Mapping Bangla Unicode Text to Keyboard Layout Specific Keystrokes

Article · January 2009

CITATIONS
0

READS
30,058

3 authors, including:

Mohammad Reza Selim
Shahjalal University of Science and Technology
5 PUBLICATIONS **21** CITATIONS
SEE PROFILE

Sabir Ismail
Stony Brook University
24 PUBLICATIONS **100** CITATIONS
SEE PROFILE

SUST Studies, Vol. 12, No. 1, 2010; P: 32-39

Mapping Bangla Unicode Text to Keyboard Layout Specific Keystrokes

Mohammad Reza Selim[1], Mahbubur Rub Talha[2] and Sabir Ismail[2]

[1,2] *Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet, Bangladesh*

Emails: [1]*selim@sust.edu,* [2]*{talha13_syl, oshorere}@yahoo.com*

Abstract

Bangla Unicode texts are becoming increasingly available from different types of sources. It is necessary for many applications to map these Unicode texts to keystrokes of a particular keyboard layout. In this paper, we analyze Bangla Unicode texts and various keyboard layouts and device algorithms to convert the Unicode texts to keyboard layout specific keystrokes. We implement and test our algorithms with sufficient data and show that they are working correctly. Our work has many applications, e.g., comparing the efficiency of keyboard layouts, predicting a word being typed to improve typing efficiency, improving the accuracy of spelling suggestions generated by a Bangla spell checker, etc.

Keywords: Keyboard Layout, Keystrokes, Bangla Unicode, Syllable.

1. Introduction

To type Bangla texts in an editor, Bangla keyboard interfaces, like Bijoy [9], Proshika [10], Proboton [14], Ekusheyr Shadhinota [15], Avro [4], etc., are used. When a key is pressed, the key code is captured by the interface. The captured codes are mapped to target codes and sent to the editor. For example, if *amader* is typed in an editor, while the Avro phonetic keyboard interface is active, it maps the sequence of keystrokes *amader* to sequence of (Uni)codes *2438 2478 2494 2470 2503 2480* which represent respectively the Bengali characters, আ ম ◌া দ ◌ের. Each keyboard interface is related to a *keyboard layout* which defines which key on the keyboard represents which Bengali character, i.e., the (Uni)code. For most of the layouts, such mapping from keystrokes to target codes is not direct one-to-one.

The generated codes by the previously used keyboard interfaces were not standardized, i.e., which code represented which Bangla character was different from interface to interface. This raised a big compatibility problem. For example, the Proshika file was not compatible with Bijoy file unless it was converted. However, all newly released keyboard interfaces follow the Unicode standards, i.e., the keystrokes are mapped to Bangla Unicode texts which are compatible from one interface to another. Because of the standardization, huge Bangla Unicode text will be prevailing in near future. For example, online version of most of the Bangla newspapers have already been started to adopt the Unicode standard. The huge collection of Bangla Unicode text will be used in many ways. Therefore, the analysis and interpretation of the Bangla Unicode text is important.

There have a number of Bangla keyboard layouts. We have categorized them into three types. In *Standard Layouts* we type a word in traditional style as if we are writing it on a paper. Such layouts are the oldest among all types. Using this type of layouts, if we want to write দেশ, we have to press key for ◌ে then for দ and finally for শ. Bijoy [9], Munir [12], Lekhoni, Gitanjali, Satyajit etc. are such type of layouts. In *Unicode Layouts* (sometimes called *Bangla Phonetic Based Layouts* [15]) the dependent vowels are typed after the consonant it modifies. For example, to write দেশ we have to press key for দ, then for ◌ে and finally for শ. UniBijoy (modern style typing) [5], Bangla Unicode [22], National [11], Munir Unicode [12], UniJoy [21], Baishakhi [23], etc are such type of layouts. *Transliteration Based Layouts* use the English to Bangla transliteration scheme [16, 17, 18, 19]. It uses the English QWERT layout. Using such layouts, if we want to

write দেশ, we have to type word like *desh* or *deS*, which will then be converted automatically to দেশ. Avro Phonetic [4], Kickkeys [24], etc., are such type of layout.

Since the number of Bangla keyboard layouts is not very few, discussing our work for each of them is not possible within the short scope of this paper. We, rather, select the most popular layout from each category discussed above, i.e., Bijoy from the first category, UniBijoy from the second category and Avro Phonetic from the third category, and discussed our work for each of them.

