7. Indic Computing

Indlinux Team, TeNeT Group, IIT Madras : Ms. Hema Murthy, *E-mail:hema@iitm.ac.in*

ABSTRACT

Although India's average literacy level is about 65%, less than 5% of India's population can use English for communication.And even though the world-wide web and computer communication has given us access to information at the click of a mouse, 95% of our population is excluded from this revolution due to dominance of English. To overcome this problem we propose to set up an Indian Language Systems Laboratory at IIT Madras. Our initial goal will be to develop a *multimodal* interface to the computer that is relevant for India, i.e., one that enables Indic computing. The components of this Indian Language interface will be:

1. Keyboard and display interface
2. Speech interface
3. Handwriting interface

1. INTRODUCTION

Imagine a villager walking into a rural Internet kiosk, who may be semi-literate or even illiterate, wanting to use the power of the Internet to either communicate with a relative somewhere else, or contact a city hospital, or get vital crop information. The currently available English-based keyboard and applications are totally unfamiliar and intimidating. As a result he or she feels shut out and not part of the ongoing information revolution. On the other hand, had the computer been able to accept speech input, and has applications in the local language, this typical villager would have been as comfortable using it as anyone else.

Computers have become an essential part of many facets of our lives. However, in the Indian context the use of computers is far less compared with that in the developed nations of the West because of the reason we have already hinted at: the language of the interface is almost always English and the communication is in the "written" form, i.e., via the keyboard. Barely 65 % of our population is literate, of which only an elite minority (~5%) can read, write, and speak the English language. This shuts out most of the Indian population from the world wide web and its huge potential. Therefore it is essential to have an interface that uses not only the local language but also speech, to cater to the needs of the semi-literate and illiterate sections of the population. Moreover, the current keyboard has been developed for English and cannot be naturally adapted for Indian languages. Hence there is a need for a handwriting interface too. We call such an interface as being *multimodal*. In the West, although speech and handwriting interfaces are available for English, these are for some specific applications, namely, dictation machines (hands-free), limited handwriting/graffiti recognition as in PDAs. This is

primarily because of the simplicity of the Roman script. Whereas, in the Indian context, these interfaces must be part of main stream applications, namely mail readers, web browsers, word processors. This is the FIRST such effort to seamlessly integrate all three interfaces.

Developing the multimodal interface is a significant multi-disciplinary effort. At IIT Madras (IITM) different groups have been working on aspects of the individual interfaces. To develop an effective solution it is essential for these groups to work

cohesively toward a long-term goal over several years. Clearly, there will also be specific deliverables each year. We have set up an Indian Language Systems Laboratory at IITM with a view to developing a multimodal interface for Indic computing, drawing upon the already available expertise and collaborating with leaders in areas where we lack the needed capability. This multimodal interface has the potential to revolutionize Indic computing. Hence our approach and design will be shaped by the need to deploy it commercially on a wide scale. The larger goal of the proposed lab is to focus on various issues related to Indic computing.

The TeNet group at IITM has pioneered the increase in connectivity to villages by developing the corDECT wireless local loop technology [7]. Already more than one million connections have been established by n-Logue communications in the Internet kiosks it has set up throughout the country. Hence the need for a multimodal interface for Indic computing is not a theoretical assessment but one resulting from practical need. Moreover, it provides the framework for widespread deployment and use.

In the following Sections we will give a brief overview of the R&D work that is needed, identify the issues, and highlight the expertise available at IITM to tackle them. We have also identified groups outside IITM with whom we wish to collaborate. The specific deliverables are given in Section 2.3.

2. THE MULTIMODAL INTERFACE(MMI)

The multimodal interface interacts with the user via several input mechanisms (such as keyboard, mouse, speech, and handwriting) and several output mechanisms (such as display and voice). Ideally, the computer should automatically choose the most appropriate one, based on the context and the user. This very challenging task is our long-term goal. Initially, to simplify the problem, we will require the user to specify the mechanism, e.g., choose speech for the output.

Even here, error-free recognition of continuous speech independent of the speaker, speaking style, microphone, and surrounding conditions still lies in the realm of Science Fiction. What is practically possible with the current technology, given the conditions in a typical kiosk, is achieving good accuracy on limited vocabulary tasks. Similarly, the quality of current synthesizers is such that they have only limited commercial acceptability. However, there is ample scope for improving the performance of these interfaces, not only by just improving the basic technology, but also by cleverly designing the man-machine interface (MMI) such that their effective performance increases, leading to wider user acceptability.

