A LINGUISTICALLY SORTABLE BENGALI CODING SYSTEM AND ITS APPLICATION IN SPELL CHECKING: A CASE STUDY OF MULTILINGUAL APPLICATIONS

M. MANZUR MURSHED and **SYED M. RAHMAN**
Gippsland School of Computing & Information Technology, Monash University
Gippsland Campus, Churchill VIC 3842, Australia

M. KAYKOBAD
Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh

1 INTRODUCTION

In the last decade, the use of Bengali scripts in daily computer usage has gained wide acceptance in Bangladesh. Although a wide range of commercial Bengali software have been developed so far to meet the ever-growing demand in the local market, a systematic and scientific efforts of integrating Bengali in modern computing systems remains in its infancy. One of the most important issues related to Bengali computing is to sort Bengali texts in order of linguistic order.

Sorting Bengali words is not same as sorting English words in ASCII. There are a number of coding schemes available in the market. Although Bangladesh Standards and Testing Institution [1] has recently standardized a coding scheme to be used in all future Bengali text processing, the other existing coding schemes are likely to dominate for a few more years because of the availability of popular commercial software. Unlike English, Bengali letters can take more than one forms and two or more letters are sometimes combined into compound characters. Although a coding scheme should assist in both displaying characters as well as sorting texts efficiently, all of the available Bengali coding schemes, including the standard one, have placed emphasis more on displaying Bengali texts as close as possible to the actual forms.

It was surprising to learn that no Bengali coding scheme was available which could sort Bengali texts in complete linguistic order until we have recently proved in [2] that no completely linguistically sorted Bengali coding scheme exists. In [2], we have further introduced an *internal coding scheme*, in addition to the primary coding scheme, to provide a linguistically sortable Bengali coding system. The internal coding scheme is made linguistically sortable by avoiding all the compound characters and introducing some artificial half-characters to compensate that. The compound letters of the primary scheme is supported by providing non-lossy transformation from the primary coding scheme to the internal scheme and vice versa.

Rahman and Iqbal [3] have recently developed a very similar sorting algorithm using lossy transformation from the primary coding scheme to the internal scheme. The difference between lossy and non-lossy transformations lies on the fact that the later transformation is reversible, i.e., non-lossy transformation can also be used to convert a word from the internal coding scheme to the primary coding scheme.

By developing an inter-scheme text conversion utility, we have established in [2] that use of non-lossy transformation instead of lossy transformation for sorting Bengali texts in linguistic order has some extra benefit. In this paper we discuss another very important application of non-lossy transformation by developing an efficient spell checking application for Bengali texts based on the internal coding scheme with non-lossy transformation. As usual, the handling of compound letters

remains the key area where a Bengali text speller differs from its counterparts in other languages. Here we establish that using of the internal coding scheme in designing the dictionary and developing suggestion generating search engine not only provides a spell checking solution which is independent of any specific primary coding scheme but also assists in designing layered solution for efficient modularization and maintenance of coding.

This paper is organized as follows. In the next section we present the basic properties of Bengali script. For the sake of completeness, some results and algorithms on sorting Bengali texts in linguistic order, developed in [2], are given in Section 3. In Section 4 we discuss various issues of developing an efficient primary coding scheme independent spell checking application based on our solution to linguistically sorting Bengali texts. Section 5 concludes the paper.

2 PROPERTIES OF BENGALI SCRIPT

Bengali alphabet is divided, like most languages, into vowels and consonants. Like many South Asian languages, all of the vowels in Bengali have two forms. The selection of forms depends on the purpose of using a vowel in a word. If a vowel sounds independently, the principal form of that vowel is used and if a vowel guides the sound of a consonant (or a group of consonants), a different form of that vowel is used. Let this non-principal forms of vowels be called the *half-vowels*. Among the half-vowels some are written on the left (*left-biased*) and some are on the right (*right-biased*) of the associated consonants. There are a few half-vowels which are written in two parts – one part on the left and the other part on the right of the associated consonants. Let these half-vowels be called the *both-biased* half-vowels. Both left-biased and both-biased half-vowels not only make Bengali unique among the South Asian languages but also create many problems, as revealed in the next section, while introduced to computing systems.

Like many Asian languages, two or more consonants can be grouped together to form a compound consonant. A compound consonant is not always just the juxtaposition of the group of consonants in their principal forms. By introducing one or two different forms of some consonants besides the principal form, it is possible to display a number of compound consonants by concatenation. Let these non-principal forms of consonants be called the *half-consonants*. But this technique of using half-consonants will not work to display a few compound consonants that have no resemblance with the group of consonants they represent. In such cases, a distinct letter has to be used which also demands distinct position in the alphabet.