The function of a keyboard interface is to map the keystrokes to Bangla texts (i.e., the Unicodes). But the main theme of this paper is the reverse mapping, i.e., conversion from Bangla Unicode text to keystrokes for different keyboard layouts. For example, if we are given the Unicode text প্রযুক্তি (i.e., *2474 2509 2480 2479 2497 2453 2509 2468 2495*), our goal is to design algorithms to produce the keystrokes *prozukti, rzwsdjgk* and *rzwsjgkd* for Avro phonetic, Bijoy and UniBijoy layouts respectively.

The rest of the paper is organized as follows. Section 2 describes our motivation to do this work. Section 3 analyzes the Unicode text in order to device the algorithms. Section 4 describes the design and development of algorithms. Section 5 and 6 contains the experimental results and discussion on that results, respectively. Finally, Section 7 concludes our work.

2. Motivation

Why is such mapping from Bangla Unicode text to keystrokes important? The reason is that it has many applications. Some of them are discussed bellow.

Since there have a number of Bangla keyboard layouts, Bengali computer users are often confused which layout is the most efficient or which will let them to be most productive. Therefore, comparing the typing efficiencies of the layouts is an important work. Besides, such comparison is also necessary to design a new keyboard layout. To compare the efficiencies, measuring and comparing the average number of keystrokes per word or per character is essential. For this purpose, conversion from Bangla text to keystrokes is necessary.

While a Bangla word is being typed, before finishing typing, if it can be predicted correctly, it can help speed up the typing process. To predict a word being typed, a dictionary of Bangla words is necessary. However, if the words in the dictionary are in Unicode format only, it is not possible always to predict the word correctly for all types of keyboard layouts, especially for the transliteration based layouts. This is because the mapping between the keystrokes and Bangla words is not one-to-one always. Therefore, a layout specific keystroke version of the words in the dictionary can help predict the world correctly.

Unlike English language, Bangla does not have upper and lower case alphabets. Yet, since Bangla has a large number of alphabets, for the shortage of keys, different keyboard layers are used. Switching from one layer to another slows down the typing speed greatly. However, we think that it is possible to provide some intelligence to the keyboard interfaces which will allow typing most of the Bangla words without switching keyboard layers. To provide such intelligence a layout specific keystroke-based dictionary of Bangla words is necessary.

Existing Bangla spell checkers do not distinguish layout specific mistakes. For example, in standard layouts, a common mistake is not to type the character Hasant between the characters of a conjunct (কিন্তু). In transliteration based layouts, the default vowel অ is often forgotten to type between two consonants (e.g., একটি is often typed as *ekTi* but the correct one is *ekoTi*). The currently available spell checkers' suggestion generators do not make use of layout specific characteristics. We think that the accuracy of the suggestions generated by a Bangla spell checker can be improved greatly if layout specific characteristics are taken into consideration. A Layout specific keystroke-based dictionary of Bangla words is also necessary here for this purpose.

As we see, all of the above applications need conversion from Bangla text to keyboard layout specific keystrokes. We have not found any work on such conversion. Some conversion utilities, which coverts files created with a specific interface (e.g., Bijoy) to Unicode standard [5, 6, 7], have been developed. Also some works are found to convert incompatible text typed using one interface to that in another interface [10, 14].

These works are not directly related to our work. Therefore, the above applications and lack of research in this direction have motivated us to do this work.

3. Analysis of Bangla Unicode Text

The effective unit of the Bangla Unicode text is called the orthographic syllable or simply *syllable* which consists of a consonant and vowel (**CV**) core. The consonant can have a canonical form like (((**C**)**C**)**C**) or can be null if the syllable starts with an independent vowel [1, 2]. For example, text অন্বেষণ (অ ন ◌্ ব ◌ে ষ ণ) have four syllables: অ, ন্বে (ন ◌্ ব ◌ে), ষ and ণ. With the help of the Unicode standard 5.0 [1, 2], we define rules of a syllable bellow. We also represent the rules using regular expressions.

Abbreviations

C′ = {ৎ, ◌ং, ◌ঃ, ◌ঁ}, set of consonants except **C′**, ***C*** = {ক, খ, গ, …, য়}, set of independent vowels ***V*** = {অ, আ, ই, …, ঔ}, set of dependent vowels ***V′*** = {◌া, ◌ি, ◌ী, ◌ু, ◌ূ, ◌ৃ, ◌ে, ◌ৈ, ◌ো, ◌ৌ, ◌ঁ}, hasant ***H*** = ◌্, ***J*** = Zero Width Joiner (ZWJ), ***J′*** = Zero Width Non-Joiner (ZWNJ).

