WebKhoj: Indian language IR from Multiple Character Encodings

Prasad Pingali
Language Technologies
Research Centre
IIIT, Hyderabad
India
pvvpr@iiit.ac.in

Jagadeesh Jagarlamudi
Language Technologies
Research Centre
IIIT, Hyderabad
India
j_jagdeesh@research.iiit.ac.in

Vasudeva Varma
Language Technologies
Research Centre
IIIT, Hyderabad
India
vv@iiit.ac.in

ABSTRACT

Today web search engines provide the easiest way to reach information on the web. In this scenario, more than 95% of Indian language content on the web is not searchable due to multiple encodings of web pages. Most of these encodings are proprietary and hence need some kind of standardization for making the content accessible via a search engine. In this paper we present a search engine called WebKhoj which is capable of searching multi-script and multi-encoded Indian language content on the web. We describe a language focused crawler and the transcoding processes involved to achieve accessibility of Indian langauge content. In the end we report some of the experiments that were conducted along with results on Indian language web content.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering, Selection process; H.3.1 [**Content Analysis and Indexing**]: Linguistic processing

General Terms

Standardization, Languages

Keywords

Indian languages, web search, non-standard encodings

1. INTRODUCTION

India is a multi-language, multi-script country with 22 official languages and 11 written script forms. About a billion people in India use these languages as their first language. English, the most common technical language, is the lingua franca of commerce, government, and the court system, but is not widely understood beyond the middle class and those who can afford formal, foreign-language education. Not only is there a large societal gap between the rich and poor, but that gap appears to be widening due the dominance of English in the society. About 5% of the population (usually the educated class) can understand English as their second language. Hindi is spoken by about 30% [5] 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. Computability of Indian languages could help bridge the societal gaps in education, economy and healthcare. However the research and development, availability of standards, support from operating systems and applications in these directions moved very slow due to language heterogeneity.

Today this phenomenon can also be observed on the world wide web. The percentage of Indian language content is very less compared to the official languages of United Nations [7]. Even within the available content, majority is not searchable and hence not reachable due to multiple encodings used while authoring such websites. Web publishers of such content were hesitant to use any available standards such as Unicode due to very delayed support from operating systems and browsers in rendering Indic scripts. Even today Hindi is rendered properly only on Windows XP and beyond. Linux has very little support for these languages. Indian languages had barely any support till Windows 2000 operating system. This creates a major bottleneck for web publishers in these languages to get viewership.

Despite all these issues, we found considerable amount of content being published on the web. However such content gets unnoticed or gets very less viewership since most of such content is not accessible through search engines due to non-standard encodings being rendered using proprietary fonts.

This paper is organized into seven sections. In the next sub-section we give an introduction to characters, glyphs and fonts in order to appreciate the complexity involved in rendering complex scripts. We then introduce to the complexity of Indic scripts in the sub-section 1.2. In Section 2 we make the problem statement and explain an implementation to solve this problem in Section 3. We report some experiments and results in Section 4, followed by a conclusion in Section 5.

1.1 Fonts, characters and glyphs

In the history of mankind the act of writing has always been considered as an activity producing visual results, namely text. The computer has brought a more abstract layer to

WWW2006, May 23–26, 2006, Edinburgh, Scotland.

ACM 1-59593-323-9/06/0005
.

it, by storing and transmitting textual data. The atomic unit of this abstract representation of text, as defined in the Unicode standard [8], is called a character. And indeed, characters prove to be useful for obtaining alternative (non-visual) representations of text such as Braille, speech synthesis, etc. The visual representation of a character is called a glyph [8]. Displaying textual contents, whether on screen or on paper, involves translating characters into glyphs, a non-trivial operation for many writing systems. Going in the opposite direction (from glyphs to characters) is known as OCR when done by a machine, or as reading when done by a human [8]. The technology trend over the last few years has been to use characters for most of the text processing and to limit glyph issues to the last stage, namely rendering. At that level, character to glyph translation is handled by increasingly “intelligent” (cf. OpenType and AAT technologies) fonts and font encodings. Unicode is an effort in this direction. At the same time, restoring the original character stream from a rendered electronic document output for operations such as searching, indexing, or copy-pasting, no general solution exists in today’s popular document formats yet. Despite the problems involved, web authors tend to use proprietary encodings due to the complex characteristics of Indic scripts as described in the following section.

1.2 Characteristics of Indic Scripts

Indic scripts are phonetic in nature. There are vowels and consonant symbols. The consonants become a syllable after the addition of a vowel sound to it. Further to compound the problem there are ‘compound syllables’ also referred as ligatures. For instance, if we consider ‘tri’ in ‘triangle’, there are three letters corresponding to three sounds ‘ta’, ‘ra’, ‘yi’. But in the case of Indic Scripts the three are built together to make a single compound consonant having a non-linear structure unlike Latin based languages.

The main problem with display of Indic scripts is dealing with their non-linear structures. Glyphs have variable widths and have positional attributes. Vowel signs can be attached to the top, bottom, left and right sides of the base consonant. Vowel signs may also combine with consonants to form independent glyphs. Consonants frequently combine with each other to form complex conjunct glyphs. Although the encoding encapsulates only the basic alphabetic characters, the number of glyphs and their combinations required for the exhaustive rendering of these scripts can be quite large [11].

Since the character to glyph mappings have to be achieved using a 256 character address space, web authors come up with an intelligent way of representing all the characters in the language using some 256 glyphs. Most of these glyphs do not have any semantic significance in the language by themselves. However when displayed together using some positional parameters, they achieve human readable characters. This situation makes the Indian language web content inaccessible for machine processing.

