla, the sound /s/ is expressed using স (/s/, /ʃ/), but sometimes also with ছ /ch/. Our solution is to encode স (/s/, /ʃ/) and ছ /ch/ the same way. Since these two letters belonged to two different groups, we combine the two groups and use the same code.

Example: The name /salam/ is usually written as সালাম /ʃalam/, but often also as ছালাম /chalam/. সালাম /ʃalam/ is phonetically more appropriate as স sounds like /s/ and /ʃ/; to make matters worse, even if /salam/ is written as ছালাম /chalam/, it is still pronounced as /salam/. Following are few more examples of names where স (/s/, /ʃ/) and ছ /ch/ are both pronounced as /s/, to justify the decision to make স (/s/, /ʃ/) and ছ /ch/ in the same group.

Table 20. Example of স and ছ

Name with pronunciation (according to rules)	Both Locally pronounced as	Encoding
বাসেত /baʃet/ , বাছেত /bachet/	/baset/	“bst”
মুকসিত /mukʃit/ , মুকছিত /mukchit/	/muksit/	“mkst”
নাফিস /nafɪʃ/ , নাফিছ /nafich/	/nafis/	“nfs”
হাসিনা /haʃina/ , হাছিনা /hachina/	/hasina/	“sn” ⁴

Transformation 35: At the beginning of a word, and if the word is অ-কারান্ত /ɔ/ or আ-কারান্ত /a/, it is pronounced as /æ/ and if there is a ই or উ after য *phalaa*, then it is pronounced as এ /e/. Both of these were encoded to “e” in section 4.1. But in case of names, vowels are *Not Coded*. So, it is *Not Coded*.

⁴Rationale for হ to be Not Coded is according to Transformation 104

Example: শ্যামা /ʃæma/ and শেমা /ʃema/ are both encoded as “sm”, which are similar sounding.

Transformations 36-92: Names are just words, so the rationale is the same as for words, see section 4.1 for its reasoning.

Transformation 93: In names, য় is almost silent; it mainly gets the sound of attached vowel and sometimes causes nasalization. So, it is *Not Coded*.

Example: মিয়া /miã/ → “m”, সায়েম /saiẽm/ → “sm”, সারিয়া /sariã/ → “saria”

Transformations 94-103: Names are just words, so the rationale is the same as for words, see section 4.1 for its reasoning.

Transformation 104: In names, হ is usually silent or almost silent. So, it is *Not Coded*.

Table 21. Example of হ

Names With হ	Names Without হ	Encoding
যাহরা /ʃafra/	যারা /ʃara/	“jr”
নাবিলাহ /nabilafi/	নাবিলা /nabila/	“nbl”
তাহমিনাহ /ʔafiminafi/	তামিনা /ʔamina/	“tmn”
ফাহমিদা /fahmida/	ফামিদা /famida/	“pmd”

Transformation 105: The equivalent of the English “.” in name abbreviations and titles is Bangla োঃ, e.g., মোঃ is the same as মোহাম্মদ /mohammød/. Since these are often ad-hoc, one-to-one transformations are used before encoding process. This set of transformations will of course be expanded as more new cases come in use. So, to encode মোঃ we will first transform it to মোহাম্মদ /mohammad/ before the final encoding /mohammad/.

Table 22. One to one transformation of োঃ

Short cut	Elaborated form	Encoding

মোঃ	মোহাম্মদ /mohammad/	“mmmD”
ডঃ	ডক্টর /dɔktor/	“DkTr”
ডাঃ	ডাক্তার /dactar/	“DkTr”
এডঃ	এডভোকেট /advokæt/	“DbkT”

Table 22. One to one transformation of ঃ lists just a few of the very common – there is quite a large number in use, and new cases do get added to the colloquial use over time.

Transformations 106-108: Names are just words, so the rationale is the same as for words, see section 4.1 for its reasoning.

5.4.1. Algorithm and performance of name searching using proposed phonetic encoding

A naïve approach is to search for the encoded string in the database, which may return a large number of names, many of which are not considered equivalent to the name being searched for. The encoding removes all the vowels and the letters marked as *Not Coded*, so the encoded string is typically much shorter than the original name. Since many other names may map to this shorter encoded string, the match returns many irrelevant names in addition to the “equivalent” ones. To avoid this problem, other figures of merit must be used to narrow this list to include only the desired set, and to rank the resulting set in order of relevance [46]. We propose one such figure of merit that uses a weighted sum of the orthographic and phonetic edit-distances to exclude dissimilar names from the query result. We give an algorithm to search for a name (in this case মরতুজা /mɔrtuʒa/). After the algorithm, **Table 23: Generating suggestions for names using name encoding and other trivial methods** shows a pre-encoded list of names to search, with various columns that are computed during the various steps.

