A Bangla Phonetic Encoding for Better Spelling Suggestions

Naushad UzZaman
BRAC University
Dhaka, Bangladesh
naushad@bracuniversity.ac.bd

Dr. Mumit Khan
BRAC University
Dhaka, Bangladesh
mumit@bracuniversity.ac.bd

Abstract

We present a phonetic encoding for Bangla that can be used by spelling checkers to provide better suggestions for misspelled words. The encoding is based on the Soundex algorithm, modified to match Bangla phonetics. We start by analyzing Soundex encoding scheme when applied to Bangla. Next we propose a new encoding that handles the case of Bangla words, including those containing conjuncts. We conclude with a demonstration of a prototype spelling checker that uses this phonetic encoding to offer suggestions for a set of misspelled Bangla words.

Keywords

Bangla phonetic encoding, Soundex, spelling suggestions.

INTRODUCTION

One of the more difficult tasks for a spelling checker is to produce "good" suggestions for misspelled words. While there have been significant research efforts in approximate string matching algorithms for English and other Western languages [1-4], similar work for Bangla has however just begun [5, 6]. An analysis of Bangla misspelled words shows that two of most common reasons for misspellings are (i) phonetic similarity of Bangla characters, and (ii) the difference between the grapheme representation and phonetic utterances [7]. This observation is the primary motivation for creating a phonetic encoding for Bangla that can be used to provide suggestions for misspelled words. While this paper focuses on the spelling checking application, the proposed encoding is equally applicable in a wide range of text-processing applications, from searching for patient records in a medical database to matching names in census records.

The basic idea behind spelling suggestions using phonetic encoding is quite simple:

1. Encode the input word using phonetic coding rules;
2. Look up a phonetically encoded lexicon for words with the same code; and
3. Create an ordered list, i.e., suggestions, from the result using some heuristic.

In this paper, we introduce a phonetic encoding for Bangla, and then demonstrate how a spelling checker would use it to produce suggestions for misspelled Bangla words. We assume that the Bangla text is encoded using Unicode Normalization Form C (NFC) [8], with its consistent logical ordering of the consonants and the dependent vowels, as well as of the large repertoire of the *juktakkhors* (compound letters or conjuncts) in Bangla..

PHONETIC MATCHING TECHNIQUES

A major class of approximate string matching algorithms is the various phonetic methods, from the eighty-year old Soundex [9, 10], to the more recent Metaphone [11, 12] and PHONIX [13]. The input to these phonetic encodings or "sound-alike" algorithms is a word, and the result is an encoded key, which should be the same for all words that are pronounced similarly, allowing for a reasonable amount of fuzziness. 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. Strings that sound similar compare equal in their respective encoded form. 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.

Of these 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 an 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. The Soundex algorithm itself, shown in Figure 1, transforms all but the first letter of each string into the code, 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 coding rules

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

1. Replace all but the first letter of *s* by its phonetic code.
2. Eliminate any consecutive repetition of codes.
3. Eliminate all occurrences of code 0 (i.e., eliminate vowels, and the letters *H*, *W* and *Y*).
4. Return the first four characters of the resulting string.

Figure 1: The Soundex algorithm

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.

Table 2: PHONIX coding rules

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

The Metaphone algorithm 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.

A drawback of these algorithms, as we pointed out earlier, is that these are language-specific, and typically designed for the English language. There have been attempts to modify these algorithms for other European languages, such as Spanish, Polish and Portuguese, but not for Bangla to the best of our knowledge. Using recent research on machine learning methods for letter-to-phoneme conversion [14, 15], application of these techniques to Bangla should be straightforward, provided that there is sufficient training data. As the first step in defining a comprehensive phonetic matching technique for Bangla, we describe an algorithm based on the widely used Soundex algorithm, suitably modified to reflect Bangla phonetics.

BANGLA PHONETICALLY SIMILAR 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 [5]. When checking the spelling of a word, we first search for its encoded version in the modified dictionary. If the entry exists, then either the original exists in the list of words corresponding to this code (in which case, the spelling is correct), or the word is misspelled and the list is offered as the set of alternatives for the original word. If the entry does not exist, then the alternatives must be suggested using one of the edit-distance algorithms (e.g., Levenshtein [15]). However, this technique does not work if an extra error occurs in the spelling, so this technique must be used with a edit-distance algorithm to be effective in a spelling checker. See [2] for a summary of the various commonly used edit-distance algorithms. This technique works for Bangla conjuncts as well, but only if we eliminate the *hasant* character from our encoded strings.

Soundex Algorithm for Bangla

The Soundex algorithm, unmodified, presents a set of difficulties when used in a Bangla spelling checker. In this section, we present some of the prominent issues.

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 [15]) 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 [2] 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 a 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* chraracter, 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, ন্ন (ণ্ 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.

Phonetic Matching Technique for Bangla

Table 3 shows the proposed Bangla phonetic codes with Letter, Name (according to unicode found at [19]) and Unicode number (from [19]) & Figure 2 shows the modified Soundex algorithm using this encoding, suitable for a Bangla spelling checker.