2. PROBLEM STATEMENT

Many information seekers use a search engine to begin their Web activity. In this case, users submit a query, typically a list of keywords, and receive a list of Web pages that may be relevant, typically pages that contain the keywords.

Today though considerable amount of content is available in Indian languages, users are unable to search such content. Because Indian language websites rely on unique encodings or proprietary extensions of existing standard encodings [11]. This plurality of encodings creates a problem for information retrieval to function as desired. Also many research groups in information retrieval and natural language processing feel the need to collect corpora in these languages from the web in the same way they obtain corpora for other languages [14], [7], [1], [10]. Therefore in order to search or process Indian language websites, we should be able to transliterate all the encodings into one standard encoding and accept the user’s queries in the same encoding and build the search index.

This task involves many steps. First step is to be able to identify the various encodings in Indian languages on the web. Since these encodings are non-standard, there is no one comprehensive list of such possible encodings. Therefore we need to somehow identify all such encodings and also be able to classify these encodings into the existing types. Second step is to build a transliteration mapping for the given encoding into a standard encoding which is UTF-8 and hence convert any page into a standard and index it. Third step is to be able to accept user’s queries in the same standard as that of the transliterated documents which is UTF-8.

3. WEBKHOJ ARCHITECTURE

In this paper we report a search engine called WebKhoj which can search web pages in the top 10 Indian languages according to the number of native speakers. WebKhoj currently supports Hindi, Telugu, Tamil, Malayalam, Marathi, Kannada, Bengali, Punjabi, Gujarati and Oriya. Before we describe the architecture of WebKhoj, it is useful to understand how a Web search engine is typically put together and then see its extensions for our task.

3.1 General web search engine

Figure 1 shows a general web search engine schematically [2]. The major modules of a web search engine are a Crawler, an Indexer, a Query Engine and a Ranking Engine. Every engine relies on a crawler module to provide the grist for its operation (shown on the left in Figure 1). Crawlers are small programs that browse the Web on the search engine’s behalf, similar to how a human user follows links to reach different pages. The programs are given a starting set of URLs whose pages they retrieve from the Web. The crawler extracts URLs appearing in the retrieved pages and give this information to the crawler control module. This module determines what links to visit next and feeds these links back to the crawler. (Some of the functionality of the crawler control module may be implemented by the crawlers themselves.) The crawlers also pass the retrieved pages into a page repository. Crawlers continue visiting the Web until local resources, such as storage, are exhausted. The indexer module extracts all the words from each page and records the URL where each word occurred. The result is a generally very large “lookup table” that can provide all the URLs that point to pages where a given word occurs (the text index in Figure 1). The table is of course limited to the pages that were covered in the crawling process. As mentioned earlier, text indexing of the Web poses special difficulties, due to its size and its rapid rate of change. In addition to these quantitative challenges, the Web calls for some special, less common, kinds of indexes. For example, the indexing

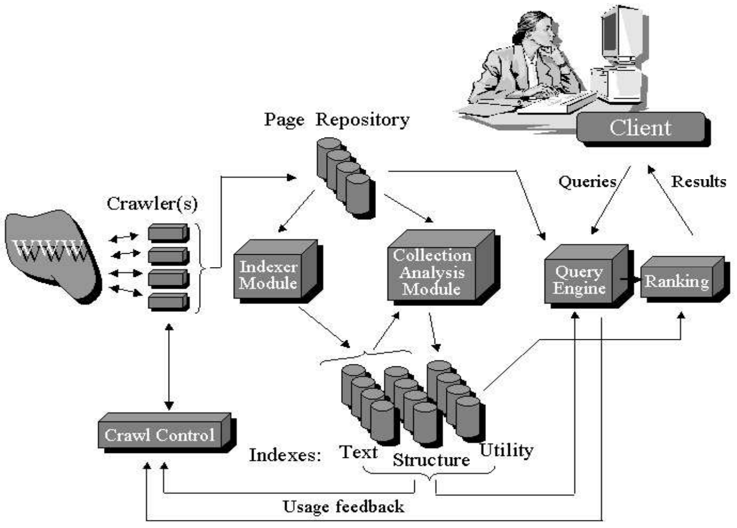


Figure 1: General web search engine architecture

module may also create a structure index, which reflects the links between pages. Such indexes would not be appropriate for traditional text collections that do not contain links. The collection analysis module is responsible for creating a variety of other indexes. During a crawling and indexing run, search engines must store the pages they retrieve from the Web. The page repository in Figure 1 represents this possibly temporary collection. Search engines sometimes maintain a cache of the pages they have visited beyond the time required to build the index. This cache allows them to serve out result pages very quickly, in addition to providing basic search facilities. Some systems, such as the Internet Archive, have aimed to maintain a very large number of pages for permanent archival purposes. Storage at such a scale again requires special consideration. The query engine module is responsible for receiving and filling search requests from users. The engine relies heavily on the indexes, and sometimes on the page repository. Due to the Web's size and the fact that users typically only enter one or two keywords, result sets are usually very large. Hence the ranking module has the task of sorting the results such that results near the top are the most likely to be what the user is looking for. In the rest of this section we describe the additional modules that were used in a general web search engine to make it work for Indian languages.

