International Journal of Computer Applications (0975 – 8887)
Volume 58– No.17, November 2012

A Finite State Transducer (FST) based Font Converter

Sriram Chaudhury
KIIT University
Bhubaneswar
India

Shubhamay Sen
KIIT University
Bhubaneswar
India

Gyan Ranjan Nandi
KIIT University
Bhubaneswar
India

ABSTRACT

This paper describes the rule based approach towards the development of an Oriya Font Converter that effectively converts the SAMBAD and AKRUTI proprietary font to standardize Unicode font. This can be very much helpful towards electronic storage of information in the native language itself, proper search and retrieval. Our approach mainly involves the Apertium machine translation tool that uses Finite State Transducers for conversion of symbolic data to standardized Unicode Oriya font. To do so it requires a map table mapping the commonly used Oriya syllables in Proprietary font to its corresponding font code and the dictionary specifying the rules for mapping the proprietary font code to Unicode font. Further some unhandled symbols that appear in the intermediate converted file are rectified by Flex scanner tool. The converted text thus obtained is in standard Unicode font and remains unchanged as Unicode font is supported by almost all the platforms.

General Terms

Font converter, Oriya font converter, FST based font converter, Rule based font converter.

Keywords

Oriya font converter, Proprietary font to Unicode font converter, A Finite State Transducer based font Converter, Font converter for Indian language, Rule based font conversion, Apertium in font conversion.

1. INTRODUCTION

When we read the e-newspapers in English they are quite accessible in different systems independent of the platform, hardware or word processor as they follow a standard encoding scheme. Whereas the e-newspapers in regional languages like Oriya, it's perfectly readable and the contents can be easily saved but when we want to access the saved contents later, it's either inaccessible or shows a lot of Latin else symbolic characters. The main reason behind this is that most of the newspapers in regional languages use their licensed proprietary font for typing regional scripts which is not standardized. SAMBAD daily newspaper that is one of the leading newspapers in Orissa uses SAMBAD proprietary font for typing Oriya script using 4Clipika software, just like Pragativadi and Dharitri which uses AKRUTI font for typing Oriya font. But these are not standardized fonts and hence not supported by most of the platforms. So there arises the need of a converter that can convert the proprietary font to a standardized font like Unicode [1] Font. To allow computers to represent any character in any language, the international standard ISO 10646 defines the Universal Character Set (UCS) [2]. It provides a unique code to each and every character of any script of any widely used language across the globe. The Unicode range for Oriya is in the range 0B00-0B7F (HEX) and 2816-2943 (DEC). The main aim behind selecting Unicode as the standard is that it encodes plain text characters (aksharas) code not glyphs. It also guarantees accurate convertibility to any other widely accepted standard and vice versa so is compatible across many platforms. It is also capable of unifying the duplicate characters with in scripts of different languages. We adopt the UTF-8 [3] encoding scheme of Unicode. UTF-8 is defined as the UCS Transformation Form (8 bit). It's a variable-width encoding that can represent every character in the Unicode character set. It was designed for backward compatibility with ASCII As a result of which we can store a huge amount of data in regional language as itself hence would be helpful in forming a large corpus and can effectively perform word search, dictionary lookup, script conversion and machine translation operations successfully. Hence would be a helpful contribution towards further linguistic research in Oriya language. Our work mainly focuses on the conversion of SAMBAD and AKRUTI font data.

2. RELATED WORK

A proposed converter for Devanagari Script by Akshar Bharati et al. [4] supports the conversion of text in unknown coding scheme to standard ACII (Alphabetic Code for Information Interchange) coding scheme either automatically or semi automatically. The converter uses a matching program which compares the glyph code with the corresponding ACII code and effectively learns the equivalence using the glyph grammar. The glyph grammar is script specific and independent of coding scheme used. It specifies the possible glyph sequence that makes an akshara. Each glyph has its equivalent byte code that is maintained in a glyph-code mapping table which is generated out of the matching program. As the next step the grammar and the map table is used to generate the converter. The second step is repeated several times to refine the conversion. The converter takes as input the sequence of byte code that is obtained when copied from e-resources and converts it to corresponding ACII code.

A different approach towards building font converters and the process of glyph assimilation for font-data conversion in Indian languages was proposed by A. Anand Arokia Raj [5]. The aksharas of Indian languages are split up into glyphs in font data and the objective of the converter is to recombine the glyphs to retrieve the valid character. The converter for the above purpose works in two steps. First it develops a glyph map table for each font type. Then defines and develops the glyph assimilation rule for the language. Glyph assimilation rules define the sequences in which glyphs can be rearranged, combined and mapped unambiguously to form the Asksharas of a language.