3 SORTING BENGALI TEXTS IN LINGUISTIC ORDER

Sorting plays significant role, both explicitly and implicitly, in every text processing systems. When a language is introduced to a computing system, the letters of that language are given specific numerical codes to represent them into the system. A list of these numerical codes is known as the *coding scheme* of that particular language, e.g., ASCII is one of the English coding schemes. A coding scheme includes all forms of letters so that it can be used to display text in the exact way it is usually written, e.g., ASCII includes both the upper-case and the lower-case English letters. It is thus obvious that a Bengali coding scheme must include not only the principal letters but also all the half-vowels and the compound consonants.

A huge number of coding schemes can be defined for a particular language, if representation of letters into the computing system is the only concern. But that is not the case in general. In fact a coding scheme forms the base knowledge for any computing system to sort texts. Most of the computing systems sort texts according to the collating sequence of the codes in the coding scheme. The users of a computing system never bother how the system performs sorting; what they care most is that sorted texts produced by the system follow linguistic order of the language. The task of sorting texts linguistically can be extremely simplified if the collating sequence of the coding scheme either follow the linguistic order completely or embed rules so that the linguistic order can be derived from the partially ordered scheme. This significantly reduces the number of possible coding schemes for a particular language. Although it is possible to define completely linguistically ordered English coding scheme, ASCII is a partially linguistically ordered set and relies on mapping upper case letters to the lower case ones, or vice versa, for producing linguistic (case insensitive) sorting of English texts.

There are two problems associated with defining a completely linguistically ordered Bengali coding scheme. First, a left-biased half-vowel or the left part of a both-biased half-vowel appears one or more positions ahead of its actual linguistic position. Second, a compound consonant that is represented by a distinct letter in the coding scheme actually represents two or more consonants. Any of the above two problems is enough to reject the existence of any completely linguistically ordered Bengali coding scheme. So, we can define, at best, a partially linguistically ordered Bengali coding scheme.

With a partially ordered coding scheme, the problem with left-biased and both-biased half-vowels can be overcome by using the fact that the actual position of a left-biased or the left part of a both-biased half-vowel can be calculated by checking the consonant to the right of it. If the consonant is a non-compound consonant or a compound consonant with distinct form, the half-vowel should be moved only one position to the right so that it is now positioned just after the consonant. If the consonant is a compound consonant that is formed by concatenation of two or more half-consonants, the half-vowel should be moved to right two or more positions so that it is now positioned just after the last half-consonant.

The above solution to handle left-biased and both-biased half-vowels need no extra information and therefore, can be considered as an embedded rule for a partially linguistically ordered coding scheme.

Unfortunately the problem with compound consonants with distinct position in the partial coding scheme cannot be solved without adding extra information to the scheme. To achieve complete linguistic order every compound consonant with distinct position must be replaced by its component consonants in any forms and there is no way to guess the components of a compound consonant simply from the coding scheme itself. To sort Bengali texts in linguistic order we must incorporate information regarding the components of the compound consonants.

In the long absence of a standard Bengali coding scheme, a number of different coding schemes gained market shares. Although recently Bangladesh Standards and Testing Institution has defined the Standard Bengali Coding Scheme [1], the non-standard coding schemes will still dominate the market, for at least a few more years, because of the availability of popular software. So, our solution to sort Bengali texts in linguistic order should not be confined with any specific coding scheme. One of the possible methods to sort Bengali texts in linguistic order is discussed below:

This method works with any coding scheme. Here, we first create artificial half-consonants for all the principal consonants. As pointed out in Section 2, a number of half-consonants are already available in the coding scheme. So, we need to introduce artificial ones only for those consonants for which no half-letter is available in the scheme. An *internal coding scheme*, in addition to the original

coding scheme, is defined which includes all forms of vowels, the principal consonants, and the half-consonants and excludes all the compound letters. The internal coding scheme is defined in partially linguistic order where each half-letter takes the position just after the corresponding principal letter. The order of the primary coding scheme thus plays no role in sorting.

A non-lossy transformation is defined which replaces every compound letter with its component half-consonants in sequence. This transformation uses a conversion table where compound letters of the primary scheme are used as key and the concatenated half-consonant components of the compound letters are stored in the second field. This transformation is called non-lossy as the original compound letters can always be retrieved from the converted texts. It is obvious that different conversion tables are required with different primary coding schemes.

Using a non-lossy transformation is not a necessary condition for deriving complete linguistic order from a partially ordered scheme e.g., in ASCII if a upper case letter is converted to a lower case one, the case information is lost unless a copy is retained. But having non-lossy property has some extra benefits as the transformation can be used in both directions, converting text from primary coding scheme to the internal coding scheme and vice versa. One of the applications is to convert Bengali texts from one primary coding scheme to another primary coding scheme. The text is first converted into the internal scheme from the first primary scheme by forward mapping, which is then again converted into the second primary scheme using reverse mapping.

In the next section we discuss various issues of developing a Bengali spell checking utility based on the above coding system. As the above coding system is independent of the primary coding scheme, the spell checker should also be able to check spelling of Bengali texts written in any of the available primary coding schemes.

