Achtung!

Dies ist eine Internet-Sonderausgabe des Aufsatzes
„Multilingual text retrieval: Requirements and solutions“
von Jost Gippert (1998).
Sie sollte nicht zitiert werden. Zitate sind der Originalausgabe in
Studia Iranica, Mesopotamica et Anatolica 3, 1997 [1998], 75-93
zu entnehmen.

Attention!

This is a special internet edition of the article
“Multilingual text retrieval: Requirements and solutions”
by Jost Gippert (1998).
It should not be quoted as such. For quotations, please refer to the original edition in *Studia Iranica, Mesopotamica et Anatolica* 3, 1997 [1998], 75-93.

Multilingual text retrieval: Requirements and solutions

Jost GIPPERT (Frankfurt)

The further development of devices for a multilingual text retrieval is a major task in the adaptation of computational means to linguistic, literary, and historical studies. The textual heritage of Old Georgian is an excellent example to show what should be aimed at in this respect: Given that most of the texts that have come down to us are translations from various sources (Greek, Armenian, Syriac, Arabic, Persian), establishing the relationship between these texts and their (presumed) originals is of peculiar importance when the informations they conceal are to be revealed. In the present paper, I shall try to show in which way and with what means these aims can be achieved[1].

Let me first summarize what in my opinion are the general aims of computational text retrieval. As far as linguistic aims are concerned, texts have to be treated as coherent sources to be analyzed as to informations about phonetic, morphological, syntactical and lexico-semantic features of the language(s) involved. From a literary point of view, texts appear as coherent structures representing ideas, attitudes, etc. to be analyzed, e.g., with respect to the relationship between their contents and the shape they are presented in; it goes without saying that literary analysis of this kind is closely related (or even intersecting) with linguistic approaches, e.g. with a view to discourse structure or metrics. The same holds true for historical investigations which, in a widest sense, include political, sociological, juridical approaches etc.: Here, texts have to be taken as coherent sources to be searched for data about certain periods of time which can be represented, e.g., by names of persons and places or other key words, i.e., linguistic elements again.

When applying approaches as the ones outlined above to digitized textual data, some typical results can be achieved with no great effort. This is true, e.g., for statistical analyses on a phonetic, lexical, or metrical level. An example can be seen in Tbl. 1 which represents the phonetic (or rather graphemic) statistics of the Old Georgian Šaṭberdi codex (Xth century). Another typical result of computational text analyses is word indices and concordances which can be either partial or total; as an example, cp. the KWIC-index („Key-Word-In-Context“) of names beginning with *O-* as appearing in the Šaṭberdi codex (cf. Tbl. 2). Beyond that,

---

[1] This is an extended version of a paper which was first read on the 2nd Tbilisi Symposium on Language, Logic and Computation (Tbilisi State University, Sept. 15-20, 1997). An abridged version will be published in the conference proceedings.

digital texts should also be applicable to grammatical analyses of various kinds, including phonological, morphological, and syntactical approaches.

(’ა’):	132098	(’ნ’):	34800	(’ღ’):	5556
(’ბ’):	14505	(’ო’):	8436	(’ყ’):	9316
(’გ’):	15896	(’პ’):	27635	(’შ’):	8105
(’დ’):	36053	(’ჟ’):	2131	(’ჩ’):	2517
(’ე’):	62739	(’რ’):	700	(’ც’):	9082
(’ვ’):	18720	(’ს’):	39813	(’ძ’):	3617
(’ზ’):	3239	(’ტ’):	58289	(’წ’):	7916
(’თ’):	1443	(’უ’):	5861	(’ჭ’):	853
(’ი’):	35149	(’ფ’):	26067	(’ხ’):	7843
(’კ’):	74355	(’ქ’):	5603	(’ჯ’):	2168
(’ლ’):	6615	(’ღ’):	4145	(’ჰ’):	1122
(’მ’):	29901	(’ყ’):	6113	(’ჱ’):	1542
(’ნ’):	47023			(’ჳ’):	74

Tbl. 1: Statistics of the Šaṭberdi codex (Xth cent.)

Computational text retrieval of the indicated type – which normally consists of searching and finding certain sequences of characters in a given text environment – requires several conditions to be fulfilled. Above all, this concerns the structuring of the electronic texts which must be much more consistent than what is produced when using a normal word processor for writing letters or articles.

