Transliteration based Search Engine for Multilingual Information Access

Anand Arokia Raj
Speech and Language Technology Lab
Bhrigus Software (I) Pvt Ltd
Hyderabad, India
rayar.anand@bhrigus.com

Harikrishna Maganti
Speech and Language Technology Lab
Bhrigus Software (I) Pvt Ltd
Hyderabad, India
hmaganti@bhrigus.com

Abstract

Most of the Internet data for Indian languages exist in various encodings, causing difficulties in searching for the information through search engines. In the Indian scenario, majority web pages are not searchable or the intended information is not efficiently retrieved by the search engines due to the following: (1) Multiple text-encodings are used while authoring websites. (2) Inspite of Indian languages sharing common phonetic nature, common words like loan words (borrowed from other languages like Sanskrit, Urdu or English), transliterated terms, pronouns etc., can not be searched across languages. (3) Finally the query input mechanism is another major problem. Most of the users hardly know how to type in their native language and prefer to access the information through English based transliteration. This paper addresses all these problems and presents a transliteration based search engine (inSearch) which is capable of searching 10 multi-script and multi-encoded Indian languages content on the web.

1 Introduction

India is a multi-language and multi-script country with 23 official languages and 11 written script forms. About a billion people in India use these languages as their first language. About 5% of the population (usually the educated class) can understand English as their second language. Hindi is spoken by about 30% (G. E. Burkhart, S. E. Goodman, A. Mehta and L. Press, 1998) of the population, but it is concentrated in urban areas and north-central India, and is still not only foreign, but often unpopular in many other regions.

Though considerable amount of Indic content is available on the World Wide Web (WWW), we can observe that search development is very less when compared to the official languages of the United Nations (UN). The primary reason for this can be attributed for much delayed standards and lack of support from operating systems and browsers in rendering Indic scripts. This caused web publishers to develop their own proprietory encodings/fonts, who are now hesitant to use available standards such as Unicode/ISCII. This creates a major hinderance in accessing Indian content through existing search engines.

Most of the search engines support Indic search in Unicode data only. But, considerable amount of content is available in ASCII based font encodings which is much larger (more dynamic also) than Unicode (Unicode Consortium - Universal Code Standard, 1991) or ISCII (ISCII - Indian Standard Code for Information Interchange, 1983) formats. Apart from this, language independent information like loan words, transliterated words, pronouns etc., are also not accessible across Indian languages. Most users are familiar with English keyboard typing than any Indian language, and would be interested to query through English transliteration. So, a meta standard transliteration scheme (IT3 sec3.1) has to be commonly defined across all the Indian languages, and the web content has to be appropriately converted. Also, the web pages need to be indexed using phonetic features like (diphone/triphones/syllables), which will be convenient to retrieve and rank the pages. In this paper, we incorporate all these aspects to make search engine as the meaningful searching tool for Indian languages.

The paper is organized into six sections. The first section explains the nature of Indic scripts. The second section details the various major encoding formats and transliteration scheme used to store and render Indic data. In section three, novel approaches for preprocessing Indic data like font-encoding identification and font-data conversion are explained. In section four, the experiments regarding stemming and grapheme-to-phoneme (G2P) for Indian-English using Classification and Regression Tree (CART) are described and stop words identification is also explained. The fifth section discusses the issues in developing a multi-lingual search engine for Indian languages. The sixth section explains the three possible ways to devlope a cross-lingual search engine. Finally the report and summary are included with conclusion.

2 Nature of Indic Scripts

The scripts in Indian languages have originated from the ancient Brahmi script. The basic units of the writing system are referred to as Aksharas. The properties of Aksharas are as follows: (1) An Akshara is an orthographic representation of a speech sound (2) Aksharas are syllabic in nature (3) The typical forms of Akshara are V, CV, CCV and CCCV, thus have a generalized form of C*V. The shape of an Akshara depends on its composition of consonants and the vowel, and sequence of the consonants. In defining the shape of an Akshara, one of the consonant symbols acts as pivotal symbol (referred to as semi-full form). Depending on the context, an Akshara can have a complex shape with other consonant and vowel symbols being placed on top, below, before, after or sometimes surrounding the pivotal symbol (referred to as half-form).

Thus to render an Akshara electronically, a set of semi-full or half-forms have to be rendered, which are in turn rendered using a set of basic shapes referred to as glyphs. Often a semi-full form or half-form is rendered using two or more glyphs, thus there is no one-to-one correspondence between glyphs of a font and semi-full or half-forms.

