Multilingual text processing difficulties

Malek Boualem, Jérôme Vinesse

CNET, France Telecom

1. Introduction

Users of more and more applications now require multilingual text processing tools, including word processors, database creation and management systems, desktop publishing systems and more recently multilingual web-based applications. In the area of automatic or machine-aided translation, multilingual text editors are a basic tool for pre-editing source text and post-editing target text. Another new area where multilingual text editors could be of great use is that of internationalization and localization of software and associated documentation for use in a multi-cultural environment. These areas were born as a direct effect of the emergence of new technology and the globalization of the Information Technology market. Many organizations and projects work to one extent or another within these areas (CEC, CEN, Esprit, Eureka, Internet, JSA, Linux International, Unicode, TEI, etc.).

Processing languages not based on the Roman alphabet poses a number of difficulties. For example:

- Arabic is written from right to left.
- Chinese contains thousands of ideograms, which obstructs a one byte coding.
- In Thai and other Indian languages, the sequence of characters does not correspond to its phonetic equivalent and one character may even be drawn encircling others.
- In Korean, characters are fused to make syllables.
- etc.

2. Multilingual text processing difficulties

Multilingual text processing difficulties occur on several levels: input (keyboard), coding, editing, printing and data exchange (see figure 1).

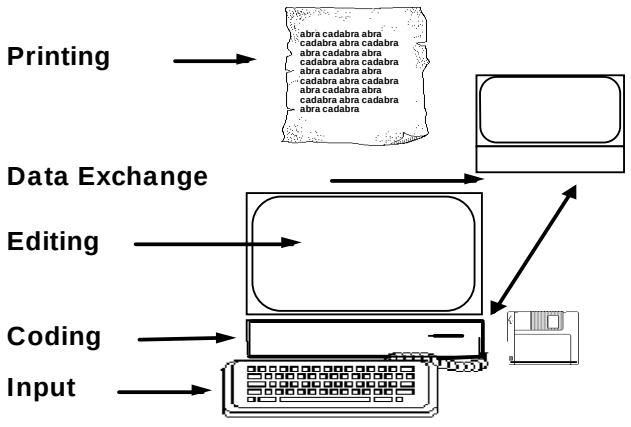

Figure 1.

2.1. Multilingual text input (Keyboard)

2.1.1. Keyboards

Though many keyboards represent only ASCII graphic characters (or ISO 646), certain localized (adapted) keyboards may also include keys for special or accented characters. For example, French keyboards generally feature keys corresponding to **"à ç é è ù"** accented characters, while characters which contain circumflexes or dieresis **"ê ï"** are input by two successive key presses. In addition, there is generally no single key on a French keyboard that allows one to produce characters that exist in other European languages, such as **"ñ"** or **"ò"**.

In a broad multilingual context one could scarcely begin to imagine a keyboard that contains all possible characters. The inclusion of such languages as Chinese (with more than 6000 ideograms) or Arabic (approximately 4 sets of 28 letters and 10 vowels) requires the definition of specific keyboard input programs. Solutions proposed by computer manufacturers are very heterogenous. Theoretically there exists a standard input method for keyboards with 48 keys (ISO/IEC 9995-3), at least for the Roman alphabet, but it is rarely used. A number of keyboard input methods for the ISO 10646 characters was recently proposed using hexadecimal codes or composition. But these keyboard input methods always require the user to know and memorize a huge number of codes and it is necessary to develop more intuitive keyboard methods and, if possible, reduce the number of key presses by the user.

2.1.2. Interpreting characters and rendering glyphs (context analysis)

Character coding standards (including Unicode) do not define glyph images, they define how characters are interpreted, not how glyphs are rendered. In this case, a special program called "Context Analyzer" is necessary for rendering glyphs on the screen. For example an abstract character such "ARABIC LETTER AIN" which has the U+0639 Unicode value can have different visual representations (called shapes or glyphs) on screen or paper, depending on context (see figure 2). Different scripts which are part of Unicode can have different writing rules for rendering glyphs and also composite characters, ligatures, and other script-specific features. The results on screen or paper can differ considerably from the prototypical shape of a letter or character. For example:

- Greek: **σ** (beginning and middle of a word), **ς** (word ending)
- German: **s + s => ß**
- Arabic:

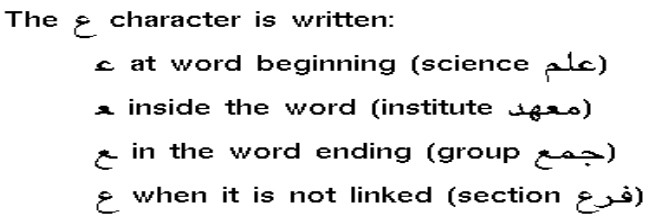

Figure 2. Arabic character associated glyphs

2.1.3. Ideogram phonetic input

Certain languages, such as Chinese, are based on a vast number of ideograms, each one representing a particular concept. Recently (1980), simplified and varied versions of Chinese have been adopted by the People's Republic of China, on one side, and by Taiwan and Hong

Kong, on the other. Several different keyboard input methods exist, such as input via character codes (e.g. **GB-2312-80** code) or the **Pinyin** method (see figure 3). , which consists of a phonetic representation of ideograms in Roman characters (420 syllables, complemented by one of five tones per syllable).

