Creating a Single South African Keyboard Layout to Promote Language[*]

Dwayne Bailey, *Translate.org.za (dwayne@translate.org.za)*

Abstract: In this case study, a description is given of a keyboard layout designed to address the input needs of South African languages, specifically Venda, a language which would otherwise be impossible to type on a computer. In creating this keyboard, the designer, Translate.org.za, uses a practical intervention that transforms technology from a means harming a language into one ensuring the creation and preservation of good language resources for minority languages. The study first looks at the implications and consequences of this missing keyboard, and then follows the process from conception, strategy, research and design to the final user response. Not only are problems such as researching the orthographies, key placement and keyboard input options examined, but strategic objectives such as ensuring its wide adoption and creating a multilingual keyboard for all South African languages are also discussed. The result is a keyboard that furthers multilingualism and ensures the capturing of good data for future research. Finally it is a tool helping to boost and bolster the vitality of a language.

Keywords: KEYBOARD, MULTILINGUALISM, VENDA, AFRIKAANS, TSWANA, NORTHERN SOTHO, ZULU, SOURCE, FREE SOFTWARE, LAYOUT

Opsomming: Die skep van 'n enkelvoudige Suid-Afrikaanse toetsborduitleg om taal te bevorder. In hierdie gevallestudie word 'n beskrywing gegee van die ontwerp van 'n sleutelborduitleg vir die hantering van die insetbehoeftes van Suid-Afrikaanse tale, veral Venda, 'n taal wat andersins onmoontlik op 'n rekenaar getik sou kon word. Deur die skep van hierdie sleutelbord gebruik die ontwerper, Translate.org.za, 'n praktiese ingryp wat tegnologie verander van 'n middel wat 'n taal benadeel tot een wat die skep en bewaring van nuttige taalhulpbronne vir minderheidstale verseker. Die studie kyk eers na die implikasies en gevolge van hierdie ontbrekende sleutelbord, en volg dan die proses van konsepsie, strategie, navorsing en ontwerp tot die uiteindelike gebruikersreaksie. Nie alleen word probleme soos die navors van die ortografieë, sleutelplasing en sleutelbordinsetkeuses ondersoek nie, maar ook strategiese doelwitte soos die versekering van 'n wye aanvaarding daarvan en die skep van 'n meertalige sleutelbord vir alle Suid-Afrikaanse tale word bespreek. Die resultaat is 'n sleutelbord wat meertaligheid bevorder en die vaslegging van nuttige gegewens vir toekomstige navorsing verseker. Uiteindelik is dit 'n werktuig wat help om die lewenskragtigheid van 'n taal te versterk en te steun.

Sleutelwoorde: TOETSBORD, VEELTALIGHEID, VENDA, AFRIKAANS, TSWANA, NOORD-SOTHO, ZOELOE, BRON, GRATIS PROGRAMMATUUR, UITLEG

* This article was presented as a paper at the Eleventh International Conference of the African Association for Lexicography, organised by the Tshivenda National Lexicography Unit, University of Venda for Science and Technology, Thohoyandou, Republic of South Africa, 5–7 July 2006.

1. Introduction

Translate.org.za is involved in the localisation of free and open source software (FOSS)[1] into the eleven official languages of South Africa. The organisation delivered the first office productivity suite (word processor, spreadsheet, presentation program) in these languages in 2005, followed by the Mozilla Firefox web browser and the Mozilla Thunderbird email program in 2006.

During this process Translate.org.za soon discovered that Venda translators could not physically type the five extra characters[2] needed in Venda orthography on their computers. Most translators chose simply to ignore these characters or to follow convoluted processes to insert them. Even more surprising was that language bodies were seriously considering revision of the language's orthography to adapt to the limitations of technology.

Of great concern to Translate.org.za were some of the methods used to circumvent this problem. These included printing and manually adding diacritics, adding characters that looked like the correct ones, and using methods that printed correctly and looked correct but were not actually the correct characters. For Translate.org.za the use of the correct characters are essential. As a character cannot be pencilled onto a piece of software, another character or even one that looks correct cannot be used. It was soon realised that Venda translators around the country were creating parallel texts that were of no use for research, that these documents looking good on paper could never be published in electronic format, and that in fact their actions would have a long-term detrimental effect on Venda as, for instance, Translate.org.za is unable to produce spell checkers using Government texts with the diacritics missing.