2.1 Convergence and Divergence

All Indian languages except English and Urdu share a common phonetic base, i.e., they share a common set of speech sounds. While all of these languages share a common phonetic base, some of the languages such as Hindi, Marathi and Nepali also share a common script known as Devanagari. But languages such as Telugu, Kannada and Tamil have their own scripts. The property which distinguishes these languages can be attributed to the phonotactics in each of these languages rather than the scripts and speech sounds. Phonotactics is the permissible combination of phones that can co-occur in a language.

This knowledge helps us in designing a common transliteration scheme, and also in identifying and converting different text encodings.

3 Indic Data Formats

Another aspect involved in the diversity of electronic content of Indian languages is their format of digital storage. Storage formats like ASCII (American Standard Code for Information Interchange) based fonts, ISCII (Indian Standard code for Information Interchange), Unicode and phonetic based transliteration schemes are often used to store the digital text data in Indian languages. Most of the text is rendered using some fonts of these formats.

3.1 Phonetic Transliteration Schemes

Transliteration is a mapping from one system of writing into another, word by word, or ideally letter by letter. It is the practice of transcribing a word or text written in one writing system into another writing system. 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. One instance of transliteration is the use of an English computer keyboard to type in a language that uses a different alphabet, such as Russian, Hindi etc. Transliterated texts are often used in emails, blogs, and electronic correspondence where non-Latin keyboards are unavailable, is sometimes referred to by special composite terms that demonstrate the combination of English characters and the original non-Latin word pronunciation: Ruglish, Hebrish, Greeklish, Arabish or Hinlish.

To handle diversified storage formats of scripts of Indian languages such as ASCII based fonts, ISCII and Unicode etc., it is useful and becomes essential to use a meta-storage format. ISO 15919 standards (Transliteration of Indic Scripts: How to use ISO 15919,) describes development of transliteration for Indic scripts. A transliteration scheme maps the Aksharas of Indian languages onto English alphabets and it could serve as meta-storage format for text-data. Since Aksharas in Indian languages are orthographic representation of speech sound, and they have a common phonetic base, it is suggested to have a phonetic transliteration scheme such as IT3 (Ganapathiraju M., Balakrishnan M., Balakrishnan N. and Reddy R., 2005) (Prahallad Lavanya, Prahallad Kishore and GanapathiRaju Madhavi, 2005). Thus, when the font-data is converted into IT3, it essentially turns the whole effort into font-to-Akshara conversion. Thus IT3 transliteration is used as common representation scheme for all Indic data formats. The same is used to get the input from the user also.

4 Indic Data Preprocessing

In search engine development, it is an absolute requirement that the content should be in an unique format to build a efficient index table. So, preprocessing the web content is unavoidable here. Most of the Indian language electronic data is either Unicode encoded or glyph based font encoded. Processing Unicode data is quite straight forward because it follows distinguished code ranges for each language and there is a one-to-one correspondence between glyphs (shapes) and characters. But this is not true in the case of glyph based font encoded data. Hence, it becomes necessary to identify the font encoding and convert the font-data into a phonetic transliteration sheme like IT3. The following subsections explain the stages in detail.

4.1 Font-Encoding Identification

The problem of font-identification could be defined as, given a set of words or sentences to identify the font-encoding by finding the minimum distance between the input glyph codes and the models representing font-encodings. Existing works (Anil Kumar Singh and Jagadeesh Gorla, 2007) addressed the same problem but with limited success.

Table 1: Font-Type Identification for Words.

Font Name	**Uniglyph**	**Biglyph**	**Triglyph**
Amarujala (Hindi)	100%	100%	100%
Jagran (Hindi)	100%	100%	100%
Webdunia (Hindi)	0.1%	100%	100%
Shree-Tel (Telugu)	7.3%	100%	100%
Eenadu (Telugu)	0.2%	100%	100%
Vaarttha (Telugu)	29.1%	100%	100%
E-Panchali (Tamil)	93%	100%	100%
Amudham (Tamil)	100%	100%	100%
Shree-Tam (Tamil)	3.7%	100%	100%
English-Text	0%	96.3%	100%

In this context, the proposed approach (A. A. Raj and K. Prahallad, 2007) use vector space model and Term Frequency - Inverse Document Frequency (TF-IDF) for font-encoding identification. This approach is used to weigh each term in the font-data according to its uniqueness. Thus it captures the relevancy among term and document. Here, *Term:* refers to a unit of glyph. In this work, experiments are performed with different units such as single glyph g_i (uniglyph), two consecutive glyphs $g_{i-1}g_i$ (biglyph) and three consecutive glyphs $g_{i-1}g_ig_{i+1}$ (triglyph). *Document:* It refers to the font-data (words and sentences) in a specific font-encoding.