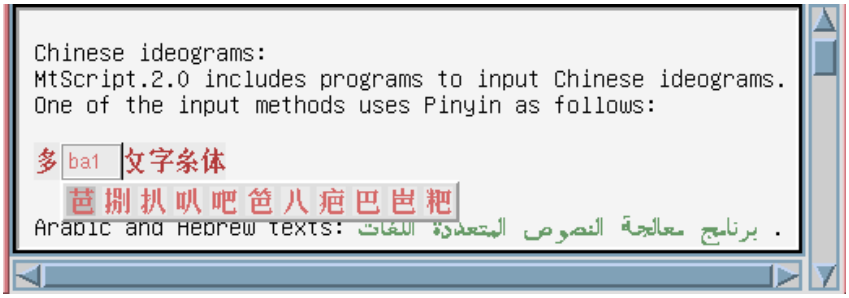

Figure 3. Chinese ideograms keyboard input using Pinyin

2.2. Multilingual text coding

2.2.1. Character coding

Computer manufacturers and software developers use numerous specific and non-compatible character codes *(MS-Windows character set for Western Europe MS CP1252, DEC Multinational character set, International IBM PC character set IBM CP850, Macintosh Extended Roman character set, Hewlett-Packard ROMAN8, etc.)*. Meanwhile other character coding norms have been standardized on an international level and are already used in some environments (ISO-8859-1/2/3/4 for Roman Alphabet, ISO-8859-5 for Cyrillic, ISO-8859-6 for Arabic, ISO-8859-7 for Greek, ISO-8859-8 for Hebrew, GB-2312-80 /BIG 5-0 for Chinese, JISX-0208-1983-0 for Japanese, KSC-5601-1987-0 for Korean, …).

More recently (1993) the ISO 10646 *(Universal multiple-octet coded character set or UCS)* proposed a universal character set including all the character sets of ISO-8859 as well as those for Chinese, Korean, Japanese, the International Phonetic Alphabet, etc. In its present form (ISO 10646-1), the UCS uses a 16-bit code (Unicode) which will be extended to a 32-bit one in future editions, thus permitting an effectively unlimited coding of characters. However, existing environments are not yet ready to implement character sets on multiple-octet code, even though the situation is rapidly improving (e.g. Windows-NT, AT&T Bell Plan 9 and Apple QuickDraw GX). Moreover SGML entities have been defined for encoding the characters of many different languages. SGML is being a standard for the multilingual document interchange.

2.2.2. Writing systems coding

In a multilingual text it is necessary to code not only individual characters but also scripts (Latin, Semitic, ...) and languages. In the case of a one-octet-based coding (e.g. ISO 8859-* character sets), it is necessary to mark the change from one set to another (e.g changing from Greek to Cyrillic). This can be done using a code such as that proposed in the ISO 2022, which includes escape sequences (<SI> (shift in) and <SO> (shift out)) that encode a transition between the "main" and the "complementary" sets. However these techniques are limited and many difficulties can arise, especially when a single document includes one-byte (e.g., ISO 8859-*) and two-byte (e.g., GB-2312-80 or BIG-5-0 for Chinese, JISX0208-1983-0 for Japanese or KSC5601-1987-0 for Korean) characters. The UCS inventory solves one part of the problem by combining all these character sets into a single set, since it is no longer necessary to implement a means for switching between character sets. However the problem

is not totally resolved because UCS does not explicitly encode some features of the character sets such as the writing direction (although bi-directional protocols have been proposed by the Unicode Consortium). Moreover language tagging is needed not only to indicate writing direction, but also to control hyphenation, ligation, font selection and character/glyph mapping.

2.2.3. Language coding

Linguistic processing of a multilingual text (segmentation, morphological and lexical analysis, etc.) requires the identification of the languages therein. Recognizing the character set or the writing system does not suffice to identify the language in which a portion of text is written: a document encoded in ISO 8059-1 could equally well be written in French, English, Spanish or even a combination of these languages.

Norms for coding the names of languages exist:

- ISO 639-1988: 2 alphabetic letters code for about 140 languages (e.g "en" for English, "fr" for French, etc.).
- ISO 639-2: 3 alphabetic letters code, alpha-3, is currently in development (e.g "eng" for English, "fra" for French, etc.).

However, in the internal code of a document, these codes cannot be used such as they are. At this time there is no established standard method for escape sequences which would permit the representation of the change from one language to another, although it has been proposed that one use the ISO/IEC 6429 set of control sequence codes with a numeric conversion of the above alphabetic codes [LANG 93]. Language markup is also currently being defined in the SGML/HTML standard used by the World Wide Web [YERG 95].