Another approach is the *IConverter* [6]: An effective tool for several code conversions. The tool uses 'isciilib' a library for code conversion. It takes as input the Configuration file and the Source code file. The configuration file defines the rules for code conversion.

H. Garg [7] has proposed two ideas for overcoming font and script barriers among Indian languages. I.e. Glyph Grammar based approach, where it uses a font glyph description file mapping glyph of the font to appropriate ISCII [8]/Unicode characters and in absence to mnemonics. Another approach is by Finite State Transducer based automatic or semi-automatic machine learning technique. These Finite State Transducers automatically learns the mapping between glyphs and equivalent ISCII/Unicode characters through parallel training corpora.

3. PROPOSED IDEA

Apart from several Font Converters being proposed and developed for many Indian languages our approach for font conversion of Oriya language follows a rule based approach. The generation of converter takes place in three steps:

(1) First a map table is created which maps the most commonly used Oriya syllables in Proprietary font to its corresponding font code.

(2) Secondly a dictionary is developed defining the rules to map the Proprietary font code to Unicode font.

@Ð → ଆ

(3) Then a Flex scanner is designed to handle the Un-handled symbols that appear in the output from step-2.

The basic block diagram of the FST based Font Converter is shown below (see Figure 1).

3.1 Mapping process

The process proceeds as follows:
(1) First a large corpus of proprietary font data is collected from the regional e resources and analyzed to extract out the most frequently used Oriya syllables.
(2) The relevant software (4clipika) for writing the regional script data is used to write the language syllables and the corresponding font code for that syllable is obtained.
(3) A mapping table is developed which maps possible Oriya-syllables with the equivalent Proprietary font code. (see Table 1)

Table 1. Sample Map Table

WX Notation	Proprietary Font Syllable	Font Code	Unicode Syllable
w_Be	ତ୍ଭେ	Ò[÷	ତ୍ଭେ
w_De	ତ୍ଢେ	Ò[çY	ତ୍ଢେ
W_h	ଥ୍ହ	\çk	ଥ୍ହ
w_Saq	ତ୍ଶୃ	[çhó	ତ୍ଶୃ

WX notation [9][10] is used to represent the Devanagari and many other Indian language alphabets in ASCII. The WX-notation for Oriya language follows the same pattern as for Devanagari. (see Table 2)

Table 2. Sample WX- notation for Oriya

Akshara	Wx Notn.	Akshara	Wx Notn.	Akshara	Wx Notn.
ଅ	a	◌ୌ	O	ଦ	x
ଆ	aA	◌ଂ	M	ଧ	X
ଇ	ai	◌ଃ	H	ନ	n
ଈ	aI	କ	k	ପ	p
ଉ	au	ଖ	K	ଫ	P
ଊ	aU	ଗ	g	ବ	b
ଏ	ae	ଘ	G	ଭ	B
ଐ	aE	ଙ	f	ମ	m
ଓ	ao	ଚ	c	ଯ	y
ଔ	aO	ଛ	C	ର	r
ଋ	aq	ଜ	j	ଲ	l
◌ା	A	ଝ	J	ୱ	v
◌ି	i	ଟ	t	ଶ	S
◌ୀ	I	ଠ	T	ଷ	R
◌ୁ	u	ଡ	d	ସ	s
◌ୂ	U	ଢ	D	ହ	h
◌େ	e	ଣ	N	◌୍	–
◌ୈ	E	ତ	w	◌ଁ	az
◌ୋ	o	ଥ	W		