Rules

1. Any consonant {**C**, **C′**} or independent vowel **V** is a syllable, e.g., অ, আ, ক, খ, etc. In regular expression: $\mathbf{S_1}$ = **C|C′|V**.
2. Hasant is used between the characters of a conjunct. Any number of (**CH**) followed by a **C** is a Syllable. For example, জ ◌্ ঞ (জ্ঞ), ত ◌্ ব (ত্ব) etc. Regular expression: $\mathbf{S_2}$ = (**CH**)*C.
3. ZWNJ is used when a hasant is explicitly used. One or more (**CH**) followed by a ZWNJ is a Syllable. Regular Expression: $\mathbf{S_3}$ = (**CH**)$^+$**J′**. For example, ক ◌্ J′ (ক্), হ ◌্ J′ (হ্), etc.
4. Single dependent vowel except ◌ঁ can follow a consonant or a conjunct, i.e., $\mathbf{S_2}$ followed by a single dependent vowel **V′** other than ◌ঁ is a syllable. Regular Expression: $\mathbf{S_4}$ = $\mathbf{S_2}${**V′** - ◌ঁ}. For example, হ ◌ৈ (হৈ), জ ◌্ ঞ ◌া (জ্ঞা), etc.
5. Certain pairs of vowels can follow a consonant or a conjunct, i.e., $\mathbf{S_2}$ followed by a pair of dependent vowels ◌োা or ◌ে ◌ঁ is a syllable. Regular Expression: $\mathbf{S_5}$ = $\mathbf{S_2}$ {◌োা | ◌ে ◌ঁ}. For example, ক◌ে ◌ঁ (কৌ), দ ◌োা (দো), etc.
6. Using ZWJ consonant-vowel ligated form is requested, while using ZWNJ it is blocked. Therefore, **C** followed by **J** or **J′** followed by **V′** is a syllable. Regular expression: $\mathbf{S_6}$ = **CJV′|CJ′V′**, e.g., র J′ ◌ু (iy), শ J ◌ু (ï), etc.
7. Consonant র followed by a hasant represents a repha, i.e., র followed by hasant **H** followed by $\mathbf{S_2}$ represent a syllable. Regular expression: $\mathbf{S_7}$ = র**H**$\mathbf{S_2}$, e.g., র ◌্ থ (র্থ), র ◌্ য ◌্ য (র্য্য).

8. র‍্য is specially considered. If (**JH**য) follows র, it is treated as র‍্য. র‍্য can be followed by a **V′**, e.g., র‍্যাম (র J H য া). Regular expression: $\mathbf{S_8}$ = র**JH**য**V′**.

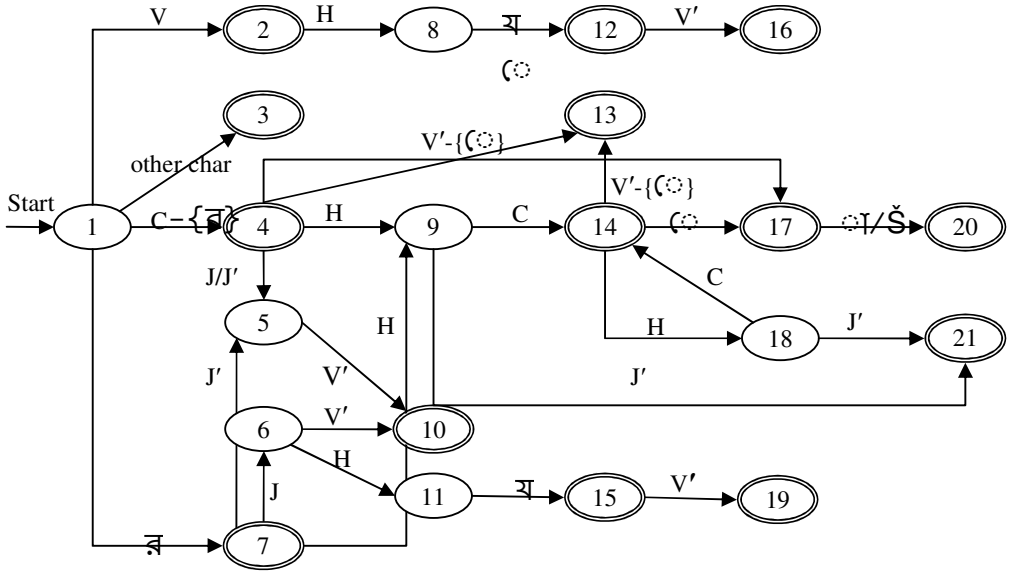


Fig. 1: DFA for syllables in Bangla Unicode text.