3.2 Language focused crawling

Since our goal is to be able to search web sites of specific languages, we are looking for a relatively narrow segment of the web. Crawlers that fetch pages related to a particular topic of interest are called topic focused crawlers [6]. While our crawler is very similar to the one mentioned in [6], we use a language identification module instead of a classifier and hence call it as language focused crawling. The language identification module returns the name of the language for a given web page. This module is aware of all the proprietary encodings and also uses a bag of words to recognize unknown encodings from meta-tag information that might be found in an HTML page. In many cases web pages contain more than one language, especially one of the languages being English. This happens since many of the website organization information such as menu items, or disclaimers and other such formatting information. In some websites such as blogs or forums majority of the content might be English, with Indian language content being a minority. The language identifier module returns a language only if the number of words in a web page are above a given threshold value.

3.3 Transcoding

Since Indian language content is being published in multiple encodings on the web, transliteration of encodings to a popular standard such as Unicode [15] is needed. In order to transliterate a non-UTF-8 encoding into UTF-8 which is a Unicode based encoding one has to come up with byte sequence mappings between source and target encodings. Such mappings are rarely one to one mappings, and involve many to one, one to many and many to many mappings of byte sequences. As it was explained in the beginning of this paper, a sequence of bytes represent a sequence of glyphs of a font, which in turn could render a single character or a ligature in the Indic script. Ideally mappings are to be created to all the unique characters in the language, which could be a large number in the order of tens of thousands. Since it would be tedious to list out all the characters and ligatures, we make use of the large number of documents collected by the crawler to come up with a semi-automatic process of generating mappings.

We use a simple heuristic to identify the potential character boundaries from byte sequences. First the text from the collected web pages is divided into words using a suitable word tokenizer. Then the algorithm lists all the possible word beginning bytes in both the source and target font encodings. Now each word is scanned from left to right until one such byte occurs in the word. Whenever a valid word beginner occurs in the word, we tokenize at that point, and the byte sequence till that point is treated as a potential character. For example in a given encoding if all the possible word beginning bytes are 'a', 'b' and 'c', a new word 'cars' is tokenized as 'c', 'ars', since neither 'r' nor 's' are

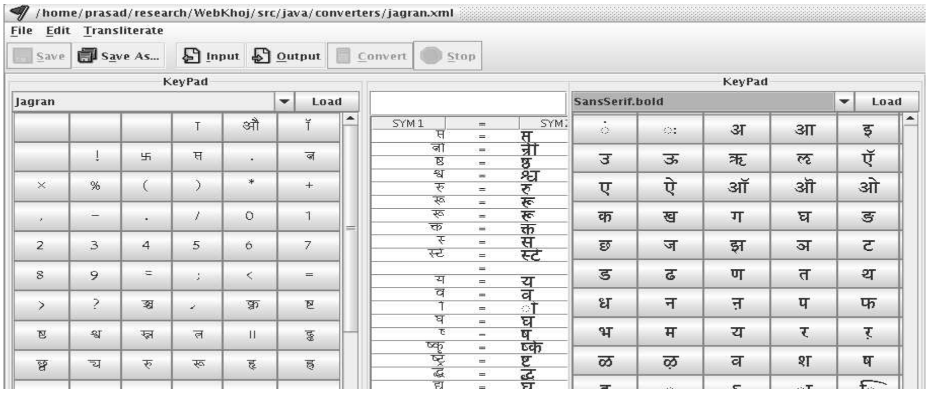


Figure 2: Transcoding from Jagran encoding to UTF-8

valid word beginners. The byte sequences thus obtained by segmentation are potential characters or ligatures in that language.

Once such segmentation is done, the frequency of such byte sequences (or potential characters) is calculated. It was found from our experiments that the ranks based on the normalized frequency of such potential characters is highly correlated (we present more details in our experiments section). Therefore we use this algorithm to come up initial suggested mappings for transcoding, and then the user would manually correct any errors by going through the font mappings as shown in the Figure 2. The transcoding tool sorts the potential characters according to their ranks, so that the user would find the equivalent match in the target encoding among the top few possibilities. Also since the mappings are ordered based on the normalized frequency found in the corpus, mapping source and target bytes in this order ensures optimal precision that can be obtained from a set of mappings.

Once such transcoder mappings are generated for all possible encodings in Indian languages, a transcoding module is called during indexing of the web documents. If a web document is encoded in an encoding other than UTF-8, the transcoder module is called to transliterate the encoding of the given web page into UTF-8 standard. In order to do this, the HTML page is parsed to obtain its document object model (DOM) using the JTidy utility[1]. All the nodes of type “font” are extracted from the DOM and the font encoding is checked against a known set of encodings on the web. Based on the font encoding, the appropriate transcoder mappings are used to transliterate the relevant text into UTF-8. One word is transcoded at a time. In order to transcode, the maximum byte sequence available in the mapping table is used to transliterate the encodings and the process is repeated to the remaining substring of the word. This transliterated document is then sent to the indexer to build the inverted index.

3.4 Retrieval Algorithm

The score of query q for document d is defined in terms of TFIDF [13] metric as shown below:

$$score(q, d) = c(q, d).q_n(q).(\sum_{t\ in\ q} tf(t\ in\ d).idf(t))$$

3.4.1 tf (term frequency)

‘tf’ (also known as term frequency) is a score factor based on a term or phrase’s frequency in a document. Terms and phrases repeated in a document indicate the topic of the document, so implementations of this score usually return larger values when frequency is large, and smaller values when frequency is small.

3.4.2 idf (inverse document frequency)

‘idf’ is a score factor based on a term’s document frequency (the number of documents which contain the term). Terms that occur in fewer documents are better discriminators of topic, so implemenations of this method usually return larger values for rare terms, and smaller values for common terms.

