Using Arabic in Blaise

Shani Abel / Shifra Har, Central Bureau of Statistics, Israel

1. Introduction

Surveys conducted by Israel's Central Bureau of Statistics must accommodate a multilingual population. The main language of our surveys is Hebrew, however, Arabic is Israel's second official language and it was therefore crucial that we find a solution for displaying Arabic, particularly once we began using CAPI. With both a large Arabic-speaking population and a large Russian-speaking population a solution was sought for incorporating both languages into the surveys. In the past, this need was answered by providing paper-based questionnaires when necessary. Thus, while most interviews were conducted in Hebrew by CATI, at times surveyors were sent to a respondent's home with a questionnaire in either Arabic, Russian or English.

The problem of displaying all three languages in a single survey was complicated by the diverse nature of these languages. Both Hebrew and Arabic are written RTL, while Russian is written LTR and each language requires a different number of characters for its alphabet, with Arabic requiring at least hundreds of characters, if not thousands.

The problem of Russian was solved by employing an interpreter that converts Cyrillic characters to encoded text understood by Blaise, which then displays Russian in Bulgarian Courier font. We have been using Russian in our CAPI surveys since 2002.

Arabic meanwhile was presented in Hebrew transcription while we continued to work on a solution for presenting Arabic fonts in the DEP. The breakthrough occurred when we realized that we could combine characteristics unique to the Hebrew-operating system (such as RTL functionality) together with characteristics unique to Blaise, to achieve a complete solution. We are pleased to report that our efforts have been successful and as of 2004 our surveys will include all three languages in the original.

In this paper, we will outline some of the problems we encountered with Arabic fonts and we will present both our solution for displaying Arabic in the Blaise DEP and its mode of deployment.

2. Problems with Arabic Fonts

Several factors contributed to the complication of displaying Arabic in the DEP.

2.1 The Arabic Alphabet

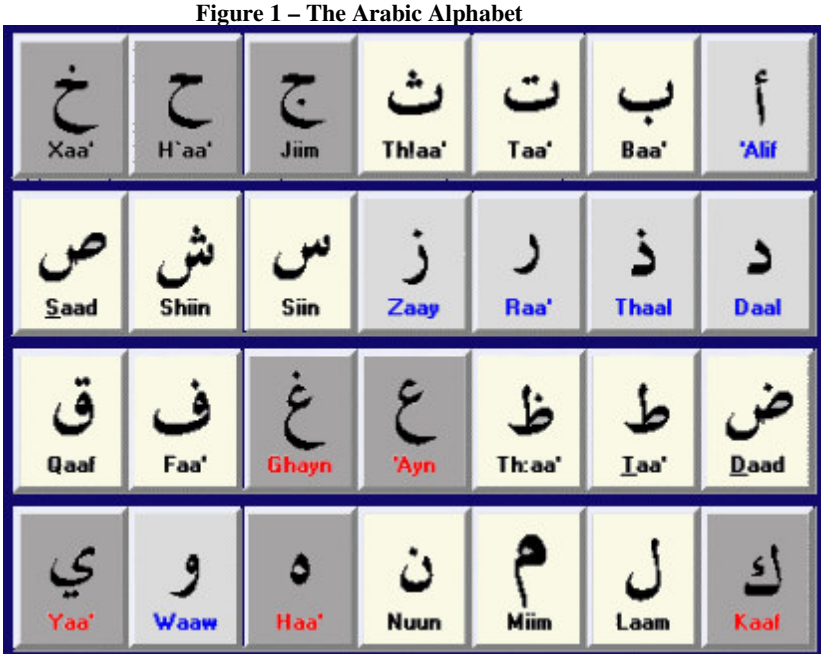
Arabic script consists of 14 basic shapes, representing 28 consonants. These are distinguished by the use of diacritics: symbols above or below the letters that signify the correct letter and the vowels.
Here are some of the main differences between Arabic and Latin characters:

1. The characters are laid out RTL, with the exception of numbers, which are laid out LTR. This change in the writing direction is usually described by the expression bidirectionality of Arabic text. Bidirectional text also emerges when Latin and Arabic text is mixed.
2. Letters change shape depending on context. Each letter has *up to* four forms: the initial form, where it is the first letter in a word, the final form, where it is the last letter in a word, the

isolated form, and the medial form. This means that as a word is typed, the shape of previously-entered letters changes as well as a new letter being added.

