

SAKHR BILINGUAL OCR (AL-QARI' AL-ALI). A USER'S INITIAL IMPRESSIONS

In this paper I would like to record some initial impressions from working with the Sakhr bilingual OCR system known as Al-Qari' al-Ali, to comment certain specific features of the program and to suggest a number of ways in which it may be improved.

It is perhaps appropriate to start with a few remarks on the origin of the product. It was first mentioned in 1990, when Dr. Efim Rezvan of the St. Petersburg Branch of the Oriental Institute of the Russian Academy of Sciences proposed the development of such a program in his report "Computer Methods in Qur'anic Studies" presented at the 2nd Conference and Exhibition on Bilingual Computing in Arabic and English in Cambridge. Originally the program was conceived as a powerful tool to facilitate the preparation of critical editions of Arabic sources by means of transferring large amounts of printed Arabic texts to computer files for subsequent processing. The immediate objective was the preparation by Valeriy V. Polosin of a critical edition of the famous "Fihrist" by Ibn al-Nadim. Dr. Rezvan considered this to be an excellent opportunity to develop and apply new techniques and software, and managed to interest a group of talented young programmers who had worked in the former Soviet high-tech military industry in the project. For a year Alexander Staryh, Mikhail Beregov, Alexander Popov and Fedor Bikov, in collaboration with Efim Rezvan, devoted nearly all their free

time to the development of the DOS prototype of the program, which was given the name MULTREC (Multi-Lingual Text Recognizer). The program was demonstrated in 1993 at the 3rd International Conference and Exhibition on Multi-lingual Computing held at Durham, where it aroused considerable interest, since it was virtually the only working program of its type. At this time the software company al-Alamiah became interested in the program, and subsequent to a visit to St. Petersburg by al-Alamiah's General Manager Dr. Ashraf Zaki, the preparation of a new Arabized version of the program was planned. The new version combined the achievements of the Russian programmers with important contributions made by specialists at al-Alamiah.

The first commercial version of Al-Qari' al-Ali was marketed in 1994. This product, although quite useful, has not yet become wide-spread, on the one hand because of its recent appearance and on the other because of its relatively high price and the powerful hardware it requires (a Pentium processor and a scanner with 600 dpi resolution are recommended). Hoping to introduce the product to my colleagues in Arabic studies who may not have had the opportunity to use it yet, I would like to report briefly on some characteristics of the program and how it may be applied.

Characteristics and area of use

Al-Qari' al-Ali works under the operating system "An-Nawafidh al-'Arabiya" 4.01 (or later), which, in turn, is installed over a Windows 3.1 operating system. It allows the transfer of scanned images of printed Arabic materials into text format, yielding 8-bit encoded text files which can be processed with al-Alamiah's word processor "al-Ustadh" or, for example, with the Arabic version of Microsoft Word for Windows 6.0. The program can be used for recognizing any Arabic printed matter. But if the text contains numerous ligatures, which is characteristic of older printed texts [1], errors at "Recognition" are practically inevitable, so the user has to correct them later during "Spell Checking". The best results are obtained from well printed mod-

ern texts with a minimum of ligatures. It is possible to transfer rather quickly a modern book or magazine into computer text with few errors (no more than 1%). As for poorly printed older books with a great many ligatures not included on the training keyboard and a variety forms for a given character, the process of recognition is regularly accompanied by errors. With such materials the production of a computer text file is extremely time consuming because of the need for careful correction of the recognized text (at first with the help of the built-in spell checker, and then by checking the corrected text in Word 6.0 or some other word processor). Even so, the production of an Arabic text is much faster than by typing, though it requires a

more highly qualified user. *The use of such a program is at any rate practically the only way for the majority of European Arabists to computerize a large amount of Arabic printed matter.* For the majority of Eastern European scholars, moreover, the services of professional Arabic typists are beyond reach, and the percentage of errors in typed text is rather high as well. It should be noted, however, that the advantage in speed becomes significant only when transferring rather considerable amounts of text (not less than ten or twenty pages), because preparing the pro-

gram to work, that is, "teaching" a new font, is a laborious process with fonts of any complexity.

The program is particularly important for the urgent task of compiling databases of medieval and modern Arabic texts, such as the database being developed at the University of Bergen under the direction of Prof. Joseph N. Bell, where I had the opportunity to work with Al-Qari' al-Ali [2]. Another particularly promising database project was begun in November 1994 in Saudi Arabia [3].