To build a model for each font-encoding scheme, we need sufficient data. So we have collected manually an average of 0.12 million unique words per type for nearly 37 different glyph based fonts. To create a vector space model for a font-encoding, primarily the term (uniglyph or biglyph or triglyph) is extracted out of the font-data. Then TF-IDF weights are calculated for all terms in the documents.

Identification Results: The steps involved are as follows. Firstly, terms from the input word or sentence are extracted. Then a query vector using those terms is created. The distance between query vector and all the models of font-encoding is computed using TF-IDF weights. The input word is said to be originated from the model which gives a maximum TF-IDF value. It is typically observed that TF-IDF weights are more sensitive to the length of query. The accuracy increases with the increase in the length of test data. Thus, two different types

Table 2: Font-Type Identification for Sentences.

Font Name	Uniglyph	Biglyph	Triglyph
Amarujala (Hindi)	100%	100%	100%
Jagran (Hindi)	100%	100%	100%
Webdunia (Hindi)	100%	100%	100%
Shree-Tel (Telugu)	100%	100%	100%
Eenadu (Telugu)	0%	100%	100%
Vaarttha (Telugu)	100%	100%	100%
E-Panchali (Tamil)	100%	100%	100%
Amudham (Tamil)	100%	100%	100%
Shree-Tam (Tamil)	100%	100%	100%
English-Text	0%	100%	100%

of test data were prepared for testing. One is a set of unique words and the other is a set of sentences. It should also be noted that the accuracy depends on various factors: a) The number of font-encodings from which the identifier has to select one b) The inherent confusion of one font-encoding with another and c) The type of unit used in modeling. For 1000 different number of inputs (words and sentences) we have identified the closest models and calculated the accuracy. It is repeatedly done for various (uniglyph, biglyph and triglyph) categories. From Tables 1 and 2, it is clear that triglyph seems to be an appropriate unit for a term in the identification of font-encoding. It can also be seen that the performance at word and sentence level is 100% with triglyph.

4.2 Font-Data Conversion

The problem of font-data conversion could be defined as a module whose input is sequence of glyph codes and whose output is a sequence of Aksharas (characters) of Indian languages.

Existing methods and solutions proposed by (Himanshu Garg, 2005) (Khudanpur S. and Schafer C., Devanagari Converters, 2003) lack in, a) Framing a generic methodology or algorithm for conversion of font-data of all Indian languages b) Since glyph codes are manually constructed, 100% accurate conversion is achievable c) Existing methods requires large amount of effort for each font-encoding d) Large number of rules have to be written for rule based system e) Large parallel corpora has to be prepared for training f) They don't exploit shape and positional information of the glyphs, thus reducing accuracy in conversion process.

Exploiting Position and Shape Information: (A. A. Raj and K. Prahallad, 2007) Characters in Indian languages change their shape where they appear (top, bottom, left, right) in the script. In this work, an unambiguous glyph code mapping is done by introducing a simple alphanumeric scheme where the alphabets denote the corresponding phoneme and the number denotes the glyph position. We followed IT3 phonetic notations and the position numbers as described below. Glyphs which could be in a) pivotal (center) position are referred by code 0/1. b) left position of pivotal symbol are referred by code 2. c) right position of pivotal symbol are referred by code 3. d) top position of pivotal symbol are referred by code 4. e) bottom position of pivotal symbol are referred by code 5.

Training: First in the training, a font-encoding for a language is selected and a glyph-map table is prepared by hand-coding the relation between glyphs (suffixed with their position and shape information) and IT3 notations. In the second stage, a simple set of glyph assimilation rules are defined (Multi-Lingual Screen Reader and Processing of Font-data in Indian Languages,). We iterated through the following steps until there are minimal errors on held-out test set of words. Results are checked for errors using human evaluation. If errors are found then the rules are updated or redefined. The above process is repeated for 3 different font-encodings of different font-families of the chosen language.

