LANGUAGE TRANSLITERATION IN INDIAN LANGUAGES – A LEXICON PARSING APPROACH

SUBMITTED BY

JISHA T.E.
Assistant Professor,
Department of Computer Science,
Mary Matha Arts And Science College,
Vemom P O, Mananthavady

A Minor Research Project Report

Submitted to

University Grants Commission
SWRO, Bangalore

ABSTRACT

Language, ability to speak, write and communicate is one of the most fundamental aspects of human behaviour. As the study of human-languages developed the concept of communicating with non-human devices was investigated. This is the origin of natural language processing (NLP). Natural language processing (NLP) is a subfield of Artificial Intelligence and Computational Linguistics. It studies the problems of automated generation and understanding of natural human languages. A 'Natural Language' (NL) is any of the languages naturally used by humans. It is not an artificial or man-made language such as a programming language. 'Natural language processing' (NLP) is a convenient description for all attempts to use computers to process natural language. The goal of the Natural Language Processing (NLP) group is to design and build software that will analyze, understand, and generate languages that humans use naturally, so that eventually you will be able to address your computer as though you were addressing another person. The last 50 years of research in the field of Natural Language Processing is that, various kinds of knowledge about the language can be extracted through the help of constructing the formal models or theories. The tools of work in NLP are grammar formalisms, algorithms and data structures, formalism for representing world knowledge, reasoning mechanisms. Many of these have been taken from and inherit results from Computer Science, Artificial Intelligence, Linguistics, Logic, and Philosophy.

Natural language communication with computers has long been a major goal of artificial intelligence, both for the information it can give about intelligence in general, and for practical utility. There are many applications of natural language processing developed over the years. They can be mainly divided into two parts, Dialogue based applications and Text-based

applications. Some of the typical examples of Dialogue based applications are answering systems that can answer questions, services that can be provided over a telephone without an operator, teaching systems, voice controlled machines (that take instructions by speech) and general problem solving systems. Text based involves applications such as searching for a certain topic or a keyword in a data base, extracting information from a large document, translating one language to another or summarizing text for different purposes and transliterating one language to another. Transliteration is helpful for many applications, such as Machine Translation (MT), Cross Language Information Retrieval (CLIR) and Information Extraction (IE), etc. There are two directions of transliteration: forward and backward. Forward Transliteration is the representation of the glyphs of a source script by the glyphs of a target script. In our description, source script is Malayalam and target script is English. Backward Transliteration is the process whereby the glyphs of a target script are transliterated into those of the source script.

First chapter is the introductory chapter of the thesis. It includes the major definitions, terms and algorithms. This chapter includes also the study of Natural language processing (NLP) as a subfield of Artificial Intelligence and Computational Linguistics.

In the second chapter of the thesis investigator presents the related literature survey in the topic of study. For collecting the literature effort has been taken to study the important text books and research papers containing terminology, definitions and algorithms.

The third chapter describes the details of the procedures adopted for the study. The chapter is divided into the following sections: overview of the project, Creation of the database, steps for Forward Transliteration, steps for Backward Transliteration and Parsing Stream of Characters into Literals and algorithms for developing the dicode (both forward and backward).

In the fourth chapter the investigator developed an algorithm for forward and backward transliteration, which is listed below. The algorithm for forward transliteration consists of mainly three steps. They are algorithm for isolating Malayalam words in to group of phonetic units, algorithms for Malayalam to HRR and algorithm for HRR to Destination Language English. The algorithm developed for backward transliteration consists of three steps namely; algorithm for Parsing Stream of Characters into Literals, algorithm for English to HRR and algorithm for HRR to Destination Language Malayalam. This chapter also includes the study of transliteration where we segment a Malayalam word into glyphs and then converted in to HRR of Malayalam based on the English transliteration of the Malayalam word. Then map these HRR to the corresponding English equivalent from the English dictionary. For backward transliteration, we segment a English word into glyphs and then converted in to HRR of English based on the Malayalam transliteration of the English world. Then map these HRR to the corresponding Malayalam equivalent from the Malayalam dictionary. The chapter also includes a graphical analysis of the algorithm.

The fifth chapter discusses directions for further research in the selected topic. In this chapter the investigator proposed and developed a model for forward and backward transliterate glyphs from Malayalam to English and English to Malayalam. We use Hepburn Romanization Representation system as the basic platform in this model. Because of the similarities between phonetic units among Indian languages, the method proposed in this work can be enhanced for transliteration between any Indian language and English. Promising results of our experiments suggest our method will be helpful to several applications, such as MT, CLIR, IE, etc. There is scope for further research to include more sophisticated transliteration model allowing insertion and deletion, and thereby establishing a more powerful language model with larger context and better smoothing. Also more research on the noise robustness and analyzing the performance of the developed algorithm under various *training set* and *test set* are left for further investigation. Investigator hope and expect that the algorithm and the method developed in the thesis may help the society, especially in government and non-government offices in the near future for transliteration of one language to another.

CONTENTS

Chapter	Title	Page
I	**INTRODUCTION**	
	1.1 LANGUAGE TRANSLITERATION	3
	1.2 TRANSLITERATION, TRANSLATION AND TRANSCRIPTION	4
	1.3 FORWARD AND BACKWARD TRANSLITERATION	4
	1.4 OBJECTIVE OF THE PROPOSED SYSTEM	5
II	**REVIEW OF RELATED LITERATURE**	
	2.1 NATURAL LANGUAGE PROCESSING	9
	2.2 LANGUAGE TRANSLITERATION	11
	2.3 STUDIES ON MALAYALAM LANGUAGE	16
	2.4 MALAYALAM PHONTICS	17
	2.5 DIFFERENT TERMS AND DEFINITIONS RELATED TO THE STUDY	21
	2.6 HEPBURN ROMANIZATION RPRESENTATION	24
III	**METHODOLOGY**	
	3.1 OVERVIEW OF THE PROJECT	25
	3.2 CREATION OF THE DATABASE	26
	3.3 STEPS IN FORWARD AND BACKWARD TRANSLITERATION	30
	3.4 PARSING STREAM OF CHARACTERS IN TO LITERALS	32
IV	**EXPERIMENTS AND RESULTS**	
	4.1 ALGORITHMS FOR FORWARD AND BACKWARD TRANSLITERATION	34
	4.2 ANALYSIS OF THE RESULT	46
V	**CONCLUSION AND SCOPE**	42
	BIBLIOGRAPHY	
	APPENDIX - SCREEN SHOT	

INTRODUCTION

Language, ability to speak, write and communicate is one of the most fundamental aspects of human behaviour. As the study of human-languages developed the concept of communicating with non-human devices was investigated. This is the origin of natural language processing (NLP).Natural language processing (NLP) is a subfield of Artificial Intelligence and Computational Linguistics. It studies the problems of automated generation and understanding of natural human languages. A 'Natural Language' (NL) is any of the languages naturally used by humans. It is not an artificial or man-made language such as a programming language. 'Natural language processing' (NLP) is a convenient description for all attempts to use computers to process natural language. The goal of the Natural Language Processing (NLP) group is to design and build software that will analyze, understand, and generate languages that humans use naturally, so that eventually you will be able to address your computer as though you were addressing another person. The last 50 years of research in the field of Natural Language Processing is that, various kinds of knowledge about the language can be extracted through the help of constructing the formal models or theories. The tools of work in NLP are grammar formalisms, algorithms and data structures, formalism for representing world knowledge, reasoning mechanisms. Many of these have been taken from and inherit results from Computer Science, Artificial Intelligence, Linguistics, Logic, and Philosophy.

Natural language communication with computers has long been a major goal of artificial intelligence, both for the information it can give about intelligence in general, and for practical utility. There are many applications of natural language processing developed over the years. They can be mainly divided into two parts; Dialogue based applications and Text-based applications. Some of the typical examples of Dialogue based applications are answering systems that can answer questions, services that can be provided over a telephone without an operator, teaching systems, voice controlled machines (that take instructions by speech) and general problem solving systems. Text based involves applications such as searching for a certain topic or a keyword in a data base, extracting information from a large document, translating one language to another or summarizing text for different purposes and transliterating one language to another. Transliteration is helpful for many applications, such as Machine Translation (MT), Cross Language Information Retrieval (CLIR) and Information Extraction (IE), etc. There are two directions of transliteration: forward and backward.

Forward Transliteration is the representation of the glyphs of a source script by the glyphs of a target script. In our description, source script is Malayalam and target script is English. Backward Transliteration is the process whereby the glyphs of a target script are transliterated into those of the source script. Here, English to Malayalam. In this work, the algorithm describes for Forward Transliteration will convert the given Malayalam text into Hepburn Romanization Representation symbols and transformation to English. For Backward Transliteration the algorithm describes to convert the English text in to Hepburn Romanization Representation symbols and transformation to Malayalam. The method proposed in this work can be enhanced for transliteration between any Indian language and English. This task made easy because of the similarities between phonetic units among Indian languages. There are a large number of peoples who cannot read script in other Indian languages than their mother tongue. In such a context, transliteration will help them to formulate a representation of words in one language using the alphabet of another language. Transliterated texts are often used in emails, blogs, and electronic correspondence. Transliteration is also used for simple encryption. The proposed project is severely practical one: civil servants, agriculturalists, administrators, businessmen, educationalists, industrialists and many others have to read documents and have to communicate in languages they do not know. Language transliteration/backward transliteration plays an important role in many multilingual speech and language applications.