Hardware requirements

Anyone who has dealt with Arabic PC software knows how complex and slow these programs tend to be, especially in comparison with similar Latin programs. Al-Qari' al-Ali is no exception to this rule [4].

The program requires at least 386 processor with 4 Mb RAM and 10 Mb available disk space, but a more powerful hardware configuration is very welcome. Working with a Pentium 90 with 16 Mb RAM produces quite acceptable results. On a 486 DX2/66 with 12 Mb RAM the processing of scanned images of Arabic text was less successful. The teaching of a font and the further recognition of scanned text were not very difficult for the smaller computer, although the processor could not provide permanent support for the keyboard layout display on the screen and restored it after each operation. But the subsequent spell checking takes far too much time. Going from one error to the next takes up to half a minute, and if the font requires further teaching, which is practically inevitable even for rather

carefully taught fonts, the total correction process for one word can take several minutes. Thus the use of a Pentium with 16 Mb RAM is to be recommended when working with the program.

However, I would like to emphasize that a powerful processor is required not primarily for recognition of the text, but first of all for the spell checking, which is most important when working with poorly printed materials and complex fonts. In the case of modern books or typewritten texts, a weaker configuration (486 or even 386 with 4—8 Mb RAM) can be used.

The scanning resolution recommended by the manual is 300 dpi. However, it seems that the scanning of rather complex fonts of small size with this resolution can cause too many errors during the recognition process. In such cases, if the hardware configuration allows normal work with a higher resolution, this would be preferable. I achieved acceptable results scanning with 600 dpi.

Some remarks on the work with the program

Al-Qari' al-Ali comes with a standard set of modern computer fonts, which the program can recognize automatically. If the font of the scanned text is not included in this set, the program, after a search which may take some minutes, reports that no built-in font coincides with the scanned one. In such a case one must teach the program the new font. For this purpose it is generally sufficient to process in "learning" mode at least one and one-half to two pages of text, after which almost all letters, ligatures, and other symbols of the font will have been taught. Afterwards, it is useful to process one or two additional pages in a separate recognition mode within the learning option.

Learning option

In the learning mode each character or ligature of the scanned text is distinguished by the program, and the user must choose its alphabetic equivalent from the four-page keyboard layout on the screen (letters are on the first page, figures and other special characters on the second, ligatures on the third and fourth). At first one must do this for every character. Eventually the program will offer its own choices, which one can accept, if correct, or replace.

In the special recognition mode within the learning option, the program, having already been taught most of the characters, automatically recognizes them, stopping only on the symbols which it can not recognize. The user can then set the alphabetic equivalent of the unknown symbol himself (as in the learning mode). This option

makes it possible to process a page quickly and to teach the program most of the remaining symbols. One should not pass to this mode too early, however, because during "recognition" the program may make mistakes which it will not be possible to discover automatically. The most usual errors are connected with diacritics, the first and main parameter analyzed by the program being the "shape" of the letter or ligature. Thus, for example, if the program has been "taught" medial "bā" but not medial "nūn", "yā" and other similar characters, it will consider every "one-tooth" letter as "bā". The same applies to such pairs as "r" — "z", "d" — "dh", and "t" — "z". This problem is particularly troublesome with ligatures. If, for example, the user has taught the program the medial ligature "nb", he must theoretically teach it another thirty-five similar ligatures, that is, all thirty-six two-letter combinations of the six "one-tooth" letters "b", "t", "th", "n", "y" and "hamza". Otherwise errors such as *kunba* for *kunya* will occur regularly throughout recognition. But the number of non-standard ligatures one may teach the program is limited to about 130, which is often insufficient for a complex font. Therefore, the user should consider very carefully whether it is necessary to teach a given ligature or not. The criterion, naturally, is the existence and frequency of similar ligatures. For example, if one has taught the ligature "bah", "yah", etc., it will be expedient to add the ligature "ah", though it is not a frequent one. If one has taught a ligature and sacrificed its uncommon variants, error when these rare combinations occur will be inevitable.

To avoid such errors the user must periodically go out of recognition mode and return to learning in order to check how correctly and completely he has taught the program the font in question. If too many errors occur, it is reasonable to remove some of the most "dangerous" ligatures. Sometimes it is better to get a symbol for an unrecognized character ("^") instead of the wrong letter or ligature. It is easier to find this symbol after recognition is complete and to correct it then. Spell checking could pass over possible errors, because it is not rare in Arabic that a change of a letter produces a new "correct" word. However, the teaching of the majority of such ligatures will usually be finished during the following step of the work, the recognition of the whole text.