Translate.org.za's own needs and the belief that the problem could be addressed lead to the research on and development of the South African keyboard layout.

Most users take a keyboard for granted; it simply works and is very reliable. If it breaks, you buy a relatively inexpensive replacement. What most people think of as the keyboard is the physical keyboard derived from the computer's typewriter heritage. A keyboard has a layout, a definition of the positioning of the keys. In South Africa, the dominant layout is the US English layout. But you will most probably have encountered a British English layout and would also have seen layouts for French, German and other languages.

Although you will have a computer keyboard with a defined physical layout, you also have a software layer that defines this layout for the computer. This usually maps one-to-one to the physical keyboard. However, as it is software driven, this can be changed to redefine the behaviour of the physical keyboard. Thus it can be defined that when you press S you get š. In this process, a new layout is being created that could in time be made into a physical keyboard. What Translate.org.za has developed is a South African keyboard layout, creating a new software mapping to the physical keyboard.

Although, from a software side, the creation of the keyboard layout was relatively simple, the strategic decisions and research into the design took much longer to formulate and test. The team devoted attention to the following aspects:

(a) Language investigation — What additional characters did each language need?

(b) Decision on one or multiple language layouts — Should each language have a keyboard layout or could/should one be created to cover all South African languages?

(c) Determination of the layout — How should the keys be positioned for ease of use and adoption, considering the different types of users (touch typists and two-finger typists)?

(d) Consideration of the keyboard — Should a physical keyboard be created?

The team settled on combining all languages into one keyboard layout with three different layouts catering for different types of users. Instead of a physical keyboard, an overlay was developed that allows any existing keyboard to be converted into a South African keyboard.

From the design phase, it was soon realised that Venda still has the problem of displaying the characters if the computer does not have a Venda-capable font installed. To overcome this, Translate.org.za extended the existing free DejaVu font to include the missing Venda characters and combined this with the keyboard software.

The final step has been to validate and test the layouts. Fortunately a number of translators have tested and evaluated the keyboards and given constructive feedback. Three layouts are not ideal as only one design would have been preferred. But for now, Translate.org.za is testing and refining with the hope that only one will remain. In the long term, this could evolve into a standard South African keyboard layout.

Translate.org.za, over and above its software localisation effort, continues to undertake work which touches on the broader issues of computers and South African languages. These include amongst others: designing spell checkers, creating and validating collation sequences and creating locale information. Translate.org.za has received funding from various organisations, including the Department of Communications, the International Development Resource Centre (IDRC), The Open Society Institute (OSI) and the Shuttleworth Foundation.

2. What lead to the creation of this keyboard?

2.1 The history of Translate.org.za

Translate.org.za is a South African non-profit organisation focused on the localisation, or translation, of free and open source software (FOSS) into South

African languages and on the creation of FOSS localisation tools such as translation memory tools, translation editors and translation management software.

Under its first mandate, Translate.org.za released the first word processor in a South African language in 2004 when the Zulu, Northern Sotho and Afrikaans versions of OpenOffice.org were introduced. Subsequently a version in all eleven official languages has been created and was released in 2005. In addition to this, the Mozilla Firefox web browser and the Thunderbird email program have been completed in all languages. Translate.org.za has added spell checkers and now a keyboard to make this software fully multilingual.

Towards its second mandate, Translate.org.za is a partner in the WordForge project which is developing FOSS localisation tools.

Translate.org.za believes that concrete action is the only way to foster language pride, challenge negative language perceptions and build a multilingual society. The keyboard is a good example of such action.

2.2 Language pride

When championing language pride what could be more debilitating than technology that cannot correctly work in a language? The perception is then that the language is in some way inferior or backward.

2.3 Observing usage patterns

Since Translate.org.za translates software into all eleven official languages, it is thus also translating into Venda. During training sessions with translators, patterns and problems were noticed that had not been seen in other languages. Venda translators were struggling to enter text in their language. The problem had been ignored for so long that most translators were surprised when questions were asked about their ability to type characters for Venda diacritics.

None of the solutions used by translators addressed the root of the problem and none of them were transportable between applications. One solution involved: type, print, then pencil in diacritics. It was clear Translate.org.za could not accept pencilled-in characters. Many other issues were found with the workarounds. For instance, solutions would work in Microsoft Word, but not in Microsoft Excel. Or the solutions would work well, yet not in Windows 98.