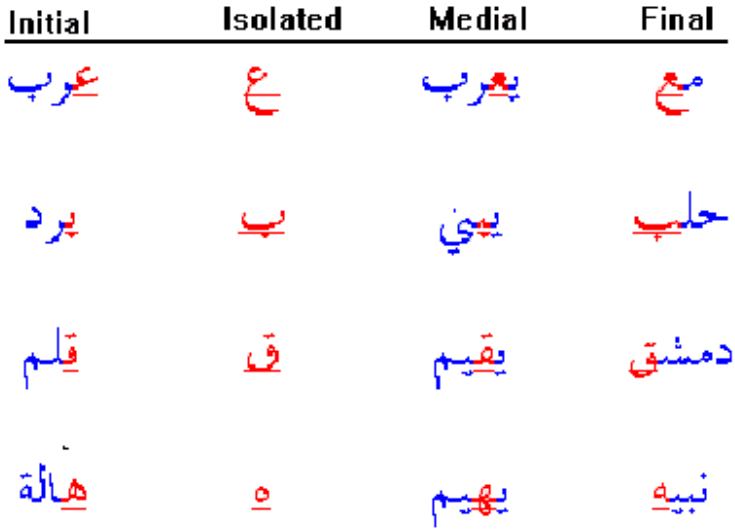
3. Arabic includes diacritics. These are marks placed above or below letters which typically represent vowel sounds or other modifiers.
4. Arabic includes ligatures. This is the process whereby two consecutive letters printed are replaced by a single new character.

Figure 1 – The Arabic Alphabet



Light Grey - Letters which cannot be joined on the left
Dark Grey - Letters which dramatically change shape according to their position in the word

Figure 2 – examples of four forms of characters ('ayn', 'baa', 'qaaf' and 'haa', respectively), according to their position in the word:



The problem thus becomes the sheer amount of graphic representations needed in order to create a complete Arabic font. Since the number of graphic representations is dramatically multiplied when using vowels, and since Arabic (to a native speaker) is easily understood without the vowels, we chose not to include vowels at this stage.

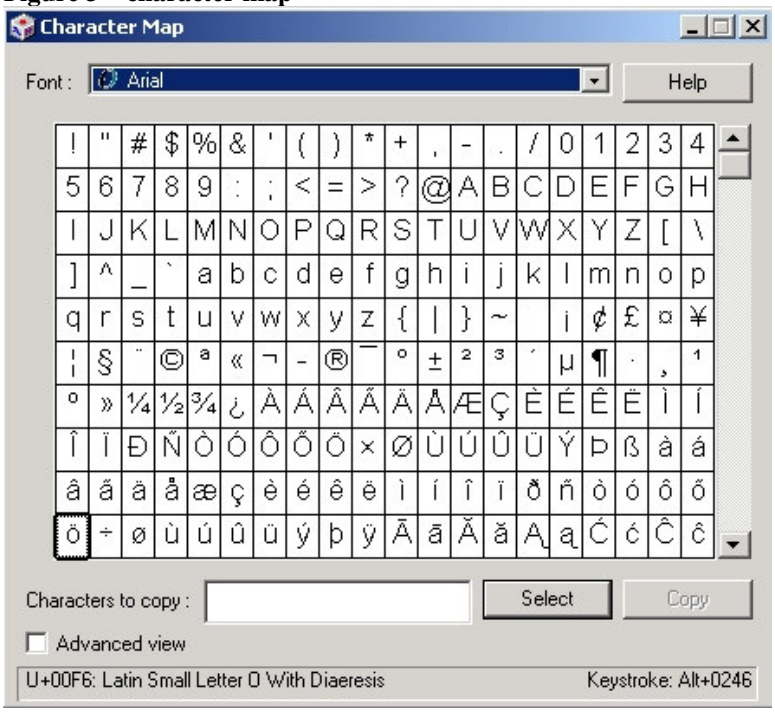
2.2 Arabic Encoding

Standard operating systems, such as MS Windows, use ANSI code for the Latin alphabet. This coding system uses a set of 8 bits to represent each character, which allows for 256 different symbols (2^8). Not all of these codes can be used, as several are reserved for special characters. Because of the special features of Arabic outlined above, the ASCII character set is not sufficient to display Arabic. It is therefore necessary to encode the characters into several character sets.