Algorithm for Name searching

- Encode the name to search for: মরতুজা /mɔɾtʊʒa/ → mrtj.
- Compute the Levenshtein edit-distance [23] (column ED) between the candidate name and each of the names from list.
- Compute the edit distance score (column EDscr) between the two strings s1 and s2 from ED: $EDscr = (\maxLen(s1, s2) - ED) / \maxLen(s1, s2)$.
- Compute the phonetic edit-distance (column PED), using the encoded versions.
- Compute the phonetic edit distance score (PEDscr) from PED: $PEDscr = (\maxLen(s1, s2) - PED) / \maxLen(s1, s2)$.
- The figure of merit (FOM) is the weighted sum of PEDscr and Edscr, with PEDscr as the dominant factor: $(PEDscr + Edscr/10)/1.1$ and value ranges from 0 to 1.

Table 23: Generating suggestions for names using name encoding and other trivial methods

Names	Encoding	ED	EDscr	PED	PEDscr	FOM
সুমিন /ʃumin/	"smn"	6	0	4	0	0
রশিদ /rɔʃid/	"rsd"	5	0.167	4	0	0.02
মুস্তোফা /mustofa/	"mstp"	5	0.375	2	0.5	0.49
বাছেত /bachet/	"bst"	6	0	3	0.25	0.23
মুকসিত /mukʃit/	"mkst"	5	0.167	3	0.25	0.24
মরতুজা /mɔɾtʊʒa/	"mrtj"	0	1	0	1	1
মুরতোজা /murtoʒa/	"mrtj"	2	0.714	0	1	0.97
মরতোজা /mɔɾtoʒa/	"mrtj"	1	0.833	0	1	0.98
মোরতুজা /mortʊʒa/	"mrtj"	1	0.857	0	1	0.99

We can use the FOM to rank the matches returned by the query, which in this case does correspond to the expected convention for the Bangla name মর্তুজা /mɔɾtʃuʒa/. We expect that a name searching algorithm will need to tailor the figure of merit to the application domain.

CHAPTER VI: CONCLUSION

We present a Double Metaphone phonetic encoding for Bangla, tailored for applications like spelling checker, transliteration, cross-lingual information retrieval and name searching. This encoding encapsulates the complex spelling rules for Bangla, and in addition, takes into account some of the dialectic pronunciation differences that are not possible to handle otherwise. The result in each case shows that it easily outperforms the current state of the art Bangla spelling checker, transliteration, name searching application and also opens new area of research on cross-lingual information retrieval.

6.1. Summary of contributors

- 6.1.1. Can be used to develop a spelling checker, which can give the words of same pronunciation in suggestion.
- 6.1.2. Can be used to develop a transliteration, which can use not only a one to one mapping but also able to give words with same pronunciation from dictionary
- 6.1.3. Can be used to develop a name searching application, where similar sounding names can be easily found and ranked in the suggestion.
- 6.1.4. Can be used as an intermediate code in multi-lingual information retrieval, where a user issues a query in one language (such as English) to search a collection in a different language (such as Bangla). More specifically, writing the pronunciation of a word in English one can search words with same pronunciation in a Bangla document.

6.2. Future research

6.2.1. Digital pronunciation dictionary can be developed from the variation of this encoding, which can be used as a stand-alone digital pronunciation dictionary and also for a Text to Speech application.