3. The present situation

3.1 Workarounds used by translators

The following patterns were found which place translators into two groups, namely those who let technology master them and those who create some workaround even if it is convoluted and inferior.

3.1.1 Those who are mastered by technology

The way they cope with these computer problems also reflects on the translators' respect for their own language. They will do one of the following:

(a) Ignore the diacritics — simply type using the 26 letter alphabet.

(b) Add the diacritics later — type, print or pencil in the missing characters.

(c) Choose characters that look suitable but are not the correct ones — N dot below (ṇ), instead of N dot above (ṅ).

The consequence of any of these actions is data unfit for further use or research. Here is a short list of possibilities lost with these actions:

(a) You cannot reuse any of the translations in a translation memory.

(b) Researchers cannot use any of this data for any language analysis.

(c) You cannot repurpose a document, e.g. create a printed copy and a copy to publish on the Internet.

(d) You cannot collaborate, i.e. you cannot email a document for review.

Thus every day translations are being created that could be enhancing the status of languages. Yet every day work is being produced that is unfit for advancing the languages in any way.

3.1.2 Those who have mastered technology

Fortunately the second group is larger than the first, people who do care about correctly representing their language even if they have invented very convoluted ways of doing so. Unfortunately they are also very averse to change. This can be a problem with some of the solutions they have designed as these do not create valid output. The following are the methods they use:

(a) Menu driven: Select, Insert → Special Character, select the Venda diacritic from a dialogue box.

(b) Special key combinations: Associate special characters to a special key combination in their word processor. These options have been removed in the newer version of Microsoft Office.

(c) Autocorrect: Create special autocorrect sequences, e.g. typing ˆn will create ṋ.

(d) Macros: Create special Office macros to add the characters. (It is uncertain whether these were valid Unicode characters or whether they merely looked correct.)

3.2 Revisiting orthographies

This, also known as the workarounds developed by linguists, has been referred to as mechanical imperialism,[3] changing the language to meet the shortcomings of technology. The extreme of this would of course be to change the complete writing style.

Orthographies do need to be revisited and updated from time to time. But this should be based on adaptations of the language, not on the shortcomings of the tools used to transcribe the language.

4. The role of a keyboard in the promotion of multilingualism

Translate.org.za is very focused on practical interventions. It is felt that not enough of these interventions are effected in the effort to promote multilingualism.

The keyboard interventions were:

(a) Simple — These are relatively easy to implement.

(b) Practical — These are easy to demonstrate.

(c) Helpful — These simplify the operation for users.

(d) Unifying — These bring languages together.

5. The keyboard in the computer process

Although this can seem quite technical, it is often important that language users obtain insight into how a computer understands their input. This helps them to identify certain types of problems and comprehend their causes, and their solutions.

5.1 Character encoding (Unicode)

Each alphabetic character is represented by a number, also known as a code point. A number of code points creates an encoding. A group of encoded characters is known as a character set or a character set encoding. In the early days of computing, there was only limited space for character storage. Thus there were different encodings for different language groups: Western European, Eastern European, Japanese, etc. The problem with this approach is that if the wrong encoding is specified, the text will be garbled, i.e. an A in the Latin alphabet will appear as another character in Cyrillic, because they have the same encoded number.

However, this was solved with the creation of Unicode, which has enough space to encode every character of every living language in the world. In fact, Venda can only be stored using the Unicode character set. An added advantage of Unicode is that since every character has a unique number, it is possible to mix languages in a single document. So it is now possible to write an English paper describing an Arabic translation of a Chinese text.

Many older computers cannot handle Unicode correctly[4]. This would explain why you cannot type Venda on a Windows 98 machine, as it is not fully Unicode compliant. Certain applications, by implementing their own system to manage Unicode, are able to give the illusion of compliance but only in the application, not across the system. This, however, would still not address the ability to type on a keyboard.

5.2 Fonts

With the ability to store the characters, the ability to see them is now also needed. For this there are fonts and each font contains a number of glyphs, a glyph containing the drawing instructions for one character. The glyph is associated with a code point. Thus if Venda text is stored on your computer, the computer will know the code point of that character. By also knowing the encoding, the computer determines that that code point represents N with a dot above (ṅ) in the Unicode encoding. Since the encoding of the font is also known, the correct glyph can be pulled from the font and thus the correct character can be drawn and displayed. With the absence of the correct glyph, you will see a small square box (□) or a question mark (?). This is an indication that you do not have the correct font.