1.1 Language Transliteration

There are two components for NLP, applied and theoretical. The applied component of NLP is more interested in the practical outcome of modeling human language use. The goal is to create software products that have knowledge of human language. The Language Transliteration of a text from one language to another language is one of such problem.

Transliteration is the practice of transcribing a word or text written in one writing system into another writing system or system of rules for such practice. From a linguistic point of view, transliteration is a mapping from one system of writing into another, word by word, or ideally letter by letter. Transliteration attempts to be exact, so that an informed reader should be able to reconstruct the original spelling of unknown transliterated words. To achieve this objective, transliteration may define complex conventions for dealing with letters in a source script which do not correspond with letters in a goal script.

Transliteration is the process of formulating a representation of words in one language using the alphabet of another language. This is also called cross language information retrieval (CLIR), in which query expressed in a source language is used to retrieve information represented in another target language. Names of people, places, and companies etc., ie, proper nouns are by far the most frequent targets in queries. Contemporary dictionary-based translation techniques are ineffective for proper noun translation. New foreign names appear almost daily; and they become unregistered vocabulary in the dictionary. Unknown language seriously affects translation as well as retrieval of information. Transliterations can be bidirectional.

Given a word pair (m, e) where ‘m’ is the original word in one language and ‘e’ is the transliterated word of ‘m’ in another language. Forward transliteration is the process of phonetic conversion of ‘m’ to ‘e’; and backward transliteration is generating possible candidates and determining correct ‘m’ given ‘e’. Transliteration systems are normally mapping of templates between the phonemes of target and source scripts.

1.2 Transliteration, Translation and Transcription

Transliteration is opposed to transcription, which specifically maps the sounds of one language to the best matching script of another language. Still, most systems of transliteration map the letters of the source script to letters pronounced similarly in the goal script, for some specific pair of source and goal language. If the relations between letters and sounds are similar in both languages, a transliteration may be (almost) the same as a transcription.

Also, transliteration should not be confused with translation, which involves a change in language while preserving meaning. Transliteration performs a mapping from one alphabet into another. In a broader sense, the word transliteration is used to include both transliteration in the narrow sense and transcription. Anglicizing is a transcription method. Romanization encompasses several transliteration and transcription methods.

1.3 Forward and Backward Transliteration

Forward / backward transliteration plays an important role in many multilingual speech and language applications. The phonetic translation from the native language to foreign language is defined as forward transliteration; conversely, the process of recalling a word in native language from a transliteration is defined as backward transliteration.

Forward Transliteration is the transliteration of a foreign name (in the case of our proposed study, Malayalam) into English. Typically, there are several acceptable transliteration candidates. For example, the Malayalam word "പ്രതികൂലം", might correctly be transliterated to "prathikoolam". Backward Transliteration is the reverse transliteration process used to obtain the original form of an English name that has already been transliterated into the foreign language. In this case, English to Malayalam. For Example, the word "prathikoolam" be transliterated to "പ്രതികൂലം".

1.4 Objective of the proposed System

In many natural language processing tasks, such as multilingual named entity and term processing, machine translation, corpus alignment, cross lingual information retrieval and automatic bilingual dictionary compilation, transliteration has become an indispensable component. Transliterations in the narrow sense are used in situations where the original script is not available to write down a word in that script, while still high precision is required. For example, traditional or cheap typesetting with a small character set; editions of old texts in scripts not used any more; some library catalogues. Transliteration in the broader sense is a necessary process when using words or concepts expressed in a language with a script other than one's own. Transliterating words and names from one language to another is a frequent and highly productive phenomenon.

Transliteration is a useful tool for any student of a language that uses a different alphabet to the student's native language. So English speakers who are learning Russian or Greek will find they need to use transliteration. Travellers to tourist orientated parts of these countries will see signs in the English alphabet, which have also undergone transliteration. Transliteration helps a student to pronounce words that at first look impossible to pronounce because of the odd looking characters. It is not exactly the same as transcription, where the sounds of the words are written down using the English alphabet, although commonly the term transliteration is used to describe both systems.

For a new learner of a language, transliteration is vital. Without transliteration, new speakers would struggle to learn how words are to be pronounced. Even if their teacher taught them how to say all of the words they came across, they would be stuck when they read a new one because they would have no reference point for how the letters were supposed to sound. Transliteration can also be used for other uses apart from teaching languages. If someone wants to type a letter in Russian or Greek on an English keyboard, they may use transliteration. This way they will not have to press a complicated sequence of keys to get the Cyrillic or Greek alphabet.

Transliteration is so useful to language learners that they would be lost without it. In fact, some people who are learning Mandarin Chinese for example, only ever learn Pinyin, which is the form of the language written in the Roman script. Although this is not the best way to be completely fluent, especially if you plan on ever reading or writing, it is a way of learning that will enable you to hold conversations and understand others without worrying about the complicated Mandarin script. And in countries like Greece, many road signs and other useful words are written in the transliterated text. This way, a tourist who does not speak Greek can at least say the word of the thing they are looking for, rather than having to point to it in a book.

So-called “transliteration” is to translate a word in one language into a word with a similar pronunciation in another language. For instance, a transliteration method is often used in translating a proper name. Previously, people usually use a bilingual lexicon to translate proper name. Such a bilingual lexicon (e.g., a bilingual proper name lexicon) is compiled by linguists or specialists in related fields, which has very high accuracy.

However, even a very large bilingual lexicon cannot cover the whole vocabulary, very often people would encounter a case in which a wanted word cannot be found in a lexicon. Furthermore, with the development of time and society, new words are emerging continuously, making the situation even worse. Therefore, for a long time, people need a method and apparatus for automatic transliteration to realize automatic transliteration between two languages. Such an automatic transliteration technology is also important to machine translation, cross language information retrieval and information extraction.

REVIEW OF RELATED LITERATURE

The new research direction of Natural Language based knowledge representation and reasoning systems emerged over the past few years. It grew out of concerns over the efficient handling of large-scale, general-purpose knowledge, reasoning, and the meaning of natural language. One motivation for this research was - and still is – the fact that a vast majority of knowledge representation and reasoning systems do not adequately reflect important characteristics of natural language and are representationally and inferentially impoverished relative to natural language. Over the course of a decade of research, and large scale development, Lucja M. Iwanska [16] reached that natural language is a powerful, general – purpose knowledge – representation system. He developed a formal, computational model, the UNO model of a natural language that closely simulates many of its uniquely advantageous representational and inferential characteristics. The UNO model is fully implemented as a large scale natural language processing system capable of processing knowledge from the real life corpora of thousands of textual documents in a number of domains.

According to Akshar Bharati, Vineet Chaitanya and Rajive Sangal [2], the goal of Natural Language Processing is to build computational models of natural language for its analysis and generation. First, there is technological motivation of building intelligent computer systems such as machine translation systems, transliteration systems, natural language interfaces to databases, man-machine interfaces to computers in general, speech understanding systems, text analysis and understanding systems, computer aided instruction systems, systems that read and understand printed or handwritten text. Second, there is a cognitive and linguistic motivation to gain a better insight into how humans communicate using natural language (NL).

Keh-Yih [13] says in all the applications of NLP, language is serving a communicative function. For example, in natural language interfaces to databases, users communicate their information need by means of natural language query. In the context of machine translation and transliteration, the writer wants to convey something to his readers through text. This is not surprising because the primary function of natural language is communication. A consequence of the above is that NLP focuses on the study of language as a means of communication. The transliteration between two languages is considered to be one of the areas of interest for scholars working in the area of Natural Language processing.

Number of Universities and colleges of national and international status have already implemented transliteration methods and investigations towards these directions are of great interest. National Institutes like IIT’s, NIT’s, Universities, Industries and other well-reputed institutes, are taken interest in this area of Natural language processing and Language Transliteration.

2.1 Natural Language Processing

Akshar Bharati, Vineet Chaitanya and Rajive Sangal [2] gives some example applications of NLP. Computers have been widely used to store and manage large amounts of data. The data might pertain to railway reservation, library, banking, management information, and so on. Normally, to use these systems, specialized computer knowledge is necessary. The goal of natural language interfaces (NLI) is to remove this barrier. The user is expected to interact in natural language (by means of a keyboard and a screen). LIFER by Hendrix (1978) and INTELLECT by Harris (1977) were some of the early systems.

Some general NL interfaces to computers have also been developed. UC, short for UNIX consultant, developed at Berkeley (Wilenskey, 1982) assists a new user to UNIX operating system. In case of a problem the user can seek its assistance. It engages him/her in a dialogue and tries to tell him/her what to do. It uses scripts and knowledge about user's goals and plans.

Several systems have been built as research vehicles in NLP which answer questions about a domain. LUNAR by Woods (1977) was an early system that answered questions about the moon rocks. It makes a sophisticated analysis of quantification in the NL sentences.