3.4.3 c (coverage of query terms)

‘c’ is a score factor based on the fraction of all query terms that a document contains. This value is multiplied into scores. The presence of a large portion of the query terms indicates a better match with the query, so implemenations of this function usually return larger values when the ratio between these parameters is large and smaller values when the ratio between them is small.

3.4.4 q_n (query normalization)

This is the normalization value for a query given the sum of the squared weights of each of the query terms. This value is then multiplied into the weight of each query term.

This does not affect ranking, but rather just attempts to make scores from different queries comparable.

3.5 User Interface

Currently there is no easy means to key-in UTF-8 queries to the search engine using the normal keyboard. So WebKhoj is provided with a soft keyboard which displays the UTF-8 character set of the language on the screen. The layout of the keys is very similar to the Keylekh layout [9]. We also tried providing a roman to local language transliteration keyboard which dynamically renders Indian language text when its phonetic equivalent is typed using roman characters. We had student volunteers from a near by village to try out the keyboards. However, we found that the students who are taught in the local language in schools are not comfortable with English symbols. Also within the local language, the way symbols are taught in schools is much

[1] JTidy is a Java implementation of Dave Raggett’s HTML tidy. JTidy can be found at http://jtidy.sourceforge.net

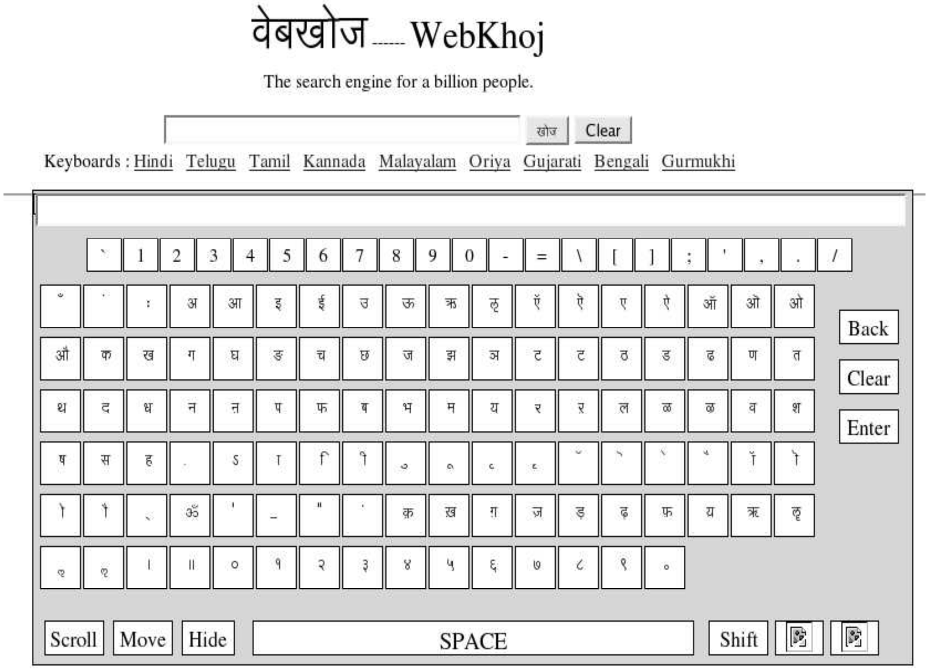
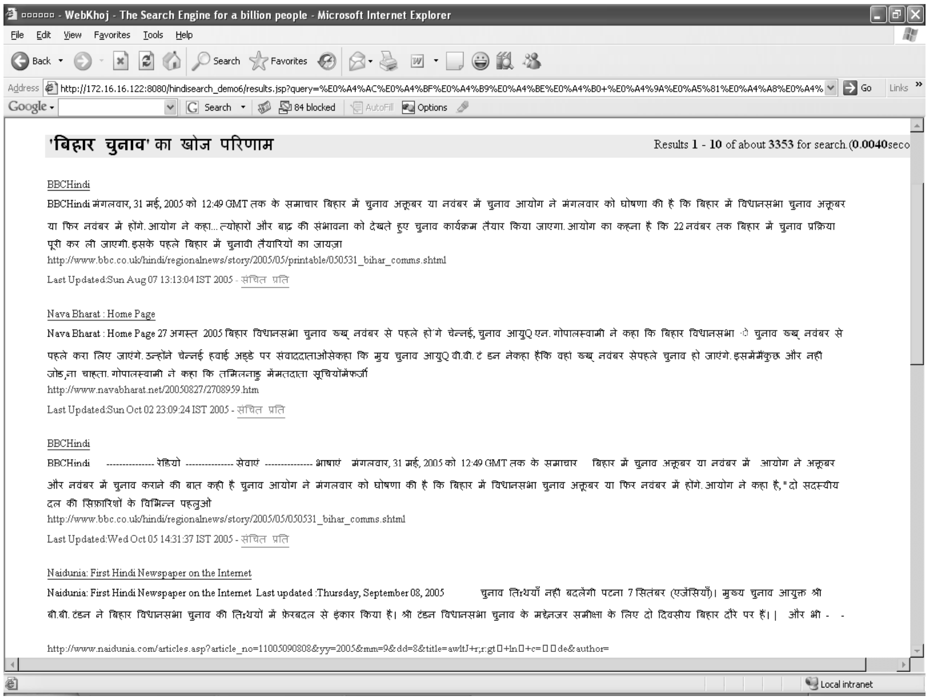