5.3 Keyboard layout

In more complex languages, such as Thai and Khmer, there are issues related to inputting of the language into the computer that are much more advanced than the simple keyboard used by Latin-based languages, thus South African languages are unaffected.

Once a character can be stored and displayed, the keyboard allows the same character to be entered. The keyboard layout ensures that when a key or a set of keys are typed, the computer can determine what code point the user requested. With the code point, the correct character can be displayed and stored. As a user, you simply see that the key you press produces the correct character in your word processor.

5.3.1 Dead keys

The name "dead key" is a hold-over from the days of typewriters. If you typed a dead key on a typewriter, the carriage would not move. Thus the second character would be typed in the same space. So, for instance, typing ˆ followed by E would create ê. Modern computer keyboards can be made to emulate this behaviour.

5.3.2 AltGr

In many languages where there are not enough keys, the right Alt key, also called AltGr, allows a keyboard to have many more characters, e.g. AltGr + E could create ê.

6. Strategic options

There were a number of options that could have been pursued in the design of the keyboard and certain constraints that caused a move in a certain direction.

6.1 Goal

This is a brief summary of the pursued goals. These evolved over time as it was realised and learnt what was possible and what was not workable.

(a) Unobtrusive — A keyboard must be created that could exist without anyone knowing it was there. If it does not get in your way, you are unlikely to want to remove it, yet it remains in place in case you or someone else requires it.

(b) Simple — The key sequences must be easy to remember with logical associations.

(c) One-off — One keyboard must be designed for all languages, so that any language can be typed on the same keyboard. If users need to exchange keyboard layouts, they would most likely not adopt a new one, but revert to their old US English layout.

(d) Cheap — The keyboard must be inexpensive so that nobody has an excuse not to use it.

6.2 Design

6.2.1 Keystroke analysis

At first, it was felt that keystroke analysis would be a good approach as it would create an optimal keyboard. By monitoring which keys are used most frequently, the high frequency characters could be placed on keys which are easy to reach. This would create the best keyboard for speed of typing in a given language. The current QWERTY keyboard layout was designed to slow people down when using the first mechanical typewriters; it was not designed for speed.

In consultation with linguists working on Nigerian languages, it was soon realised that people are hard to change. Once you can use a QWERTY keyboard, you are unlikely to use another. Such a demanding change would fail on a number of the pursued goals.

6.2.2 Keyboard format

How to present the keyboard to users?

6.2.2.1 A physical keyboard

The option of a physical keyboard would allow the localisation of keys such as Enter, Shift, Caps Lock, etc. However, it is costly with a relatively small market. The question though is this: Do you read or need to read your keyboard? Once users know the key associations, they no longer need the hints from a physical keyboard.

This option fails dramatically as an inexpensive goal. It would also make it difficult to adopt. Would someone supply you with a Venda keyboard when it would add a considerable sum to the price of your computer? In the first stages, it was decided that this was not the best strategy, although it is still being investigated as an option.

6.2.2.2 Labels

Another option is simple stick-on labels that can work with an existing English US keyboard. The labels are not essential but allow relatively easy conversion of any existing keyboard into a South African keyboard. Production costs, however, are still high.

6.2.2.3 Configuring the keyboard

The keyboard can be presented as a software-only solution. For now, this is what Translate.org.za is doing, supplying the software layout that allows users to configure and operate the keyboard without any physical crutches such as a keyboard or keyboard labels.

6.2.3 Layout

The decisions on layout revolved around what keyboard style to use:

(a) Remap unused keys.

(b) Use AltGr.

(c) Use dead keys.

With remapping of keys, a key or character that is unused in a language is used for a different character. For example, if the character C is unused in the language, it can be remapped to another character such as š. As most South African computer users need to type English, this was, however, not an option.

With each of the last two possibilities, AltGr and dead keys, there is the decision around which keys to associate with each diacritic character. In the use of dead keys, the decision concerns which key should be used to represent the caron or the dot above. In the use of AltGr, it is to be decided which letter to associate with the character that contains the diacritic. For instance: N needs to be used twice in Venda for N with a dot above (ṅ) and N with a circumflex below (ṋ). It must be decided which character should occupy the N, i.e. AltGr + N, and where the other character should be placed.