2.3 Hebrew-Enabled Operating-System

Since our primary survey language is Hebrew, we use a Hebrew-enabled operating system. On the one hand, this solves the problem of RTL writing in Arabic, as Hebrew is also written RTL. On the other hand, it further complicates the issue, by using a slightly different character set than that of a standard Latin alphabet. Some of the codes available in the Latin character set are reserved in the Hebrew character set and are therefore unavailable for use.

Figure 3 – character map



3. The Breakthrough

After careful analysis of the problems, we realized that some of the problematic features could in fact be used to our benefit. Since we anyway work with a Hebrew operating system there would be no need to account for Arabic's RTL direction. Furthermore, while Blaise cannot understand Unicode and requires an interpreter for such fonts, it has one unique feature that could aid us in our solution: Blaise allows mixing various font-types in a single line by enclosing each font in its correct font definitions respectively. Thus we were able to develop three separate fonts that together make up the complete Arabic alphabet and use the interpreter to translate Arabic texts into a string made up of all three fonts together.

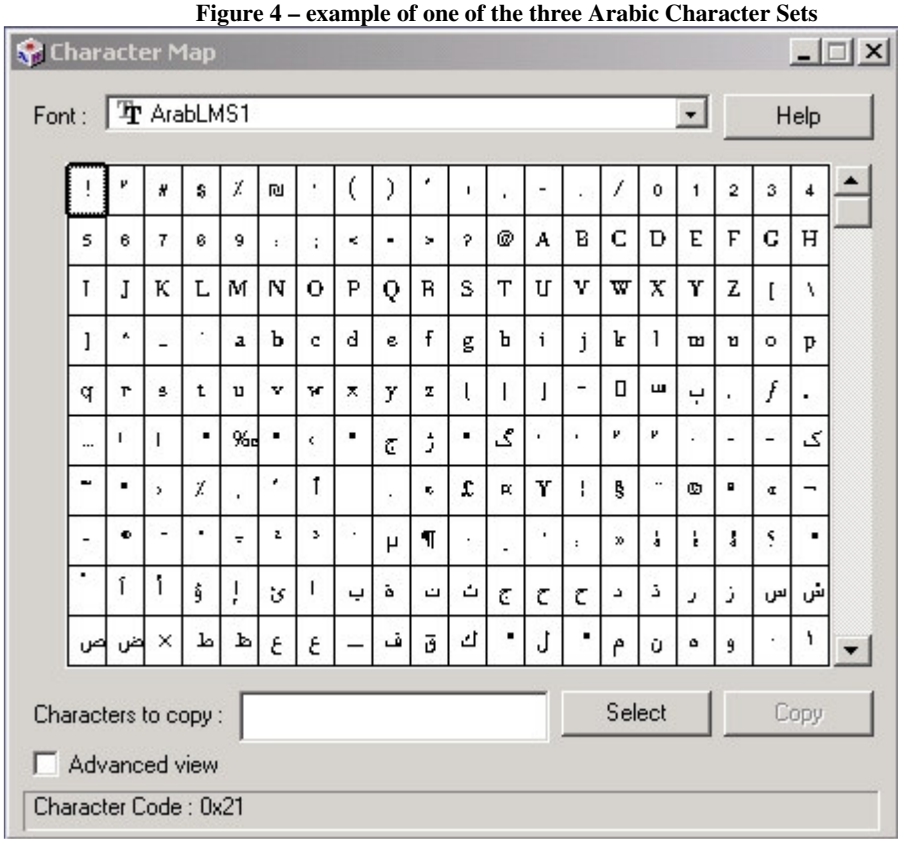
4. Our Solution for Displaying Arabic in Blaise

Our solution for displaying Arabic fonts in Blaise consists is twofold: First, designing three unique fonts that together make up a complete Arabic alphabet and, second, converting Arabic characters from MS Word into the new fonts designed for Blaise, including their Blaise font definitions.