वेबखोज ------ WebKhoj
The search engine for a billion people.
खोज
Clear
Keyboards : Hindi Telugu Tamil Kannada Malayalam Oriya Gujarati Bengali Gurmukhi
` 1 2 3 4 5 6 7 8 9 0 - = \ [ ] ; ' , . /
ँ ं ः अ आ इ ई उ ऊ ऋ ऌ ऍ ऎ ए ऐ ऑ ऒ ओ
औ क ख ग घ ङ च छ ज झ ञ ट ठ ड ढ ण त
थ द ध न ऩ प फ ब भ म य र ऱ ल ळ ऴ व श
ष स ह ़ ऽ ा ि ी ु ू ृ ॄ ॅ ॆ े ै ॉ ॊ
ो ौ ् ॐ ' _ " ' क़ ख़ ग़ ज़ ड़ ढ़ फ़ य़ ॠ ॡ
ॢ ॣ । ॥ ० १ २ ३ ४ ५ ६ ७ ८ ९ ॰
Back
Clear
Enter
Scroll
Move
Hide
SPACE
Shift

Figure 3: Hindi soft keyboard user interface for WebKhoj search engine


□□□□□□ - WebKhoj - The Search Engine for a billion people - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links
Google
Search
84 blocked
AutoFill
Options
Local intranet

http://172.16.16.122:8080/hindisearch_demo6/results.jsp?query=%E0%A4%AC%E0%A4%BF%E0%A4%B9%E0%A4%BE%E0%A4%B0+%E0%A4%9A%E0%A5%81%E0%A4%A8%E0%A4%

'बिहार चुनाव' का खोज परिणाम

Results **1** - **10** of about **3353** for search.(**0.0040**seco

BBCHindi

BBCHindi मंगलवार, 31 मई, 2005 को 12:49 GMT तक के समाचार बिहार में चुनाव अक्तूबर या नवंबर में चुनाव आयोग ने मंगलवार को घोषणा की है कि बिहार में विधानसभा चुनाव अक्तूबर या फिर नवंबर में होंगे. आयोग ने कहा... त्योहारों और बाढ़ की संभावना को देखते हुए चुनाव कार्यक्रम तैयार किया जाएगा. आयोग का कहना है कि 22 नवंबर तक बिहार में चुनाव प्रक्रिया पूरी कर ली जाएगी. इसके पहले बिहार में चुनावी तैयारियों का जायज़ा

http://www.bbc.co.uk/hindi/regionalnews/story/2005/05/printable/050531_bihar_comms.shtml

Last Updated:Sun Aug 07 13:13:04 IST 2005 - संचित प्रति

Nava Bharat : Home Page

Nava Bharat : Home Page 27 अगस्त 2005 बिहार विधानसभा चुनाव ख्ख् नवंबर से पहले हो ंगे चेन्नई, चुनाव आयुQ एन. गोपालस्वामी ने कहा कि बिहार विधानसभा ो चुनाव ख्ख् नवंबर से पहले करा लिए जाएंगे. उन्होंने चेन्नई हवाई अड्डे पर संवाददाताओंसेकहा कि मुय चुनाव आयुQ वी.वी. टं डन नेकहा हैकि वहां ख्ख् नवंबर सेपहले चुनाव हो जाएंगे. इसमेंमेंकुछ और नहीं जोड,ना चाहता. गोपालस्वामी ने कहा कि तमिलनाडु मेंमतदाता सूचियोंमेंफर्जी

http://www.navabharat.net/20050827/2708959.htm

Last Updated:Sun Oct 02 23:09:24 IST 2005 - संचित प्रति

BBCHindi

BBCHindi -------------- रेडियो -------------- सेवाएं -------------- भाषाएं मंगलवार, 31 मई, 2005 को 12:49 GMT तक के समाचार बिहार में चुनाव अक्तूबर या नवंबर में आयोग ने अक्तूबर और नवंबर में चुनाव कराने की बात कही है चुनाव आयोग ने मंगलवार को घोषणा की है कि बिहार में विधानसभा चुनाव अक्तूबर या फिर नवंबर में होंगे. आयोग ने कहा है,"दो सदस्यीय दल की सिफ़ारिशों के विभिन्न पहलुओं

http://www.bbc.co.uk/hindi/regionalnews/story/2005/05/050531_bihar_comms.shtml

Last Updated:Wed Oct 05 14:31:37 IST 2005 - संचित प्रति

Naidunia: First Hindi Newspaper on the Internet

Naidunia: First Hindi Newspaper on the Internet Last updated :Thursday, September 08, 2005 चुनाव तिथियाँ नहीं बदलेंगी पटना 7 सितंबर (एजेंसियाँ)। मुख्य चुनाव आयुक्त श्री बी.बी. टंडन ने बिहार विधानसभा चुनाव की तिथियों में फ़ेरबदल से इंकार किया है। श्री टंडन विधानसभा चुनाव के मद्देनजर समीक्षा के लिए दो दिवसीय बिहार दौरे पर हैं।| और भी - -

http://www.naidunia.com/articles.asp?article_no=11005090808&yy=2005&mm=9&dd=8&title=awltJ+r;r:gt□+ln□+c=□□de&author=

Figure 4: Search results being displayed for a Hindi query in UTF-8

different from the way UTF-8 characters need to be typed in. However, with some training these students were able to adapt to the soft keyboard.

Currently soft keyboards for 10 Indian languages are provided in the searching interface. One language is shown to the user at any given instance. The user can change the keyboard to a different language by clicking on the desired language hyperlink displayed on the interface as shown in Figure 3. After thus framing the query, the user can search for the web documents, and the results are ranked and displayed much like Google as shown in Figure 4.