There are question – answering systems which, given a story in a specified domain, answer questions about it; for example, about going to restaurants. Much of this work was carried out at Yale under Schank and Abelson (1977). They firmly established the need for domain knowledge, lots of it, for understanding. They also made interesting models of goals, intentions and plans among human beings.

There has been much renewed interest in machine translation (MT) since the early 80s. There have been several large efforts: Eurotra for European languages, Mu for Japanese and English, KBMT between English and Japanese at Carnegie Mellon University, Anusaraka among Indian languages at IIT Kanpur, etc.

Encarta uses Microsoft Research Group's [6] technology to retrieve answers to user questions; Intellishrink uses natural language technology to compress cellphone messages; Microsoft Product Support uses their machine translation software to translate the Microsoft Knowledge Base into other languages.

2.2 Language Transliteration

Transliteration is the writing or spelling of words or letters in another alphabet. Transliteration is the practice of transcribing a word or text written in one writing system into another writing system or system of rules for such practice. From a linguistic point of view, transliteration is a mapping from one system of writing into another, word by word, or ideally letter by letter. Transliteration attempts to be exact, so that an informed reader should be able to reconstruct the original spelling of unknown transliterated words. To achieve this objective, transliteration may define complex conventions for dealing with letters in a source script which do not correspond with letters in a goal script.

The idea of transliteration is not new. For more than a century, printed books used a suitable transliteration scheme with Roman letters and diacritics to display text in Indian scripts. Early attempts at using transliteration with computers started with TeX, where ASCII letters were used to specify the aksharas of Indian languages. TeX has the potential to display a nearly complete set of aksharas on account of its ability to deal with fonts which could in principle have as many as 250 Glyphs.

Transliteration History

1885 [*See* 32, 33, 34, 35] — The American Library Association [ALA] creates a system for representing Cyrillic characters. No diacritics are used. (e.g. zh, kh, tch, sh, shtch, ye [for jat], yu, ya) Reverse transliteration is not considered.

1898 [*See* 29, 34, 35] — The Prussian Instructions (Preussische Instruktionen [PI]) are created, which use a system of transliteration based on the Croatian model (with diacritics).

1905 [See 29, 30, 32, 33, 34, 35] — Library of Congress creates their system, which is virtually identical to what is used today.

1909 [*See* 29, 30, 31, 32, 33, 34,35]— The ALA and British Library Association [BLA] allow for two systems, the ALA system and one based on Croatian.

1917 [See 29, 30, 31, 32, 33, 34, 35] — The British Academy creates its own system. Like many other systems. It does not take into account reverse transliteration.

1930s [See 29, 30, 31, 34, 35] — Central European and Scandinavian countries adopt the Prussian Instructions [PI]. This system was based on the Croatian model. Exceptions were made for German speaking countries, where "ch" was used instead of "h" for Cyrillic "x"
In France the Bibliotheque Nationale adopts a purely phonetic rendering following French spelling conventions (transcription rather than transliteration).

1953 [See 31, 32, 33, 34, 35] — The British Royal Society [BRS] creates another system, covering Russian, Serbian & Bulgarian (but not Ukrainian, Macedonian or Belorusian). It uses only two diacritical marks — for "?" and "?". It is closer to the LC system (minus many of the diacritics), but with "ya" and "yu" for "?" and "?"

1954 [See 29, 35] — The International Organization for Standardization [ISO] creates IS0/R9. Based on Croatian, this transliteration system is very close to the PI system.

1959 [*See* 29, 30, 31]— The British Standards Institution [BS/BSI] rejects ISO/R9 (because of its reliance on multiple diacritics) and comes up with its own system: BS 2979. Very close to the British Royal Society system. (This system is used by Chemical Abstracts).

1968 [See 29, 33, 34, 35] — ISO/R9:1968 is relaxed to allow for the ANSI and BS 2979 systems (in certain countries).

1976 [See 29, 32, 33, 34] — The American National Standards Institute [ANSI] publishes their system, nearly identical to the BSI system.

1995 [See 29, 30, 34, 35] — ISO/R9:1995 reverts to to its initial standards, doing away with allowing "ch" or "kh" for Cyrillic "x."

During the past several years, different methods have been introduced to prepare Indian language documents by entering the text through specific transliteration schemes. Data entry through transliteration is quite close to phonetic mapping of Indian language characters to the letters of the Roman alphabet. Notable among these methods are:

The ITRANS [29] package developed by Avinash Chopde makes use of an approach for printing documents through LATEX. ITRANS now has the support for several Indian languages but the transliteration scheme is not uniform for all Indian languages. However it is a highly recommended package for printing documents. Substantial number of Sanskrit and Hindi documents has been developed using ITRANS.

The RIT [39] package developed by RamaRao Kanneganti and Ananda Kishore enabled to prepare Telugu text. RIT is not unlike ITRANS but offers greater flexibility during data entry. RIT also relies on LATEX to get an output. A large number of Telugu documents (Poetry and Texts) are available in RIT format.

The ADAMI and ADHAWIN [35] packages of Dr. Srinivasan are exclusively for Tamil and are based on the principle of using TrueType fonts to view documents under Windows or the Mac. They constituted one of the earliest approaches (1995) to dealing with Tamil text on a PC.

The MYLAI [36] fonts software developed by Dr. Kalyana Sundaram of the Swiss Federal Institute of Technology, is again for use with TAMIL. Data entry for this scheme is based on Transliterated Tamil using Roman letters but the transliteration scheme is different from that of Adhawin.

A review of the archives of Indian language documents on the net reveals several other schemes of Transliteration and fonts. The Indology [40] site in England has electronic texts of Sanskrit Documents prepared in CSX format, a special input method recommended in 1990 for Sanskrit data entry using a Dos feature called Code page switching.

The Tamil archives of the Institute of Indology and Tamil Studies in Germany [41] (IITS) has an archive of texts of Tamil Sangam literature and many Sanskrit documents. These archives are based on the transliteration scheme recommended by the University of Madras, a fairly well known and accepted standard.

The Mahabharata and Ramayana [36] texts have been prepared by Prof. Muneo Tokunaga of Kyoto university in Japan and the massive amount of effort that has gone into preparing the archives deserves special mention as also the dedication and patience with which the project was undertaken and completed.

Mehdi M. Kashani, Fred Popowich, School of Computing Science, Simon Fraser University & Fatiha, Sadat[21], National Research Council of Canada, provided a brief introduction to the transliteration problem, and highlighting some issues specific to Arabic to English translation, a three phase algorithm is introduced as a computational solution to the problem on their work titled "THE CHALLENGE OF ARABIC FOR NLP/MT, Automatic Transliteration of Proper Nouns from Arabic to English".

In "Direct Orthographical Mapping for Machine Transliteration" paper, a novel framework for machine transliteration/*bac*k-transliteration that allows us to carry out direct orthographical mapping (DOM) between two different languages is presented by ZHANG Min, LI Haizhou, SU Jian [22,23,24],Institute for Infocomm Research 21 Heng Mui Keng Terrace, Singapore 11961.

Guo Yuqing, Wang Haifeng, [18,24,25]Toshiba (China) Research and Development Center 5/F, Tower W2, Oriental Plaza, No.1, East Chang An Ave., Dong Cheng District Beijing, 100738, addresses the problem of backward translating person name from Chinese to its English counterpart on their paper titled "Chinese-to-English Backward Machine Transliteration".

Om Transliteration mapping scheme [37, 38, and 42]: This transliteration scheme is designed as an improvement over the ITRANS scheme for typing Indian language characters using the standard keyboard.

Varamozhi [42] is a set of software programs that enable your computer to read and write Malayalam. For writing these programs use transliteration: you can type in Manglish and you will see in real Malayalam. Transliteration scheme used by Varamozhi is called Mozhi. It uses a unique English character sequence for each Malayalam letter.

2.3 Studies on Malayalam Language

Malayalam is a Dravidian language closely related to Tamil [*See* 36, 37, 42], although it is more influenced by Sanskrit than the latter. Speakers of

Malayalam are called Malayalis. It is estimated that the ancestral language that gave rise to both Tamil and Malayalam split sometime in the 9th century AD, giving rise to Malayalam as a language distinct from Tamil. The earliest writings in Malayalam are from the end of the 9th century, and the first literary text dates to 1125–1250 AD. The early literature of Malayalam included classical songs and poetry. Modern Malayalam literature is rich in poetry, fiction, drama, biography, and literary criticism.

Malayalam is spoken by 35 million people primarily in the state of Kerala and in the Laccadive Islands in southern India. It is one of the 22 official languages of India. It is also spoken in Bahrain, Fiji, Israel, Malaysia, Qatar, Singapore, United Arab Emirates, and United Kingdom. Today, Malayalam is coming into its own as the language of administration and as the medium of instruction in schools and colleges.