9. Independent vowel **V** followed by ya-phalla (**H**য) followed by a dependent vowel **V′** is a syllable, e.g., অ ্ য া (অ্যা), এ ্ য া (এ্যা). Regular expression: $\mathbf{S_8}$ = **VH**য**V′**.

Based on the above rules, we build a deterministic finite automaton (DFA) to represent a syllable as shown in Fig. 1. The purpose of this representation is to write, for each type of keyboard layouts, algorithm which take one syllable at a time from the Bangla Unicode text and convert it into keystrokes in that layout. The automaton could be simplified if we would use non-deterministic finite automaton (NFA) instead of DFA. But such automaton would make the automata conversion algorithm complex.

Table 1: Bangla character to keystroke(s) mapping.

Unicode	Bangla Character	৹ PPhonetic	Bijoy	UniBijoy	Unicode	Bangla Character	Avro Phonetic	Bijoy	UniBijoy	Unicode	Bangla Character	Avro Phonetic	Bijoy	UniBijoy
58	:	:`	:	:	2460	জ	j	u	u	2488	স	s	n	n
2404	।	.	G	G	2461	ঝ	jh	U	U	2489	হ	h	i	i
2433	ঁ	^	&	@	2462	ঞ	NG	I	I	2494	া	a	f	f
2434	ং	ng	Q	q	2463	ট	T	t	t	2495	ি	i	d	d
2435	ঃ	:	|	^	2464	ঠ	Th	T	T	2496	ী	I	D	D
2437	অ	o	F	F	2465	ড	D	e	e	2497	ু	u	s	s
2438	আ	a	gf	gf	2466	ঢ	Dh	E	E	2498	ূ	U	S	S
2439	ই	i	gd	gd	2467	ণ	N	B	B	2499	ৃ	rri	a	a
2440	ঈ	I	gD	gD	2468	ত	t	k	k	2500	র্ফ	rr	A	A
2441	উ	u	gs	gs	2469	থ	th	K	K	2503	ে	e	c	c
2442	ঊ	U	gS	gS	2470	দ	d	l	l	2504	ৈ	OI	C	C
2443	ঋ	rri	ga	ga	2471	ধ	dh	L	L	2507	ো	O	c,f	x
2447	এ	e	gc	gc	2472	ন	n	b	b	2508	ৌ	OU	c,X	X
2448	ঐ	OI	gC	gC	2474	প	p	r	r	2509	্	empty	g	g
2451	ও	O	x	gx	2475	ফ	f	R	R	2510	ৎ	t"	\\	&
2452	ঔ	OU	gX	gX	2476	ব	b	h	h	2519	ঁ	(*)	X	(*)
2453	ক	k	j	j	2477	ভ	v	H	H	2524	ড়	R	p	p
2454	খ	kh	J	J	2478	ম	m	m	m	2525	ঢ়	Rh	P	P
2455	গ	g	o	o	2479	য	z	w	w	2527	য়	y	W	W
2456	ঘ	gh	O	O	2480	র	r	v	v	8204	ZWNJ	,,	g	g
2457	ঙ	Ng	q	Q	2482	ল	l	V	V	8205	ZWJ	(*)	(*)	(*)
2458	চ	c	y	y	2486	শ	S	M	M					

2459	ছ	ch	Y	Y	2487	ষ	Sh	N	N					

(*) – Not supported

Note that in addition to valid syllables, the DFA can accepts some invalid syllables also, e.g., ইা, ঙা, ে, etc. The DFA would be over complex if it would adopt every possible constraint. Still such constraints can be adopted in the code during implementation.