4 SPELL CHECKING OF BENGALI TEXTS

A typical spell checking utility has two important components – a dictionary of words and a search engine to generate suggestions for misspelt words. The spell checking of a text document is done in the following way:

The text is scanned word by word, starting from the beginning. For each word in the text, it is looked up in the dictionary. If the word is found in the dictionary, nothing needs to be done. But if the word is not found in the dictionary, two things may happen. First, the word may be a correct word which is not included in the dictionary (this suggests for a mechanism to add words in the dictionary). Second, the word may indeed be a misspelt word. In both the cases, the search engine should be able create new valid words, which differ from the original word slightly, as suggestions. These suggestions should be presented in linguistic order so that a user can select quickly.

The design of a dictionary should consider not only storing of all the possible words in the language in an efficient manner but also providing a way to look up whether a word is in the dictionary extremely fast. Two of the popular data structures used in dictionary design are B^+ trees and hash tables.

Developing a spell checking utility, independent of primary coding scheme, also demands that the word of the dictionary should not be stored in any particular primary coding scheme. We have two options – either to keep separate dictionaries for each of the primary coding schemes or to keep a scheme independent dictionary. If we consider the enormous size of a single dictionary that has to include most of the words of a language, the second option must be the preferred one.

The strength of the internal coding scheme with non-lossy transformation of compound consonants, developed for sorting Bengali texts in linguistic order, can again be realized when we use this scheme to design a Bengali dictionary, independent of any specific primary coding scheme. Checking of a word in any primary scheme is first converted into the internal scheme by forward mapping. This converted word is then looked up in the dictionary. If the word is not in the dictionary, the search engine then produces a number of suggestions in internal coding scheme. Each of these suggestions is then converted back to the primary coding scheme using reversed mapping. Because of this reverse mapping, the non-lossy transformation of compound consonants is essential.

Why should the suggestion generating search engine be primary coding scheme independent? Before answering to this question we need to understand the underlying technique of generating effective suggestions. Given a misspelt word, a set of words is generated by replacing a letter with another one, by inserting a letter, by deleting a letter, and by swapping successive letters. All the words in this set is very close to the misspelt word in the sense that difference between on of these words and the misspelt word is within one or at most two characters. So, this set represents the very best candidate for suggestion. The only problem with the set is that not all the words in it are valid. The solution lies in looking up each of the words in this set in the dictionary, and retain only those as suggestions which are found in the dictionary.

If the set of possible suggestions is generated in a specific primary scheme, each of these possible suggestions needs to be converted into the internal coding scheme before looking at the dictionary. This conversion is not necessary if the suggestion pool is generated in the internal coding scheme. On the contrary, we do not need to convert the suggestions from the internal coding scheme to a specific primary scheme, if the suggestions are already in primary coding scheme. But choosing a primary coding scheme independent suggestion generator lies elsewhere. The number of possible suggestions is much higher than that of actual suggestions. So, mapping time with primary scheme independent solution is insignificant.

Besides this time efficiency, primary scheme independent solution also allows us to implement the spell checking utility into two separate layers for better modularity and maintenance. The top layer deals with all primary coding scheme dependent codes e.g., code conversions. This layer should not be a new one. All the codes of our solution in sorting Bengali texts in linguistic order can be reused. The bottom layer deals with all primary coding scheme independent codes e.g., the dictionary, the suggestion search engine, and sorting of suggestions.

5 CONCLUSION

So far sorting Bengali texts in linguistic order remained a challenging task.. In [2], we have shown for the first time that a completely linguistically ordered Bengali coding scheme does not exist. By introducing an internal coding scheme along with the primary coding scheme and non-lossy transformation from the primary coding scheme to the internal one and vice versa, we have further developed an algorithm to sort Bengali texts in linguistic order. In this paper, we have discussed a very significant added benefit to our solution of sorting in linguistic order. Here, we have presented ideas of developing a spell checking utility for Bengali texts based on the internal coding scheme and non-lossy transformation used in solving the sorting problem. A commercial spell checker, developed by one of the author, based on the ideas presented in this paper is currently enjoying the lions share of the local market. Because of the proprietary nature of this commercial product, we refrain from detailing any implementation issues.

ACKNOWLEDGMENT

We wish to thank Rezwan Al Bakhtiar, A. T. M. Zakaria Swapan, Faisal Ahmed, and Badrul Munir Sarwar for their valuable comments and suggestions.

REFERENCES

[1] *Bangladesh Standards in Coding Bengali Alphabet*. Bangladesh Standards and Testing Institution, 1996 (in Bengali).

[2] M. M. Murshed and M. Kaykobad. Linguistically Sorting Bengali Texts: A Case Study of Multilingual Applications. *Proceeding of the 9th International Conference of the Information Resources Management Association*, Boston, Massachusetts, USA, pp.795—797, 1998.

[3] M. S. Rahman and M. Z. Iqbal. Bangla Sorting Algorithm: A Linguistic Approach. *Proceedings of International Conference on Computer and Information Technology (ICCIT'98)*, Dhaka, Bangladesh, pp.204—208, 1998