4.1 Designing an Arabic Font

Since we are unable to use an Arabic operating system for the surveys, we required that a unique font be designed for Blaise. Over 500 separate symbols were needed (even without using vowels) and it was therefore necessary to assign 3 separate fonts to get a complete set of Arabic characters. The fonts also include numbers (Arabic numerals – 1,2,3…) and letters in English, since we sometimes have need for them. These fonts were designed uniquely for us and must be installed on all the computers that require Arabic in Blaise for both programming and surveying.

Figure 4 – example of one of the three Arabic Character Sets

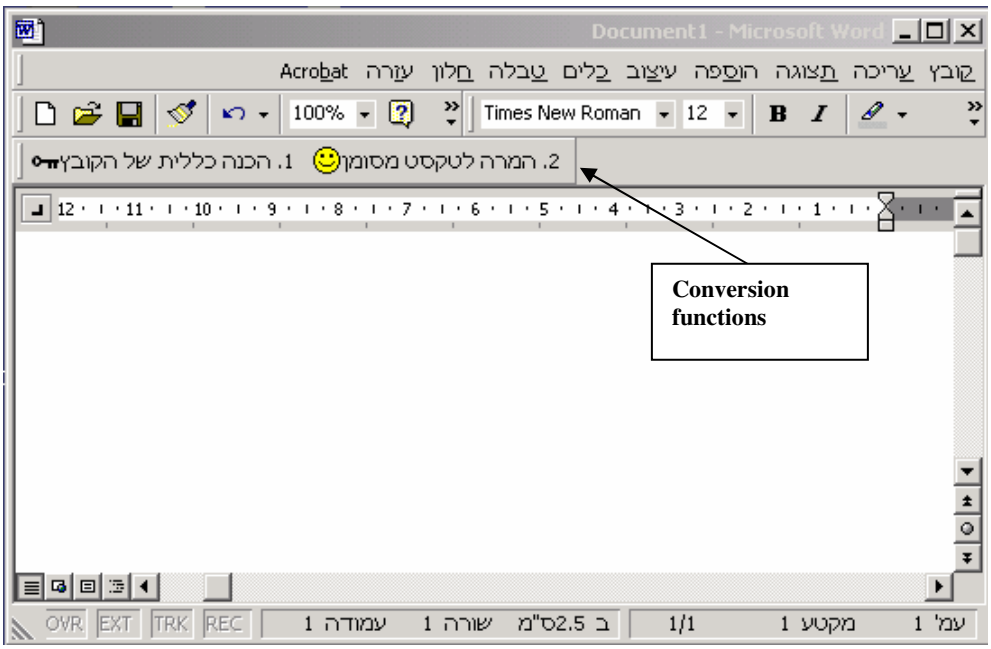


4.2 Converting Arabic from MS Word into the designated fonts for Blaise

In addition, it was necessary to develop an encoding program for Arabic, as it cannot be copied directly from MS Word into Blaise. This was achieved by developing a Word template that is able to convert the font as requested.
This file must be installed on all the computers that require Arabic for programming in Blaise. (Note that it is unnecessary to copy this file onto the laptop computers used for surveying.). The file is copied to:
C:\Documents and Settings*User*\ApplicationData\Microsoft\Word\STARTUP

Now, when we open a document in Arabic we have an extra toolbar with two functions necessary for converting the fonts.

