LINGUISTICALLY SORTING BENGALI TEXTS: A CASE STUDY OF MULTILINGUAL APPLICATIONS

M. MANZUR MURSHED*
Computer Sciences Lab, Research School of Information Sciences & Engineering
The Australian National University, Canberra ACT 0200, Australia
E-mail: murshed@cslab.anu.edu.au, Tel: +61-2-6279-8636, Fax: +61-2-6279-8651

M. KAYKOBAD
Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh
E-mail: kaykobad@csebuet.agni.com, Tel: +880-2-9665612, Fax: +880-2-863026

1 INTRODUCTION

In the last decade, the use of Bengali scripts in daily computer usage has gained wide acceptance in Bangladesh. Wide ranges of Bengali software have been developed so far to meet the ever-growing demand in the local market. From the very beginning, Bangladeshi software developers followed two different paths. One group started writing software from the scratch, while the other group tried to embed Bengali scripts in popular international software. But it is now well established that due to the limited market size and massive development and upgrading cost involved in writing software from the scratch, embedding Bengali scripts is the most feasible way.

This paper focuses primarily on developing a Bengali scripting system capable of sorting Bengali texts linguistically. Although the solution presented in this paper puts no restriction over the method of implementation, we have preferred, for obvious reasons, to embed our solution in Microsoft Windows.

Here, in this paper, we have proved that no completely linguistically sorted Bengali coding scheme exists. We have further proved that it is also not possible to define any rule to derive the complete linguistic order from any partially linguistically ordered Bengali coding scheme. Based on the nature of the mapping functions, whether any information is lost in transformations or not, two solutions are suggested. Both of the solutions employ conversion tables to handle the complexity associated with the compound letters. In the second solution we have introduced an internal coding scheme, in addition to the conventional coding scheme, to provide non-lossy transformations. This solution gives us some extra benefits. Bengali texts, written in a completely unordered coding scheme, can now be sorted. Moreover, based on the fact that non-lossy transformations are reversible, we have developed an application to convert Bengali texts among different coding schemes.

This paper is organised as follows. In the next section we present the basic properties of Bengali script as well as the definition of the problem of sorting Bengali texts in linguistic order. Solutions to this problem are given in Section 3. In Section 4 we briefly describe the implementations of our solutions. An additional application of one solution is presented in Section 5. Section 6 concludes the paper.

2 PRELIMINARIES

For the sake of completeness, we briefly describe the properties of Bengali script and give the definition of the problem of linguistically sorting Bengali texts.

2.1 Properties of Bengali Script

Bengali alphabet is divided, like most languages, into vowels and consonants. Like the European languages, Bengali is written from left to right direction. Although mostly the principal letters are included in the alphabet, but like most of the Asian languages, a large number of compound letters are widely used in Bengali. In most of the cases, a compound letter is not just the juxtaposition of some principal letters and thus it demands distinct position in the alphabet.

Like many South Asian languages, all the vowels in Bengali have two forms. When a vowel is used to guide the sound of a principal or a compound consonant, a different form of letter, other than the principal letter of that vowel in the alphabet, is written. Let this forms of vowels be called *half-letters*. Although the half-letters are written on either the left or right or top or bottom of the associated consonant, a particular half letter is always written in the same position w.r.t. the consonant it guides. The half-letters which are written on the left of the associated consonants not only make Bengali unique among the South Asian languages but also create problems, as revealed in the next section, while introduced to a computing systems. Let these half-letters be called the *left biased half-letters*.

2.2 Problem Definition

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. ASCII is one of the English coding schemes. A coding scheme includes the principal letters in the alphabet as well as all the forms of the letters so that it can be used to display text in the exact way it is written *viz*. ASCII includes both the upper-case and lower-case English letters. It is thus obvious that a Bengali coding scheme then must include not only the principal letters but also all the compound letters as well as the half-letters.

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

* Corresponding author.

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 case insensitive sorting of English texts.

The following theorem not only makes a clear impression of the difficulty of the problem of sorting Bengali texts in linguistic order but also suggests a way to solve the problem.

Theorem 1 *It is not possible to define a Bengali coding scheme which either follows the linguistic order completely or embeds rules so that the complete linguistic order can be derived from the partially ordered scheme.*

Proof. It is obvious that the left biased half-letters makes it impossible to define a completely linguistically ordered Bengali coding scheme. But definitely a partially linguistically ordered scheme can be used with an associated mapping function which swaps every left biased half-letter with the associated consonant. So, suppose a partially linguistically ordered Bengali coding scheme exists with embedded rules so that the complete linguistic order can be derived from that scheme. Now, any compound letter contradicts the existence of any rule to derive its position in the linguistic order as the scheme contains no information regarding the components of that compound letter. ′

3 THE SOLUTIONS

The proof of Theorem 1 suggests that to sort Bengali texts in linguistic order we must incorporate information regarding the components of the compound letters. The nature of the mapping functions also plays important role in deriving solutions. If lossy functions like the case changing functions on ASCII, where information of the case of the letters is lost, are used the solution becomes little bit easier. But using non-lossy functions enables us to sort Bengali texts written in any arbitrarily ordered coding scheme and to develop additional application as covered in Section 5.

3.1 Lossy Mapping

In this method a partially linguistically ordered coding scheme is defined where at least the principal letters are in linguistic order. The rest of the solution depends on defining the following mapping functions:

i) A non-lossy mapping function is defined to swap every left biased half-letter with its associated consonant.

ii) A lossy mapping function is defined which replaces every compound letter with its components in sequence. This function uses a conversion table where compound letters are used as key and the concatenated components of the compound letters are stored in the second field.