REFERENCES

- [1] Definition of phonetic encoding available online at <http://www.nist.gov/dads/HTML/phoneticEncoding.html>.
- [2] Facts about the World's Languages: an Encyclopedia of the World's Major Languages, Past and Present, Jane Garry and Carl Rubino (ed.), New York/Dublin: H. W. Wilson Press, 2001.
- [3] P. Kundu and B.B. Chaudhuri, "Error Pattern in Bangla Text", *International Journal of Dravidian Linguistics*, 28(2), 1999.
- [4] The Soundex Algorithm, available online at http://www.archives.gov/research_room/genealogy/census/soundex.html.
- [5] Lawrence Phillips, "Hanging on the Metaphone", *Computer Language*, 7(12), 1990.
- [6] Lawrence Philip's Metaphone Algorithm, available online at <http://aspell.sourceforge.net/metaphone/index.html>
- [7] Lawrence Phillips, "The Double Metaphone Search Algorithm", *C/C++ Users Journal*, 18(6), June 2000, available online at <http://www.cuj.com/documents/s=8038/cuj0006philips/>.
- [8] T. N. Gadd, "PHONIX: The Algorithm", *Program*, 24(4), pp. 363-366, 1990.
- [9] Naushad UzZaman and Mumit Khan, "A Bangla Phonetic Encoding for Better Spelling Suggestion", *Proc. 7th International Conference on Computer and Information Technology*, Dhaka, December, 2004.
- [10] Md. Tamjidul Haque and M. Kaykobad, "Coding System for Bangla Spell Checker", Page 186 – 190, *Proc. 5th International Conference on Computer and Information Technology*, Dhaka, December, 2002.

- [11] Md. Tamjidul Haque and M. Kaykobad, "Use of Phonetic Similarity for Bangla Spell Checker", Page 182 – 185, *Proc. 5th International Conference on Computer and Information Technology*, Dhaka, December, 2002.
- [12] J. Zobel and P. Dart, "Finding Approximate Matches in Large Lexicons", *Software - Practice and Experience*, 25(3), pp. 331-345, March, 1995.
- [13] The Unicode Consortium, *The Unicode Standard, Version 4.0*, Addison-Wesley, 2003.
- [14] Bangla Unicode Chart, available online at <http://www.unicode.org/charts/PDF/U0980.pdf>.
- [15] B. B. Chaudhuri, "Reversed word dictionary and phonetically similar word grouping based spell-checker to Bangla text", *Proc. LESAL Workshop*, Mumbai, 2001.
- [16] Daniel Jurafsky and James H. Martin, "Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition", ISBN – 0-13-095069-6, Prentice Hall, 2000.
- [17] F.J. Damerau, "A technique for computer detection and correction of spelling errors", *communication of ACM*, 7(3), 171-176, 1964.
- [18] Arif Billah Al-Mahmud Abdullah and Ashfaq Rahman, "A Different Approach in Spell Checking for South Asian Languages", *Proc. 2nd International Conference on Information Technology for Applications (ICITA)*, China, 2004.
- [19] Bangla Uccharon Obidhan, Bangla Academy, Dhaka, Bangladesh.
- [20] Bangla Banan Obidhan, Bangla Academy, Dhaka Bangladesh.
- [21] R. Ishida's Bengali script notes [Draft], available online at <http://people.w3.org/ishida/scripts/bengali/bengali-script/>.
- [22] Bangla Banan Obidhan, Dr. Khurshid Alam, Mirnava, Dhaka, Bangladesh.

- [23] Levenshtein edit distance algorithm, available online at <http://www.nist.gov/dads/HTML/Levenshtein.html>.
- [24] <http://www.cs.virginia.edu/~cyberia/presentations/EuDL/ppt/sld011.htm>.
- [25] <http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/Edit/>.
- [26] <http://www.cs.sunysb.edu/~algorithm/files/longest-common-substring.shtml>.
- [27] Wikipedia description on Transliteration, available online at <http://en.wikipedia.org/wiki/Transliteration>.
- [28] Wikipedia description on Transcription, available online at http://en.wikipedia.org/wiki/Transcription_%28linguistics%29.
- [29] Kevin Knight and Jonathan Graehl, "Machine Transliteration", available online at <http://acl.ldc.upenn.edu/J/J98/J98-4003.pdf>.
- [30] Yaser Al-Onaizan and Kevin Knight, "Machine Transliteration of Names in Arabic Text", available online at <http://acl.ldc.upenn.edu/acl2002/SEMITIC/pdfs/Semitic027.pdf>.
- [31] Nasreen AbdulJaleel and Leah S. Larkey, "English to Arabic Transliteration for Information Retrieval: A Statistical Approach", available online at <http://ciir.cs.umass.edu/pubfiles/ir-261.pdf>.
- [32] Leah S. Larkey, Nasreen AbdulJaleel, Margaret Connell, "What's in a Name?: Proper Names in Arabic Cross Language Information Retrieval", available online at <http://ciir.cs.umass.edu/pubfiles/ir-278.pdf>.
- [33] Nasreen AbdulJaleel and Leah S. Larkey, "Statistical Transliteration for English-Arabic Cross Language Information Retrieval", available online at, <http://ciir.cs.umass.edu/pubfiles/ir-293.pdf>.
- [34] GAO Wei, "Phoneme based Statistical Transliteration of Foreign Names for OOV problem", MSc Thesis, Chinese University of Hong Kong, 2004, available online at <http://compling.ai.uiuc.edu/webpage/projects/thesis.pdf>.
- [35] ITRANS, available online at <http://www.aczoom.com/itrans/>.
- [36] ITRANS table, available online at <http://sanskrit.gde.to/web-interface/bengali.html>.
- [37] Aksharmala mapping, available online at