Figure 5 – MS Word with conversion functions

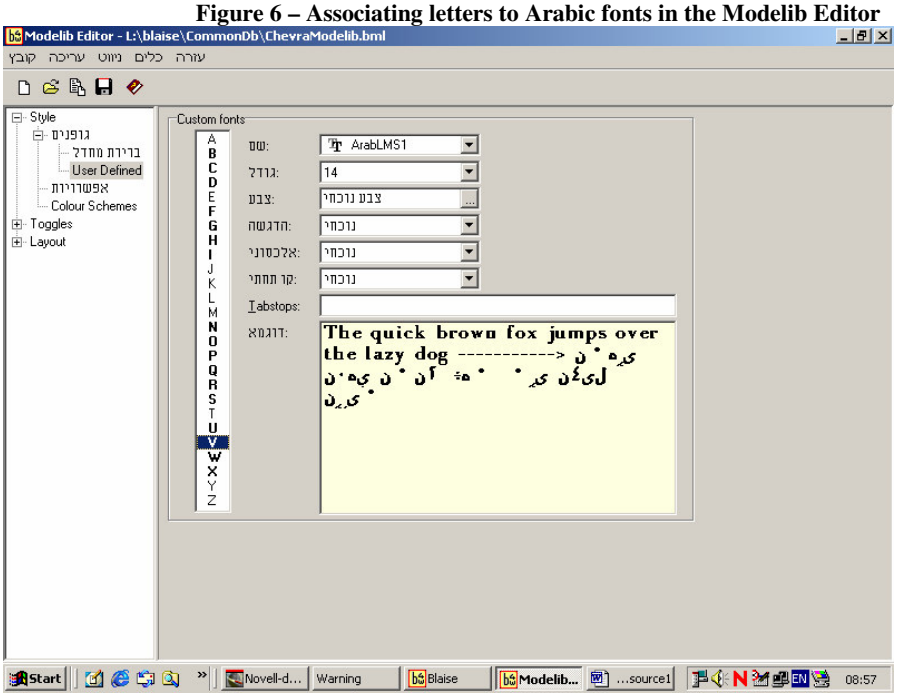


5. Using Arabic with Blaise

5.1 Defining The fonts in Blaise

We Associate the following three font letters in the Modelib Editor with the corresponding Arabic fonts:
X – ArabLMS1
Y – ArabLMS2
Z – ArabLMS3