3.6 Word spelling normalization

Indian language words face standardization issues in spelling, thereby resulting in multiple spelling variants for the same word. For example we found widely used spelling variations for the hindi word ‘angrezi’ as shown below

अँगरेज़ी, अँगरेजी, अँग्रेज़ी, अँग्रेजी, अंगरेज़ी, अंगरेजी, अंग्रेज़ी, अंग्रेजी

The major reasons for this phenomenon can be attributed to unavailability of proper website authoring tools equipped with spell checkers for Indian languages and multiple dialects of spoken language, transliteration of proper names and words borrowed from foreign languages whose spellings are not standardized. While we have to handle Indian language words with spelling variations and errors, we also showed that a considerable percentage of foreign language words mainly English have entered into Indian language usage which cannot be ignored. While such words are being frequently used by people, there is no standardization in spelling for such words thereby resulting in huge variations due to transliteration. Given such variations in spelling it becomes difficult for web Information Retrieval applications built for Indian languages, since finding relevant documents would require more than performing an exact string match. It was shown that normalization rules for specific languages work best with spelling normalization problems. We make use of a set of rules [12] to normalize the words before indexing them or looking them up from the index. These rules are language specific and we describe the rules for Hindi in the next sub-sections. We achieve normalization of word spellings by mapping the alphabet of the given language L into another alphabet L' where $L' \in L$. We use the following rules to achieve such a normalized mapping.

3.6.1 Mapping chandrabindu *to* bindu

Often people tend to use *chandrabindu* (a half-moon with a dot) and *bindu* (a dot on top of alphabet) interchangeably. Lots of confusion exists in common language usage on which to use when. In order to equate all such words we convert all occurrences of *chandrabindu* to *bindu*, which would equate all the words shown below.

अँगड़ाइयाँ , अँगड़ाइयां , अंगड़ाइयाँ , अंगड़ाइयां

3.6.2 nukta *deletion*

Unicode contains 10 consonant characters with *nukta* (a dot under consonant) and one *nukta* character itself. We delete all occurrences of nukta character and replace all consonants with *nukta*s with their corresponding consonant character. This would equate words like the ones shown below.

अँग्रेज , अँग्रेज़

3.6.3 halanth *deletion*

Hindi and many other Indian languages face the problems of *’schwa’* (the default vowel ’a’ that occurs with every consonant) deletion. Lots of spelling variations occur due to *’schwa’* deletion. In order to normalize such words we delete all the *halanth* characters in the given word before making a string match. This operation would normalize words as shown in the example below.

अदरक , अद्रक

3.6.4 vowel shortening

Many times in written script people use shorter vowels instead of longer ones or vice versa. Therefore in our application we convert all the longer vowels to their corresponding shorter ones. Using this feature we can normalize words as shown in this example.

अदिलाबाद , अदीलाबाद

3.6.5 chandra *deletion*

’chandra’ (half-moon) is used for vowel rounding. Usually words borrowed from English at times require vowel rounding operation. For example the word ”documentary”. But this character is used inconsistently many times. Therefore deleting such a character would normalize the words where vowel rounding has been used.

These rules were compared with many approximate string matching algorithms are were found to result in a better f-measure [12].

4. EXPERIMENTS AND DISCUSSION

We report here some experiments that were conducted in transcoding the proprietary encodings and present some statistics from our language focused crawl about the Indian language web.

The transcoding tool was designed to generate mappings between two encodings in a semi-automatic fashion. In order to achieve this the tool automatically gives some mapping suggestions based on the rank correlation of the two encodings in question. We found that the byte sequences from two encodings of same language correlate very well, by looking at the Spearman’s rank correlation coefficient. Intuitively this phenomenon can be understood as the convergence of unique lexicon from two encodings from sufficiently large corpus, since they both belong to the same language. To find the amount of correlation, we experimented with two different encodings from Hindi. We ran the character segmentation algorithm and computed the normalized frequencies as mentioned above and ranked the character sequences in both the encodings from a corpus of 2,000 documents from each of these encodings. We manually marked the corresponding frequency based rank positions of a given character or a ligature from these encodings and calculated the Spearman's rank correlation coefficient. We then plotted a graph with the Spearman's correlation coefficient on y-axis and the number of mappings on x-axis as shown in Figure 5. We observed that the rank correlation is 100% for the first 25 mappings that were automatically generated, and are close to 90% for the first 200 mappings which can achieve a transcoding precision of above 90%.

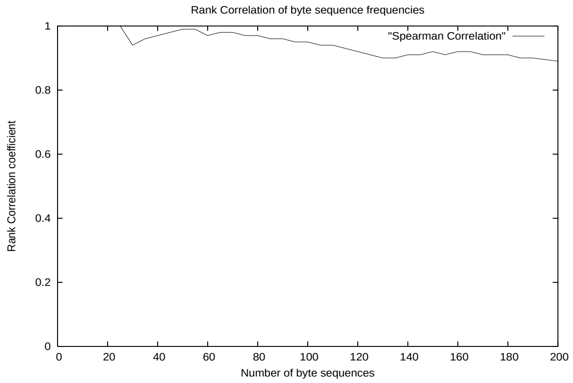


Figure 5: Spearman's rank correlation for number of byte sequences between Jagran and Webdunia font encodings

Since these byte sequences are an ordered set, ordered by their normalized frequency, the precision of transliteration obtained by providing mappings between encodings in the order provided in the ordered set is optimal. We have observed that with about 2,000 encoding mappings for each encoding on average once can achieve around 99% precision. However this number also depends on the language complexity. For instance, the number of encodings required in Telugu transliteration is more than the number of encodings required in Hindi to obtain the same amount of precision.