One thing that has to be cared for is a unique encoding of linguistic elements, i.e. letters or words, in correlation with the script system to be represented. There will often be several ways to produce a certain letter; e.g., letters like the German „umlauted“ *ä, ö, ü* can either be typed in as such (if present on a keyboard and in the script font used) or, as combinations of their basic element (vowels *a, o, u*) plus a diacritic trema, ¨, set above. While the results may look just equal on the screen and on the printer (which depends on the correct positioning of the diacritic), the encoding remains totally different, each character being represented by different bytes. When searching a given text file for such characters, there is hardly any way to allow for both encoding types to be found on the same searching condition. This is why the encoding of linguistic elements to be searched for has to be unique throughout.

For similar reasons, a unique encoding is also required for the structural elements of the texts to be analyzed. Firstly, this concerns all kinds of formatting prescriptions such as tabulators, indenting, and the like. If, e.g., the first line of a paragraph is to be indented, this can be done in different ways, using either a certain sequence of spaces or the indenting function of the text processor, both resulting in a very different encoding. If the indenting is meant to serve as an indicator of a certain text section such as, e.g., a paragraph boundary, one and the same encoding should be used throughout.

#ოზია (1)
268:22 – #იორამ, #იორამს – **#ოზია**, #ოზიას – #იოათამ, #იოათამს
#ოზია-ცა (1)
198:2 წელ; #აზარია, რომელსა **#ოზია-ცა** ერქუა, რომელი განკეთრდა – ნბ~
#ოზიას (1)
268:22 #იორამ, #იორამს – #ოზია, **#ოზიას** – #იოათამ, #იოათამს – #აქაზ, აქა
#ოზიაჲსა (1)
338:28 მამის–მამისა ჩემისა **#ოზიაჲსა**, ვითარმედ: "იგივე
#ოზიელ (1)
197:40 #ელისე, #აბდია, #იოველ, **#ოზიელ**, #ელიაზარ, #აზარია; #იორამ – ც~
#ოლდად (1)
198:14 #იერემია, #სოფონიას, **#ოლდად**, #ბარუქ; #იოაქას – გ~ თთუე.
#ომეროს (1)
196:18 ასოჲ. ამისთჳს–ცა მიამსგავსა **#ომეროს** რიცხჳ იგი კბ~თა მათ
#ონორი (1)
201:38 – იე~ წელ, #არკადი – კგ~ წელ; **#ონორი** – იგ~ წელ; #თევდოსი მცირც
#ოსე (2)
198:3 წინაჲსწარმეტყუელებდეს **#ოსე**, #ამოს, #ესაია, #იონას; #იოვათამ
198:6 მოუჯდა #ისრაჱლსა და წარტყუენა **#ოსე** მეფც #ისრაჱლისაჲ ათსა
#ოსჱსითგან (1)
198:25 ყოველნი წელნი ტყუენვითგან **#ოსჱსითგან** მეფისა #ისრაჱლისაჲთ და
#ოსია (1)
198:13 – ნე~ წელ; #ამოს – ბ~ წელ; **#ოსია** – ლა~ წელ,
#ოსოტერ (1)
200:19 – კგ~ წელ; #პტოლემეოს **#ოსოტერ** – კზ~ წელ; #პტოლემეოს
#ოქოზია (1)
197:40 #აზარია; #იორამ – ც~ წელ; **#ოქოზია** – ა~ წელ; #გოდოლია,
#ოძრაჯე (1)
320:5 ქალაქი, #კასპი, #ურბნისი და **#ოძრაჯე**, და ციხენი მათნი: ციხც
#ოძრაჯისაჲ (1)
320:6 ციხც #კასპისა, #ურბნისსა და **#ოძრაჯისაჲ**. დაუკჳრდა #ალექსანდრეს

Tbl. 2: KWIC-index of names beginning with *O-* from the Šaṭberdi codex

Encoding considerations of the latter type are especially important because with a view to a computational analysis, texts will always have to be clearly divided into units that can serve as a basis for reference. Such divisions can consist of all kinds of reasonable structuring levels such as, e.g., pages, chapters, paragraphs, lines, strophes, verses, sentences, phrases, speakers (in a dramatic play), etc. Normally, only those level divisions can be entered automatically that depend on the structure of the text itself (this holds true, e.g., for lines in a page or pages in a text); this presupposes, though, that the encoding of the text reflects its underlying structure in a unique way as indicated above.