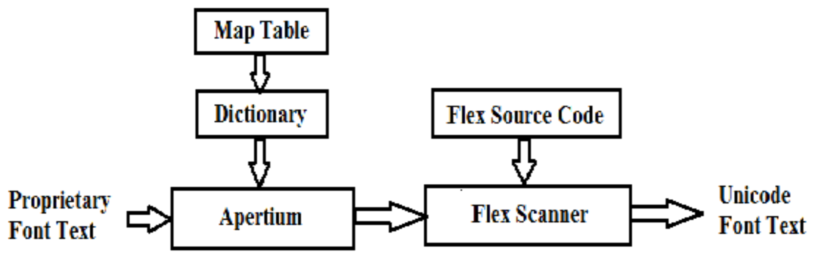

Figure 1: Basic Block Diagram of Font Converter

3.2 Formulating the Dictionary

The Dictionary forms the soul of the system. It consists of hand written rules mapping the Unicode font data of the corresponding Proprietary font code for the collected Oriya syllables. It helps the FST to work on a pattern-result basis, i.e. when a known pattern of font code is found the corresponding Unicode font is generated. The dictionary consists of two basic components Alphabet part and Section part. Alphabet part introduces all the alphabets, symbols, numbers and special characters going to be used in Section part. Where as in the Section part its defined the basic mapping between the given proprietary fonts code to the Unicode font character or combination of characters.

```
<?xml version="1.0" encoding="UTF-8"?>

<dictionary>

<alphabet>ABCD…abcd…012…ଁଂଃଅଆଇ…ିୀୁ…୦୧
୨୩…ÀÁÂ</alphabet>

<section id="main" type="standard">

<e><p><l>ଶ</l><r>h</r></p></e>

<e><p><l>ଦ</l><r>]</r></p></e>

…. </section>

</dictionary>
```

The Converter uses the Apertium [9] machine translation tool. It takes as input a .txt or .xml file containing the proprietary font code (saved contents of e-news papers of regional language) comprising of Latin characters and symbols and converts it to Unicode Oriya font by the help of the dictionary. The Apertium follows an approach to lexical processing based on the use of letter-transducers (a class of FST). A letter-transducer [9] is a Finite State Machine consisting of states (single initial and one or more acceptance state) and a finite set of state transitions with given input letter or symbol to the output letter or symbol. The transducer takes the font code for a particular syllable at a time and tries to read the longest pattern recognized by the dictionary (left-to-right longest match mode), which matches with the given token. On matching of each symbol transition takes place from one state of FST to another. The FST terminates when all the symbols of the font code is matched to corresponding symbols specified in the dictionary. Hence an acceptance state is encountered which yields the associated output specified by that dictionary entry. That is nothing but that syllable in Unicode font. If no match for the syllable is found in the dictionary the transducer simply copies the input data in the output file. This situation may arise when the syllable encountered is rarest to appear in day to day conversation and can be rectified by adding the appropriate new rules to the dictionary.

3.3 Generating the Scanner

The basic need of further refinement of the converted text arises as Apertium can't handle some symbols like @, \, _ so these symbols appear as it in the text of the output data file obtained from the second step. Thus a scanner is introduced which uses a wrapper file. The wrapper file is developed with the help of Flex programming to handle such issues and rectify all those errors.

Flex is an exclusive tool used for further refinement of expert systems. It takes as input set of descriptions of possible tokens and produces a scanner. The input to the Flex tool is a text file containing regular expressions and the corresponding action to be taken when each expression is matched. The Flex source file consists of three basic parts divided by a single line starting with %%:

```
Definitions

%%

Rules

%%

User Code
```

3.3.1 *Definitions*

The definition section occurs before the first %%. It contains two things. First, any C code that must be inserted external to any function should appear in this section between the delimiters %{ and %}. Secondly, the definitions section contains declarations of simple name definitions to simplify the scanner specification and declarations of start conditions.

3.3.2 *Rules*

The lexical rules section of a Flex specification consists of a set of regular expressions (pattern) and actions that are executed when the scanner matches the associated regular expression.

3.3.3 *User code*

The user code section is simply copied to"lex.yy.c" (output file generated by Flex). The presence of this section is optional.
Sample Flex programming for solving error;

```
%%

_ï          {printf("ନ୍ତ");}

\[          {printf("ତ");}

\[Ð         {printf("ତା");}

\[Þ         {printf("ତି");}
```

4. ERROR ANALYSIS

Error in font conversion though Apertium tool mainly occurs because of the following two cases:

(1) The syllable of proprietary font text is rare and does not appear normally in day to day life. As the raw corpus is collected from e-news paper sources so it mostly reflects the common syllables. In case such a syllable is encountered the tool founds no match for it in the rules specified in the dictionary. So the same unconverted text appears in the output file.

(2) Some symbols cannot be processed by Apertium like \, Ð, @, _ so they appear as it in the output file.