2.1. Input Mechanisms

2.1.1. The Keyboard and Character Encoding

Keyboards that are easy to use for Indian languages are possible only if character encoding is based on a well-thought-out design. The design issues become even more critical if we have to deal with multiple languages.

Today, applications in local languages are beginning to appear. But they are not only language specific but also dependent on the character encoding and font used! We have twenty two official languages and many more unofficial ones. With the existing approach, developing language-specific applications is a herculean task. Instead, the correct approach will be to develop applications *independent of the language, font, and encoding by making changes at the operating-system level*. If this is done, all applications will inherit the new interface, obviating the need to recompile whenever the language is changed.

We have already begun work on such an interface in which modifications have been made at the kernel level (see Figure 1), X-lib level (see Figure 2), eliminating the need to recompile an application whenever there is a language change [6].Changing the language is simply a matter of changing configuration files. All input passes through a language filter. Further improvements will be done as part of the proposed work. The support provided at the X-lib level supports a number of encoding schemes and fonts.

The display mecahnism at the console is taken care by the console, the tty and the video device drivers in the kernel. Provision of variable-width font support in all respects, will require the modification of all these drivers. The solution adopted by us it to display multiple glyphs for a single character code in order to display wider fonts. In this case, the glyphs are still of fixed size and a one-to-many mapping mechanism is introduced in the display pipeline. In this design, only the console and TTY drivers need to be changed.

The X window system based solution is to provide a virtual terminal similar to xterm, which supports Indian scripts in addition to the normal English script. The work has been based on the design and implementation of rxvt-idev [6], which is an extension of RXVT-2.4.5. A library providing the necessary funtions to handle the keyboard layout and parse rules for each supported language is maintained for this purpose. RXVT 2.6.2, a virtual termainal has been extended to support the Indian scripts.

Non-uniform encoding is another important issue for Indian languages. In English, one keystroke corresponds to one letter and there are no character clusters. But in a typical Indian language there are roughly 3000 character clusters. Therefore, the number of keys on the keyboard will become unmanageably large if we want a single keystroke for each cluster. Instead,we will have to make do with a sequence of keystrokes. As a result, the encoding will not be uniform for all character

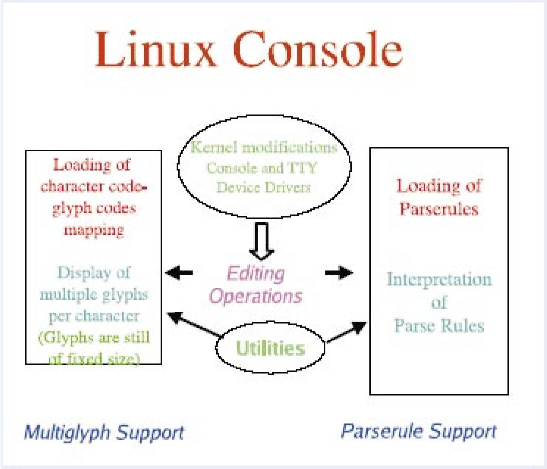

Linux Console
Kernel modifications Console and TTY Device Drivers
Loading of character code-glyph codes mapping
Display of multiple glyphs per character (Glyphs are still of fixed size)
Editing Operations
Utilities
Loading of Parserules
Interpretation of Parse Rules
Multiglyph Support
Parserule Support

Fig. 1. Kernel level modifications

clusters—it can vary from a single byte to three bytes. Since the number of characters in the script corresponding to every language is different, we propose to support different encoding schemes—for example, mapping could be made dependent on the frequency of usage of a cluster in written text.

Finally, the present keyboard is designed for English. The efforts for developing keyboards for Indian languages has been along the lines of remapping the keys of the QWERTY keyboard. This is a very unnatural adaptation. Instead, we propose the keyboard be designed from scratch, keeping in mind the similarities that exist between all the Indian languages. If there are important differences, those must also be taken into account and localized accordingly. For example, Tamil does not have aspirated consonants, which frees up keys that can be used for some of the most frequently occurring character clusters.

2.1.2. Speech Recognition

The goal of this input interface is to accept speech and take appropriate action by recognizing what is spoken. The action could be anything that the current keyboard interface is capable of, e.g., change directory, open a file, etc. However, certain kinds of tasks are more easily carried out using the keyboard. Hence the speech interface will coexist with the keyboard and not replace it. Clearly continuous speech recognition is the core technology for the input interface.