Recognition option

Having taught the program the font, one can pass to the next step, namely recognition of the text. After recognition of a given page one should spell check the recognized text. The parallel movement of the cursor, which highlights a block of the text in two windows (*Text* and *Image*) makes it possible to correct errors rather easily. While correcting one should continue teaching the font, since from the *Spell Checking* window it is possible to switch to learning mode and teach incorrectly recognized characters. Usually, such errors are caused by ligatures, so at this step one will face the serious problem of selecting ligatures to be removed, because the number of ligatures that can be taught in one font is limited.

Spell checking (including the *final stages of teaching a font*) takes from ten to fifteen minutes up to one and a half hours per page initially, depending on the complexity of the font. As one progresses in a text, this time is reduced as the number of characters that must be taught decreases. However, for complex fonts the process of re-teaching continues, practically as long as one is working with the text.

When the user passes from the *Spell Checking* window to the learning option, he can also correct errors noticed in

the *Text* window (unfortunately in this version of the program it is impossible to pass directly from the *Spell Checking* window to the *Text* window). Here it is necessary to be very cautious. Firstly, the position of the cursor on the screen after moving it in the *Text* window does not correspond to its real position (the difference is some three or four places), so in order to find out where the cursor really is, it is necessary to perform some operation in the *Text* window or simply to move the text in the window slightly. Secondly, it is not a good idea to correct text below the last place checked, because it could cause loss of connection between the *Text* and *Image* windows before the text is fully corrected.

Once spell checking is completed, it is necessary (using a special option) to detect unrecognized characters (designated by the symbol "^") and to correct them.

The result is a quite readable text with rather few errors. For a font of average complexity they will number from ten to twenty per page. For simple modern fonts the number will be very small (as in similar Latin OCR programs), but for complex fonts the amount of errors can be rather considerable. Subsequent manual correction of the text in Arabic Word 6.0 (or other word processor) is recommended in all cases.

Once the processing of a page is finished, it must be added to the text file, which should be saved in the OCR program as Arabic MS-DOS Code Page 720, and opened as the same in Word 6.0. The most convenient way is to use files with 7—10 pages, since the further processing of larger files (more than ten pages) in Word will be quite slow.

If one has to convert a rather short text (no more than ten pages), it would be useful to reduce the time of teaching the font and to correct inevitable errors by using the spell checking application. In such a case it will be enough to teach only one page and then to process another in the recognition mode within the learning option. Thereafter one can turn to the recognition of the whole text with a minimum teaching of the font during spell checking.

Recommendations for improvement of the program

Since the program will continually be improved, we would like to point out some problems which we hope the developers will take into consideration in future upgrades.

1. Switching From Spell Checking to Text Window

The most needed improvement would be to provide direct switching from the *Spell Checking* window to the *Text* window without closing the former (a similar function exists, for example, in Word 6.0). This is especially important because of certain peculiarities of the Arabic script. In a great many old Arabic printed texts the spaces between words are not indicated, and spaces often occur in the middle of words, rather than between them. As a result, the word could sometimes be cut in two. For example, the word 'arsala could be cut in two, if the break between "rā" and "sīn" is too large. Such an error, although quite typical, will never be found by the spell checker, since both *sall* and *'ara* exist in Arabic [5]. The second half of the incorrectly divided word 'aw][dahnā will be discovered by spell checking, since the word "d-h-n" does not exist in Arabic, but nevertheless one will not be able to correct this error from the *Spell Checking* window. The user has to

delete the space between 'aw and d-h-n, but for this purpose it is necessary (1) to close the *Spell Checking* window, (2) to pass to the *Text* window, and (3) to delete the space. Afterwards the user has to start spell checking again, so the operation will take considerable time. The fact that many scanned texts will contain a large number of such errors is the main reason why easy and fast switching to the *Text* window is desirable.

A second method to solve this problem would be to provide "Delete Space Back" and "Delete Space Forward" options inside the *Spell Checking* window.

2. Zooming of the image

In the present version of the program, zooming of the text image enables the user to enlarge it (in the learning mode and while spell checking), but does not allow him to diminish it. However, if only a very small part of the image is visible in the window, there is often a need to zoom the text out [6]. So, it would be useful to add to the zoom feature one or two options less than 100%, including at least one as low as 75%.