In the early thirteenth century /vattezhuthu/ (round writing) [*See* 9, 19, 37], traceable to the pan-Indian brahmi script, gave rise to the Malayalam writing system, which is syllabic in the sense that the sequence of graphic elements means that syllables have to be read as units, though in this system the elements representing individual vowels and consonants are for the most part readily identifiable. Malayalam now consists of 53 letters including 20 long and short vowels and the rest consonants. The earlier style of writing is now substituted with a new style from 1981. This new script reduces the different letters for typeset from 900 to less than 90. This was mainly done to include Malayalam in the keyboards of typewriters and computers.

English stands only second to Sanskrit in its influence in Malayalam. Hundreds of individual lexical items and may idiomatic expressions in modern Malayalam are of English origin. As the language of administration and as the medium of instruction in schools and colleges, Malayalam is coming into its own. A scientific register in the language is slowly evolving.

2.4 Malayalam Phonetics

Phonetic characters are usually falls in the groups, Vowels, Consonants, Consonant Vowel units, conjunct consonants and Chillukal [*See* 9, 19, 28, 36, and 42]. Table 2.4.1 to 2.4.8 mentioned combinations are the maximum possible elements.

Table 2.4.1

Vowels (V) which is a set at the most it consist of 15 elements.

അ ആ ഇ ഈ ഉ ഊ ഋ എ ഏ ഐ ഒ
ഓ ഔ അം അഃ

Vowels in Malayalam

Table 2.4.2

Consonants (C) 36 elements are there in this class. They are usually represented as /k//g//t/…..

ക് ഖ് ഗ് ഘ് ങ്
ച് ഛ് ജ് ഝ് ഞ്
ട് ഠ് ഡ് ഢ് ണ്
ത് ഥ് ദ് ധ് ന്
പ് ഫ് ബ് ഭ് മ്
യ് ര് ല് വ്
ശ് ഷ് സ് ഹ്
ള് ഴ് റ്

Consonants in Malayalam

Table 2.4.3

Conjunct Consonants, this is a combination of two consonant units. Maximum value of this set is 36*36=1296 but most of these combinations are absent, means few hundred combinations are commonly used.

ക്ക ഗ്ഗ ങ്ക ങ്ങ ച്ച ഞ്ച ഞ്ഞ ട്ട ണ്ട ത്ത ത്ഥ
ദ്ദ ന്ത ന്മ പ്പ മ്പ മ്മ ള്ള സ്സ ബ്ദ ഹ്ന ന്ധ
ജ്ജ ന്ഥ ശ്മ ത്മ ഗ്ന ന്റ ന്ന ല്ല റ്റ ശ്ശ ബ്ബ

some examples for conjunct consonant in Malayalam

Table 2.4.4

Chillukal” are special set of alphabets in Malayalam. This set consist 5 letters. All other languages that have not “chillukal” are filled with reliable alphabet with similar pronunciation.

ൺ ൻ ർ ൽ ൾ
“Chillukal” in Malayalam

Table 2.4.5

Symbols in Malayalam. Each vowel has its own symbol representation. In addition to this, Malayalam has also some symbols with consonants.

ാ ി ീ ു ൂ ൃ െ േ ൗ
ം ഃ ്യ ൄ
്ര ൈ
Symbols in Malayalam

Table 2.4.6

The table given below contains all the possible symbols in Malayalam with its corresponding vowel or the consonant units. Vowel and its symbol representation

Vowel	Symbol
അ	
ആ	ആ
ഇ	ഇ
ഈ	്യ
ഉ	ഉ
ഊ	ഷ
ഋ	ൃ
എ	എ
ഏ	ഏ
ഐ	എെ
ഒ	െ/C/ാ
ഓ	േ/C/ാ
ഔ	യ
അം	ട
അഃ	ഃ

Here /C/ indicates the position of the consonant unit in the case of Consonant Vowel unit with ഒ and ഓ, left and right side of the consonant unit contains the symbols. Malayalam has also some symbols for representing conjunct consonants; this will be used with consonant unit. Symbol used to representing Conjunct Consonant.

Table 2.4.7

്വ	്യ	്ര
വ്	യ്	റ്

Eg.

ക്വ -----> ക് + വ് + അ

ക്ര -----> ക് + റ് + അ

ക്യ -----> ക് + യ് + അ

Also group these symbols by occurrence of the position of these symbols with consonant unit. That is,

1. Pre Consonant symbols

2. Post Consonant symbols
3. Infix Consonant symbols

In first group, Consonant unit preceded by the symbols.

Eg. കാ

In second group, consonant unit succeeded by the symbol, ie, first symbol and then Consonant unit.

Eg. കെ

Third group Consonant unit are in between the symbols. ie, left and right are symbols and Consonant are in the middle of the symbols. This will occur only for Consonant unit with Vowel ഒ and ഓ.

Eg. കൊ and കോ

‘്യ‘ and ‘⌟‘ are Pre Consonant symbols and ‘ ⌞‘ is Post Consonant symbols.

From the above discussed five groups of characters in Malayalam namely, Vowels, Consonants, Conjunct Consonants, Chillukal and Symbols, and in addition to these phonetic characters there are some special symbols in Malayalam. The combination of the Vowel units gives Consonant Vowel units but which are usually not connected with “chandrakkala” instead of some special symbols can be used.

Table 2.4.8

Consonant Vowel units (CV), it is formed from the combination of a consonant unit with vowel (eg. /ka/=/k/+/a/ and /gi/=/g/+/i/). In this set we can expect at the most 540 elements.

ക കാ കി കീ കു കൂ

കൃ കെ കേ കൊ കോ

കൗ കം കഃ

Some examples of consonant vowel units

In short, Malayalam language consists 15 Vowels, 36 Consonants, Conjunct Consonants, 5 “Chillukal” and Symbols for representing Consonant Vowel.

2.5 Different Terms and definitions related to the study.

Following are the terms and definitions that we are use in developing the software [42].

Letter:

Letters are the basic phonetic building blocks of a language. Along with the sound it has one or more graphical forms also.

Literal:

Literal is a sequence of letters which has a single fused graphical form when written either independently or combined with any other letters (ex: *`n'*, *`nn'* and *`yu'*). Usually it includes all letters and all conjunct forms consisting of two or more letters. Literals representing the letters are called base literals and those representing the conjunct forms are called derived literals. Size of a literal is the number of characters being used to represent that literal in the given transliteration scheme. |x| operator denotes the size of the literal x with respect to that scheme. A literal can also be classified as vocal or consonantal depending on whether it can be pronounced independently or not.

Concept of literals is closely linked to fonts or some code for information interchange in general. Whether a character sequence has a single fused graphical form or not, can be decided only by analyzing that code.

Glyph:

Glyph refers to the graphical form of a literal.

Text:

It refers to the sequence of characters representing transliterated Malayalam. It can also be interpreted as a sequence of literals and symbols represented by that character stream.

Script:

Script is a sequence of glyphs denoting a text.

Conjunct:

It is a sequence of literals such that all the literals other than the last are consonantal literals. Last literal can be either vocal or consonantal. (eg: In the conjunct *`sva'*, *`s'* is the first literal, *`v'* is the second and *`a'* is the third. More over, *`svar'* is not a conjunct because the vocal literal *`a'* does not come as the last literal.)

Context:

Context of a literal refers to its neighbourhood in a text. There are two different contexts for a literal. One is the lexical context and other is the phonetical context. By default context means phonetical context.

Different Glyphs of a Literal:

A literal assumes different glyphs depending on its context. They are classified as following:

Independent glyph

A vocal literal assumes independent glyph in null context. (eg: graphical form to represent a single *`o'* sound). Any consonant literal X assumes independent glyph in the conjunct X + *`a'*. (eg: graphical form of *`ka'*)

Sign glyph or Partial glyph

It is a Graphical form of the literal when it appears last in a two literal conjunct. (eg: graphical form of *`o'* in *`ko'*; form of *`ya'* in *`kya'* and form of *`na'* in *`sna'*). It can have two parts which come on left and right of the independent glyph of the first literal. They are called "left sign glyph" and "right sign glyph" respectively. (eg: in *`ko'*, *`o'* has both sign glyphs. In *`kya'*, *`y'* has only right sign glyph and in *`kra'*, *`r'* has only left sign glyph). Moreover, all the vocal literals except *`a'* have sign glyphs.

Chillu glyph

Only *`N'*, *`n'*, *`m'*, *`r'*, *`l'*, *`L'* and *`rr'* have single fused glyph in null context. Those glyphs are called chillu glyphs. (In old orthography *`k'* and *`y'* also had chillu glyphs.

Transliteration:

It is the representation of the glyphs of a source script by the glyphs of a target script. In our description, source script is Malayalam and target script is English.

Backward transliteration:

It is the process whereby the glyphs of a target script are transliterated into those of the source script (English to Malayalam).

2.6 Hepburn Romanization Representation

Romanization is a system [10, 20, and 43] used to write a language with the Roman alphabets. The earliest Japanese Romanization system was based on the orthography of Portuguese. It was developed around 1548 by a Japanese Catholic named Yajiro. The systems used today all developed in the latter half of the 19th century.