The following chart for Apertium (see Figure 2) specifies the unhandled cases that accounts for the inefficiencies that encountered in the converter. This happens because of the above specified reasons. The basic solution to the first inefficiency is further improvement of the dictionary by adding new rules for all those syllables that are new or rarely appear. This may be done by collecting a large corpus related to domains like literature or any other domain specific like science and training the machine on those corpuses also. It reduces the chances of error but can't absolutely guarantee its absence.

As stated above the second cause of error can be checked by allowing the output text from Apertium to pass through the Flex scanner which can easily encounter such cases. The chart (see Figure 3) for Flex scanner output specifies those improvements made on the converter which results as almost 72% conversion accuracy on the test corpus. The statistics presented in the following table (see Table 3) specifies the performance of the converter on a small test corpus of 4000 words of SAMBAD font data collected from e-news paper. Out of which 1216 words are successfully converted after the first step with an accuracy of 30.4%. Secondly 2882 words are accurately converted after the second step with an accuracy of 70.05%. Further 1118 words remain unconverted or wrongly converted. The phase wise comparison of the Font Converter is given in Figure 4.

Table 3. Statistics of Conversion

Total words	**Correctly Converted**		**Wrong/Un handled**
	After Apertium	**After Flex**	
4000	1216(30.4%)	2882(72.05%)	1118

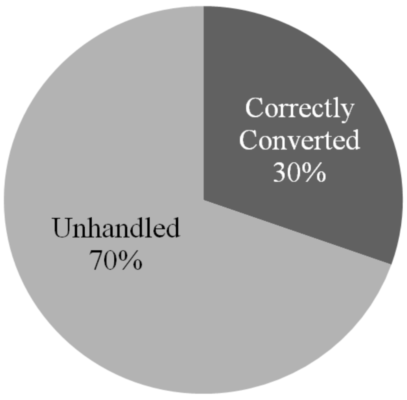

Figure 2: Result of first phase of conversion

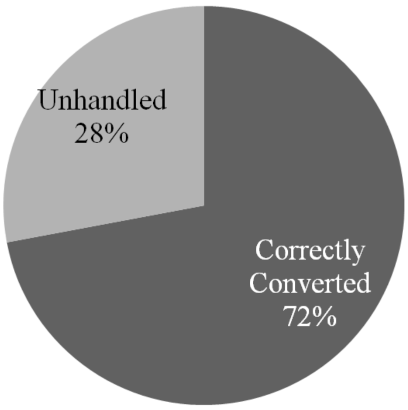

Figure 3: Result of second phase of conversion

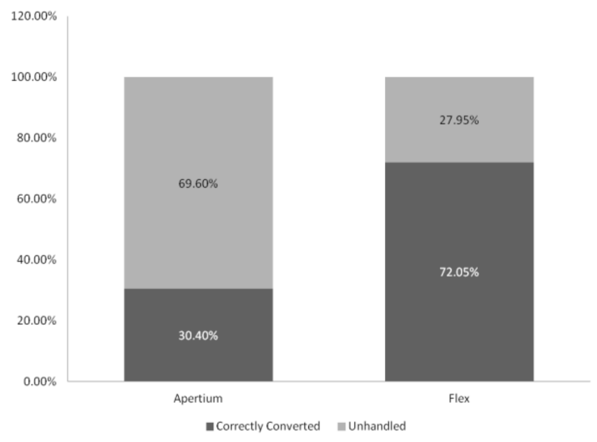

Figure 4: Phase wise improvement of converter

Table 4. Example Conversion

Symbolic code	Correct Syllable	O/p Apertium	O/p Flex
@Ð	ଆ	ଅ	ଅ
Òk	ହେ	Òହ	ହେ
`Ð	ଘା	ଘ	ଘ
BÜ	ଇଁ	ଇÜ	ଇଁ
¦ÞÆ	କ୍ତି	କ୍ତିÆ	କ୍ତି
q	ଓ୍ଚ	q	ଓ୍ଚ
[Ð	ଡା	ଡ	ଡା
$¼ç	ଫ୍	ଫ।ç	ଫ୍

6. COMPARATIVE STUDY

There is no FST based Font Conversion technique present for Oriya script. However the Flex Programming based Walkman-Chanakya Font Converter for Devanagari script of Hindi language may be considered as an existing approach for comparative study. The Walkman-Chanakya Font Converter converts the Chanakya Font data to WX notation only. To get the standard Unicode font data from WX notation it needs a further processing of data. Basic comparison between the accuracy of two Font Converters on a small corpus of 4000 words chosen from different domains shows the following outcome. (see Table 5)