We now report some of our experiments on the Indian language focused crawling. We ran a daily crawl for 6 months period. Our crawler was focused to fetch content in the top 10 spoken languages in India, namely Hindi, Telugu, Tamil, Bengali, Marathi, Gujarati, Kannada, Malayalam, Oriya and Punjabi. In another experiment, in order to find the effectiveness of language focused crawling, we executed the crawler in two modes with a set of 100 seed URLs which constitute popular Indian based web portals, news sites and home pages of people of Indian origin. In the first mode it was executed without language focus restriction using a pure FIFO crawl queue while the second mode was with language focus restriction using a priority queue from which the crawler fetched the next crawl URL. We plotted the number of relevant pages fetched in the first 50,000 URLs in both the runs as shown in the Figure 6. The relevance of the fetched pages was calculated by checking the encoding on that page. It can be clearly seen that language focus restriction on the crawler helps in downloading more relevant pages.

From the 6 month crawl, about half a million unique documents were collected from all the languages. Unique web pages were picked after eliminating approximate duplicate pages using shingling technique [4]. These half a million pages were distributed across the 10 languages as shown in the Figure 7. Figure 8 shows the population of people speaking the various Indian languages [3]. It can be observed that even within India there is a divide in the web publishing activity in various languages. For instance it can be observed that the content is actively getting published in south Indian languages like Telugu, Tamil and Malayalam when compared to the northern languages such as Marathi, Gujarati, Oriya, Bengali and Punjabi. Hindi has the majority of content published on the web but Hindi is also the language spoken by majority of Indian population.

It can be seen from Figure 10 that a very few websites publish content using a global standard such as Unicode. This explains the reason for most of the Indian language not being indexed or searchable by the present day popular web search engines. On the other hand it can be seen from Figure 9 and Figure 11 that the number of unique encodings found on the web for these languages is almost equivalent to the number of websites. This observation suggests that every web publisher is coming up with their own proprietary encodings to publish web content. We did not consider the websites that publish using images in this study, but our preliminary study suggests that there are a large number of websites that publish content as images as well.

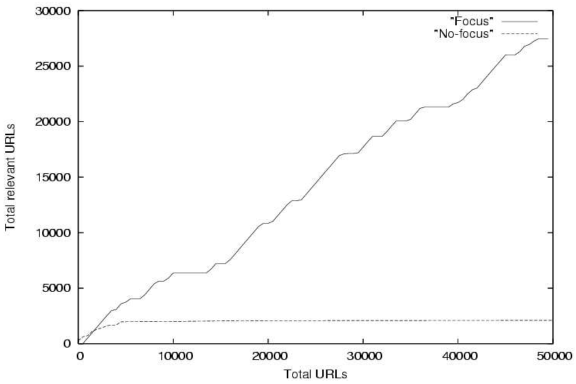


Figure 6: Crawl with and without language focus

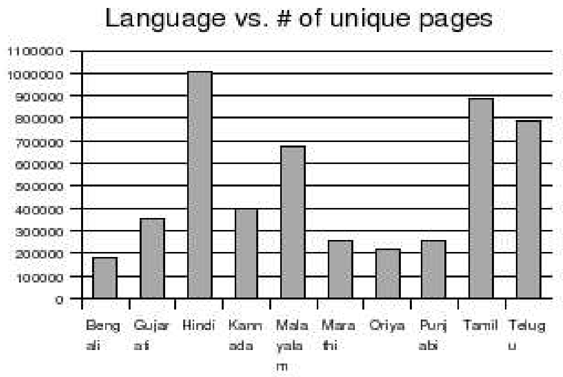


Figure 7: Languages on x-axis and number of unique web pages on y-axis

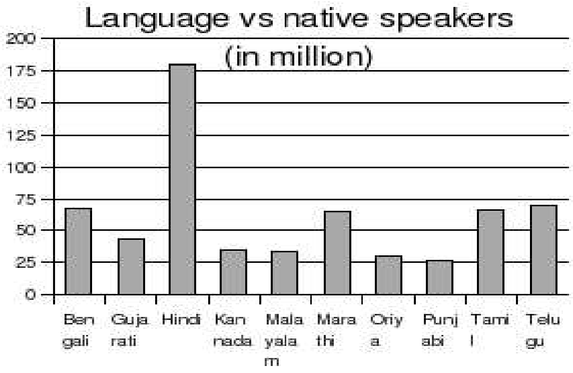


Figure 8: Languages on x-axis and number of native speakers on y-axis

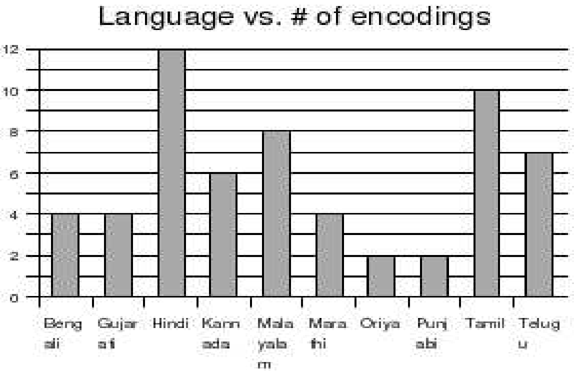


Figure 9: Languages on x-axis and number of encodings found on web including UTF-8 on y-axis