To be able to make progress in speech recognition having standardized databases is an essential pre-requisite. One of the major issues in Speech Recognition for Indian languages is the lack of such databases. In the US, a variety of databases for English have been collected over two decades, in a variety of conditions (clean speech, telephone, cellular, etc.), spanning different applications. By contrast, we do not have a collection that is even remotely close for even one Indian language. Collecting such databases is a significant effort and we need a concerted effort to rectify this basic deficiency. We would like to pioneer the Indian language speech database collection. Initially we would like to collect data for two Indian Languages, viz. Hindi and Tamil, and then work toward data collection for all Indian Languages. Databases are also required for speech

synthesis, but the requirements are different from those for handwriting recognition.

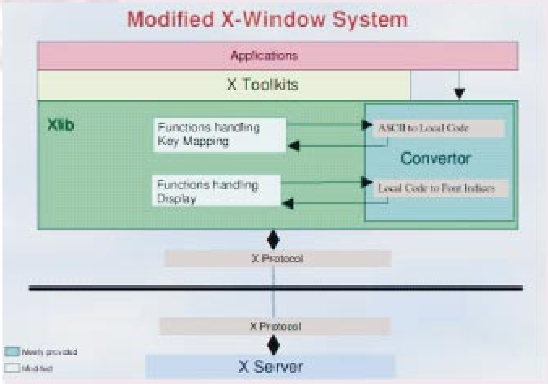


Fig. 2. Modification to X-Lib to support multiple encoding schemes

Before collecting a database, it has to be carefully designed by an expert linguist, taking the application into consideration. Annotation of the collected data requires careful development of a number of tools (2.3).

2.1.3. Handwriting Recognition

Although Indian language computing will significantly increase number of non-English speaking public to use the computer, the corresponding keyboard is very cumbersome to use. As we pointed out earlier, a typical Indian language has roughly 3000 character clusters (here by character cluster we mean the C*V or V(C stands for "consonant" and V

for "vowel"). On a keyboard that is restricted to 256 possible scancodes, it is not possible to avoid multiple keystrokes for most of the clusters except perhaps the most frequently occurring ones. An attractive alternative is the handwriting interface.

Handwritten character recognition is of two varieties depending on the means used to produce it: (i) scanned image of handwritten text on paper, and (ii) handwritten text produced by an electronic pen, where the pen trajectory on a special tablet is processed by the computer. The latter is known as Online Handwritten Character Recognition (OHCR). A handwritten character is composed of a set of pen strokes. A stroke is a line drawn by a pen between the time when a pen touches the writing surface and the time when it is lifted. Therefore, identifying the component strokes of a character is the first step to character recognition. The following issues need to be addressed:

- Database of Strokes: Stroke databases for all Indian scripts is not available. We would like to collect the needed databases. We would like to exploit the structural similarities among various Indian languages by using stroke databases shared among multiple languages and come up with a smaller-sized solution.
- Output format: Currently ISCII is a widely accepted standard for representing Indian language characters. More recently different standards have started emerging for different Indian languages. In this context, we will borrow from the effort on keyboard and display interfaces to produce output based on the encoding used.
- Interfaces: An OHCR system consists of four main components. The 1) input area where the writer enters content, 2) the stream of stroke IDs, 3) the stream of characters, and finally 4) the output. Between every stage and its successive one, interfaces must be defined and standardized. Standard interfaces are the key to the longevity of a software.

2.2. Output Mechanisms

In this section we discuss different output mechanisms. Although humans have five different sensors the focus here is on the visual and auditory sensors.

2.2.1. Display

Unlike in English, the display corresponding to a keystroke is not only dependent upon the past keystrokes but also on the future. Enabling such support requires a language model for the representation of graphemes.1 For this, the kernel must be enhanced to support such language models. Additionally, the widths of the graphemes vary significantly, making variablewidth fonts essential (fixed-width fonts will give ugly results). There are a number of issues associated with variable-width fonts that we propose to address. For example, the width of the grapheme that represents h in Tamil is four times as large of the grapheme that represents r (h and r are symbols taken from the International Phonetic Alphabet).