4. Algorithms

The DFA for syllables is presented in the previous section. In this section, for each type of keyboard layouts, we design an algorithm which converts one syllable at a time from the Bangla Unicode text to layout specific keystrokes. In order to convert a Bangla character to stroke(s) in a layout, we use the conversion table Table 1 [3]. The table contains the key(s) need to press to type each Bangla character in selected keyboard layout. Some characters such as numerals (০ to ৯), symbols and punctuations (except ':') is not shown in the table, because these characters need same keys pressed in all layouts. In Avro phonetic layout, a number of characters can be typed in several ways, e.g., শ can be typed by keystrokes *sh* or *S*, গ can be typed by g or G, etc. However, we have not shown it in the table. We have shown only keys we have used in conversion process.

```
/* input contains input text, output will contain the converted output,
startPos and endPos are start and end indexes of the syllable in the
input and state is the final state of the DFA where the syllable is
identified */

terminate (startPos, endPos, input, output, state)
    if layout='B' then //'B' for Bijoy layout
        replaceAll(input, 'ো', "ো") //ো not supported in Bijoy
        replaceAll(input, 'ৌ', "ে ঁ") //ৌ not supported in Bijoy
        toBijoy(state, startPos, endPos, input, output)
    else
        // "েো" and "ে ঁ" are not used in Unicode layouts
        replaceAll(input, "েো","ো")
        replaceAll(input, "ে ঁ", 'ৌ')
        if layout='U' then //'U' for UniBijoy layout
                toUniBijoy(state, startPos, endPos, input, output)
        else If layout='A' then //'A' for Avro Phonetic layout
                toAvro(state, startPos, endPos, input, output)
end
```

As we see from the DFA, the syllable that terminates at state 10, is to request or block the consonant-vowel ligature [1, 2], e.g., if a (**J′V′**) is found after a consonant, it should be considered as non-ligated character. However, in current keyboard layouts we have not found support for it. We therefore have not considered it in our algorithms.

Implantation of DFA is very trivial. Therefore, it is not shown in algorithms. For each of the identified syllable in the input text, the procedure *terminate()*, which is shown in Fig. 2, is called. This procedure does some preprocessing on the input text and ultimately calls the appropriate procedure which converts the syllable to keystrokes in specific layout.

4.1 Mapping to Bijoy Keystrokes

Bijoy (classic version) is a traditional style keyboard layout. The 'ি'and 'ে', although phonetically come after the consonant/conjunct they modify, they are placed before them. But the Unicode text follows the phonetic order. For example, the Unicode text syllable ক ্ষ ে (ক্ষে) is represented in Bijoy as *cjgN* (ে ক ্ ষ). In Bijoy, modifiers য-ফলা and র-ফলা are represented by *Z* and *z* respectively.

```
/*convert(x, L) function uses the conversion table Table 1 and convert
x to keystroke(s) in L layout and returns the keystrokes */
toBijoy (state, startPos, endPos, output, input)

        if input[endPos] = 'ি'| 'ে' then
                output = output + convert(input[endPos], 'B');
                //B means Bijoy layout
        else if input[endPos-1] = 'ে'then
                output = output + convert(input[endPos-1], 'B');
        end if
        for I = startPos to endPos do
                if input[I] = 'ি'| 'ে' then
                    Next I //skip the rest
                else output = output + convert(input[I], 'B')
        //post processing
        //য-ফলা (gw), র-ফলা (gv) represented by Z, z
respectively
        replaceAll(output, "gw,'Z')
        replaceAll(output, "gv", 'z')
end
```

In Unicode text, the position 'ি' is at the end of the syllable because after 'ি' no other vowel is used. On the other hand, 'ে'is found at the end or just before end of the syllable, because some other vowels (া, ঁ) may be used after 'ে'. Therefore the main task of the algorithm is to reposition the 'ি'and 'ে' and convert from Unicode to ASCII code (i.e., keystroke) with the help of the conversion table Table 1. The final

step (post processing) is placing *Z* where a য-ফলা is found and *z* where a র-ফলা is found. The procedure *toBijoy()* shown in Fig. 3 does this task.

4.2 Mapping to UniBijoy Keystrokes

Conversion from Unicode text to UniBijoy (modern style typing [5]) layout is straight forward. It is a one to one mapping. Unlike Bijoy, no rearrangement of characters in the syllable is necessary. Because, in UniBijoy all dependent vowels are typed after the consonant or conjunct it modifies. Therefore, the algorithm *toUniBijoy* shown in Fig. 4 takes one character from the Unicode text syllable at a time and convert it to keystroke(s) with the help of the conversion table. For example, the Unicode text syllable ক ্ষ ো (ক্ষো) which is represented by the codes 2453 2509 2487 2503 is converted to *jgNc* (ক ্ষ ো). Here, 2453 is mapped to *j,* 2509 to *g*, and so on.