The first system to be developed was the Hepburn system, developed by James Curtis Hepburn for his dictionary of Japanese words and intended for foreigners to use. There are a number of different Romanization systems in use: the three main ones are Hepburn, Kunrei-shiki (ISO 3602), and Nihon-shiki (ISO 3602 Strict). Hepburn (long-vowel omitted) is the most widely used. Modified Hepburn, which uses a macron to indicate some long vowels and an apostrophe to note the separation of easily confused syllables, is widely used in Japan and among foreign students and academics.

The Hepburn Romanization system was devised by an American missionary doctor in the 1860s to transcribe the sounds of the Japanese language into the Roman alphabet (in Japanese, “Romaji"). It is widely used today both in the English-speaking world and in Japan, where many younger people are most familiar with the Roman alphabet through the study of English and thus find its spelling conventions more comfortable than the official Monbusho Romanization standard. Hepburn generally follows English phonology and so gives the best indication to Anglophones of how a word is pronounced in Japanese. It was standardized as American National Standard System for the Romanization of Japanese, but this status was abolished on October 6, 1994.

METHODOLOGY

This chapter deals with details of the procedures adopted for the study. The chapter is divided into the following sections: overview of the project, Creation of the database, steps for Forward Transliteration, steps for Backward Transliteration and Parsing Stream of Characters into Literals and algorithms for developing the dicode (both forward and backward).

We discuss all the existing terminology and important techniques we adopt to develop the software and Algorithm.

3.1 Overview of the Project

The Project has been developed using MS Access as Back-End and Visual C++ 6.0 as Front-End. Microsoft Visual C++ has always been one of the most comprehensive and sophisticated software developments available. Visual C++ is known for its outstanding flexibility. You literally do anything with it without contorting your code into a mass of spaghetti. Microsoft Access is an application used to create small and midsize computer desktop databases for the Microsoft Windows family of operating systems. In this project, the Open Database Connectivity (ODBC) is used to connect Visual C++ Program to access data from the database. ODBC is one of the database interface technologies that Microsoft has introduced. The project uses the Akruthi Malayalam Multifont Engine –Licensed to Malayalam Software tools MCIT 1.0.0.8 presented by Ministry of Communications and Information Technology, Government of India for Malayalam Unicode compliant open type fonts for source language Malayalam.

This project takes text input in Malayalam alphabets and produces output in English for forward transliteration. But in the backward transliteration the source language is English and target language is Malayalam. When you transliterated in to the Roman alphabet, the process is called Romanization. First the source language is converted in to its Hepburn Romanization representation, which is the common platform and searching for pattern in HRR dictionary, transliteration techniques ranged from simple mapping routines and context based searches and parsing. After converting source language to HRR and then converted in to the destination language. Here takes place a detailed study in different languages and which characters corresponded to which consonants and vowels, and how different combinations of them yielded different pronunciations. This information was intended for transliteration.

The design process involves,

1. Creation of dictionary for Malayalam
2. Creation of HRR dictionary for Malayalam
3. Creation of the dictionary for English
4. Creation of HRR dictionary for Malayalam
5. Various Steps in Forward and Backward Transliteration
6. Developing Algorithms for Forward and Backward Transliteration

3.2 Creation of the Database

The major steps involved in the Database design are

1. Creation of dictionary for Malayalam
2. Creation of HRR dictionary for Malayalam
3. Creation of the dictionary for English
4. Creation of HRR dictionary for Malayalam

Using Table 2.4.1 to 2.4.8, the Investigator designed to establish the Malayalam dictionary which consists of all combinations of alphabets. After development of this dictionary the investigator developed the HRR dictionary for Malayalam. HRR dictionary contains all corresponding representation of Malayalam Phonetic unit. In Malayalam language, each character has its own phonemes. Phonemes are the smallest units of the sound system of a language. So the conversion from Malayalam to Romanization is very simple. Romanization is based on standard language pronunciation.

Usually phonetic units are classified as vowels and consonants. In addition of vowels and consonants Malayalam language contains “Chillukal” and conjunct consonants. “Chandrakkala” is used to connect consonant + vowel (for consonant vowel unit) and consonant + consonant + vowel (for conjunct consonants). In this project, translation to Romanization is based on English pronunciation. Malayalam pronunciation always has either one the following structures.

- a vowel
- a consonant
- a consonant + a vowel
- a consonant + a consonant + a vowel
- “chillukal”

The following tables 3.2.1 to 3.2.5 contain Malayalam alphabets and corresponding Romanization representation.

Table 3.2.1: Vowel and its HR Representation

Vowel	HR rep
അ	a
ആ	A
ഇ	i
ഈ	I
ഉ	u
ഊ	U
ഋ	Ru
എ	e
ഏ	E
ഐ	ai
ഒ	o
ഓ	O
ഔ	au
അം	aM
അഃ	aH

Table 3.2.2: Consonants and its HR representation

ക്	K	ഖ്	K	ഗ്	g	ഘ്	G	ങ്	~g
ച്	C	ഛ്	C	ജ്	j	ഝ്	J	ഞ്	~j
ട്	T	ഠ്	Th	ഡ്	D	ഢ്	Dh	ണ്	N
ത്	T	ഥ്	th	ദ്	d	ധ്	Dh	ന്	n
പ്	P	ഫ്	P	ബ്	b	ഭ്	B	മ്	m

യ്	Y	ര്	r	ല്	l	വ്	v
ശ്	S	ഷ്	Sh	സ്	s	ഹ്	h
ള്	L	ഴ്	zh	റ്	R	-	-

Table 3.2.3: Chillukal and its HR representation

Mal	HRR
ൺ	N^
ൻ	n^
ർ	r^
ൽ	l^
ൾ	L^

Table 3.2.4: Consonant + Vowel unit

Vowel	HRR of V	Consonant+ Vowel	HRR of CV
അ	a	ക	ka
ആ	A	കാ	kA
ഇ	i	കി	ki
ഈ	I	കീ	kI
ഉ	u	കു	ku
ഊ	U	കൂ	kU
ഋ	Ru	കൃ	kRu
എ	e	കെ	ke
ഏ	E	കേ	kE
ഐ	ai	കൈ	kai
ഒ	o	കൊ	ko
ഓ	O	കോ	kO
ഔ	au	കൌ	kau
അം	aM	കം	kaM
അഃ	aH	കഃ	kaH

The combination of vowel unit with consonant units gives the consonant vowel units but which are not connected with “chandrakkala”, instead of that some special symbols are used. This table representing a consonant unit ക് with all combinations of vowel units.

Table 3.2.5 contains some examples for conjunct consonant.

Two consonant units are combined to form conjunct consonant. The conjunction of two consonant units can be done with the help of a special symbol called “chandrakkala” (്).

Malayalam conjunct consonant	HRR	Division of conjunct consonant
ക്ക	kka	ക്+ക
ത്ത	tta	ത്+ത
ന്ന	nna	ന്+ന
മ്മ	mma	മ്+മ
ള്ള	LLa	ള്+ള
ത്സ	tsa	ത് +സ
ന്ഥ	ntha	ന് +ഥ
ത്ഭ	tBa	ത് +ഭ
ന്ധ	ndha	ന് +ധ
ത്ഥ	ttha	ത്+ഥ
ഹ്ന	hna	ഹ് +ന
ക്ത	kta	ക് +ത
ബ്ബ	bba	ബ് +ബ
ച്ച	cca	ച്+ച

Let the word ‘അനുകൂലം’ and corresponding Romanization representation is ‘anukUlaM’.

According to the above tables of representation of Romanization of Malayalam letters,

‘അ’ is converted to ‘a’

നു -----> nu,

കൂ -----> kU and

ലം ----->laM

From these Malayalam and corresponding HRR dictionaries the investigator developed the Destination language dictionary for all combinations of Malayalam glyphs. For Backward transliteration developed a HRR dictionary for destination language, English.

3.3 Steps in Forward and Backward Transliteration

This project consists of four files;