As mentioned earlier, a sequence of keystrokes may have to be pressed to generate a single character cluster. Further, as one types, the cluster will have to be modified to ensure that it corresponds to the key sequence in progress. This brings in the related issue of the positioning of the cursor. The rendering is being taken care by using an operating system that supports UNICODE. Internally all data is represented in UNICODE.

Currently cursor positioning is handled by the rendering engine. Once the interface is available, applications need to be customized for each language.

2.2.2. Speech Synthesis

The speech output interface uses speech synthesis as its core technology. Similar to its input interface counterpart, it will not replace the monitor but supplement it. This is because there are many instances where the output is not text; even for text output it is more convenient in many cases to see the display on the screen rather than having it converted to speech and played out (especially when the size of the text message exceeds a few words).

Our goal is to synthesize natural sounding speech in all the official Indian languages. We have used a common diphone database to synthesize speech in multiple Indian languages. A data driven approach was used to extract the prosody from labeled Doordharshan news bulletins for Telugu and Hindi. The prosodic rules obtained were plugged in the freeware Festival (see ??) . It is observed that prosody makes a significant difference to perception. We are currently looking at alternate models, namely, a syllable-like unit for speech synthesis.

Similar to the speech recognition effort in India although synthesis of intelligible speech is a simpler task than recognition, the major lacuna seems to be the official onavailability of databases and a systematic linguistic analysis of languages. See Section ?? for more details.The databases required for synthesis are quite different from those required for recognition. Nonsense words spoken by a single speaker with minimal variations in pitch is typical.[2]

Significant linguistic expertise is needed for language analysis and design of databases. Some of the issues that need to be addressed are:

- Choice of basic unit: How large should the basic unit be to enable unrestricted speech synthesis? Can the unit be a common choice for all Indian languages?
- Development of natural sounding speech: The naturalness of the synthesized speech crucially depends upon the prosody that is associated with it. What kinds of prosodic models have to be developed to produce natural-sounding speech?

Although a large number of studies are available from language departments across the country, there are no studies that are driven from the requirements of synthesizers. To this end, we plan to collaborate with The International School of Dravidian Linguistics (ISDL), Trivandrum, and Central Institute of Indian Languages (CIIL), Mysore, and other Institutes, wherever such effort is available, to develop prosodic models for synthesis.

2.3. Current Status at IIT Madras – Integration of MMI into applications

Recently, we have integrated the Multimodal interface with KDE based applications. The speech interface is primitive, primarily owing to the lack of comprehensive databases. As we pointed out in Section 2.1.2 one cannot overstate the importance of comprehensive, high-quality databases for making headway in speech recognition, synthesis, and andwritten character recognition. The success of Indic computing highly depends on the performance of multimodal interface, which in turn is dependent on the quality and quantity of the databases. We strongly feel that our immediate priority would be to create speech and handwriting databases in all the major Indian languages.

We have collected speech data in Tamil for Isolated Word Recognition and Continuous Speech Recognition. More about the collection of data and its' implementation is given in section (2.3.1)

So far, we have also integrated handwriting recognition in Tamil with various applications. This process also involves sufficient data collection from various people who know the script of the language. Although, we have collected a database of Telugu, Hindi and Malyalam scripts, we are yet to integrate it to various applications. This integration is not too far off, as it is matter of implementing what has already been done in Tamil.

Figure 5 shows the integration of a multimodal interface with a word processor. On the top right hand corner, icons are provided for speech and handwriting input. On clicking the speech icon, the speech interface is overlayed (figure.3), the speaker is asked to record word by word. The recorded word is recognized and placed at the cursor position in the word processor. Similarly clicking the handwriting icon (figure.6), displays a write strip. The user writes continuously giving sufficient blank spaces between words. The recognized words are then placed at the cursor position in the word processor.

Information on the speech and handwriting interfaces and their integration with various applications have been discussed in detail below.

---

[1]Graphemes are the set of units of a writing system (as letters and letter combinations) that represent a phoneme.

[2]Whereas, for speech recognition we want the database to contain as many speakers as possible, speaking naturally.

2.3.1. Speech Recognition and Its' Integration to Some Applications

With respect to the speech interface, we have already collected about 500 words in Tamil with a particular reference to Cricket from approximately 20 speakers in a laboratory environment. We are aware that the database for the isolated word recognizer is small and are working towards increasing the size of the database, both in terms of the number of words, as well as the number of speakers. We intend expanding this database to 5000 words in Tamil with a reference to sports. We have also collected continuous read speech in Tamil (news) from speakers in a kiosk environment.