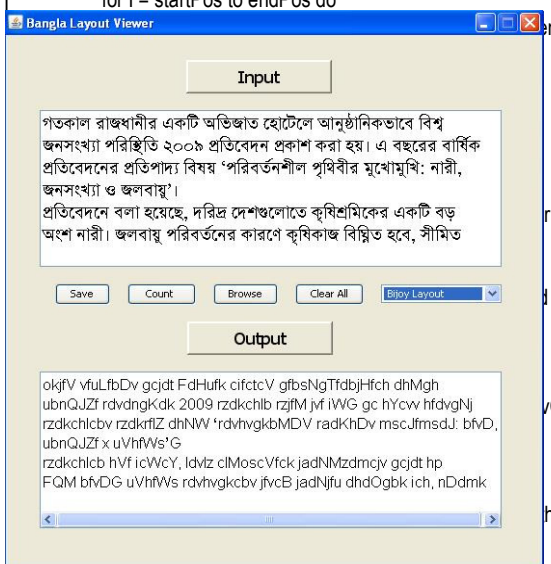
```
toUniBijoy (state, startPos, endPos, output, input)
        for I = startPos to endPos do
                output = output + convert(input[I], 'U')
        //post processing
        //য-ফলা (gw), র-ফলা (gv) can be represented by
        // Z, z respectively. It reduces the no. of strokes
        replaceAll(output, "gw, 'Z')
        replaceAll(output, "gv", 'z')

end
```

Fig. 4: *toUniBijoy()* procedure

4.3 Mapping to Avro Phonetic Keystrokes

Since Avro is a transliteration based layout, converting from Unicode syllables to Avro keystrokes is not as easy as that to Bijoy and UniBijoy keystrokes. In Bangla, a consonant by default has an inherent vowel অ [1, 2]. For example, কর is implicitly similar to কঅরঅ. Unlike Unicode text, transliteration based layouts use this strategy, i.e., they represent কর as *koro/kor* (the last 'o' is not necessary) where *k* is for ক and *r* is for র. On the other hand, in Unicode text, it is coded as 2453 2480 (ক র) without any vowel inside them. Unlike Unicode standard, in transliteration based layouts, to represent conjuncts extra character for hasant is not necessary, only the default vowel অ is omitted from the consonant. For example, the word অন্তর is represented as *ontoro/ontor*. Note that there is no অ or hasant between *n* and *t* because they are the characters of a conjunct. However, in Unicode it is coded as 2437 2472 2509 2468 2480 (অ ন ্ ত র) which contains a hasant between ন and ত indicating that they form a conjunct.

```
//Abbreviations C, V, V' etc. has been described in previous section
toAvro (state, startPos, endPos, output, input)
        len = endPos – startPos + 1
        if (state = 11 or state = 12) and match(startPos+1, "্য") then //if য-ফলা comes after
V
                output = output + convert(input[startPos], 'A')+ "Z"
                if state = 12 then output = output + convert(input[endPos], 'A')
                        //A for Avro Phonetic, Z is to force য-ফলা in Avro
                return
        repha = false
        for I = startPos to endPos do
```




```
                r = '্' //য-ফলা
        nextChar = '্' then //রেফ
        vChar = '্' then //ব-ফলা
        hen //C defined before
        repha = false
```

Fig. 6: Unicode text (part of a news paper) to Bijoy Keystrokes

In Unicode text, য-ফলা is represented by hasant and য (্ য). However, in Avro, য-ফলা is represented by *y* if it follws a consonant. But if it follows a vowel (e.g., অ্যা), a *Z* is used instead of *y*. As shown in Fig. 1, such case happens if a syllable terminates at state 11 or 12.

If a hasant is followed by a র it is considered as repha, e.g., অর্থ (অ র ্ থ). Repha is represent as *rr* in Avro. However, in Unicode text, to seperate repha from য-ফলা in a character sequence (র ্ য), an extra ZWJ is used after the র. Therefore, as shown in

Fig. 1 and Fig. 5, if (ZWJ ◌্ য) is found after a র, a য-ফলা is enforced by using *Z*. When a ব-ফলা (◌্ ব) is found in a Unicode syllable, it is coded as *w* in Avro.

All other characters are simply converted using Table 1. For example, a ◌ি is translated as *e*, a ZWNJ that appears after a hasant (shown in Fig. 1) is translated as ,, which is used to represent the hasant explicitly e.g., আহ্লাদ (converted as *ah,,lad*).