3. Learning option

a. When switching from learning mode to recognition (and vice versa), it is necessary first to stop the learning process, then switch to recognition mode, and then start the learning process again (altogether 3 steps). This switching operation takes a great deal of time, and in the second stage of teaching a font it must be carried out quite often. Direct switching from learning to recognition (and back) by pressing a key would be a considerable improvement.

b. A space symbol should be included in the keyboard character set. Sometimes it is senseless to provide meaning for a "symbol" distinguished by the program (for example, if it is a printing error, non-text mark, just a paleographical or paper defect, etc.).

c. A feature making it possible to "erase" such unnecessary elements from the scanned image would also be useful. An eraser is likewise necessary when adjusting the frame before the recognition of text (especially when dealing with poor quality printing, where the Arabic text frame is accompanied by a "dust" cloud), because the options "Selection of Text for Recognition" and "Marking Zones to Be Excluded from Recognition" are sometimes too cumbersome.

4. Ligatures

a. The window listing optional ligatures gives them in the order of creation rather than alphabetically, which in most cases makes it more difficult to find the ligatures one is looking for. Therefore, if possible, the ligature window should include an optional alphabetical sorting button.

b. In the current version, after a ligature window is opened and, then closed or removed, the cursor moves back to the top of the ligature list. In order to continue work with the ligatures, one has to place the cursor in the window of the ligature list and click it. Then one must start looking through the ligature list from the very top (this is especially important when the number of non-standard ligatures has already been exceeded and one has to select ligatures to be removed). It would be much more convenient if the cursor returned to the former position in the ligature list after closing or removing an opened ligature.

c. It would be helpful if the following standard ligatures, or at least some of them, were included in the third and fourth pages of the keyboard layout display (a fifth page would be a useful addition):

- 1) "لك + كم": كك; ككلا; ككما; كككم;
- 2) "حى + حى": حى; حى; حى (three variants — حى; حى; حى); حى; حى (two variants);
- 3) "به/بة": twelve variants (although the actual number of quite frequent ligatures is smaller) + عه (four variants); مه (two variants); له (two variants); كه (two variants);
- 4) "... + م": مم; مم (two variants); عم (two variants); حم (three variants); سم; سم (two variants);
- 5) "لد": three variants;
- 6) "... + ح": سد (six variants); صد (six variants); فد (six variants); عد (two variants);

7) بنى; على; محم; لله 7)

8) The dialogue boxes for some standard ligatures (for example, "b-y") offer only two standard positions (final and isolated), but in some fonts other positions occur, so these non-standard positions should be included as well. In the current version of the program the only way to deal with such positions is to create three-character non-standard ligatures, such as "b-y-keshida".

d. The fact that the program separates poorly the following ligatures should be considered:

- 1) "Consonant + alif" (for example, "nā", "fā", etc.)
 - 2) "Consonant + rā" (for example, "l-r", "m-r")
 - 3) "Consonant + wāw" (for example, "bū", "fū")
 - 4) "Consonant + sin" (for example, "yas", "fas")
 - 5) "Consonant + "single-tooth" consonant" (for example, "kan", "man")
 - 6) Two "single-tooth" consonants (for example, "yab").
- This is probably a function of the peculiarities of the fonts, on which I tried the program, but nevertheless it would be useful if the separation of such frequent quasi-ligatures were improved in the next version of the program.

e. It would be helpful to have two operating modes:

- 1) one mode with a minimum of non-standard ligatures, some fifty to seventy (for a simple modern font);
- 2) one mode with a maximum of non-standard ligatures, some 200 to 300, or more (for complex fonts).

The availability of two such operating modes would, on the one hand, simplify and accelerate using the program with modern texts. On the other hand, it would also facilitate work with complex fonts, because the amount of permitted non-standard ligatures (about 130) is not always enough, even for unvowelled texts, and it is obviously not sufficient for work with fully vowelled texts such as poetry.

Of course, an indefinite number of allowed non-standard ligatures would be welcome, since work with complex multi-ligature fonts would become much easier, though increasing the number of ligatures taught, if possible with the current algorithm, will slow the program down correspondingly.

f. There are also some problems related to the recognition of separate consonants:

1) The program unsatisfactorily distinguishes a medial "hā". Despite the presence of a specific variant of this glyph among the variants taught, the program often fails to recognize it. When this happens, increasing the number of variants taught provides little help.