2.3. Multilingual text editing

2.3.1. Bi-directional texts

The majority of languages are written horizontally from left to right. Some languages, such as Arabic or Hebrew, are written from right to left. Other languages, such as Chinese or Japanese, can even be written from top to bottom (especially in ancient texts). As a consequence, the co-existence of languages in the same document, and particularly on the same line of the text, poses huge problems when inserting or deleting text zones. The example in figure 4 shows that it is often necessary to rearrange words to maintain the semantic coherence of a sentence.

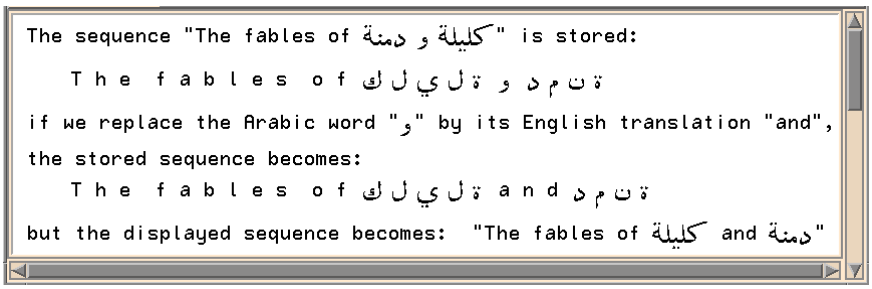

Figure 4. Editing aspects in a multilingual text

2.3.2. Insertion / deletion of characters in bi-directional texts

A significant problem with displaying multilingual texts is the co-existence of opposite writing directions on the same line of text. Insertion and deletion of characters must take into account their writing direction, according to quite complex rules. The text editor may allow the user to define interactively a main and a secondary writing direction for a text-zone

(paragraph). The cursor moves only in the direction specified as the main one. When a sequence of characters is entered in a language written in the secondary direction, the cursor stays put and the characters are written in an insertion mode (see figure 5).

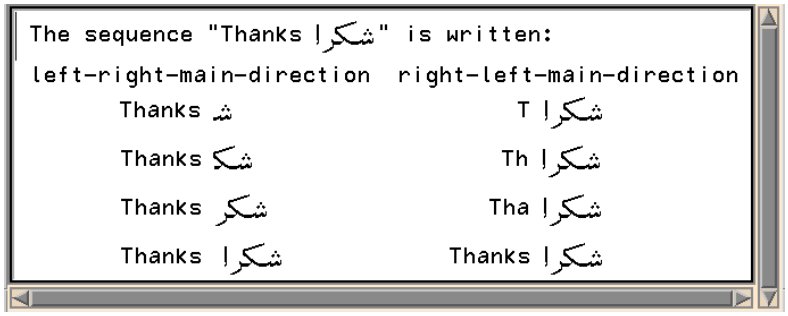

Figure 5. Bi-directional texts

2.3.3. Multilingual text justification

The justification of right-to-left written texts or bi-directional multilingual texts is a major problem. Contrary to Latin alphabet based texts which can be justified by using extra spaces, the justification of cursive right-to-left written texts requires the use of **dynamic fonts**, where most of the characters can be contracted or stretched dynamically, according to the line length. In the example of figure 6, the last character "Ba" of the word "KaTaBa" (to write) is drawn differently according to the line length. A special character such as "_" can also be used to extend the character drawings.

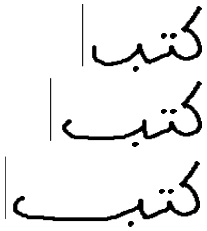

Figure 6. Dynamic font for Arabic text justification

2.4. Multilingual text printing

Printing multilingual texts suffers most obviously from the lack of printer fonts (essentially PostScript fonts). Many PostScript fonts are now available (freely or not) for Roman characters, but only a few fonts have been developed for the other character sets. Significant efforts in designing multilingual Unicode-based PostScript fonts for editing and printing multilingual electronic documents include particularly the OMEGA project activities.

2.5. Multilingual data exchange

With the rapid growth in the use of the Internet, the electronic transfer of multilingual documents is becoming more and more necessary. Until recently, only one part of the standard invariant characters of the ISO 646-IRV (ASCII) could allow a non-corrupted electronic text exchange, and multilingual documents could be transmitted safely only with the assistance of coding utilities such as *UUENCODE* and *BINHEX*. However the situation is improving: standards have been adopted on the Internet which allow the transfer of 8-bit characters without corruption in the TCP/IP protocol (for example, applications such *TELNET*

and *FTP* are "8-bit clean"). In addition, the MIME norm *(Multi-purpose Internet Mail Extension: RFC-1521 and RFC-1522)* allows uninterrupted data transfer in any circumstance by compressing and decompressing the files. Moreover the emerging general standard for text data interchange is SGML (and for certain areas TEI), although these standards are not yet universal and some transfer problems persist. In saying this, one must point out that the current guarantee for data transfer without corruption does not extend to the transfer of multilingual data. It is necessary that both parties involved in the transfer, the sender and the receiver, have the same systems of encoding characters, documents, languages and writing systems.

…/