Barriers to Localisation

a case study of Nepal

Pat Hall

brief background to Nepal

language diversity in South Asia

linguistic history and politics

Writing through the computer

an old mechanical typewriter
direct connection between key and type
early computers
based on typewriters
dot matrix printer
electrical signals
internal codes
communications/applications
ASCII
60-70 Roman
letters
5-10 control
7+1 bit
ISO646
EBCDIC
8 bits
more characters
internal codes
communications/applications

indic “hack font”
developers
keyboards, codes, fonts dependent
1. design your keyboard
2. accept whatever internal code arises
3. use tool to create fonts
key
conceptual
inkmage
one-to-one correspondence
font table
code -> bitmap
disaster
NO communication

encodings
• ISCII
  – IIT Kanpur, then CDAC Pune
  – letters, not glyphs
  – like Arabic
  – needs renderer
  – based on Brahmi view
  – single code table with language switch to render “same” letter appropriately
  – sequence of characters as spoken
• Unicode
  – based on ISCII but separate tables for each script
  – still controversial
    – new Tamil standard
modern computers
separate issues
key mapping
key -> code
font table
code -> bitmap
laser or inkjet printer
key
conceptual
inking
one-to-one correspondence
internal Unicodes
communications/applications
letters and characters
not graphics

Nepali language into Computers
• Desk top publishing with PCs
  – hack fonts to give some representation of the writing
  – 128 places not enough for everything
  – implicitly defined an internal code
  – fonts copied widely and then changed slightly
  – result – inability to exchange data
  – 1997 attempts to standardise in context of emerging Unicode
  – improved by ISCII, but not international
  – some fonts also developed for other languages
• today, accept Unicode Devanagari for Nepali
  – 1997 IDRC and later UNDP UNESCO funded keyboard drivers and fonts
  – can use Indian open type fonts, though not liked
  – still no agreed keyboard layout for Nepal

two key issues in technology projects
Research versus application
• doing ICT4D projects requires new understanding
• what understanding?
  – ACM paper claims must deliver computer science research
Technology transfer versus knowledge transfer
• do we do it for our beneficiaries?
• or teach them how to do it for themselves?
Could academic objectives cloud development objectives?
Nepalinux at MPP funded by IDRC

- 1997, 2000 produced Nepali Unicode-driver add on to Windows
- 2004 joined PAN localisation project
- Debian Linux with GNOME desktop, KDE desktop, OpenOffice, ...
  - release 1 December 2005
  - release 1.1 October 2006
- FOSS critically important here
- then PDAs, mobile phones, OCR
- in parallel Microsoft produced UP for Nepali
  - launched November 2005

what is involved in localisation?

- translation of all text in screens, menus, and help
  - available in separate 'resource' files
  - agree terminology in national committees
  - store past translations in 'translation memory'
- develop spell checker
  - need standard spellings
  - morphology
- other capabilities that might be needed
  - code converters = hack fonts to unicode
  - develop fonts
  - text-to-speech
  - OCR
  - is a grammar checker needed?

NeLRaLEC - funded by the EU

for Nepali, even though not (yet) endangered

- Nepali National Corpus
  - text (50M words, 100K parallel) and speech (4hrs=130K words)
- Nepali dictionary - aiming at 100,000 words
  - word list and entries for most frequent
- linguistic tools
  - fonts for Nepali
  - and Maithili
  - speech generation
  - trials in schools and universities
  - localised schools management software
  - computational and corpus linguistics course in University

Nepali National Corpus - what people actually write

advice from Lancaster University

- wanted material from 1991/2 to match corpora in US and UK
  - not much material then, just after end of repressive Panchayat era
- wanted in fixed genres with strong western bias
  - very little science and technology, 1 science fiction, no westerns (or Kung Fu)
- CORE - aimed a 1 million words
  - collected 500 documents, had them typed
  - some got lost in civil disturbances and lack of records
- General – opportunistic, at least 4 million words
  - already digitzed but in hack fonts, needed conversion
  - when to stop?
  - English dictionaries used 200 million words or more
  - eventually stopped at 13 million words, good enough will do

Nepali Corpus-based Dictionary

first ever in South Asia

- needed software to store entries as produced
  - multiple lexicographers
  - could not find suitable existing system
- developed own system to meet needs of Nepali and linguists
  - 2 software engineers in exploratory development
- developed entries using Oxford’s Xiara concordance system
  - linguists still learning, each did things differently
  - at around 20,000 entries realised quality problem
- started again, reusing earlier entries or producing new ones
  - only reached 8,000, okay for an on-line dictionary, but not in print
  - expected linguists to continue after project
  - but didn't because not paid

Nepali Font Development

- hiring font developers
  - original person went to UK to do masters
  - recruited two graphic designers, arranged training
- short training from Reading in Kathmandu, open to public
  - some hazards from civil disturbances
- font development process
  - draw many examples of characters needed, including ligatures
  - scan into computer, add rules of combination
  - test and improve constantly
- eventually agreed needed further training in UK
  - two months in Reading
- outputs
  - font for Nepali in two versions
  - font for Maithili MithilaKsha style of writing
### Nepali TTS speech generation