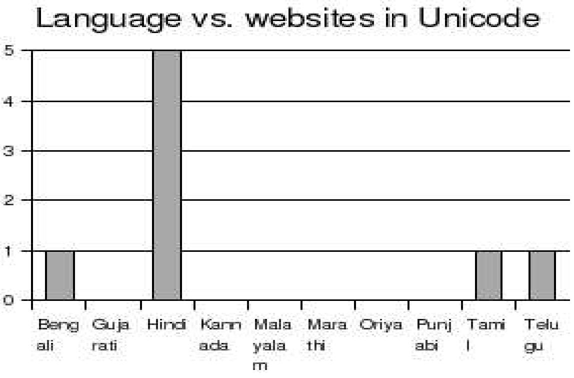


Figure 10: Languages on x-axis and number of UTF-8 websites on y-axis

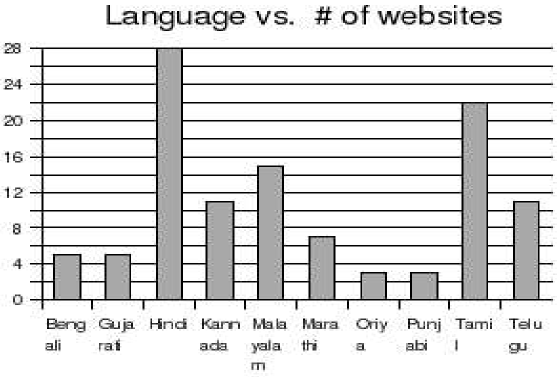


Figure 11: Languages on x-axis and number of websites (web servers) on y-axis

5. CONCLUSIONS

In this paper we discussed the importance of being able to search the Indian language web content and presented a web search engine which takes the UTF-8 queries from a soft keyboard and capable of searching 10 most spoken Indian languages' web pages encoded in multiple encodings. We presented a language focussed crawler which can fetch web pages of specific languages and also the distribution of the Indian language content on web based on the pages that were crawled. This distribution clearly shows the need for processes and algorithms to transcode non-Unicode encodings to Unicode. Hence we have discussed a semi-automatic algorithm to generate the mappings between different encodings. This shows that transcoding of proprietary encodings into a standard encoding makes Indian language web content accessible through search engines.

6. ACKNOWLEDGMENTS

We would like to thank the Department of Science and Technology, Ministry of Communications and IT, Government of India for funding this project.

7. REFERENCES

[1] J. Allan, J. Aslam, N. Belkin, C. Buckley, J. Callan, B. Croft, S. Dumais, N. Fuhr, D. Harman, D. J. Harper, D. Hiemstra, T. Hofmann, E. Hovy, W. Kraaij, J. Lafferty, V. Lavrenko, D. Lewis, L. Liddy, R. Manmatha, A. McCallum, J. Ponte, J. Prager, D. Radev, P. Resnik, S. Robertson, R. Rosenfeld, S. Roukos, M. Sanderson, R. Schwartz, A. Singhal, A. Smeaton, H. Turtle, E. Voorhees, R. Weischedel, J. Xu, and C. Zhai. Challenges in Information Retrieval and Language Modeling: Report of a Workshop held at the Center for Intelligent Information Retrieval, University of Massachusetts Amherst, September 2002. *SIGIR Forum*, 37(1):31–47, 2003.

[2] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the Web. *ACM Trans. Inter. Tech.*, 1(1):2–43, 2001.

[3] G. B. *14th ed. Ethnologue: Languages of the World.* SIL International, Dallas, TX, 2003.

[4] S. Brin, J. Davis, and H. Garcia-Molina. Copy Detection Mechanisms for Digital Documents. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 398–409, New York, NY, USA, 1995. ACM Press.

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

[6] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated Focused Crawling through Online Relevance Feedback. In *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, pages 148–159, New York, NY, USA, 2002. ACM Press.

[7] F. Gey, N. Kando, and C. Peters. Cross Language Information Retrieval: A Research Roadmap. *SIGIR Forum*, 36(2):72–80, 2002.

[8] Y. Haralambous and G. Bella. Injecting Information into Atomic Units of Text. In *DocEng '05: Proceedings of the 2005 ACM Symposium on Document Engineering*, pages 134–142, New York, NY, USA, 2005. ACM Press.

[9] A. Joshi, A. Ganu, A. Chand, V. Parmar, and G. Mathur. Keylekh: a Keyboard for Text Entry in Indic Scripts. In *CHI '04: CHI '04 Extended Abstracts on Human Factors in Computing Systems*, pages 928–942, New York, NY, USA, 2004. ACM Press.

[10] L. S. Larkey, M. E. Connell, and N. Abduljaleel. Hindi CLIR in thirty days. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(2):130–142, 2003.

[11] D. P. Madalli. Unicode for Multilingual Representation in Digital Libraries from the Indian Perspective. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 398–398, New York, NY, USA, 2002. ACM Press.

[12] P. Pingali and V. Varma. Word Normalization in Indian Languages. In *ICON05: Proceedings of the 2005 International Conference on Natural Language Processing*, 2005.

[13] G. Salton and C. Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Process. Management*, 24(5):513–523, 1988.

[14] S. Strassel, M. Maxwell, and C. Cieri. Linguistic Resource Creation for Research and Technology Development: A Recent Experiment. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(2):101–117, 2003.

[15] F. Yergeau. *UTF-8, a transformation format of ISO 10646*. RFC Editor, United States, 2003.