Evaluation: While testing, a new font from the same language is selected and a glyph-mapping table is prepared. It has to be noted that for new font, we don't update or add any glyph assimilation rules, and thus we use the existing rules obtained during training phase. A random set of 500 words from that font-data is picked-up. The conversion accuracy is evaluated using human evaluation. We have built converters for 10 Indian languages and 37 different font-encodings. The evaluations results in Table 3 indicate that the font-data conversion performs consistently above 99% for a new font-encoding across languages except for Telugu. Thus in our approach the effort of building rules is limited to three different fonts of a language to build the converter. To add a new font, only glyph-map table is required and no more repetition of rule building process.

In Table 3, we can observe inferior performance for Telugu. It is due to the number of glyphs and their possible combinations are higher than other languages. Also it is common for all Indian languages that the pivotal character glyph comes first and other supporting glyphs come next in the script. But in Telugu the supporting glyphs may come before the pivotal glyph which creates ambiguity in forming assimilation rules.

5 Experiments and Discussion

In this section, the experiments performed to build the tools/modules are explained. Most of them used the CART tool to train and test. These modules/tools are integrated and used for development of the proposed search engine.

5.1 CART (Classification and Regression Tree)

CART is a decision tree procedure introduced by Breiman et al., in 1984. CART uses an exhaustive, recursive partitioning routine to generate binary splits that divide each parent node into two child nodes by posing a series of yes-no questions. CART searches for questions that split nodes into relatively homogenous child nodes. As the tree evolves, the nodes become increasingly more homogenous, identifying segments. The basic CART building algorithm is a greedy algorithm which chooses the locally best discriminatory feature at each stage in the process.

Stop Parameter: The stop parameter specifies the minimum number of samples necessary in the training set before a question is hypothesized to distinguish the group. Normally with smaller stop value the model may become over-trained. The optional stop value may differ for different datasets of different languages.

Predictee: In a given feature set, the feature that is to be predicted as the output is termed as the predictee. By default, the first feature in the feature-set is taken as the predictee, but always the predictee can be specified while giving the description of the data. Some times CART is over-fit with training data, which may reduce the performance.

Feature Selection: Many experiments were conducted for different problems like grapheme to phoneme conversion (G2P) for English (Indian-

Table 3: Font-Data Conversion Results (Precision Values).

Language	**Font Name**	**Training / Testing**	**Result**
Hindi	Amarujala	Training	99.2%
	Jagran	Training	99.4%
	Naidunia	Training	99.8%
	Webdunia	Training	99.4%
	Chanakya	Testing	99.8%
Marathi	ShreePudhari	Training	100%
	ShreeDev	Training	99.8%
	TTYogesh	Training	99.6%
	Shusha	Testing	99.6%
Telugu	Eenadu	Training	93%
	Vaarttha	Training	92%
	Hemalatha	Training	93%
	TeluguFont	Testing	94%
Tamil	ElangoValluvan	Training	100%
	ShreeTam	Training	99.6%
	ElangoPanchali	Training	99.8%
	Tboomis	Testing	100%
Kannada	Shree Kan	Training	99.8%
	TTNandi	Training	99.4%
	BRH Kannada	Training	99.6%
	BRH Vijay	Testing	99.6%
Malayalam	Revathi	Training	100%
	Karthika	Training	99.4%
	Thoolika	Training	99.8%
	Shree Mal	Testing	99.6%
Gujarati	Krishna	Training	99.6%
	Krishnaweb	Training	99.4%
	Gopika	Training	99.2%
	Divaya	Testing	99.4%
Punjabi	DrChatrikWeb	Training	99.8%
	Satluj	Training	100%
Bengali	ShreeBan	Training	97.5%
	hPrPfPO1	Training	98%
	Aajkaal	Training	96.5%
Oriya	Dharitri	Training	95%
	Sambad	Training	97%
	AkrutiOri2	Training	96%

English) and stemming. These experiments were conducted with different possible features and stop values. Features for English G2P conversion were manually prepared for each letter and for stemming, the roots were manually identified for each word. The features vary from experiment to experiment and consequently the dimension of the features also vary.

Evaluation: For each experiment, we have considered 'N' number of words per language and we have generated 'M' number of features out of them. From the available features we have segregated 'X' number of features for training and 'Y' number of features for testing in 80:20 ratio. Using these sets, we have evaluated the training and testing performance for various stop values.