5. Experimental Results

We have implemented our algorithms using Java and tested with the Unicode text available at online news paper Prothom Alo [25]. We separate the detail news of one day and save them into a file. We feed the file to our program and generate and save outputs for Bijoy, UniBijoy and Avro Phonetic layouts. We then typed the generated keystrokes (output) using each of the Bijoy, UniBijoy and Avro layouts in MS Word. In each case, we get the original input text (detail news). Fig. 6, Fig. 7 and Fig. 8 show the screen shots of the program showing the conversion of part of the input

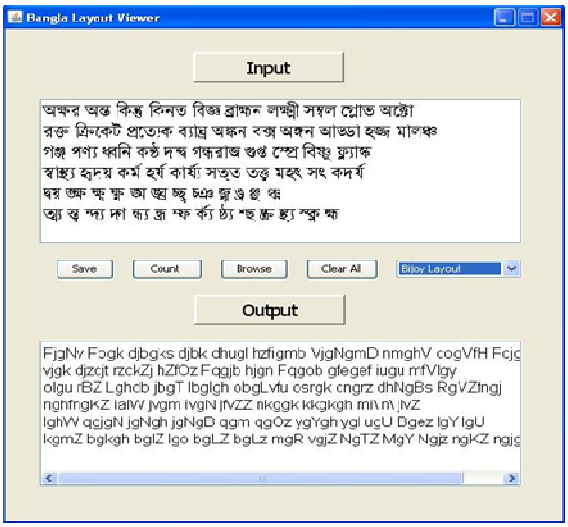


Fig. 10: Unicode text (conjuncts) to Bijoy Keystrokes

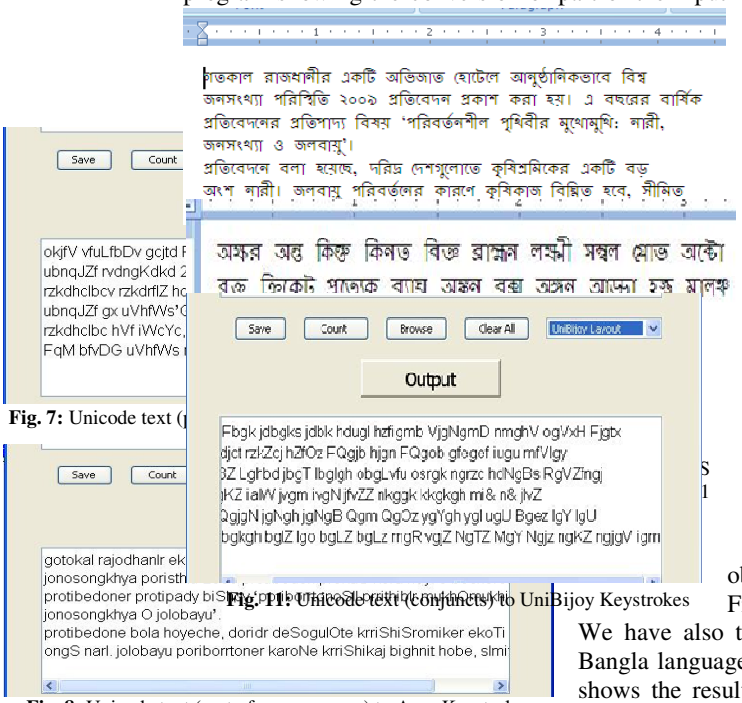


Fig. 7: Unicode text (

Fig. 8: Unicode text (part of a news paper) to Avro Keystrokes

Fig. 12: Unicode text (conjuncts) to Avro Keystrokes

Fig. 11: Unicode text (conjuncts) to UniBijoy Keystrokes

(detail news) to keystrokes for each of the layouts. Fig. 9 shows the result of the reverse process, i.e., the text obtained after keying the output of the program shown in Fig. 6, Fig. 7 and Fig. 8.

We have also tested our program with the conjuncts [4, 20] used in Bangla language as shown in Fig. 10, Fig. 11 and Fig. 12. The Fig. 13 shows the result of the reverse process. In all cases we could exactly regenerate the input text by keying the output.

6. Discussion

We have tested our algorithms with sufficient data. It seems that our algorithms work correctly. Yet, if the text does not follow the Unicode standard exactly, it could produce unexpected result. It is sometimes seen that some Unicode characters are not used by the keyboard interfaces properly. For example, sometimes in place of ZWJ, a ZWNJ is used. One such example is র‍্যাম. In Unicode standard, this word should be written as (র (ZWJ) ◌্ য া ম) [1, 2] but our used keyboard interface generates (র (ZWNJ) ◌্ য া ম). As a result, our program could not convert this word correctly.