Figure 6 – Associating letters to Arabic fonts in the Modelib Editor

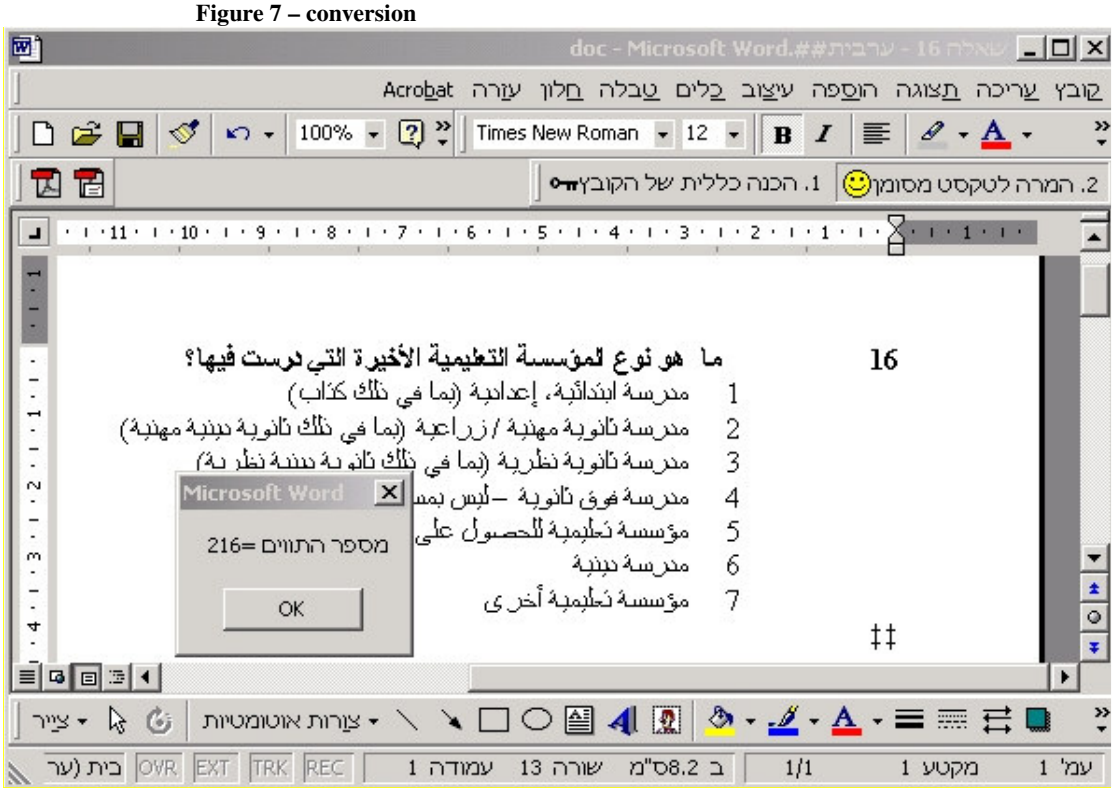


5.2 Copying Arabic text from Word to Blaise

In order to copy Arabic text from Word to Blaise we must use the special features developed for the conversion in word.
The first function *prepares* the entire text for conversion, so it is only necessary to perform this action once for any given document in Arabic.

The second function copies the selected sentence into the required font for Blaise. A popup window informs us of the length of the string:

Figure 7 – conversion

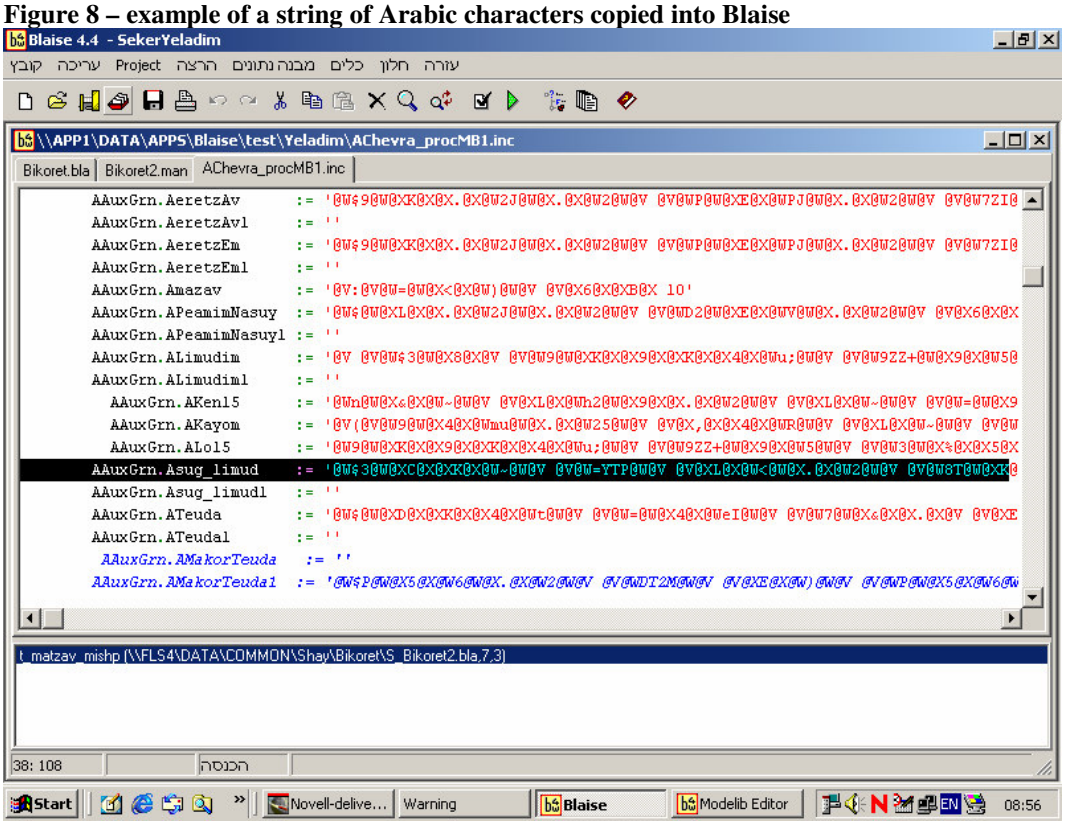


The string conversion in Word already includes not only the character itself but its corresponding font definitions for Blaise: @Xcharacter/s@X@Ucharacter/s@U@Wcharacter/s@W

Since the Arabic characters are split into 3 separate fonts, and each font requires a separate definition in Blaise, the output string resulting from the conversion is much longer than the original. Every section, or even a single letter at times, is surrounded by its definitions.

