A NOVEL TAMIL LANGUAGE PROCESSING IN ANDROID OPERATING SYSTEM

Dr. J. Indumathi[1] P. Joseph Pravin[2] Shelbina A. Jeya[3]
Department of Information Science and Technology,
College of Engineering, Anna University, Guindy,
Chennai-600025, Tamilnadu, India.
[1]indumathi@annauniv.edu, [2]josephpravinn@gmail.com, [3]ashelfina@gmail.com

ABSTRACT

In this fast growing world where technology has become inevitable in everyone's life, Android-base applications are growing at a very fast rate. At this run, if Tamil language could be implemented in this environment it would be a great progress in Natural language processing. The main objective of this paper is to discuss how Tamil language can be deployed in an Android environment. The paper aims at discussing how a lay user can enter the contents of Tamil using English scripts and how it gets converted into Tamil font. The language used for implementation is Java. The rich API set in Java is utilized to its maximum extent and it is implemented in Android platform. The main functionality of this would be, it gives us the ease of typing Short Messages in our mother tongue. Along with this, grammar checking and font style properties are checked and set by default. All the necessary details are clearly discussed and verified throughout the paper.

Keywords- Tamil encoding, Unicode, Android App

1. INTRODUCTION

For more than half a century, immigrants from the Indian subcontinent and the West Indies have added variety and diversity to the rich patchwork of accents and dialects spoken in the UK. British colonizers originally exported the language to all four corners of the globe and migration in the 1950s brought altered forms of English back to these shores. Since that time, especially in urban areas, speakers of Asian and Caribbean descent have blended their mother tongue speech patterns with existing local dialects producing wonderful new varieties of English, such as London Jamaican or Bradford Asian English.
To be more specific the impact of English language for globalization let most of the emerging technologies to be in its specific nature. Although English is spread worldwide, people all over the world feel convenient to use their own mother tongue which they practiced from their childhood.

2. REASON FOR MAKING E→T ANDROID APP

Smart phones have revolutionized the cellular world and has become an inevitable part of people's life. Most of the smart phones run on android kernel. The reason, why smart phone users prefer android OS is because of its vast applications which makes their task easier and is saves time. Most Android applications are designed such that it meets almost all user requirements[1].
In the same way, users all around the world find it convenient to send text messages in their mother tongue. Android phones do not support all languages spoken in different parts of the world. So users started typing their vernacular language in English script which occasionally led to perplexion. It would be better if there is an application that transliterates the English scripts to the corresponding scripts of their mother tongue [2]. For better description users with their mother tongue Tamil requires an app that transliterates the English scripts to the corresponding Tamil scripts [3].

3. TRANSLITERATION

Transliteration refers to the conversion of scripts written in one language to another language. not the translation that is finding the meaning of a word in one language to converting to another language. For this the dictionary from English to Tamil, Tamil to English is required for references.

3.1. HOW TRANSLITERATION OCCURS

For the transliteration, a separate program is to be written. In Transliteration is associated with the encoding. As far as Tamil is concerned each script is assigned a specific unicode. The letters of one script and the corresponding English scripts are archived in a property file if it is coded in Java. The encoding happens when Tamil script is typed in English letters. English scripts are actually encoded in ASCII 7 bit encoding. But Tamil and other native languages are encoded in ASCII 16. The encoded English scripts are matched with the encoded Tamil scripts so that the corresponding Tamil word is retrieved.

4. ENCODING

It is the process of converting data into particular format required for program compilation and execution, data transmission, storage, Compression/Decompression, file conversion. Encoding is the process of assigning particular code for letters, symbols and numbers to data for conversion into equivalent cipher. The standard encodings used are

- ASCII(American Standard code for Information Interchange)
- Unicode
- Tamil All Character Encoding

4.1. ASCII

ASCII encoding schemes are usually for the files with the content in text. This assigns a separate number for each character. The printable and the non printable characters of the ASCII code represents uppercase, lowercase, letters, symbols, punctuation marks, numbers. The character positions in the ASCII character encoding is 0 to 127. The representations after 127 from 128 to 255 cannot be represented with the ASCII encoding. To solve this problem Unicode encoding was introduced.