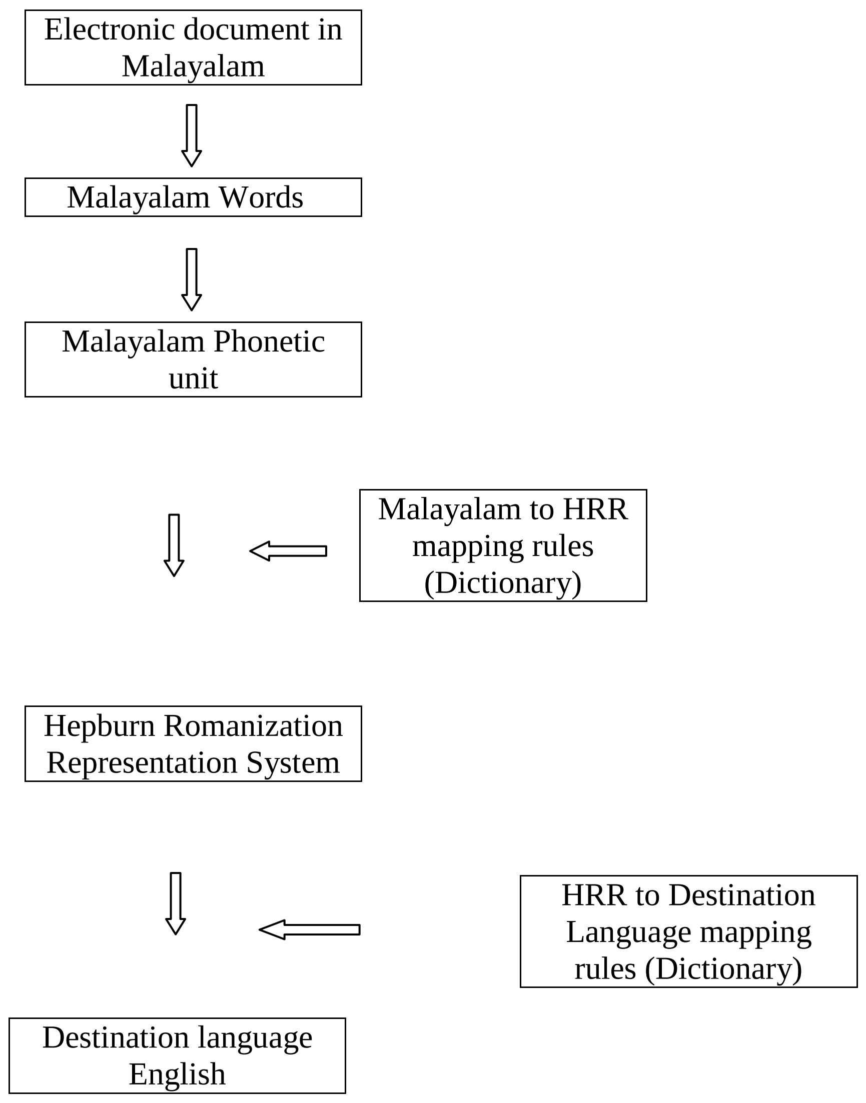
1. Malayalam file: contains Malayalam words
2. MalHrr: contains HRR of Malayalam alphabets.
3. English file:contains english words equivalent to Malayalam words
4. EngHrr: contains HRR of English alphabets.

3.3.1 Steps for forward Transliteration

Forward transliteration has 5 steps.

1. Assign Malayalam words in to a String array named Mal [].
2. Get the length of the array using the function Mal.Getlength();
3. Set the type of the letters in the database.
   CONSONANT=0;
   VOWELS=1;
   SYMBOLS=2
   CHILLUKAL=3
   OTHERS=4
4. Isolate Malayalam words in to groups of phonetic units.
5. Find the HRR (Hepburn Romanization System) of or glyphs of Malayalam phonetic units.
6. Find the English (destination language) phonetic units equivalent to HRR of Malayalam.

Fig 3.3.1.1 Architecture of the forward transliteration

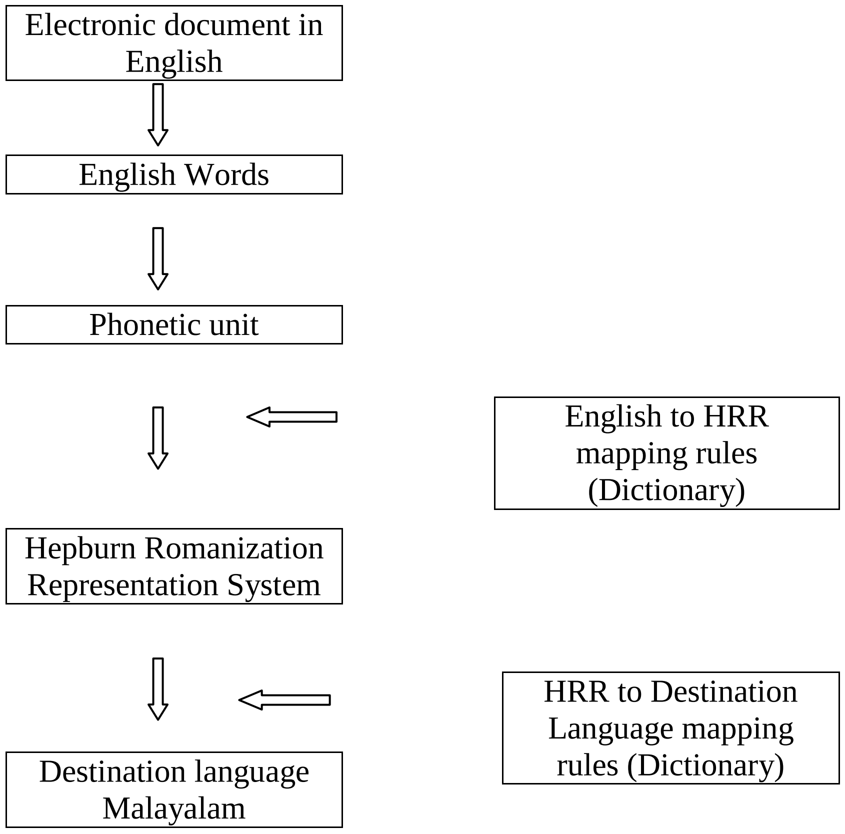

Electronic document in Malayalam
Malayalam Words
Malayalam Phonetic unit
Malayalam to HRR mapping rules (Dictionary)
Hepburn Romanization Representation System
HRR to Destination Language mapping rules (Dictionary)
Destination language English

3.3.2 Steps for Backward Transliteration

1. Assign English words in to a String array named Eng[].
2. Get the length of the array using the function Eng.Getlength().
3. Set the type of the letters in the database.
   CONSONANT=0;
   VOWELS=1;
4. Parsing the character stream of transliterated Malayalam text into sequence of literals

5. Generating the glyph or find the HRR of sequence corresponding to sequence of literals
6. Find the Malayalam (destination language) phonetic unit equivalent to HRR of English.

Fig 3.3.2.1 Architecture of the backward transliteration



Electronic document in English
English Words
Phonetic unit
English to HRR mapping rules (Dictionary)
Hepburn Romanization Representation System
HRR to Destination Language mapping rules (Dictionary)
Destination language Malayalam

3.4 Parsing Stream of Characters into Literals

Reverse transliteration has mainly two steps. Parsing the character stream of transliterated Malayalam text into sequence of literals and then generating the glyph sequence corresponding to the sequence of literals by looking at the context of each literal.

In the model we use, sequence of one or more characters represent each Malayalam letter. Hence there arises the problem of splitting the character stream representing Malayalam text into corresponding literals. For example, if the stream is *`thn'*, we can view it as *`t'* + *`hn'* or *`th'* + *`n'* or *`t'*+*`h'*+*`n'* or *`thn'*. We have to choose correct one from these different options. The generalised rule is as follows:

Let the character stream S be $a_1 + a_2 + ... + a_n$ where a_i ($1 <= i <= n$) is a character.We can consider the non-trivial case where S can be split in two ways as $S_1 = x_1 + x_2 + ... + x_p$ and $S_2 = y_1 + y_2 +...+ y_q$ such that $x_1 != y_1$ and $x_p != y_q$ where $x_i(1<=i<=p)$ and $y_i(1<=i<=q)$ are character sequences representing single literals. Without loss of generality S_1 will be chosen if either of the following conditions is true:

1. $p = 1$ and $q > 1$
2. $p > 1$ and $q > 1$ and x_p and y_q are base literals and $| x_p | > | y_q |$

Next step is to generate the glyph sequence from the sequence of literals obtained.

EXPERIMENTS AND RESULTS

4.1 Algorithms for Forward and Backward transliteration

The investigator developed an algorithm for forward and backward transliteration, which is listed below. The algorithm for forward transliteration consists of mainly three steps. They are algorithm for isolating Malayalam words in to group of phonetic units, algorithms for Malayalam to HRR and algorithm for HRR to Destination Language English. The algorithm developed for backward transliteration consists of three steps namely; algorithm for Parsing Stream of Characters into Literals, algorithm for English to HRR and algorithm for HRR to Destination Language Malayalam.