Next, the string is pasted into Blaise. There is no need to enclose it with font definitions as these are already part of the conversion. The length is defined according to the number displayed at the time of conversion.

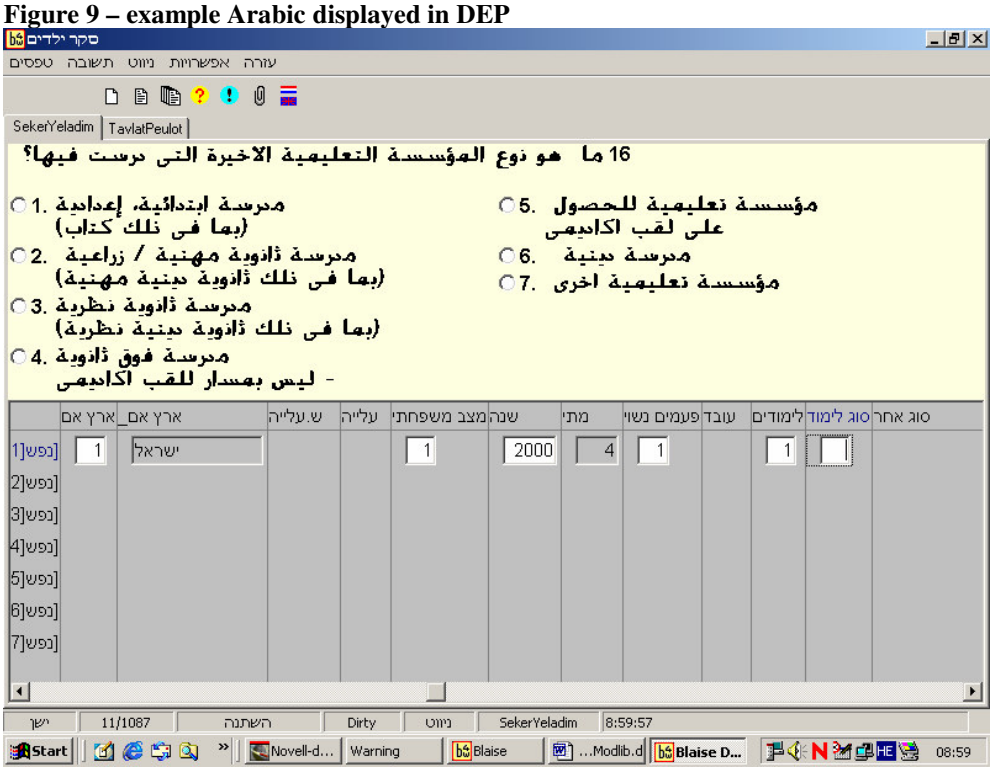
Figure 8 – example of a string of Arabic characters copied into Blaise



5.3 Displaying Arabic in the Data Entry Program

The above example of a string of Arabic characters will appear thus in the questionnaire itself:

Figure 9 – example Arabic displayed in DEP



Both Arabic and Russian are defined as parallel language blocks, so it is possible during the survey to switch between each of the languages at any given point.

6. Summary and Conclusions

Finding a solution for displaying Arabic in Blaise has been a top priority for Israel's Central Bureau of Statistics over the past few years. Arabic is one of Israel's two official languages and it became crucial that we be able to display it properly once we began using CAPI. The complications arose from the number of symbols required for encoding Arabic characters, which is greater than those available in a standard 256 ANSI character set. The difficulties, however, were overcome by designing b`oth a set of fonts and a converter from Word into Blaise. We have begun using Arabic characters in our surveys as of 2004 and are pleased with the results.