Bangla keyboard interfaces usually provide more than one way to type a Bangla character. For example, in UniBijoy য-ফলা can be written by pressing *Z* or *gw*, র-ফলা by *z* or *gv*, in Avro ফ can be written by *f* or *ph*, গ by *g* or *G*, etc. Therefore, many of the words used in Bangla have multiple representations in keystroke form. However, our algorithm performs conversion to only one of a number of different forms in a keyboard layout. Multiple forms of a converted word can be necessary in the applications we have described in section 2. However, conversion to other forms from our generated keystrokes is not difficult. It only needs some pattern matching and replacement operations on the output of our program.

7. Conclusion

In this paper, we have analyzed the Unicode standardized Bangla text and identified the syllables. We have developed algorithms to convert the syllables into keystrokes in different Bangla keyboard layouts. There is no previous research work on such type of conversion. We have also implemented our work using Java and shown that our algorithm works correctly. As we have described, our work can be used in many applications including comparing the efficiencies of keyboard layouts, predicting a word while being typed to improve typing efficiencies, improving the accuracy of spelling suggestions generated by a Bangla spell checker, etc.

References

[1] *South Asian Scripts-I*, http://Unicode.org/versions/Unicode5.0.0/ch09.pdf.

[2] *South and Southeast Asian Scripts*, http://www.Unicode.org/book/ch09.pdf.

[3] *Bengali Unicode Chart*, http://www.Unicode.org/charts/PDF/U0980.pdf.

[4] Md. Mehedi Hasan, *User Manual for Avro Keyboard 4*, March 26, 2007, http://www.omicronlab.com/avro-keyboard.html.

[5] *Avro Converter 0.6.0*, http://www.omicronlab.com/avro-converter.html.

[6] *Rupantor Bangla Converter*, http://www.apona-bd.com/Unicode-converter-rupantor/.

[7] *Data Converter*, Bijoy Bayanna, www.bijoyekushe.net.

[8] *Bijoy Bayanna*, *Broshure,* www.bijoyekushe.net.

[9] *Bijoy Keyboard Interface*, www.bijoyekushe.net.

[10] *Proshika Shabda*, http://www.proshikanet.com/.

[11] *Unicode Complient National Keyboard User Manual*, http://www.bcc.net.bd/nkb/usermanual.pdf.

[12] *Munir Keyboard Layout*, http://ekushey.org/?page/munir_layout.

[13] *Software, Bangla*, http://www.banglapedia.org/httpdocs/HT/S_0449.HTM.

[14] *Prabartan-2000*, http://prabartan-2000.software.informer.com/.

[15] *Ekusheyr Shadhinota*, http://ekushey.org/?page/shadhinota.

[16] Naushad UzZaman, Arnab Zaheen and Mumit Khan, *A Comprehensive Roman (English)-to-Bangla*

[17] *Transliteration Scheme*, International Conference on Computer Processing on Bangla (ICCPB-2006), Dhaka, Bangladesh, 17 February, 2006

[18] *Bengali Transliteration System*, http://www.prabasi.org/Literary/ComposeArticle.html

[19] Naushad UzZaman, *Phonetic Encoding for Bangla and its Application to Spelling Checker,*

[20] *Transliteration, Cross language Information Retrieval and Name searching*, Undergraduate thesis (Computer Science), BRAC University, May 2005.

[21] *Transliteration*. In Wikipedia, The Free Encyclopedia. Retrieved 13:15, December 12, 2009, from http://en.wikipedia.org/w/index.php?title=Transliteration&oldid=330460039

[22] *ব্যবহারিক বাংলা অভিধান*, ডিসেম্বর ২০০০, বাংলা একাডেমী, ঢাকা।

[23] *Ekusheyr Shadhinota "UniJoy" Layout*, http://ekushey.org/?page/uni_joy_layout.

[24] *Ekusheyr Shadhinota "Bangla Unicode" Layout*, http://ekushey.org/?page/bangla_Unicode_layout.

[25] *Baishakhi Keyboard Layout*, http://nltr.org/snltr-software/readme/keyboard_layout_readme.txt

[26] *KickKeys*, http://www.kickkeys.com/index.htm

[27] *The Daily Prothom Alo*, Online version, http://www.prothom-alo.com.

Submitted: 13th December, 2009; **Accepted for Publication:** 10th March, 2010.