2) The same problem occurs with "kāf" and its ligatures ("كما"; "كل"; "ك" and others). In the case of "kāf", the problem is probably caused by the link of the "tail" of the "kāf" with the previous letter, but in the case of "hā" the reason is not clear. Perhaps this peculiarity of the program should simply be accepted and the user should not continue to teach these two letters if the program fails to learn them.

g. It would be useful to increase the number of variants allowed for two letters: "alif" (isolated and final) and "lām" (initial and medial). At least twice the normally permitted eleven variants are required, since confusion of these two letters is common in many fonts. In the current version of the program one must select very carefully which variants are to be kept or removed, a process which is time-consuming, but still produces only meagre results.

مادام بطناً وأذا بس حمايته وهو مسم قائل وقيل هي حجرة نارياً تشبه الضريح وقال ابن ٤٤٩ كسان هو طعام يضربون عنده

ولم يكن الذين كفروا من أهل الكتاب والمشركين منفكين حتى تأتيهم البينة رسول من الله بشي لو صحفا
 مطهرة فيم كتب بقرآنها وتقرى الذين أوتوا الكتاب الا من بعد ما جاءتهم البينة اعلم ان في الآية
 مسائل (الاشكال الاولى) قال الواحدي في كتاب البسيط هذه الآية من اصعب ما في القرآن نظماً
 وتفسيراً وقد تحفظ في الكتاب من العلماء ثم انه رحمه الله تعالى لم يلخص كيفية الاشكال فيها وأنا أقول وجه
 الاشكال ان تقدير الآية لم يكن الذين كفروا من المشركين حتى تأتيهم البينة التي هي الرسول ثم انه تعالى لم
 يذكر انهم من المشركين من اذ لم يردوا الكفر الذي كانوا عليه فصار التقدير لم يكن الذين
 كفروا من المشركين عن كفرهم حتى تأتيهم البينة التي هي الرسول ثم ان كلمة حتى لانها انما هي في الآية
 تقتضي انهم صاروا من المشركين عن كفرهم عند انزال الرسول ثم قال بعد ذلك وما تقرى الذين أوتوا الكتاب
 الا من بعد ما جاءتهم البينة وهذا يقتضي ان كفرهم قد ازداد عندهم والرسول عليه السلام حينئذ يحصل
 بين الآية الاولى والاشكال الثانية منافية في الظاهر هذا منتهى الاشكال فيما أظن (الجواب) عنه من
 وجود (أولها) وأصحها الوجه الذي نخصه صاحب الكتاب وهو ان الكفار من الفريقين أهل الكتاب
 وعبدة الاوثان كانوا يقولون قبل بعث محمد صلى الله عليه وسلم لانك علمنا نحن عليه من ديننا ولا نتركه
 حتى بعث النبي الوعد الذي هو مكتوب في التوراة والانجيل وهو محمد عليه السلام فكيف الله تعالى
 ما كانوا يقولونه ثم قال وما تقرى الذين أوتوا الكتاب يعني اسم كانوا يدعون اجتماع الكلمة والاتفاق على
 الحق اذ اذاهم الرسول ثم ما قرعهم عن الحق ولا أقدم على الكفر الا على الرسول ونظيره في الكلام ان
 يقول القبر العاسق ان به ظاهرا استمتع بما آتاه من الاقوال القبيحة حتى برز في الله التي فمارزقه
 الله التي ازيد اذ فاقول واعظه لم تكن من المشركين الفسفي حتى توبوا وما عشت وأسلت في الفسفي الابد
 اليسار بكروه ما كان يقوله توبوا لولا ما حصل هذا الجواب برجع الى حرف واحد وهو ان قوله لم يكن
 الذين كفروا من المشركين عن كفرهم حتى تأتيهم البينة كوزح كانه عنهم وقوله وما تقرى الذين أوتوا
 الكتاب هو اخبار عن الواقع والمعنى ان الذي وقع كان على خلاف ما دعوا (وثانيها) ان تقدير الآية لم
 يكن الذين كفروا من المشركين عن كفرهم وان جاءتهم البينة وعلى هذا التقدير يزول الاشكال هكذا ذكره
 القاضي الان تفسيرا لفظه حتى لم يرد البس من الآية في (وثالثها) ان الاحتمال قوله من المشركين على الكفر
 بل على كونهم من المشركين عن ذكر محمد بالمناقب والصفات والى ما لم يكن الذي كفروا من المشركين عن ذكر
 محمد بالمناقب والصفات حتى تأتيهم البينة قال ابن عرفة أي حتى أتتهم فاللفظ لفظ المناقب وهو معناه
 الماضي وهو كقوله تعالى ما تتلوا الساطين أي ما تاتت والمعنى انهم ما كانوا من المشركين عن ذكر مناقبه ثم
 لما جاءهم محمد تفرقوا فيه وقال كل واحد في قول آخر يرد بان نظيره قوله تعالى وكانوا من قبل يستفتحون
 على الذين كفروا لما جاءهم ما عرثوا كفروا به والاقول المختار في هذه الآية هو الاول وفي الآية وجه
 رابع وهو انه تعالى حكى عن الكفار انهم ما كانوا من المشركين عن كفرهم الى وقت يحيى الرسول وكلمة حتى
 تقتضي ان يكون الحال بعد ذلك بخلاف ما كان قبل ذلك والامر هكذا كان لان ذلك الجموع ما به واعلى
 الكفر بل تفرقوا عنهم من صاره وثمانون منهم من صار كافرا واسلم ببق حال أولئك الجمع بعد يحيى والرسول
 كما كان قبل بعثته كفي ذلك في اهل بابل لفظ حتى هو فيم اوجه خامس وهو ان الكفار كانوا قبل بعث
 الرسول من المشركين عن التردد في كفرهم بل كانوا حازمين به معتقدين حقيقته ثم زال ذلك الجزم بعد بعث
 الرسول بل قرأوا كبر مقتدرين في ذلك الذين وفي سائر الاديان ونظيره قوله كان الناس أمة واحدة
 فبعث الله النبيين مبشرين ومنذرين والمعنى ان الذين كفروا كانوا على صاركانه اختلط بلعهم ودمهم
 طاهر ودي كان حازما في يهوديته وكذا النصراني وعابد الوثن فلما بعث محمد عليه الصلاة والسلام اضطربت
 الباطل والافكار وتشكك كل أحد في دينه ودينه ومماته وقوله تعالى من المشركين مشبههم هذا لان
 تشكك النبي عن النبي وانفساله عنه فمنا أن قلوبهم ما خلت عن تلك الامم قائل وما انفصلت عن