Currently we display Arabic with no vowels. Further development may be required in order to include vowels: a complete set of Arabic characters with vowels will require several *thousand* symbols. However, at present we find there is no need for displaying the vowels.

Another aspect of our present solution which might be problematic is the length of strings of Arabic characters. Since each string consists of three different fonts intermingled, and each font must be enclosed within its Blaise definitions, the strings end up being quite lengthy. The texts in Arabic, therefore, require more memory than the parallel text in our other survey languages.

7. References

Groeneveld, R., (2003) “Using non-Latin alphabets in Blaise”, Proceedings of the 8th International Blaise Users Conference

Cheung, G.Q. and Liu, Y., (2003) “Displaying Chinese Characters In Blaise”, Proceedings of the 8th International Blaise Users Conference

http://diwww.epfl.ch/w3lsp/conferences/ridt98/decotype.html

http://members.aol.com/gnhbos/meissues.htm

Codification automatique en ligne

HUMBERT Stéphane, INSEE, France

1. Introduction

L'ensemble des enquêtes auprès des ménages de l'INSEE sont réalisées grâce à une application CAPI qui permet d'intégrer les différents questionnaires écrits en Blaise. Certaines variables telles que la profession sont collectées dans des zones de textes qui permettent à un enquêteur de relever la réponse exacte donnée par l'enquêté. Elles doivent donc être codifiées systématiquement pour pouvoir être exploitées.

L'opération était jusqu'à présent réalisée à l'INSEE après la collecte en deux étapes :

- une codification automatique à l'aide d'un outil (SICORE) assurant d'une part la reconnaissance des libellés et d'autre part l'application de règles de codage s'appuyant sur d'autres variables appelées variables annexes,
- puis une reprise dans un service de l'INSEE pour les libellés qui ne peuvent être codifiés automatiquement.

Il y avait alors un nombre important de libellés envoyés en reprise et l'ensemble des variables annexes pouvant éventuellement servir à la codification devaient être collectés. Pour réaliser des gains de productivité et de délai, la codification automatique de ces variables a été insérée sur le poste de collecte.

Après une rapide présentation du fonctionnement de l'outil SICORE, nous verrons quelles fonctionnalités ont été développées sur le portable des enquêteurs. Nous expliquerons ensuite comment ces fonctionnalités ont été implantées d'un point de vue technique. Enfin, nous exposerons quelques applications de cette codification.

2. Présentation de SICORE :

SICORE est un outil qui permet de codifier dans une nomenclature des libellés saisis en toutes lettres en s'appuyant sur des bases de connaissance. Cette codification s'effectue en deux étapes. La première étape est une reconnaissance du libellé qui s'appuie sur une recherche en bigramme. En cas de succès de cette première phase, un précode est obtenu. La deuxième étape consiste alors à affecter un code final à partir d'une table de décision indiquée par le précode obtenu lors de la première étape. Cette table de décision donne les règles à suivre pour réaliser la codification à l'aide d'éventuelles variables annexes. Si toutes les variables annexes utiles ont bien été collectées, on obtient en final une codification.

Un "serveur SICORE" a été installé sur le portable de l'enquêteur pour permettre à Blaise d'accéder à ces fonctionnalités. Ce serveur fonctionne de la façon suivante : au démarrage, les bases de connaissance sont chargées en mémoire vive. Le serveur reste alors en attente des demandes de connexion de postes clients. Lorsqu'une telle connexion est établie, il reçoit les requêtes en provenance du client, effectue ces requêtes et renvoie le résultat au client. Du fait du chargement de l'environnement en mémoire vive la réponse est quasi-instantanée sur un portable enquêteur.

La requête et la réponse à la requête sont formées de chaînes de caractères aux formats prédéfinis. La chaîne de la requête contient notamment le nom de l'environnement SICORE à utiliser pour effectuer la codification, le libellé à codifier et la liste des variables annexes renseignées avec leur valeur. La chaîne de la réponse contient quant à elle le résultat de la codification, et éventuellement le précode, le code final et la liste des variables manquantes (variables dont SICORE a eu besoin et qui n'étaient pas dans la liste précédente).