4.2. UNICODE

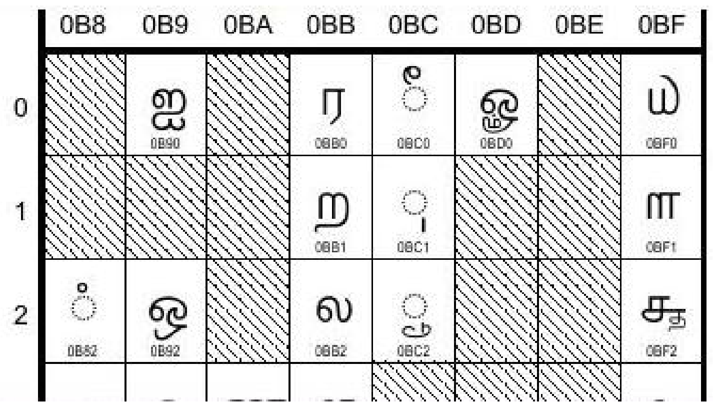
Computers and Operating system can understand only binaries. Unicode was devised to be a system capable of storing encoded representations of the plain text character of every human language that has ever been existed. The first step in the Unicode representation is the formation of the Universal Character Set by setting out all the characters of the language [4]. The codepoints (plain text character in Unicode Language) is mapped to code values (encoded text in the Unicode Language) with the UTF and UCS.

The persistence of the Unicode encoding is because there is no other encoding techniques that could contain enough characters to represent a single language. The main drawback is that there occurs several code for representing same characters and the same code for different characters. To overcome this and maintain consistency one significant encoding should be globalized [5]. For this purpose the Unicode encoding representation was considered and had undergone several revisions in its features.

Unicode characters										
BMP		SMP		SIP		SSP	PUA			
0000–0FFF	8000–8FFF	10000–10FFF	18000-18FFF	20000–20FFF	28000–28FFF	E0000–E0FFF	F0000–F0FFF	F8000–F8FFF	100000–100FFF	108000–108FFF
1000–1FFF	9000–9FFF	11000–11FFF	19000-19FFF	21000–21FFF	29000–29FFF		F1000–F1FFF	F9000–F9FFF	101000–101FFF	109000–109FFF
2000–2FFF	A000–AFFF	12000–12FFF	1A000-1AFFF	22000–22FFF	2A000–2AFFF		F2000–F2FFF	FA000–FAFFF	102000–102FFF	10A000–10AFFF
3000–3FFF	B000–BFFF	13000–13FFF	1B000-1BFFF	23000–23FFF	2B000–2BFFF		F3000–F3FFF	FB000–FBFFF	103000–103FFF	10B000–10BFFF
4000–4FFF	C000–CFFF	14000-14FFF	1C000-1CFFF	24000–24FFF	2C000–2CFFF		F4000–F4FFF	FC000–FCFFF	104000–104FFF	10C000–10CFFF
5000–5FFF	D000–DFFF	15000-15FFF	1D000–1DFFF	25000–25FFF	2D000–2DFFF		F5000–F5FFF	FD000–FDFFF	105000–105FFF	10D000–10DFFF
6000–6FFF	E000–EFFF	16000–16FFF	1E000–1EFFF	26000–26FFF	2E000–2EFFF		F6000–F6FFF	FE000–FEFFF	106000–106FFF	10E000–10EFFF
7000–7FFF	F000–FFFF	17000-17FFF	1F000–1FFFF	27000–27FFF	2F000–2FFFF		F7000–F7FFF	FF000–FFFFF	107000–107FFF	10F000–10FFFF

Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), WML, etc., and is the official way to implement ISO/IEC 10646. It is supported in many operating systems, all modern browsers, and many other products. The emergence of the Unicode Standard, and the availability of tools supporting it, is among the most significant recent global software technology trends. Use of legacy character set is not preferable as it may lead to high cost. Instead incorporating the Unicode character set into the client-server machine or websites would significantly reduce the cost. Unicode comes handy in enabling a software product or a website to be active in the global market without being re-engineered[6]. It also supports transmission of data through various systems without corruption.