3.2 Non-lossy Mapping

This method works with any coding scheme. Here, we first create artificial half-letters for all the principal consonants. An *internal coding scheme*, in addition to the original coding scheme, is defined which includes all forms of vowels, the principal consonants, and the artificially created half consonants and excludes 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. The rest of the solution depends on defining the following mapping functions:

i) A non-lossy mapping function is defined to swap every left biased half-letter with its associated consonant.

ii) A non-lossy mapping function is defined which replaces every compound letter with its sequenced components in half-letter forms. This function uses a conversion table where compound letters of the primary scheme are used as key and the concatenated half-letter components of the compound letters are stored in the second field. The mapping function 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.

4 IMPLEMENTATIONS OF THE SOLUTIONS

The stand-alone implementations of the solutions in Section 3 are straightforward. Any Bengali software, which is developed from the scratch, should be able to incorporate our ideas easily with little effort. But our prime interest lies in embedding the solutions in a popular international computing system. For obvious reason we have preferred the Microsoft Windows system.

A Dynamic Data Exchange (DDE) [1] server application is written to provide the necessary mapping functions. Any Windows based software, capable of connecting itself as a DDE client, can now be able to linguistically sort Bengali texts by using that DDE server as shown in Fig. 1.

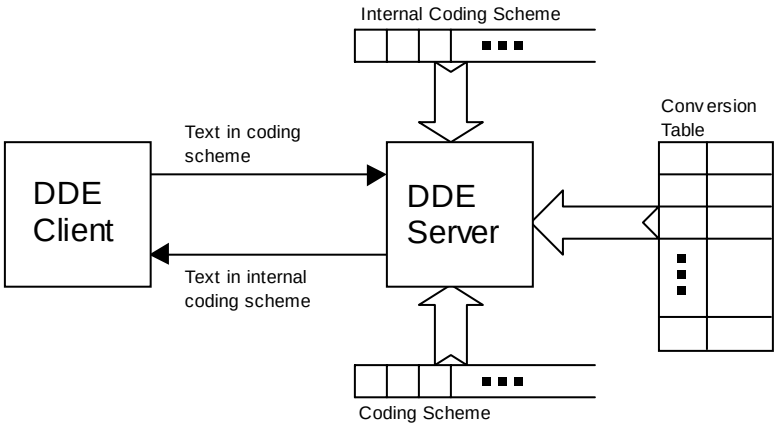

Fig.1: DDE Client-Server interactions during sorting Bengali texts.

5 MORE APPLICATION OF THE SOLUTIONS

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 [2], the non-standard coding schemes will still dominate the market, for at least a few more years, because of the availability of popular software. Even if all the non-standard schemes are eliminated from the market, an enormous volume of works based on these schemes will become obsolete unless these are converted to the new standard scheme. Thus, an intra-scheme text conversion application seems to be a perfect solution.

5.1 Inter-Scheme Text Conversion

Lossy mapping functions are sufficient to provide facilities to sort Bengali texts linguistically. But using non-lossy mapping functions has some extra benefits as they can be used in both directions. In fact we have developed an intra-scheme text conversion application based on the non-lossy mapping. Separate conversion table is provided to the DDE server system for each individual coding scheme. To convert a piece of text ***T*** in coding scheme ***A*** into the text in coding scheme ***B***, ***T*** is first converted into the text in internal coding scheme. This text in internal coding scheme is then again converted into the text in coding scheme ***B*** as shown in Fig. 2.

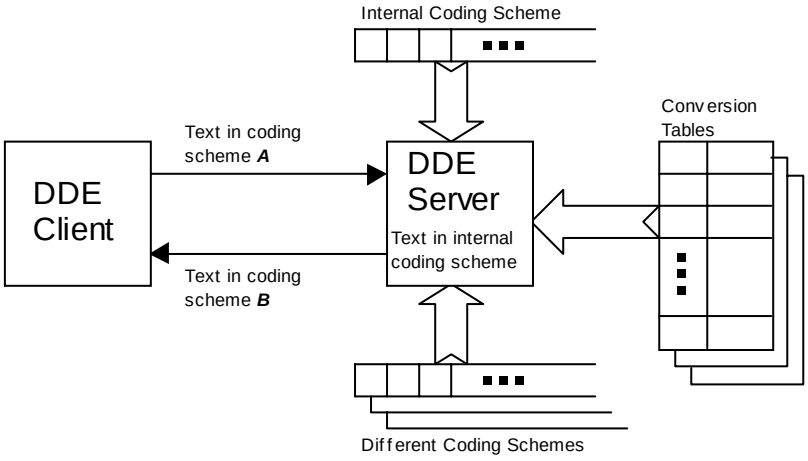

Fig.2: DDE Client-Server interactions during Bengali text conversion.

6 CONCLUSION

So far sorting Bengali texts in linguistic order remained a challenging task. Here, in this paper, we first have shown that neither a completely linguistically ordered Bengali coding scheme exists nor it is possible to define any rule to derive the complete linguistic order from any partially linguistically ordered Bengali coding scheme. By utilising conversion tables to overcome the complexity associated with the compound letters, we thus have proposed two solutions to sort Bengali texts in linguistic order. The first solution uses lossy transformation while the second one uses non-lossy transformation.

Based on the reversible nature of lossy transformation, we further have developed an inter-scheme text conversion utility.

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] Microsoft Windows 3.1 Programmer's Reference, Volume 3: Messages, Structures, and Macros, Redmond WA: Microsoft Press, 1992.

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