4.1.1 Algorithms for Forward Transliteration

I. Algorithm for isolating Malayalam words in to group of phonetic units.

```
1. Start.
2. len=Mal.Getlength();
3. Cstring h, i=0;
4. Repeat through step6 until(i<len)
5. i=i+1
6. Return the type of the letters in the array Mal[] using
getType(Mal[i]).
7. if  Mal[i]=VOWEL
           h=firstVowel();
   else if Mal[i]=SYMBOL
           h=firstSymbol();
   else if Mal[i]= CONSONANT
           h=firstConsonant();
   else if Mal[i]=CHILLUKAL
           h=findHrr(mal[i]);
```

```
        else Mal[i]=OTHERS && Mal[i]!= ‘ ് ‘
                h=Mal[i]
    8.  Stop.
```

II. Algorithms for Malayalam to HRR

II.1 Algorithm for Mal[i] is a vowel

```
    1.  Start.
    2.  if Mal[i]=VOWEL && Mal[i+1]= ം, ഃ, ൗ, ാ then
                return HRR(VOWEL+SYMBOL);
        else return HRR( VOWEL);
    3.  Stop.
```

II.2 Algorithm for Mal[i] is a symbol

```
    1.  Start.
    2.  if Mal[i]= െ, േ, ൈ && Mal[i+1]= CONSONANT
                return HRR( SYMBOL+ CONSONANT);
        else if if Mal[i]= െ, േ && Mal[i+1]= െ, ൈ &&
        Mal[i+2]= CONSONANT
                return HRR(SYMBOL+ SYMBOL+CONSONANT);
        else if Mal[i]= െ, േ && Mal[i+1]= CONSONANT &&
        Mal[i+2]= ാ, ൃ, ⌋
                return HRR(SYMBOL+ CONSONANT+ SYMBOL);
    else if Mal[i]= െ, േ && Mal[i+1]= െ, ൈ && Mal[i+2]=
CONSONANT && Mal[i+3]= ാ, ൃ, ⌋
                return HRR(SYMBOL+ SYMBOL +CONSONANT+
        SYMBOL);
```

```
            else if  Mal[i]= െ, േ && Mal[i+1]= CONSONANT  &&
Mal[i+2]= ു, ൂ && Mal[i+3]= ാ
            SYMBOL +
                    return HRR(SYMBOL+ CONSONANT+ SYMBOL+
            SYMBOL) ;
            else return HRR(SYMBOL+ SYMBOL+ SYMBOL+
CONSONANT);
        3. Stop.
```

II.3 Algorithm for Mal[i] is a Consonant

```
        1.  Start
        2.  if Mal[i]= CONSONANT && Mal[i+1]= SYMBOL &&
            Mal[i+2]= SYMBOL
            return HRR(CONSONANT+ SYMBOL+ SYMBOL) ;
            else if Mal[i]= CONSONANT && Mal[i+1]= SYMBOL
            return HRR(CONSONANT+ SYMBOL) ;
                    else return HRR( CONSONANT)+ ‘a’ ;
        2   Stop.
```

II.4 Algorithm for Mal[i] is a Chillukal

```
        1.  Start
        2.  if Mal[i]= CHILLUKAL
                    return HRR(CHILLUKAL);
        3.  Stop.
```

II.5 Algorithm for Mal[i] is Others

```
        1.  Start
        2.  if Mal[i]= OTHERS && Mal[i]!= ‘ ് ‘
                    return Mal[i];
        3.  Stop.
```

III Algorithm for HRR to Destination Language English

```
        1.  Start
```

```
2.    len= hrr[i].Getlength();
3.   Repeat through step4 until (i=VOWEL)
4.            if MalHrr [i]=  Eng[i]
                   return Eng[i];
 5.  Stop
```

4.1.2 Algorithms for Backward Transliteration

I. Algorithm for Parsing Stream of Characters into Literals

```
1.   Start.
2.   len=Eng.Getlength();
3.   Cstring h, i=0;
4.   Repeat through step6 until ((i==VOWEL|| i == (len - 1))
5.   i=i+1
6.   Return the type of the letters in the array Eng [] using
getType(Eng[i]).
7.   if Eng[i]=VOWEL
             h=findVowelHrr();
     else if Eng[i]=CONSONANT
             h=findConsonantHrr();
8.  Stop.
```

II. Algorithm for English to HRR

II.1 Algorithm for HRR of Vowel

```
1. Start
2. if Eng[i]= Eng.CompareNoCase(vowel) == 0 && getType ==
1)
             return EngHrr[i];
3.   Stop
```

II.2 Algorithm for HRR of Consonant

```
1. Start
```

```
2. if Eng[i]= Eng.CompareNoCase(Consonant) == 0 && getType
== 0)
            return EngHrr[i];
3. Stop
```

III Algorithm for HRR to Destination Language Malayalam

```
1.    Start
2.     len= EngHrr[i].Getlength();
3.    if EngHrr[i]=Mal[i]
      return Mal[i];
5.    Stop.
```

4.2 Analysis of the result

In this paper, for forward transliteration, we segment a Malayalam word into glyphs and then converted in to HRR of Malayalam based on the English transliteration of the Malayalam word. Then map these HRR to the corresponding English equivalent from the English dictionary. For backward transliteration, we segment an English word into glyphs and then converted in to HRR of English based on the Malayalam transliteration of the English world. Then map these HRR to the corresponding Malayalam equivalent from the Malayalam dictionary. These mappings are illustrated in Figure 4.2.1 and Figure 4.2.2.

അ	നു	കൂ	ലം
a	nu	kU	laM
a	nu	koo	lam
പ്ര	വാ	സി	
pra	vA	Si	
pra	vaa	si	
മൂ	ല്യ	ച്യു	തി
mU	lya	cyu	ti
moo	lya	chyu	thi

പ	ക്വ	ത	
pa	kwa	ta	
pa	kwa	tha	
വ	ല	യു	ക
va	la	yu	ka
va	la	yu	ka

Figure 4.2.1 Forward Transliteration

a	nu	koo	la	m
a	nu	kU	la	aM
അ	നു	കൂ	ല	ം
pra	vaa	si		
pra	vA	Si		
പ്ര	വാ	സി		
moo	lya	chyu	thi	
mU	lya	cyu	ti	
മൂ	ല്യ	ച്യു	തി	
pa	kwa	tha		
pa	kwa	ta		
പ	ക്വ	ത		
va	la	yu	ka	
va	la	yu	ka	
വ	ല	യു	ക	

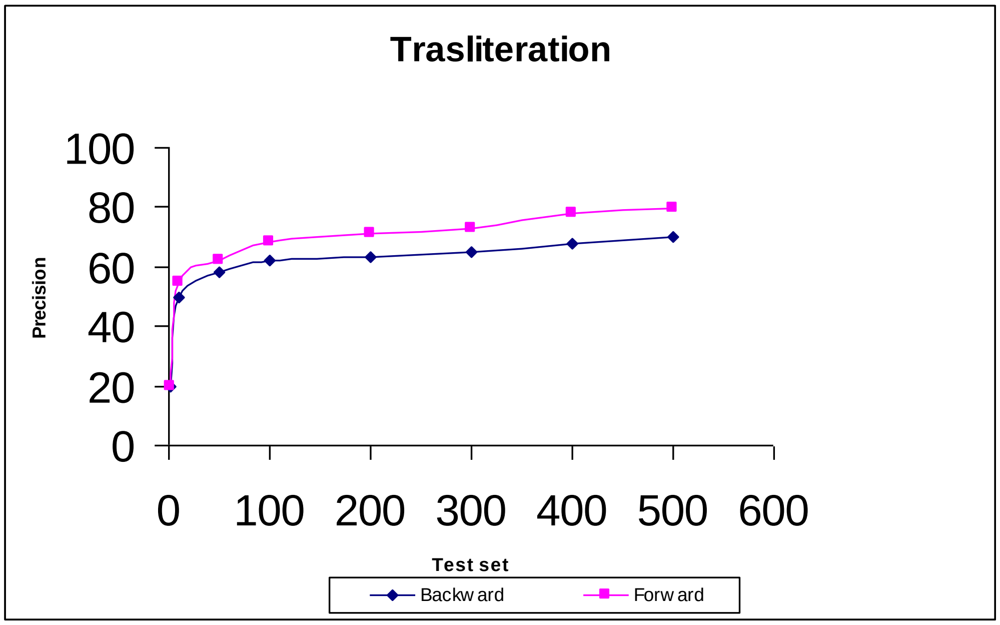
Fig. 4.2.2 Backward Transliteration

In order to evaluate the performance of transliteration results, we define the Precision of transliteration as in Equation (4.2.1).

$$\textit{Precision} = \frac{\#\text{Correct Transliterated Words}}{\#\text{ All Words}} \times 100\% \qquad (4.2.1)$$

Where, *# All Words* is the total number of words in the test set (500 pairs), and *#Correct Transliterated words* is the number of the correct transliterated words occurring in the top *n* candidates. We choose top 1, 10, 50, 100, 200, and 500 as the evaluation points.

Investigator has taken interest in analyzing of the precession test of equation 4.2.1 using graphs. In this analysis the total number of words in the test set has been taken as 500 pairs and the Correct Transliterated words as the number of the correct transliterated words occurring in the top *n* candidates and 1, 10, 50, 100, 200, and 500 as the evaluation points.



The study reveals that satisfactory results *Precision* = 79.6%, in forward transliteration and Precision =70%, in backward transliteration could be reached when chosen more candidates *(n* = *50*0). Also we can re-score the top candidates with other clues, to find the best transliterated word from the top *n* candidates with higher efficiency. Experiments also show that our transliteration process is robust when the noise is produced in the forward and backward transliteration.

Investigator strongly feel that a better algorithm can thought for further investigation as in the developed algorithm and method one may face some problems in the conversion process of certain words from English to Malayalam. For example, consider the forward transliteration, from Malayalam to English, the word ‘കൃതി’ is converted to ‘kruthi’, whereas, in backward transliteration, from English to Malayalam, the word ‘kruthi’ is converted to ‘ക്രുതി’. Such problems the investigator leaves for further scope for research.