This database is used for various applications that have already been implemented in our lab. Details of the applications with respect to the speech interface follow:

- IsolatedWord Recognition:
  - Interface with KOffice: Approximately 500 words have been recorded from 20 speakers so far. The unique words were obtained from news articles with a reference to cricket. Hidden Markov Models (HMMs) were generated using HMM ToolKit (HTK). The application is now loaded on to KOffice to be used with KWord, KSheet etc. On clicking the speech icon which is included in the KOffice appications, the speech interface is overlayed (figure.3), the speaker is asked to record word by word. The recorded word is recognized and placed at the cursor position in the word processor.

    Figure 3 is a snapshot of speech recognition interfaced to the KOffice word processor KWord. A GUI shown in the center of the image is opened by clicking an icon on the KOffice appliction. The icon is also seen in the image as the one with a microphone. The opened GUI shows options that indicate "start recording", "stop recording" and "close". These options are used by the user to start, stop or close the GUI.
  - Speech interface for web forms: The aim in this case is to fill in a web based form using speech interface. For this application, a local proxy and a plugin have been designed to first check if the webpage requested by the user is a web form, and if so, an interface for speech input is also displayed along with the web form. A database of isolated words needs to be identified, for each form, so as to develop isolated word models for each of these forms. More information about this process has been discussed in [1]. We have implemented this design on the railways Speech Input Method reservation form which is available on the net. HTK has been used to generate models of certain city names, train numbers, dates etc., to be used in the railways reservation form. The cursor is placed in the appropriate box, where data entry needs to be made. A button, generated as part of the plugin, aids in recording of the input utterance. A back end word recognizer, recognizes the word uttered by the user and displays the word in the box intended. In this manner the whole form can be filled with no input from the keyboard.

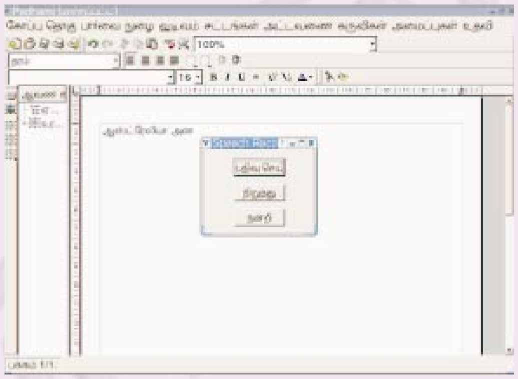


Fig. 3. Speech interface to the word processor

- Continuous Speech Recognition:

Continuous speech recognition is used to transcribe continuously uttered speech. This kind of speech in turn can be grouped under various categories, like, read speech, conversation, etc. Our immediate aim is to build a speech recognizer for read speech. The task in this case is considerably more difficult than building an isolated speech recognizer. We have collected continuous read speech in Tamil in a kiosk environment. The data collected from various speakers is brought back to the lab, where we automatically segment and label the various syllable-like units encountered. A number of tools for segmenting and labeling the speech data at the level of syllable-like units have been developed (see Fig.4). The segmenting and labeling performance is about 50% for Doordharshan News Bullettins. We are working at improving the performance of the segmentation and labelling tools.

Once segmented and labeled, the syllable-like units are clustered together based on similarity of the various units. Models of the syllable-like units are then generated using Hidden Markov Modeling ToolKit. When a test utterance is inputted in to the system, it is segmented using the segmentation tool in to syllable-like units. This units would then be compared to the

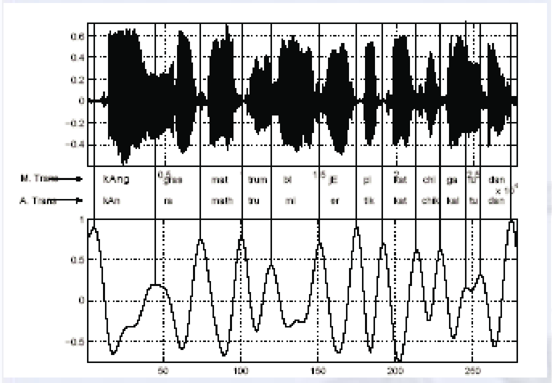


Fig. 4. Segmentation and labeling of continuous speech

models already available in the database. The label for the most likely model is then outputted on the screen, resulting in transcription of the read speech. This recognizer will further be used for building a spontaneous or conversation speech recognizer.