<http://aksharamala.com/help/chm/Input%20Schemes/ITRANS/Bengali/quick.html>

[38] Iwrite32, available online at <http://members.tripod.com/~sbiswas/IWrite32/IWrite32.html>.

[39] Bornosoft, available online at <http://www.bornosoft.com/>.

[40] Kickkeys, available online at <http://www.kickkeys.com/>.

[41] Lekho, available online at <http://lekho.sourceforge.net/>.

[42] Bengali Transliteration System by prabashi.org, available online at <http://www.prabasi.org/Literary/ComposeArticle.html>.

[43] Sadikur Rahman, "Apar shontaner prio naam", Salahuddin Boi Ghar, Bangla Bazar, Dhaka, September, 2003.

[44] Anis Ahmed, "Bisshe shreshto 110 monishi", Dhaka, Bangladesh.

[45] List of Contributors, "BANGLAPEDIA: National Encyclopedia of Bangladesh", Dhaka, Bangladesh, 2003.

[46] NameX Technology, available online at <http://www.imagepartners.co.uk/Thesaurus/AboutNameX.htm>.

APPENDICES

A. Bangla Alphabet

Soro Barna

অ আ ই ঈ উ ঊ ঋ ঌ এ ঐ ও ঔ

Banjan Barna

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 ঝ ঞ ঃ ৎ

Bangla Number

১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯ ০

B. Bangla Unicode Chart

Bengali

Range: 0980–09FF

This file contains an excerpt from the character code tables and list of character names for the Unicode Standard, last updated for *The Unicode Standard, Version 4.0*.

This file may be updated as necessary to reflect errata without notice. For an up-to-date list of errata, see <http://www.unicode.org/errata/>

Disclaimer

These charts are provided as the on-line reference to the character contents of the Unicode Standard, Version 4.0 but do not provide all the information needed to fully support individual scripts using the Unicode Standard. For a complete understanding of the use of the characters contained in this excerpt file, please consult the appropriate sections of *The Unicode Standard, Version 4.0* (ISBN 0-321-18578-1), as well as Unicode Standard Annexes #9, #11, #14, #15, #24 and #29, the other Unicode Technical Reports and the Unicode Character Database, which are available on-line.

See <http://www.unicode.org/Public/UNIDATA/UCD.html> and <http://www.unicode.org/reports/>

A thorough understanding of the information contained in these additional sources is required for a successful implementation.

Fonts

The shapes of the reference glyphs used in these code charts are not prescriptive. Considerable variation is to be expected in actual fonts. The particular fonts used in these charts were provided to the Unicode Consortium by a number of different font designers, who own the rights to the fonts.

See <http://www.unicode.org/charts/fonts.html> for a list.

Terms of Use

You may freely use these code charts for personal or internal business uses only. You may not incorporate them either wholly or in part into any product or publication, or otherwise distribute them without express written permission from the Unicode Consortium. However, you are welcome to provide links to these charts.

The fonts and font data used in production of these Code Charts may NOT be extracted or otherwise used in any commercial product without permission or license granted by the typeface owner(s).

The information in this file may be updated from time to time. The Unicode Consortium is not liable for errors or omissions in this excerpt file or the standard itself. Information on characters added to the Unicode Standard since the publication of Version 4.0 as well as on characters currently being considered for addition to the Unicode Standard can be found on the Unicode web site.

See <http://www.unicode.org/pending/pending.html> and <http://www.unicode.org/alloc/Pipeline.html>.

Copyright © 1991-2003 Unicode, Inc. All rights reserved.