CONCLUSION AND SCOPE

The investigator proposed and developed a model for forward and backward transliteration; transliterate glyphs from Malayalam to English and English to Malayalam. We use Hepburn Romanization Representation system as the basic platform in this model. Because of the similarities between phonetic units among Indian languages, the method proposed in this work can be enhanced for transliteration between any Indian language and English. Promising results of our experiments suggest our method will be helpful to several applications, such as MT, CLIR, IE, etc. There is scope for further research to include more sophisticated transliteration model allowing insertion and deletion, and thereby establishing a more powerful language model with larger context and better smoothing. Also more research on the noise robustness and analyzing the performance of the developed algorithm under various *training set* and *test set* are left for further investigation. Investigator hopes and expects that the algorithm and the method developed in the thesis may help the society, especially in government and non-government offices in the near future for transliteration of one language to another.

BIBLIOGRAPHY

1. AbdulJaleel, Nasreen and Leah S. Larkey, *Statistical Transliteration for English-Arabic Cross Language Information Retrieval*, CIKM 2003: Proceedings of the Twelfth International Conference on Information and Knowledge Management, New Orleans, LA (2003).
2. Akshar Bharathi, Vineet Chaitanya and Rajeev Sangal, *Natural Language Processing – A Paninian Perspective*, Prentice-Hall of India Private Limited, New Delhi (1995).
3. Al-Onaizan, Yaser and Kevin Knight, *Machine Transliteration of Names in Arabic Text*, ACL Workshop on Computational Approaches to Semitic Languages (2002).
4. Barkakati, Nabajyoti *Visual C++2 Developer's Guide Second Edition*. Sams Publishing (1995).
5. L.Berger, S.A. Della Pietra and V.J. Della Pietra, A maximum entropy approach to natural Language Processing, Computational Linguistics, March (1996), Vol. 22(1):39-72,
6. **Micro Soft research group (**Bill Dolan et.al., 2008), *Natural Language Processing*, Microsoft Corporation.
7. Bonnie G. Stalls and Kevin Knight, *Translating names and technical terms in Arabic text*, In Proceedings of 17th COLING and 36th ACL Workshop on Computational Approaches to Semitic Languages, Montreal, Canada, (1998), pp. 34-41.
8. Freeman, Andrew T., Sherri L. Condon and Christopher M. Ackerman, *Cross Linguistic Name Matching in English and Arabic: A "One to Many Mapping" Extension of the Levenshtein Edit Distance Algorithm*, HLT-NAACL, Newyork (2006).
9. Gundert, **Malayalam-English-Malayalam Dictionary**.

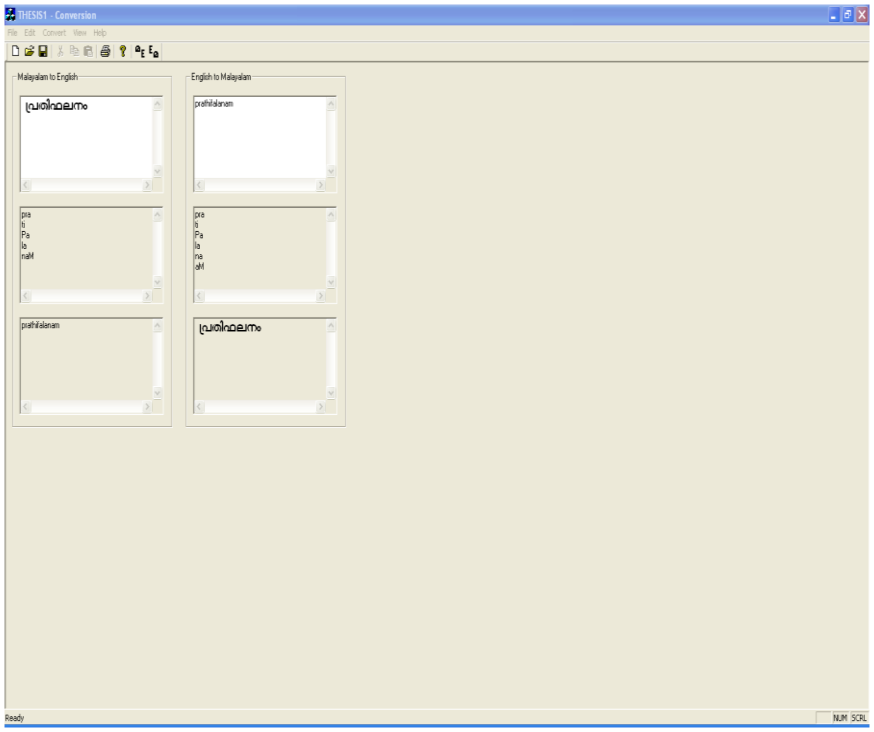
10. Hepburn System – Method of transcribing Japanese into Roman Alphabet, Japan -101, Information Resource (2005).
11. Kevin Knight and Jonathan Graehl, *Machine Transliteration.* Computational Linguistics, (1998). *Vol.*24(4): pp 599-612.
12. Kevin Knight and Jonathan Graehl, *Machine transliteration*. In Proc. of the Meeting of the European Association of Computational Linguistics, (1997), pp128–135.
13. Keh-Yih, Natural *Language Processing – IJCNLP, Springer* (2004).
14. Klementiev, Alexandre and Dan Roth, Named *Entity Transliteration and Discovery from Multilingual Comparable Corpora*, COLING-ACL, Sydney, Australia (2006).
15. Kruglinski, David J., *Inside Visual C++*. Microsoft Press (1993).
16. Lucja M Iwanska and Stuart C Shapiro, *Natural Language Processing and Knowledge Representation*, Universities Press(India) Limited, Hyderabad (2001).
17. Murray, William H. and Pappas, Chris H., *The Visual C++ Handbook Second Edition*. Osborne (1995).
18. Paola Virga and Sanjeev Khudanpur, *Transliteration of Proper Names in Cross-Lingual Information Retrieval*. In Proceedings of 41st ACL Workshop on Multilingual and Mixed-language Named Entity Recognition, Sapporo, Japan, (2003), pp57-64.
19. A.R. Rajarajavarma, Keralapaniniyam, Second Ed.
20. P. Roochnick, *Computer – based Solutions to Certain Liguistic Problems Arising from the Romanization of Arabic Names*, Ph.D Dissertation, Georgetown University, Washington, DC (1993).
21. Sadat, Fatiha, Howard Johnson, Akakpo Agbago, George Foster, Roland Kuhn and Aaron Tikuisis, *Portage: A phrase-base Machine Translation System*. In Proceedings of the ACL Workshop on Building and Using Parallel Texts, Ann Arbor, Michigan, (2005*)*.

22. Samy, Doaa, Antonio Moreno and Jose M. Guirao, *A Proposal for an Arabic Named Entity Tagger Leveraging a Parallel Corpus*, International Conference RANLP, Borovets, Bulgaria (2005).
23. R. Schwartz and Y. L. Chow, *The N-best algorithm: an efficient and exact procedure for finding the N most likely sentence hypotheses*. In Proceedings of International Conference on Acoustics Speech and Signal Processing, Albuquerque, NM, (1990) pp.81-84.
24. Sekine, S. (2002). *OAK System (English Sentence Analyzer) Version 0.1* [online].[cited 2006-7-10], <http://nlp.cs. yu.edu/oak/>.
25. Stalls, Bonnie G. and Kevin Knight, *Translating Names and Technical Terms in Arabic Text*. In Proceedings of the COLING/ACL Workshop on Computation Approaches to Semitic Languages. Sproat, Richard, Tao Tao, ChengXiang Zhai, *Named Entity Transliteration with Comparable Corpora*, COLING-ACL, Sydney, Australia (2006).
26. Stephen Wan and Cornelia Maria Verspoor, *Automatic English-Chinese name transliteration for development of multilingual resources*. In Proceedings of 17th COLING and 36th ACL, Montreal, Canada, (1998), pp.1352-1356.
27. SungYoung Jung, SungLim Hong and Eunok Paek, *An English to Korean transliteration model of extending markov window*. In Proceedings of 18th International Conference on Computational Linguistics, Saarbrucken, Germany, (2000). pp.383-389.
28. Suranad Kunjan Pillai (1965 version), Malayalam Lexicon.
29. http://www.aczone.com/.
30. http://www.ala.org/.
31. http://www.ansi.org/.
32. http://www.britac.ac.uk/portal.
33. http://www.bnf.fr/.

34. http://www.bsi-global.com/.
35. http://www.geocities.com/.
36. gopher://gopher./gopher.cc.columbia.edu.
37. http://www.kerala.org/culture/music/mal/scripts/
processQuery.cgi?song_name=alphabet
38. http://www.la-hq.org.uk/.
39. http://www.teluguworld.net/.
40. http://www.ucl.ac.uk/.
41. http://www.uni-koeln.de/.
42. http://www.varamozhi.sourceforge.net/.
43. http://Wapedia - Wiki Romanization of Japanese.htm.

APPENDIX

SCREEN SHOT



Malayalam to English

പ്രതിഫലനം

pra
ti
Pa
la
naM

prathifalanam

English to Malayalam

prathifalanam

pra
ti
Pa
la
na
aM

പ്രതിഫലനം