Table 5. Comparative Statistics

	Correctly Converted	**Wrong/ Unhandled**	**% Accuracy (aprox.)**
FST Based FC	2882	1118	72.05
Chanakya FC	3410	590	85.25

The chart below represents a comparative study of the FST based Converter and Chanakya Converter. (Figure 5)

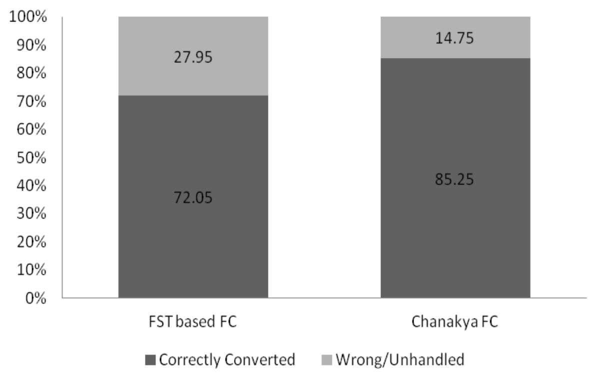

Figure 5: FST FC vs. Chanakya FC

7. CONCLUSION AND FUTURE WORK

In the current age standardization of any language script across the world is a basic need for electronic availability of regional font data and for any research to carry on. In this paper we have defined the basic steps for developing a converter for conversion of SAMBAD and AKRUTI Proprietary font data of Oriya language to standardized Unicode font using a rule based approach. The converter is based on apertium machine translation tool; the output is further smoothened using the flex scanner. The final converted text thus obtained is quite accurate but not absolute. The result also suggests the Chanakya Font Converter to be more accurate. It's mostly because of some un-handled cases and for those cases where the syllables appear quite rarely in day to day common conversation so may remain un-handled. But the idea behind the FST based Font Converter is a quite different one. Hence a scope for further improvement is always there. Mostly by improving the dictionary rules as well as the Flex source code. This will be addressed in future.

8. ACKNOWLEDGMENTS

And at this juncture behind to thank one and all who have helped us, advised us, guided us, stood by us and given us the motivation and faith to carry this research work onto successful consummation. First and foremost, we express profound and heartfelt gratitude to SAMBAD and PRAGATIVADI for providing us the required Oriya Proprietary font data and the software. We thank CIIL, Mysore for its notable contribution towards this research, by providing the Oriya corpus.

We acknowledge those who have worked really hard for the unification and standardization of Indian scripts and inspired us to make this trial successful. We are also thankful for the co-operation received from those who have extended their friendly hands for the successful completion of this job. I will always be indebted to you all.

9. REFERENCES

[1] Steve Comstock, September 2011. An Introduction to Unicode.

[2] Markus Kuhn, "UTF-8 and Unicode FAQ for Unix/Linux". http://www.cl.cam.ac.uk/~mgk25/unicode.html.

[3] F. Yergeau, November 2003, "UTF-8, A transformation format of ISO 10646", RFC 3629, The Internet Society (2003).

[4] Akshar Bharati, Nisha Sangal, Vineet Chaitanya, Rajeev Sangal and G Uma Maheshwara Rao,1998. "Generating converters between fonts semi-automatically". In Proceedings of SAARC conference on Multi-lingual and Multi-media Information Technology, (CDAC, Pune, India).

[5] A. Anand Arokia Raj, 2008. "Multi-lingual Screen Reader and Processing of Font-data in Indian languages". MS Thesis at International Institute of Information Technology Hyderabad, India.

[6] IConverter, A utility program for various code conversions.http://www.cse.iitk.ac.in/users/isciig/iconverter/ main.html.

[7] Himanshu Garg, 2005. "Overcoming the font and script barriers among indian languages". MS Thesis at International Institute of Information Technology Hyderabad, India.

[8] Indian Script Code For Information Interchange – ISCII, 1991, Bureau of Indian Standards(BIS), http://varamozhi.sourceforge.net/iscii91.pdf.

[9] Wx notation overview, http://sanskrit.inria.fr/DATA/wx.html.

[10] WX- notation for Devanagari script alphabet. http://mirror.umd.edu/mozdev/indicime/wx_keyboard.html

[11] Mikel L. Forcada et al., 2010. "Documentation of the Open-Source Shallow Transfer Machine Translation Platform *Apertium*", Departament de Llenguatges i Sistemes Inform`atics Universitat d'Alacant.