5.2 Stemming

Stemming is the use of linguistic analysis to get to the root form of a word. Search engines that use stemming compare the root forms of the search terms to the documents in its database. For example, if the user enters "viewer" as the query, the search engine reduces the word to its root ("view") and returns all documents containing the root - like documents containing view, viewer, viewing, preview, review etc. Since our training data is very small it fails for out-of-vocabulary words. And also, it fails for homographs (a homograph is one of a group of words that share the same spelling but have different meanings).

For stemming in Indian languages, inflections of the words are formed mostly by suffixes than prefixes. So considering the first 5 phones of a word would help to predict the root of the word. But, for English prefixes as well as suffixes are equally used to form inflections. So prefixes are separated and considered as a single unit like a phone here. So we have selected the features like

- First 6 phones for English and
- First 5 phones for Indian languages

Stemming results for various languages are shown in Table 4. It shows that sto-value 1 would be optimal, when we used training and testing features in the ratio 907:227 for English, 3606:902 for Tamil and 987:247 for Telugu.

Table 4: Stemming Performance.

Language	Stop Value	Training	Testing
English	1	100%	99.55%
	2	98.78%	96.01%
	3	94.59%	90.26%
	4	86.64%	82.3%
Tamil	1	93.25%	77.69%
	2	84.74%	75.24%
	3	80.63%	74.49%
	4	77.08%	73.47%
Telugu	1	100%	93%
	2	100%	92%
	3	100%	93%
	4	100%g	94%

5.3 English G2P Conversion

Our search uses phonetic features like syllables. In cross-linguagl search support for English input is neccessary. So we need a mechanism to convert the query from its grapheme form to phoneme form. It is very challenging since English words doesn't follow one-to-one correspondence between its letters and its phonemes.

For G2P conversion of English words, the letters of the word are used as features. We hypothesize that the first and last letters of the word and previous and next letters of the current letter help much to predict its phoneme. So we have selected the features like

- First and Last letters of the word and Previous and Next letters of the Current letter

The G2P conversion results for Indian-English is shown in Table 5. It shows that stop-value 1 would be optimal for a training feature set of 106896 and testing feature set of 26724.

5.4 Stop Words Identification

Stop words, is the name given to the words which are filtered out prior to, or after processing of natural language data (text). There is no definite list of stop words which all natural language processing tools incorporate. Some search engines don't index/record extremely common words in order to save space or to speed up searches. The list of stop

Table 5: English G2P Conversion Performance.

Stop Value	Training	Testing
1	95.89%	85.56%
2	92.15%	85.37%
3	90.79%	85.56%
4	89.73%	85.53%

words for Indian languages have not been identified yet. So, we tried to generate the list by the basic idea that the most common words of a language might have occurred more frequently than other words in the corpus. We generated a list of top 500 frequently occurred words in a language. Then stop words list was produced with the help of a linguist who manually cleaned it.

6 inSearch - Search Engine for Indian Languages

Most information seekers use a search engine to begin their web activity (Prasad Pingali, Jagadeesh Jalagarlamudi and Vasudeva Varma, 2006). In this case, users submit a query (typically a list of keywords) and receive a list of web pages that may be relevant. In conventional information retrieval systems (Google, Live, Yahoo, Guruji etc.) the user must enter a search query in the language/encoding of the documents in order to retrieve it. This restriction clearly limits the amount of information to which an user will have access.

Developing a search engine for Indian languages faces many challenges. Some of them are, identifying the text-encoding of the web content, converting them into a transliteration scheme, developing stemmers and identifying the stop words etc. Also one need to design a good mechanism/tool (A. Joshi, A. Ganu, A. Chand, V. Parmar and G. Mathur, 2004) to accept user's query in transliteration scheme or standard encoding like UTF-8 and even in English also. **inSearch** is a search engine for Indian languages developed by considering all the above discussed issues and solutions. Fig 1 shows the basic architecture and the following sub-sections explain them further.

6.0.1 Web Crawling

Our web crawling is language focused. It takes a list of identified URLs per language for which we have converters. Then it crawls those pages and stores the documents locally for further processing. It maintains the same directory structure as on the web and ordered by date.

6.0.2 Indexing

The effectiveness of any search engine mainly depends on its index structure. The structure should be capable of catering sufficient and relevant information to the users in terms of their needs. To serve users with the contexual information in their own language, the system needs to index on meaning representation and not on plain text. Also, the size of the index should not be too large.