(٥٧ - نغرامان) شوق الى مطهروم ما أو التناذير عند الكل واستغناءه عن الغير واستفادة قوة فهمه ان وكذا عظمهم عبارته

Fig. 1

(لم يكن الذين كفروا من أهل الكتاب والمشركين منفكين حتى تأتيهم البينة رسول من الله يتلو صحفا
 مطهرة فيها كتب قيمة وما تفرق الذين أوتوا الكتاب إلا من بعدما جاءتهم البينة) اعلم ان في الآية
 مسائل (المسئلة الاولى) قال الواحدي في كتاب البسيط هذه الآية من أصعب ما في القرآن نظما وتفسيرا
 وقد تخطب فيها الكبار من العلماء ثم انه رحمه الله تعالى لم يلخص كيفية الاشكال فيها وانا أمول وجه
 الاشكال أن تفديرا لآية لبيكن الذين كفروا منفكين حتى تأتيهم البينة التي هي الرسول ثم انه
 تعالى لم يذكر انهم منفكون عن ما ذاك لكنه معلوم ان المراد هو الكفر الذي كانوا عليه فصارا لتقدير
 لم يكن الذين كفروا منفكين عن كفرهم حتى تأتيهم البينة التي هي الرسول ثم ان كلمة حتى لا نتها
 الغاية فهذه الآية تقتضي انهم عاروا منفكين عن كفرهم عند اني ان الرسول ثم قال بعد ذلك وما
 تفرق الذين أوتوا الكتاب إلا من بعدما جاءتهم البينة وهذا يقتضي أن كفرهم قد ازداد عند مجيء
 الرسول عليه السلام فحينئذ يحصل بين الآية الاولى والآية الثانية منافضة في الظاهر هذا
 منتهي الاشكال فيما أظن (والجواب) عن من وجوه (أولها) وأسناها الوجه الذي لخصه صاحب
 الكشف وهو ان الكفار من الفريقين أهل الكتاب وعبداء الاوه ثان كانوا يقولون قبل مبعث محمد
 صلي الله عليه وسلم لا ننكح ما نحن عليه من ديننا ولا نتركه حتى يبعث النبي الموعود الذي هو
 مكتوب في التوراة والانجيل وهو محمد علي السلام فحكى الله تعالى ما كانوا يقولونه ثم قال وما
 تفرق الذين أوتوا الكتاب يعني انهم كانوا يعدون اجتماع الكلمة ولا تفاق علي الحق اذا جاءهم
 الرسول ثم ما فرقهم عن الحق ولا أقرهم علي الكفر الا مجيء الرسول ونظيره في الكلام أن يقول
 لفقيرا لفاسق لمن يعظه لست أمتنع مما أنا فيه من الافعال القبيحة حتى يرزقني الله الغني فلما
 رزقه الله الغني ازداد فسقا فيقول واعظه لم تكن منفكا عن الفسق حتى توسر وما غمت رأسك في
 الفسق الا بعد ان ليسار يذكره ما كان يقول تو بيحا والزاما وحاصله ذا الجواب يرجع الي حرف واحد
 وهو في قوله لم يكن الذين كفروا منفكين كفرهم حتى تأتيهم البينة مذكور حكاية عيه م وقوله
 وتفرق الذين أوتوا الكتاب هو اخبار عن الواقع والمعني ان الذي وقع كان علي خلاف ما اد
 عوا (وثانيها) ان تقدير الآية لبيكن الذين كفروا مفكين عن كفرهم وانجاءهم البينة وهلي هذا
 لتقدير يزول الاشكال هكذا ذكره الفاني الا أن تفسير لفظة حتى بهذا لليس من اللغة في شيء (وثا