In all these decisions, the layout needs to remain logical, easy to use and easy to remember.

7. Implementing the South African keyboard

7.1 What characters are needed

This would seem to be one of the simplest tasks. However, it proved to be quite difficult, because it was necessary to ensure that everything had been covered. The problems encountered were changes in orthography, and remnants of old orthography.

It was discovered that, although orthographies had changed, the old orthography remained in, for instance, people's names. Thus it had not completely disappeared, and there would be instances in which the old orthography was needed such as population registration and legal contracts. Over time, this will become less of a problem.

In researching the characters, a number of people were consulted, but the vagueness of most answers made it difficult finding a reliable resource. There seemed to be confusion about what was old and what was new, and what was required for the current orthographies. Dictionaries, grammars and various study materials were consulted. Some presented problems as they included many tone marks not used in the written form of the language. Thus extracting a simple list of characters was troublesome.

In addition to this, old orthographies in texts such as the Tswana Bible were still prevalent. In each instance of a new authoritative source that presented new characters, it had to be investigated whether this was current or old orthography.

7.2 Understanding the targets and tools

The primary operating systems targeted are Linux and Windows. The first prototype was produced on Linux, but subsequently most of the development was done on Windows as the tools are more easily used for prototyping keyboard layouts. The layouts will be ported back to Linux once they are stable.

Windows proved to be problematic in its own way. Since Windows did not start supporting Unicode properly until Windows 2000, it took much research to discover that proper keyboard input for Venda on the Windows 9x platform will never be attained.

On Windows, the Microsoft Keyboard Layout Creator (MSKLC) is employed. This is a useful tool as it is quite quick to prototype layouts. It was frustrating that the tools do not actually compile a usable install package which Translate.org.za could handle itself.

On Linux, the process of developing the layout is quite cumbersome with little documentation of a clearly powerful system. Work is under way to improve the keyboard handling on Linux's windowing system. Keyboard layouts will have to be proved before undertaking the porting task.

No work has been done on creating a layout for the Apple Mac range of computers. The process is relatively simple but the need is not as great as the Windows keyboard layout.

7.3 The final outcome

As it was found that there are different sets of users, it was actually not possible to make one design, which would have been preferred. Thus these keyboards with the following user profiles were created.

AltGr	Infrequent user, computer used by multiple people, public terminal, user is not a touch typist
Intl	User types at some speed and finds the AltGr combinations awkward
Super	Touch typist

It is felt, however, that the AltGr layout will prevail as it is simple to understand and use, even though it is not ideal for speed.

7.3.1 One keyboard

It was hoped that a single keyboard could be created. Afrikaans with its many diacritics was a concern as these quickly fill up the layout. But since there is very little overlap between Venda, Northern Sotho and Afrikaans, it was found that all characters could be catered for on one keyboard layout.

7.3.2 Key associations and how these were arrived at

Logical associations are needed between keys and their diacritics. On the AltGr keyboard, the circumflex and the caron are the dominant symbols. Although the diacritics are not the same characters, they do look the same. Venda has a circumflex below, Afrikaans a circumflex above and Northern Sotho and Tswana S caron. Thus this was chosen as the dominant character. So if you type AltGr + N, you will get N with a circumflex below (ṋ). AltGr + S results in S caron (š).

In Afrikaans, the diaeresis was chosen as the dominant character as it appears on all vowels, while the circumflex is needed on only some of the vowels.

Other diacritics occurring on the same key were placed as close to the key as possible and, if possible, on a key of related shape. Thus AltGr + M will give N with a dot above (ṅ). M was chosen because it is shaped like N.

In Afrikaans, the circumflexes proved a problem as the keyboard around U, I and O became rather crowded. But since the AltGr keyboard is designed to be used with labels, it was concluded that this pattern can be learnt.

The Intl (International) keyboard was a compromise in that many expert Afrikaans computer users make use of it. It is also an easy layout to learn as there are logical associations for diacritic characters. All diacritics are made with dead keys, so this is not a non-intrusive keyboard. A person unfamiliar with the layout will be frustrated. The US Intl keyboard layout was merely taken and South African characters added. Thus ˆ + S will give S caron (š), ˆ + N will give N circumflex below (ṋ) and ˆ + E will give E circumflex (ê).