Conventional search engines use stemming technology to get the root of the word and index the document about it. Thus, it will search not only for the search terms, but also for its inflexions and similar to some or all of those terms. But in case of Indian languages, there is no effective algorithm or tool to do stemming. So we used phonetic features like syllables to index the pages. We extract the first two syllables (since they are almost equal to the root of the word most of the times) of the word and index about it. Since, we have identified a method for stemming, we used them also for indexing. The detailed experiments are provided in the above section 5.2. Our index structure includes syllables, stem, word, term-frequency, language, date and doc-id. This structure enables efficient multi-lingual and cross-lingual search.

6.0.3 Retrieval

At first, begining two syllables of the words of the query are extracted. Then the words begining with those syllables are retrieved from the database. Hence the common words across languages are captured here. These words are ranked according to their phonetic relativeness to the query calculated by DTW method. The words fall under threshold are discarded, so that the documents containing the most related words pop-up. Then the documents are re-ranked about their term frequency (TF) values (G. Salton and C. Buckley, 1988) and contextual information.

6.0.4 User Interface

Presently, there is no standard/convenient notation or keyboard layout to input the query in Indian languages. Even with UTF-8 encoding most of the users don't know how to type the query. So, for cross-lingual search we provide a phonetic mapping table to be refered by the user to type the query in IT3 notation. But for language specific search, we provide a query typing tool. This tool has buttons for characters of the selected language. By clicking the buttons, user can type the query in his native script since most of the queries won't be more than a word/phrase. After forming the query, user can search the web and the ranked results are displayed much like the standard search engine's results. Here the cached pages for even font encoded pages are displayed in UTF-8 encoding.

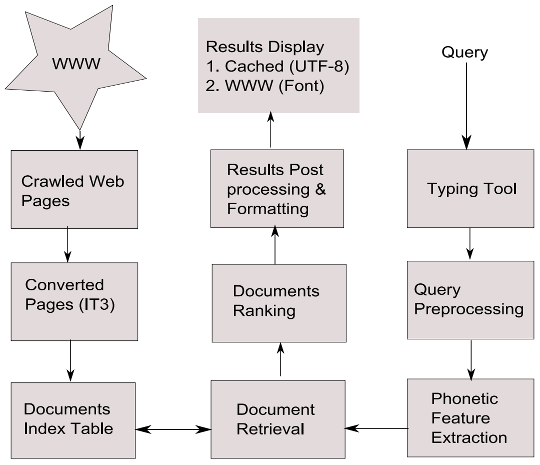

Figure 1: Search Engine Architecture.

7 Cross-Lingual Search

The development of digital and online information repositories has created many opportunities and new problems in information retrieval. Online documents are available Internationally in many different languages. This makes it possible for users to directly access previously unimagined sources of information. However in conventional information retrieval systems, the user must enter a search query in the language of the documents in order to retrieve it. This restriction clearly limits the amount and type of information which an individual user really has access to. Cross Language Information Retrieval (CLIR) (F. Gey, N. Kando and C. Peters, 2002) (L. S. Larkey, M. S. Connell and N. Abduljaleel, 2003) enables users to enter queries in languages they are fluent in, and uses language translation methods to retrieve documents originally written in other languages.

The aim of this attempt is to extend the search capability to search across all Indian languages. The users are ordinary Indians who master one of the Indian languages, but have only passive knowledge in the other neighbour languages. This means that they can read a text but not search for it since they do not have active knowledge of how the different concepts in the other languages are written or spelled. This will also strengthen the use of the Indian languages on the Internet and further avoid unnecessary use of the English language. We are trying to achieve it step-by-step by using the below mentioned methods.

1.Phonetic Relativeness Measure: In this approach the phonetic distance (how many insertions/substitutions/deletions occured) between the query words and the available words is calculated. Then the closest words are considered as query related words and the results are produced for those words. There are many methods to calculate the phonetic distance and we used DTW (Dynamic Time Warping) method to calculate the phonetic distance for our experiments. We used equal weightage (i.e 1) for insertion, substitution and deletion here.