0980

Bengali

09FF

	098	099	09A	09B	09C	09D	09E	09F
0		ঐ 0980	ঔ 09A0	র 09B0	ী 09C6		ঋ 09E0	ঌ 09F1
1	ঁ 0981		ড 09A1		় 09C1		ঙ্ 09E1	ঞ 09F1
2	ং 0982		ঢ 09A2	ণ 09B2	্ন 09C2		্ 09E2	় 09F2
3	ঃ 0983	ও 0983	ঞ 09A3		় 09C3		্ 09E3	্ত 09F3
4		ঙ 0984	ত 09A4		় 09C4			় 09F4
5	অ 0985	ক 0985	খ 09A5					খ 09F5
6	আ 0986	খ 0986	দ 09A6	শ 09B6			০ 09E6	ঙ 09F6
7	ই 0987	গ 0987	ষ 09A7	ষ 09B7	ে 09C7	ো 09D7	১ 09E7	। 09F7
8	ঈ 0988	ঘ 0988	ন 09A8	স 09B8	ৈ 09C8		২ 09E8	৷ 09F8
9	উ 0989	ঙ 0989		হ 09B9			৩ 09E9	০ 09F9
A	ঊ 098A	চ 098A	প 09AA				৪ 09EA	৵ 09FA
B	ঋ 098B	ছ 098B	ফ 09AB		ৌ 09CB		৫ 09EB	
C	ৱ 098C	জ 098C	ব 09AC	় 09BC	ৌ 09CC	ড় 09DC	৬ 09EC	
D		ঝ 098D	ভ 09AD	হ 09BD	় 09CD	ঢ 09DD	৭ 09ED	
E		ঞ 098E	য 09AE	া 09BE			৮ 09EE	
F	এ 098F	ট 098F	য 09AF	ি 09BF		য় 09DF	৯ 09EF	

0981

Based on ISCII 1988

Various signs

0981	◌̂	BENGALI SIGN CANDRABINDU
0982	◌̃	BENGALI SIGN ANUSVARA
0983	◌̄	BENGALI SIGN VISARGA

Independent vowels

0985	অ	BENGALI LETTER A
0986	আ	BENGALI LETTER AA
0987	ই	BENGALI LETTER I
0988	ঈ	BENGALI LETTER II
0989	উ	BENGALI LETTER U
098A	ঊ	BENGALI LETTER UU
098B	ঋ	BENGALI LETTER VOCALIC R
098C	ৠ	BENGALI LETTER VOCALIC L
098D	◌̅	<reserved>
098E	◌̆	<reserved>
098F	এ	BENGALI LETTER E
0990	ঐ	BENGALI LETTER AI
0991	◌̇	<reserved>
0992	◌̈	<reserved>
0993	ও	BENGALI LETTER O
0994	ঔ	BENGALI LETTER AU

Consonants

0995	ক	BENGALI LETTER KA
0996	খ	BENGALI LETTER KHA
0997	গ	BENGALI LETTER GA
0998	ঘ	BENGALI LETTER GHA
0999	ঙ	BENGALI LETTER NGA
099A	চ	BENGALI LETTER CA
099B	ছ	BENGALI LETTER CHA
099C	জ	BENGALI LETTER JA
099D	ঝ	BENGALI LETTER JHA
099E	ঞ	BENGALI LETTER NYA
099F	ট	BENGALI LETTER TTA
09A0	ঠ	BENGALI LETTER TTHA
09A1	ড	BENGALI LETTER DDA
09A2	ঢ	BENGALI LETTER DDHA
09A3	ণ	BENGALI LETTER NNA
09A4	ত	BENGALI LETTER TA
09A5	থ	BENGALI LETTER THA
09A6	দ	BENGALI LETTER DA
09A7	ধ	BENGALI LETTER DHA
09A8	ন	BENGALI LETTER NA
09A9	◌̅	<reserved>
09AA	প	BENGALI LETTER PA
09AB	ফ	BENGALI LETTER PHA
09AC	ব	BENGALI LETTER BA = Bengali va, wa
09AD	ভ	BENGALI LETTER BHA
09AE	ম	BENGALI LETTER MA
09AF	য	BENGALI LETTER YA
09B0	র	BENGALI LETTER RA