2.3.2. Handwriting recognition and Its' interface to various Applications

As with speech recognition, the applications of handwriting recognition seem endless. As mentioned earlier, many applications have been enabled with a speech interface, providing a fairly good performance. Figure 5 shows KWord with icons to enable both speech and handwriting interface. For the handwriting interface, a good database of strokes in the language of interest is required. We collected handwriting data from 10 people (each person giving 10 samples of each character), from which a database of strokes was generated. This database was useful in identifying the strokes inputted by user (and thereby identifying the character set inputted by the user) in the following applications:

Interface with Koffice: Currently we have interfaced KOffice with handwriting recognition. An icon on Koffice applications like KWord or KSheet etc., helps open the GUI for handwriting interface. When the user writes on this GUI, the strokes written by the user and consequently the character itself is recognized and displayed on the application. A snapshot of the GUI and KWord is shown in figure 6. The GUI is invoked by using the handwriting interface icon shown on the top right hand corner of the image. We have integrated a list of approximately 350,000 words obtained from IIIT, Hyderabad on to the *aspell* spell checker [5]. Also integrated are the 500 words with a particular reference to cricket and words used for the railways reservation form. Additionally, for a small list of words it is easy to generate a word complete algorithm. This algorithm has been implemented for the 500 words with reference to cricket. The GUI has a button which enables this word completion algorithm.

Padhami-A Tamil Wordprocessor with multi model input

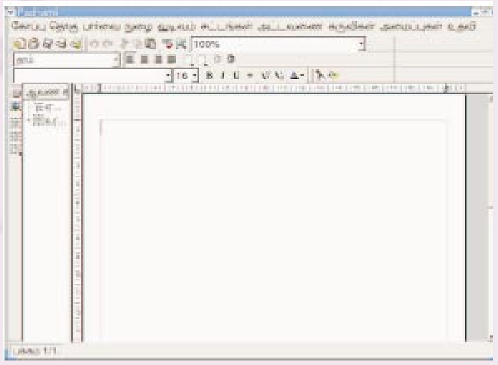


Fig. 5. Word processor enabled with speech and handwriting

- Search Engine: The Tamil language like any other language, can be shown in many different fonts. There are many search engines available that do effective searching in the Tamil language. However, to view all pages that are displayed by the search engine, fonts for each of those pages need to be downloaded. This is an inefficient way of viewing the fonts. The issue can be circumvented, by having a back end which downloads the requested page, converts it to Unicode font and displaying the original webpage in Unicode font [3]. For this application, maptables need to be maintained, which show mapping of all available Tamil fonts to Unicode. The webpage obtained from the search engine is downloaded, checked for the type of font, converted to Unicode and then displayed. One major problem that we face is constructing an error proof maptable and maintaining a good database of maptables. Currently, the search engine has also been interfaced with handwriting recognition [2]. A handwriting interface is provided at the client end via a GUI. A snapshot of this GUI is shown in figure (7). The user uses to mouse to write on the handwriting area on the GUI. The coordinates of the points of the characters drawn by the user, are collected by a simple applet. On clicking a send button on the applet, the points are sent to a server where the handwriting recognizer is invoked. The recognized word is displayed on the search engine on the client side, and additionally, a list of best matches is also sent back to the client. Conversion of webpages from various fonts to Unicode takes place and the client can read the contents of various webpages with ease. A snapshot of results obtained for a word called "Arasu" is shown in figure (8).

Handwritten Character Recognition

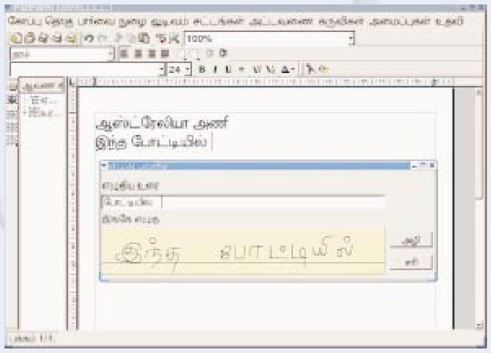


Fig. 6. Handwriting interface to the word processor

_ Handwriting interface for Web forms: We have integrated handwriting recognition to enable filling forms available on the web. Much of the technology has already been explained in detail in the previous section. Stroke (and thereby character) recognition is performed in the same manner as handwriting recognition in the other cases. Just as we have integrated Tamil language handwriting recognition on to KOffice and search engine, we intend to integrate as many languages as possible. We have already collected such data for Telugu, Hindi and Malayalam.