Table 3: Bangla phonetic coding rules

Code	Letter	Name	Unicode
0 (zero)	্	Virama/Hasant	“09CD”
Not	ো	Sign O	“09CB”
Coded	ঁ	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”
	ঊ	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”
	র	RA	“09B0”
	ড়	RRA	“09DC”
	ঢ়	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, candrabindu, sign O*).
3. Return the resulting string.

Figure 2: The Soundex algorithm for Bangla

SUMMARY OF SOUNDEX FOR BANGLA

Transformations

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

Vowels: 5 codes

Consonants: 17 codes

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 a conjuncts. And individually has No Sound in words.

This will also reduce one extra character error. For example, if someone miss the ্, then its 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: দগ্ধ = 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.

Example of Error Correction Using Phonetic Matching

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

Table 4: Suggestions for misspelled words

Input	*Encoded*	*Suggestion*
খুমাড়	kumar	কুমার
পাসান	pasan	পাষাণ
দগধ	dgd	দগ্ধ (দগ ্ ধ)

CONCLUSION

We present a preliminary effort at creating a phonetic matching encoding for Bangla based on Soundex, and tailored for a spelling checker. We describe a prototypical phonetic coding rules and the associated algorithm that produces "good" suggestions for misspelled Bangla words. There are however many complex spelling rules from [17, 18] that are not yet addressed in this encoding, such as

1. The use of Folas (i.e., BA fola, MA fola, YA fola, RA fola, LA fola).
2. Conjuncts with unusual pronunciations. (i.e., ক্ষ, ঙ্গ, etc.) Lets consider ক্ষ.

ক্ষ = ক+্ +ষ ; ক্ষত, sounds as খত. Should be encoded as kt. But in our encoding it will be encoded as kst.

3. Different pronunciation on different context. Lets consider again ক্ষ.

At the beginning: Sounds as খ. So it should be encoded to k. (ক্ষত->খত->kt) .

At the middle / end: Sounds as কখ. So it should be encoded to kk. (দক্ষ->দকখ->dkk).

The approaches used by PHONIX and Metaphone variants do provide some context, and our future work in this area will concentrate on creating transformation maps to reduce strings to canonical forms before the table-driven encoding step.

ACKNOWLEDGMENT

This work has been partially supported through PAN Localization Project (www.PANL10n.net) grant from the International Development Research Center, Ottawa, Canada, administered through Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.

REFERENCES

1. K. Kukich, "Techniques for automatically correcting words in text", *Computing Surveys*, 24, (4), 377–440, (1992).
2. J. Zobel and P. Dart, "Finding Approximate Matches in Large Lexicons", *Software - Practice and Experience*, 25(3), pp. 331-345, March, 1995.
3. F. Damerau, "A technique for computer detection and correction of spelling errors", *Communication of the ACM*, 7(3), pp. 171-176, 1964.
4. V. Hodge and J. Austin, "A Novel Binary Spell Checker", *Proc. International Conference on Artificial Neural Networks*, Vienna, August, 2001.
5. B. B. Chaudhuri, "Reversed word dictionary and phonetically similar word grouping based spell-checker to Bangla text", *Proc. LESAL Workshop*, Mumbai, 2001.
6. 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.
7. P. Kundu and B.B. Chaudhuri, "Error Pattern in Bangla Text", *International Journal of Dravidian Linguistics*, 28(2), 1999.
8. The Unicode Consortium, *The Unicode Standard, Version 4.0*, Addison-Wesley, 2003. Also available online at http://www.unicode.org/versions/Unicode4.0.1.
9. D. E. Knuth, *The Art of Computer Programming*, Vol. 3, Addison-Wesley Publishing Company, Reading, Massachusetts, 2nd edition, 1982.
10. The Soundex Algorithm, available online at http://www.archives.gov/research_room/genealogy/census/soundex.html.
11. Lawrence Phillips, "Hanging on the Metaphone", *Computer Language*, 7(12), 1990.
12. Lawrence Phillips, "The Double Metaphone Search Algorithm", *C/C++ Users Journal*, 18(6), June, 2000. Also available online at http://www.cuj.com/documents/s=8038/cuj0006philips/ .
13. T. N. Gadd, "PHONIX: The Algorithm", *Program*, 24(4), pp. 363-366, 1990.
14. W. M. Fisher, "A statistical text-to-phone function using n-grams and rules", *Proc. ICASSP-99, the 1999 IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 649-652, March 1999.
15. K. Toutanova and R. C. Moore, "Pronunciation modeling for improved spelling correction", July 2002.
16. V. L. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals", *Soviet Physics Doklady,* 10, 1966.
17. Bangla Uccharon Obidhan, Bangla Academy, Dhaka, Bangladesh.
18. Bangla Banan Obidhan, Bangla Academy, Dhaka Bangladesh.
19. Bangla Unicode Chart, available online at http://www.unicode.org/charts/PDF/U0980.pdf