Bengali

09D7

09B1	◌̅	<reserved>
09B2	লা	BENGALI LETTER LA
09B3	◌̅	<reserved>
09B4	◌̅	<reserved>
09B5	◌̅	<reserved>
09B6	শ	BENGALI LETTER SHA
09B7	ষ	BENGALI LETTER SSA
09B8	স	BENGALI LETTER SA
09B9	হ	BENGALI LETTER HA

Various signs

09BC	◌̅	BENGALI SIGN NUKTA • for extending the alphabet to new letters
09BD	৞	BENGALI SIGN AVAGRAHA

Dependent vowel signs

09BE	◌̅	BENGALI VOWEL SIGN AA
09BF	◌̅	BENGALI VOWEL SIGN I • stands to the left of the consonant
09C0	◌̅	BENGALI VOWEL SIGN II
09C1	◌̅	BENGALI VOWEL SIGN U
09C2	◌̅	BENGALI VOWEL SIGN UU
09C3	◌̅	BENGALI VOWEL SIGN VOCALIC R
09C4	◌̅	BENGALI VOWEL SIGN VOCALIC RR
09C5	◌̅	<reserved>
09C6	◌̅	<reserved>
09C7	◌̅	BENGALI VOWEL SIGN E • stands to the left of the consonant
09C8	◌̅	BENGALI VOWEL SIGN AI • stands to the left of the consonant

Two-part dependent vowel signs

These two-part dependent vowel signs have glyph pieces which stand on both sides of the consonant. These vowel signs follow the consonant in logical order, and should be handled as a unit for most processing.

09CB	◌̅	BENGALI VOWEL SIGN O = 09C7 ◌̅ 09BE ◌̅
09CC	◌̅	BENGALI VOWEL SIGN AU = 09C7 ◌̅ 09D7 ◌̅

Various signs

09CD	◌̅	BENGALI SIGN VIRAMA = hasant (Bengali term for halant)
09CE	◌̅	<reserved>
09CF	◌̅	<reserved>
09D0	◌̅	<reserved>
09D1	◌̅	<reserved>
09D2	◌̅	<reserved>
09D3	◌̅	<reserved>
09D4	◌̅	<reserved>
09D5	◌̅	<reserved>
09D6	◌̅	<reserved>
09D7	◌̅	BENGALI AU LENGTH MARK

09DC

Bengali

09FA

Additional consonants

09DC ঙ BENGALI LETTER RRA
= 09A1 ঙ 09BC ঙ
09DD ঢ BENGALI LETTER RHA
= 09A2 ঢ 09BC ঙ
09DE ণ <reserved>
09DF য BENGALI LETTER YYA
= 09AF য 09BC ঙ

Generic additions

09E0 ঝ BENGALI LETTER VOCALIC RR
09E1 ঞ BENGALI LETTER VOCALIC LL
09E2 ঐ BENGALI VOWEL SIGN VOCALIC L
09E3 ঊ BENGALI VOWEL SIGN VOCALIC LL

Digits

09E6 ০ BENGALI DIGIT ZERO
09E7 ১ BENGALI DIGIT ONE
09E8 ২ BENGALI DIGIT TWO
09E9 ৩ BENGALI DIGIT THREE
09EA ৪ BENGALI DIGIT FOUR
09EB ৫ BENGALI DIGIT FIVE
09EC ৬ BENGALI DIGIT SIX
09ED ৭ BENGALI DIGIT SEVEN
09EE ৮ BENGALI DIGIT EIGHT
09EF ৯ BENGALI DIGIT NINE

Bengali-specific additions

09F0 ঝ BENGALI LETTER RA WITH MIDDLE
DIAGONAL
• Assamese
09F1 ঞ BENGALI LETTER RA WITH LOWER
DIAGONAL
= BENGALI LETTER VA WITH LOWER
DIAGONAL
• Assamese
09F2 ৳ BENGALI RUPEE MARK
09F3 ৳ BENGALI RUPEE SIGN
09F4 ৳ BENGALI CURRENCY NUMERATOR
ONE
• not in current usage
09F5 ৳ BENGALI CURRENCY NUMERATOR
TWO
• not in current usage
09F6 ৳ BENGALI CURRENCY NUMERATOR
THREE
• not in current usage
09F7 ৳ BENGALI CURRENCY NUMERATOR
FOUR
09F8 ৳ BENGALI CURRENCY NUMERATOR
ONE LESS THAN THE DENOMINATOR
09F9 ৳ BENGALI CURRENCY DENOMINATOR
SIXTEEN
09FA ৳ BENGALI ISSHAR