2.Dictionary Lookup: Here bilingual/multilingual dictionaries are used to get the translation of the keywords. Creating such dictionaries for all the words of a language is time consuming process. Instead, creating dictionaries for the stems of the words alone will reduce the effort. Unfortunately there are no such dictionaries available or methods to create the stems for all Indian languages. So we developed CART based decision trees to produce the stems. We have created such stem based bilingual dictionaries for 5 Indian languages. Also, we have created a multilingual dictionary (Table 6) for 8 Indian languages by keeping English words as keys.

3.Machine Translation: This is considered as an appropriate solution for cross-language search (Dr. Pushpak Bhattacharyya, 2006). The query in source language gets translated into the destination language and the results will be produced for it. In this context, there is a close synergy between the fields of

Table 6: Multi-lingual Dictionary.

Language	Words
Bengali	2028
Gujarati	6141
Hindi	22212
Kannada	22695
Malayalam	23338
Oriya	7287
Tamil	5521
Telugu	8148
English	43185

Cross Language Information Retrieval (CLIR) and Machine Translation (MT). But such systems for Indian languages are under development. We are also focussing our effort in the same direction to use it with our engine in the future.

8 Conclusion

In this paper we discussed the importance of being able to search the Indian language web content and presented a multi-lingual web search engine inSearch capable of searching 10 Indian languages. The nature of Indic scripts, Indic data storage formats and how to preprocess them efficiently are detailed. It explained about how language identification, grapheme to phoneme conversion for English and stemming can be achieved using CART. This shows that transcoding of proprietary encodings into a meta standard transliteration scheme makes Indian language web content accessible through search engines.

9 Acknowledgments

We like to thank Speech and Language Technologies Lab, Bhrigus (India) Pvt Ltd, Hyderabad, India and our collegues Ms.Bhuvaneshwari, Mr.Prasad and others for all their support and encouragement.

References

A. Joshi, A. Ganu, A. Chand, V. Parmar and G. Mathur. 2004. Keylekh: a keyboard for text entry in indic scripts. *CHI '04 Extended Abstract on Human Factors in Computing Systems, ACM Press*.

A. A. Raj and K. Prahallad. 2007. Identification and conversion of font-data in indian languages. In *In International Conference on Universal Digital Library (ICUDL)*, Pittsburgh, USA.

A. K. Singh and J. Gorla. 2007. Identification of languages and encodings in a multilingual document. In *Proceedings of the 3rd ACL SIGWAC Workshop on Web As Corpus*, Louvain-la-Neuve, Belgium.

Dr. P. Bhattacharyya. 2006. White paper on cross lingual search and machine translation. *Proposal to Government of India*.

F. Gey, N. Kando and C. Peters. 2002. Cross language information retrieval: A research roadmap. *SIGIR Forum*, 36(2):72–80.

G. E. Burkhart, S. E. Goodman, A. Mehta and L. Press. 1998. The internet in india: Better times ahead? *Commun. ACM*, 41(11):21–26.

G. Salton and C. Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Process. Management*, 24(5):513–523.

M. Ganapathiraju , M. Balakrishnan , N. Balakrishnan and R. Reddy 2005. Om: One tool for many (indian) languages. *Journal of Zhejiang University Science*, 6A(11):1348–1353.

H. Garg. 2005. Overcoming the font and script barriers among indian languages. *MS Thesis at International Institute of Information Technology Hyderabad, India*.

ISCII - Indian Standard Code for Information Interchange. 1983. http://tdil.mit.gov.in/standards.htm.

S.Khudanpur and C.Schafer , Devanagari Converters. 2003. http://www.cs.jhu.edu/cschafer/jhu devanagari cvt ver2.tar.gz.

L. S. Larkey, M. S. Connell and N. Abduljaleel. 2003. Hindi clir in thirty days. *ACM Trans. on Asian Language Information Processing (TALIP)*, 2(2):130–142.

P. Lavanya, P. Kishore and G. R. Madhavi. 2005. A simple approach for building transliteration editors for indian languages. *Journal of Zhejiang University Science*, 6A(11):1354–1361.

P. Prasad , J. Jagadeesh and V. Varma. 2006. Webkhoj: Indian language ir from multiple character encodings. *International World Wide Web Conference*.

Transliteration of Indic Scripts: How to use ISO 15919. http://homepage.ntlworld.com/stone-catend/trind.htm.

Unicode Consortium - Universal Code Standard. 1991. http://www.unicode.org.

Multi-Lingual Screen Reader and Processing of Font-data in Indian Languages. http://speech.iiit.net/speech/publications/Anand-Thesis-Final.pdf.