Based on Festival/Festvox concatenative synthesis
- Expected help from UK, Roger Tucker and Ksenia Shalapova
  - Ksenia refused to travel to Nepal to give training and guidance
- Had to help our 2 software engineers in other ways
  - Spent 1 month in Hyderabad with Kishore and Rajeev Sangal
- Recorded voices
  - Selected words containing all 1764 diphones, chose 1,200 sentences
  - Had to research speech – eg “Schwa deletion”
- Developed TTS through several versions, adding prosody
  - Recorded voices had to help software engineers in other ways
  - Soiware engineers in other ways
  - Expected help from UK, Roger Tucker and Ksenia Shalapova

### Evaluation of use of Nepali software

Nepalinux and MS LIP launched late 2005
- 1000 CDs distributed, 500 attended demos
  - Heard not being used, interesting novelty
- Conducted surveys to find out
  - What was really going on, and why?
  - Grounded theory, analysed with NVivo
  - Done by Ganesh Ghimire and Maria Newton

### 1. First impressions of computers

**Problem:** Hardware not localised
- Cabinets labeled in English
- Keyboards not marked in local script

**Example:** Delivered computers to schools
- Could only give keyboard layout charts
  - Not well produced
  - Technicians claimed phonetic keyboards easy
  - But nobody spoke or typed English!
- Ironical – claim systems for non-speakers of English.

**Solution:** Get keyboards marked for Nepali
- Keyboards in Thailand are marked in Thai!

### 2. Translation quality

**Problem:** Technical terms not understood
- Nepali terms agreed by committee

**Example:**
- “I have used all its functions because I am a writer. Some of the words like "radditokari" and "anuprayog" seem to be the unusual ones. They look like they have been directly borrowed from Sanskrit and that make Nepali even more difficult than English.” – Banking officer
- “It’s good for people who are trained in Nepali that don’t have exposure to English. For people like us who have already started to use one system it becomes difficult to switch over” – Singh from Afghanistan.

**Solution:**
- Don’t translate, English terminology often bizarre, maybe transliterate
- Listen to users, standardise terminology
- Will they get used to it anyway?

### 3. Teacher/trainer inertia

**Problem:** Trainer not familiar with the localised software
- Example: at Sankhu telecentre
  - Language of instruction - Nepali
  - Started teaching Nepali interfaces
  - Switched back to English interfaces

**Solution:**
- Train the trainers
  - In software as well as hardware
  - Don’t give option of English interface

### 4. Critical mass of users

**Problem:** Want to get help from others

**Examples:**
- Nepal’s national library
- One user, only used to catalogue Nepali books
- Government officer taught himself

**Solution:** Need to create critical mass,
- Train complete organisation
  - Restrict opportunity to use English
    - Example: Telephone, video tape formats

**Sociology**
- “Social interaction” Brock and Durlauf
- “Social embeddedness”
5. language shift and social mobility

Problem: people change to dominant language
- see economic advantage in English or Nepali or ...

Example:
- “Yeah I liked, but when we used Nepali windows that time I feel we are going to forget English language” – computer academic
- “I have daughter and I will not ask my daughter to use the Nepali interface because I want my daughter to be good in English” – computer academic

Solution: accept that for some languages
- not worth localising the OS, but enable content
- supporting the language may reduce the shift

Sociology
- “sanskritisation” caste mobility - Srinivas

6. software eco-system

Problem: software works with other software
- cannot use Unicode for information exchange!
- other software is not Unicode compliant, use hack fonts
- but can do Desk Top Publishing

Example: journalists
- “But it has font problem. In publication houses mainly they use preeti, kantipur so it’s not worthy for nepali computing” – journalist

Solution: raise an Open Source project to produce compliant publishing software.

7. support all communities

Problem: language communities very small
- commercial development not viable

Example: Lohung Rai in Nepal
- subject of socio-linguist study by Jens Alfahren, Yogendra Yadava, and Bhim Regmi
- 1,207 ‘mother-tongue’ speakers
- language not yet written, want Roman system.

Solution:
- localise for local lingua franca
- create writing close to that of lingua franca
- only enable content in local language

8. cost of entry for new language

Problem: Gnome interface cost can be significant
Example: Gnome interface for Linux
- 40,000 ‘strings’ = 500,000 words approx
- grows by 5 to 10% a distribution
- at 1,500 words per day, this takes 300 days
- 30 days for a new release

Solution: avoid manual translation
- machine translation?
- be smart technically
- language generation from model of software

Conclusions

localization must be TOTAL
- all hardware including keyboards
- all software in use by a community
- translations sensitive to language politics
- otherwise it will rejected

entry cost must be minimal
- cheap and easy to localize for a new language
- base interaction by language generation
- part of s/w development process
- sometimes only enable content

next step – harmonised writing and encoding
- then machine (assisted) translation