C. IPA (International Phonetic Alphabet)

THE INTERNATIONAL PHONETIC ALPHABET (revised to 1993)

CONSONANTS (PULMONIC)

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap				ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

CONSONANTS (NON-PULMONIC)

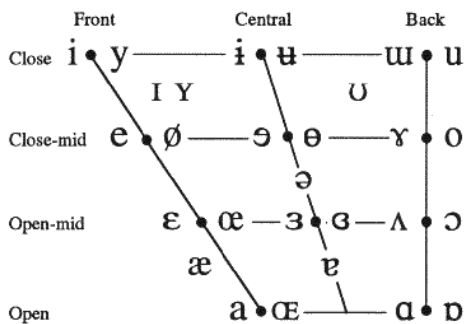
Clicks	Voiced implosives	Ejectives
◌ Bilabial	ɓ Bilabial	ʼ as in:
Dental	ɗ Dental/alveolar	ɓ' Bilabial
! (Post)alveolar	ɟ Palatal	ɗ' Dental/alveolar
≠ Palatoalveolar	ɠ Velar	k' Velar
Alveolar lateral	ɣ Uvular	s' Alveolar fricative

SUPRASEGMENTALS

	TONES & WORD ACCENTS
	LEVEL
ˈ Primary stress	ˈ founəˈtʃən
ˌ Secondary stress	ˌ eɪ
ː Long	eː
ˑ Half-long	eˑ
ˑ Extra-short	e̞
· Syllable break	ii.ækt
Minor (foot) group	è
Major (intonation) group	è̇
˘ Linking (absence of a break)	˘

	LEVEL	CONTOUR
˥ Extra high	˥	˥ or ˨ Rising
˦ High	˦	˥ or ˨ Falling
˧ Mid	˧	˥ or ˨ High rising
˩ Low	˩	˥ or ˨ Low rising
˩ Extra low	˩̇	˥ or ˨ Rising-falling etc.
˩ Downstep	˩↓	˥ or ˨ Global rise
˩ Upstep	˩↑	˥ or ˨ Global fall

VOWELS



Where symbols appear in pairs, the one to the right represents a rounded vowel.

OTHER SYMBOLS

ɱ Voiceless labial-velar fricative	ɕ ʑ Alveolo-palatal fricatives
ɸ Voiced labial-velar approximant	ɺ Alveolar lateral flap
ɸ Voiced labial-palatal approximant	ɧ Simultaneous ʃ and X
ħ Voiceless epiglottal fricative	Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary.
ʕ Voiced epiglottal fricative	
ʔ Epiglottal plosive	

k͡p t͡s

DIACRITICS

Diacritics may be placed above a symbol with a descender, e.g. ɲ̥

◌ Voiceless	◌ Breathy voiced	◌ Dental
◌ Voiced	◌ Creaky voiced	◌ Apical
◌ Aspirated	◌ Linguolabial	◌ Laminal
◌ More rounded	◌ Labialized	◌ Nasalized
◌ Less rounded	◌ Palatalized	◌ Nasal release
◌ Advanced	◌ Velarized	◌ Lateral release
◌ Retracted	◌ Pharyngealized	◌ No audible release
◌ Centralized	◌ Velarized or pharyngealized	
◌ Mid-centralized	◌ Raised	
◌ Syllabic	◌ Lowered	
◌ Non-syllabic	◌ Advanced Tongue Root	
◌ Rhoticity	◌ Retracted Tongue Root	

D. Bangla IPA Chart

Consonants									
	Bilabial	Labiodental	Dental	Alveolar	Retroflex	Palatoalveolar	Palatal	Velar	Glottal
Plosive	p b		t d		ʈ ɖ			k g	
	ph bh		th dh		ʈʰ ɖʱ			kh gh	
Affricate						tʃ ʃʒ			
						tʃʰ ʃʒʱ			
Nasal	m			n				ŋ	
Lateral				l					
Roll/Trill				r					
Flap/Tap				ɾ	ɽ				
					ɽʱ				
Fricative	ɸ β	f v		s z		ʃ			h ɦ
Approximant	ɔ̃						ɛ̃		
Vowels									
							Front	Back	
Close							i ĩ	u ũ	
Close-mid							e ě	o ɔ̃	
Open-mid							æ æ̃	ɔ ɔ̃	
Open							a	ã	