The last keyboard was designed to meet the needs of touch typists. When broken down, there is very little need to have multiple dead keys. In this layout, a relatively comfortable position was chosen for the dead keys. Comfort rather than associations was considered. When you look at the characters that are needed, no language makes use of more than two diacritic marks (ignoring of course acute and breve in Afrikaans). Since it was already found in the AltGr that there is a common visual association around caron and circumflex, it was decided to treat these similarly. Thus in the super keyboard, if you type < + S, you get S caron (š), < + E = E circumflex above (ê), while > + N = N dot above (ṅ) and > + E = E diaeresis (ë).

The keys ; and ' handle acute and breve as dead keys because they are easily located. It is still uncertain whether these two are the best choice as dead keys are wanted that are not too invasive and these effect the single quotation mark.

7.3.3 The need for fonts

Venda needs fonts and not everyone has them (unless of course you think typing documents in Tahoma is readable). Most people working with Microsoft Office will have support in Arial Unicode if they have installed them. There are a number of fonts that will cover the Venda characters; most notable are those from SIL (that has just begun to release its fonts under terms that would make them freely distributable).

In order to make sure that everyone has access to at least one set of fonts that are freely distributable, Venda characters were added to the DejaVu font. This gives a serif, sans serif and monospace font. Strangely enough, users are satisfied with such a limited choice. Choice is, however, very much a part of language pride.

8. Outcomes

Extensive testing has now been performed with a number of translators. The following, sometimes unexpected, results were obtained.

8.1 Simple can be hard

The AltGr layout meets all of the intended goals, but according to the feedback from translators it was at times awkward to use. AltGr + Shift + N can be so awkward that some users in fact just reverted to the MS Word shortcut associations or to the Insert menu.

8.2 The difficulty of change

For many people changing from the familiar to something new was impossible to even consider. Thus they abandoned the keyboard quite easily.

8.3 No ownership

This is a phenomenon that open source developers have always found rather baffling. If there is a problem, an open source user tries to figure it out and get it fixed. Translators, who are probably a good reflection of average Windows users, would simply consider a problem as a fault of the software and thus abandon it as inferior, ignore the problem or work around it.

9. Further developments

Looking forward, there is still work to be done on the keyboard. The layouts should remain relatively unchanged, but there are a number of additional items that are considered:

(a) Porting to Linux and Mac — AltGr works on Linux but the other layouts are still to be migrated so that they can be used on these platforms.

(b) Collaboration with SABS and CSIR on standardisation — Discussions have been started that may lead to some form of standardisation work on the layouts.

(c) Creation of a real keyboard — A physical keyboard needs to be created for those who believe firmly in multilingualism.

(d) Investigation of a Venda-only layout — Due to the awkwardness of the layout, further investigation should be done on the concept of a Venda-only layout that could work without dead keys.

(e) Completion of orthographies — For researchers in the language field, the International layout should be expanded to cover all characters that have been used in all orthographies, past and present, of all South African languages.

(f) Installation of more fonts — Venda-speaking people should be given the ability to use any font with their characters.

10. Conclusion

Translate.org.za has succeeded in its goals of creating a keyboard layout for South Africa. It is one practical step in raising the issue of language pride in South Africa and in restoring dignity to languages ignored by computers.

Further testing, checking, adoption and adaptation need to be performed and, most importantly, stability achieved before this has a chance of becoming a South African standard. But in the long run, it is hoped that people will be using keyboards that work in their mother tongue.

Acknowledgements

Translate.org.za would like to thank the Afrikaans users who tested the various incarnations of the keyboard and who criticised the AltGr keyboard for its awkwardness. Although more feedback is needed, the Venda translators who have been working with this keyboard must be congratulated for starting what will hopefully become a concerted fight against mechanical imperialism.

Notes

1. FOSS is an area software development that focuses on the rights of users of software. It has the right to distribute, modify, use and share modifications. It is a rapidly growing area of software development that is well known to most through the Linux operating system. OpenOffice.org, Firefox and Thunderbird are all part of FOSS.
2. These characters are: ḓ, ḽ, ṋ, ṱ and ṅ.
3. Many thanks to Ronald Madiba from whom I first heard this phrase.
4. There is a Unicode layer for Windows 9x. However, this is only a translation layer on top of the existing codepage system of the 9x series. Thus, if your characters do not appear in any codepage, they will not work in the Unicode layer. Since Venda's characters do not appear in any codepage, they cannot be managed by a Windows 9x computer.