Spell Checking

1. An on/off button for "Suggestions" would be useful (a similar option exists, for example, in Word 6.0). The function *Suggestions* is practically unnecessary, as the user can see the right word in the *Image* window, and it is moreover only of use when working with simple fonts. With complex fonts, the errors are so unpredictable that the user would hardly ever accept the suggestions proposed, although the program uses a great deal of time to produce them. However, spell checking with *Suggestions* in Word 6.0, when most recognition errors have already been corrected, is generally quite helpful.

2. The possibility to add new words to the spell checking custom dictionary would be of much use. A number of foreign borrowings, proper names, geographical names, and the like occur frequently, causing the spell checker to search again and again for the same unlisted item. With slower machines, the time spent can be up to ten or fifteen minutes per page.

3. Sometimes it is necessary to undo a correction or to see a corrected word again (as is possible, for example, with the "Undo Last" function in Word 6.0). The connection to the *Image* window in the case of the given correction will be lost, but the opportunity to go back would nevertheless be useful.

4. There are problems with the placement of the *Spell Checking* window. This window sometimes covers the text, and if one moves it, it will return to the former position after the next operation. The need to consult the context of a word to be corrected arises frequently. There are two possibilities here: (1) automatically moving the window to the top of the screen when spell checking reaches the middle of the page, or (2) saving the window position (i. e. not returning to the former position after the next operation). (It may be useful to create a button "Save Spell Checking window position"). The same problem exists, incidentally, with the *Find* and *Replace* windows.

5. It would be helpful to solve the problem mentioned above of the improper position of the cursor in the *Text* window, which occurs when the user during spell checking switches from the *Learning* window to the *Text* window.

My last recommendation concerns the program as a whole and the very principle of the recognition of Arabic symbols. Perhaps the technical implications are too great and would cause a considerable slowing down of the program, but it would be helpful if the program could take into account the position of a letter in a word (or a "block") more precisely, considering the previous letter as well as the subsequent one. What I am suggesting is that not only the "shape" of a glyph should be taken into consideration, but its "position" in the word as well. Many errors could be avoided if position was taken into account.

For example, in a number of fonts in poorly printed texts, there is practically no difference between the shape of medial "ayn" and that of final or isolated "hā". The reader can only understand the meaning of such a symbol according to its position in the word [7]. A similar analysis should presumably be done by the program. If a symbol being analyzed is followed by a medial or final variant of a letter, it means that the symbol cannot be final or isolated "hā", but only medial "ayn", even if the program can detect no difference in their shape.

However, such an analysis will not only require that the already recognized previous symbol be taken into account, but that the not yet recognized subsequent one be considered as well. The analysis of one symbol would thus consist of at least three additional steps (analysis of the letters on either side and of the group together). Since this would further complicate the program and would require even more powerful hardware, it is unclear whether such an innovation is feasible at the present time.

Notes

1. For example, works printed in the late nineteenth century in relatively complex fonts such as al-Razi's *Tafsir* printed in Cairo in 1308/1890—1891, on which I tried the program (see fig. 1).