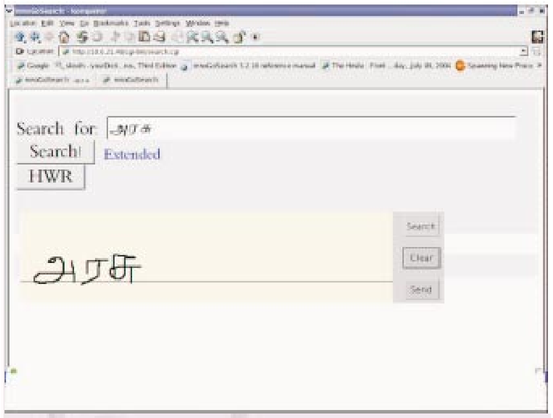
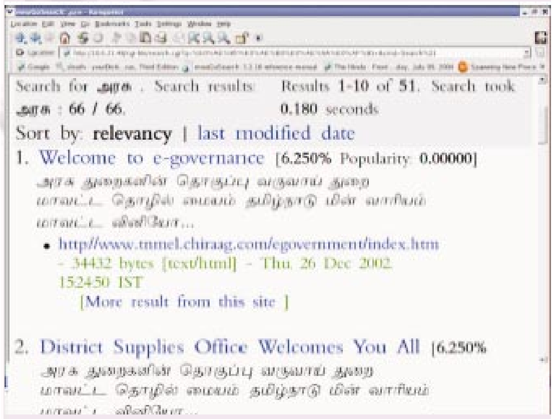


Fig. 7. Handwriting Interface to Tamil Search Engine



Search for அரசு . Search results: Results 1-10 of 51. Search took
அரசு : 66 / 66. 0.180 seconds

Sort by: **relevancy** | last modified date

1. Welcome to e-governance [6.250% Popularity 0.00000]
   அரசு துறைகளின் தொகுப்பு வருவாய் துறை
   மாவட்ட தொழில் மையம் தமிழ்நாடு மின் வாரியம்
   மாவட்ட விளிம்பொ...
   - http//www.tnmel.chiraag.com/egovernment/index.htm - 34432 bytes [text/html] - Thu, 26 Dec 2002 152450 IST
     [More result from this site]

2. District Supplies Office Welcomes You All [6.250%
   அரசு துறைகளின் தொகுப்பு வருவாய் துறை
   மாவட்ட தொழில் மையம் தமிழ்நாடு மின் வாரியம்

Fig. 8. Results Obtained from Tamil Search Engine

3. CONCLUSION

It is hoped that development of a multimodal interfaces to the computer, will bridge the gap between the haves and the havens and will go a long in the empowerment of the rural folk in India. With the proliferation of the Internet across the countryside,Mahatma Gandhi's dream of Village Swaraj can indeed become a *reality* (at least in the Information Technology sphere).

4. REFERENCES

[1] N. Hemalatha, M. Kasirajan, M. Palaniselvam, M.S. Vidhya, G. Vijayaprakash, T. Nagarajan, and Hema A. Murthy

”Multimodal Interface EnabledWeb-based Form Filling Applications” ICON-2004 (communicated)

[2] Sowjanya, S., Viji, ”Handwriting Interface Enabled Tamil Search Engine” NCC-2004, IIT Kharagpur, India (Communicated).

[3] http://www.mnogo.ru

[4] K.H. Aparna, Vidhya Subramanian, M. Kasirajan, G. Vijay Prakash, V.S.Chakravarthy, Sriganes Madhvanath, Online Handwriting Recognition for Tamil, To be presented at Ninth International Workshop on Frontiers in Handwriting

Recognition (IWFHR-9 2004), Kokubunji, Tokyo, Japan, October, 2004.

[5] http://www.iiit.net/ltrc/morph/

[6] Anitha Nalluri, Bala Saraswathi A, Bharathi S, Hema A Murthy, Patricia J, Timothy A Gonsalves, Vidhya M S,

Vivekanathan K, ” Indian Language Support for X - Window System,” in *Proceedings of the ICON 2002*, (SP-P6.4, May 2004).

[7] http://tenet.res.in/

[8] http://www.cstr.ac.uk/projects/festival