4.2.1 CODE POINTS



	0B8	0B9	0BA	0BB	0BC	0BD	0BE	0BF
0	▨	ஐ 0B90	▨	ர 0BB0	ீ 0BC0	ௐ 0BD0	▨	௰ 0BF0
1	▨	▨	▨	ற 0BB1	ு 0BC1	▨	▨	௱ 0BF1
2	ஂ 0B82	ஒ 0B92	▨	ல 0BB2	ூ 0BC2	▨	▨	௲ 0BF2

5. TAMIL ALL CHARACTER ENCODING

Tamil All Character encoding is the encoding specially for the Tamil Language. As our App deals with the transliteration of the Tamil language written in English scripts to the corresponding Tamil scripts it is better way to go with the TACE.

6. ANALYSIS OF UNICODE WITH TACE

- Tamil characters are 247 in number including all mei, uyirmei and ayutham. But there occurs only 31 positions in Unicode for representing Tamil characters. This led to the situation of using the same code for different characters which leads to ambiguity. There occurs lot of confusions with different words in Tamil typed in English especially in the mei This means that only 10% of the Tamil Characters have the corresponding code in the Standard Unicode Representation.

- The sequence of all the characters in the Unicode representation is not in the natural order of sequence. This requires the complex collation algorithm for ordering them in sequence.
- It encodes 23 Vowel-Consonants (23 consonants + Ü) and calls them as consonants, against Tamil grammar.
- Unnatural for Speech to Text/Text to Speech.
- Inefficient to store, transmit and retrieval (For example, File reading and writing, Internet, etc.).Complex processing hinders development.
- Need normalization for string comparison.
- For the Unicode representation the characters are counted more than the actual characters present. This way of counting more and processing includes more processing and consumes more time and the CPU cycles.
- As Tamil is the complex language more complex processing is required. So many of the characters are not supported in Unicode as it deals with the encoding in all the languages.

6.1. ADVANTAGE OF TACE OVER UNICODE

- Tamil All Character encoding is unique for Tamil language. Therefore it includes all character in Tamil represented as the separate code.
- As there are separate code for all the characters of Tamil language each letters are identified with the unique code. Hence unambiguous
- There is no ambiguity in encoding which naturally makes TACE preferable.
- A computing machine takes less processing cycle when encoded with TACE rather than Unicode encoding. Text encoded in TACE can be efficiently parsed using simple arithmetic operations. Basic Tamil grammar is followed in TACE but not in Unicode because it needs extra framework development.

6.2. METHODS FOR TAMIL ALL CHARACTER ENCODING

6.2.1. METHOD 1(BY SIMPLE ARITHMETIC OPERATIONS)

க் + இ = கி

E210(க்) + E203(இ) = 1C413

1C413 - E200(Constant) = E213(கி)

6.2.2. METHOD 2

க்(E210) + இ(E203) = கி(E213)

E210 (க்) | (E203(இ) & 000F(Constant)) = E213(கி)

To check whether a character is vowel:

((c >= E201) && (c <= E20C)) == True // => Vowel

To check whether a character is consonant:

x = (c & '000F (Constant)')

((x == 0) && ((c > E200) && (c < E390))) == True // => Consonant

To check whether a character is Vowel-consonant (UyirMei):

x = (c & '000F (Constant)') // => Unique number for each vowel starting from 1

```
      ( ( ( x >= 1 ) && ( x <= 12 ) ) && ( ( c >= E211 ) && ( c < E38D ) ) ) == True
     // => Vowel-Consonant(UyirMei)
```

To check whether a character is Tamil number:

```
      x = (c &‘000F (Constant)' )
      ((c & 'E18F(Constant)' == c )&& ( x <= 12 ) ) == True // => Tamil Number
```

7. CONCLUSION

Android-base applications are growing at a very fast rate. At this run, if Tamil language could be implemented in this environment it would be a great progress in Natural language processing. The main objective of this paper is to discuss how Tamil language can be deployed in an Android environment. The paper aims at discussing how a lay user can enter the contents of Tamil using English scripts and how it gets converted into Tamil font. The language used for implementation is Java. The rich API set in Java is utilized to its maximum extent and it is implemented in Android platform. The main functionality of this would be, it gives us the ease of typing Short Messages in our mother tongue.

8. REFERENCES

- Mike McGrath, *Building Android Apps in Easy Steps: Using App Inventor*, 2012
- Lee Wei-Meng, *Beginning Android 4 Application Development*, 2012
- http://www.unicode.org/faq/tamil.html
- http://www.alanwood.net/unicode/tamil.html
- http://tamil.somasundaram.us/ta-unicode.html
- http://www.unicode.org/