2. I would like to use this opportunity to express my gratitude to the Research Council of Norway, to the University of Bergen, and personally to Prof. Joseph N. Bell for giving me a chance to participate in this very interesting project.

3. The Research Institute for Computer and Electronics (RICE) at King Abdulaziz City for Science and Technology started in November 1994 a project to compile a large database of Arabic texts of different types (classical, modern, scientific, etc.) which will be available to all researchers doing Natural Language Processing research.

4. A common Latin OCR program can rather easily disassemble a word into vertical segments (characters), as the characters are separated by blanks, but for Arabic text this is an extremely sophisticated problem. Absence of blanks between the characters, overlapping of two (or more) characters or the parts thereof in one vertical segment (for example, a "tail" of a "kāf" and a previous letter), a multitude of diacritical marks, variant forms of the same letter, standard and non-standard ligatures, and so on make it necessary to compute many parameters at once and require a powerful processor and considerable RAM.

5. Another characteristic example of this kind can be seen in the accompanying "recognized" page from al-Razi's *Tafsir*, where 'alayhi as-salām has turned in 'aly h as-sa lām" (see fig. 2).

6. If the user scans with 300 dpi resolution, this is not so important, but when he works with 600 dpi resolution it becomes necessary.

7. And, of course, from the context, but unfortunately we cannot use this criterion in the program.

Illustrations

Fig. 1. Page from al-Razi's *Tafsir* (Cairo, 1308/1890—1891).

Fig. 2. The same page as "recognized" and spelled by Al-Qari' al-Ali.

RUSSIAN ACADEMY OF SCIENCES
THE INSTITUTE OF ORIENTAL STUDIES
ST. PETERSBURG BRANCH



Manuscripta Orientalia

International Journal for Oriental Manuscript Research

Vol. 1 No. 3 December 1995

THESA
ST. PETERSBURG—HELSINKI

Typed
DLG.

RB NB! Correspondence Round
table Arabic/Farsi OCR

CONTENTS

<i>TEXTS AND MANUSCRIPTS: DESCRIPTION AND RESEARCH</i>	3
L. Menshikov. A Fragment of an Unknown <i>Leishu</i> from Tunhuang	3
T. Sultanov. The Structure of Islamic History Book (The Method of Analysis)	16
<i>TO THE HISTORY OF ORIENTAL TEXTOLOGY.</i>	22
K. Kepping. The Official Name of the Tangut Empire as Reflected in the Native Tangut Texts	22
<i>PRESENTING THE COLLECTIONS.</i>	33
T. Pang. Rare Manchu Manuscripts from the Collection of the St. Petersburg Branch of the Institute of Oriental Studies, Russian Academy of Sciences	33
<i>ORIENTAL MANUSCRIPTS AND NEW INFORMATION TECHNOLOGIES</i> <i>Correspondence Round table: Arabic/Farsi OCR</i>	47
A. Matveev. Sakhr Bilingual OCR (Al-Qari' al-Ali). A User's Initial Impressions	48
J. Bell & P. Zemanek. Test of Two Arabic OCR Programs	55
P. Roochnik. Itisalal OCR Discussion	58
<i>PRESENTING THE MANUSCRIPT</i>	63
O. Akimushkin. <i>Muraqqa'</i> . Album of the Indian and Persian Miniatures of the 16—18th Centuries and the Models of the Persian Calligraphy of the Same Period	63
<i>BOOK AND SOFTWARE REVIEW</i>	68

Color plates: *Muraqqa'*. Album of the Indian and Persian Miniatures of the 16—18th Centuries and the Models of the Persian Calligraphy of the Same Period (see p. 63—67).

Front cover:

Fol. 17a. Portrait of a Man by Ridā-yi 'Abbāsī, 11.8×8.2 cm.

Back cover:

Plate 1. Fol. 16a. Portrait of Timūr Khān Turkmān by Šādiqī beg Afshār, 19.3×11.6 cm.

Plate 2. Fol. 36a. The Darvishes Picnic in the Mountains. Probably Isfahan school, 25.5×14.5 cm.

Plate 3. Fol. 6a. The Shaykh and the Harlot by Muḥammad Yūsuf Muṣavvir, 18.2×11.3 cm.

Plate 4. Fol. 1a. Portrait of Mīrzā Jalālā by 'Alī Qulī beg Jabbādār, 16.0×9.1 cm.