A retrieval software that is to be applied to well-encoded texts of this type must be able to cope with several further problems. First of all, it is expected to be adaptable to the script(s) involved. While it is always possible (and often reasonable) to represent a given script by a substituting transliterative or transcriptive system, there is no reason why nowadays' computers should keep restricted to the plain A-to-Z representation many of us had to work with some years ago when personal computers began to emanate from main frame systems. We should also bear in mind that even transcriptions are not normally restricted to the basic letters of the Latin alphabet, diacritics of all kinds constituting script reservoirs to be managed in their own right. Digraphs such as, e.g., *c$* for Georgian ß / *č* may, if applied in a unique way, be an interim solution for the encoding of foreign characters, but they will not be easily applicable to an investigation into the phonological structure of Georgian texts if kept as such.

Another necessary feature of a text retrieval system is the ability of sorting and arranging, both with respect to structural text elements such as chapters, paragraphs, and the like, and to the linguistic elements contained. In the latter case, this will mostly mean the application of alphabetic orders, depending on the script system used. Within computation, sorting of characters may be an intrinsic feature of the encoding, but normally, this holds true for the Latin A-Z characters only, the alphabetic position of which is reflected by their „byte" value within the so-called „ASCII" code today's computers use. Diacritic combinations such as *ä, ö, ü*, however, will require a special treatment, and for some languages, even di- or trigraphs will have to be accounted for in a special way (cp., e.g., Czech *ch* which has to be sorted as a special character after *h*, or Spanish *ll* as following simple *l*).

While phonological investigations can easily be performed on the basis of the encoding of the text itself, their objects being represented by unique items to be searched for, morphological and syntactical analyses will require additional informations which do not form part of the text proper. Normally, these will consist in some kind of „tagging", i.e., entering grammatical qualifications to specify text elements such as, e.g., verbal forms. The text retrieval system must then be able to distinguish the text elements proper from their tagging, albeit keeping the coherence between them. Without tagging, only those morphological features that have a unique representation in terms of uniquely encoded characters (such as, e.g., certain verbal endings) can be retrieved easily. A special case is the distinction of proper names and other nouns, which is equally important both for linguistic and non-linguistic analyses, and which could be done automatically in an (alphabetical) writing system that uses capital letters for proper names only; but such a writing system seems not to exist, most alphabets using capital letters at least also for sentence-initial words. Note that for Georgian which when written in *mxedruli* script has no capital letters at all, a special marking of proper names is

required in any way (cf. Tbl. 2 where the „number sign“, *#*, is used for this purpose).

A text retrieval problem which applies to all languages that have morphological features such as word inflexion, ablaut and the like, is what might be called the „lemma dilemma“: Word forms that belong together in a paradigm cannot be easily grouped together for searching because their structures are too different formally; cp., e.g., ablauting verbal forms such as German *schreiben* vs. *geschrieben* or even „suppletive“ paradigm items such as English *(I) am* vs. *(you) are* vs. *(he/she/it) is* etc. If these are to be treated as variant forms of one underlying verb, e.g. in establishing a word index, a special tagging will be necessary again.

The same holds true for the opposite case: Word forms that look identical although they do not belong to one lemma. This may be a problem of encoding as is the case with doublets like Russian мука „pain“ vs. мука „flour“ which are identical only graphically: The different accentuation of the two words could easily be accounted for by using a pronunciation-oriented transcription (*múka* vs. *mukáˊ*). But the problem remains with real homonyms such as German *Leiter* „ladder“ vs. *Leiter* „manager“, and whenever searching routines apply in a computational text analysis, it extends even to partial homonymity of word elements such as Georgian *šen* appearing as a personal pronoun „you“, but also as part of the possessive pronoun *šen-i* „your“ or within the verbal form *a-šen-ebs* „he/she constructs (it)“. Here too, tagging will be necessary in order for a retrieval software to be able to distinguish the respective lemmata.

As was stated above, the basis of all kinds of text retrieval is the searching of a certain code or group of codes („strings“) in a given text which consists of a sequence of codes itself. It goes without saying that searching procedures of this type will be the more time-consuming, the longer the text and the more frequent the searched-for string is. It may therefore be reasonable for a text retrieval system not to search sequentially within a given text but to use a preindexation by which all informations as to the positioning of words and other text elements are stored in an extra file. A software product which uses this procedure for all kinds of text retrieval is the „WordCruncher“ program which has been developed by Brigham Young University (Utah)[2]. Let me now evaluate the applicability of this program

[2] The WordCruncher program has existed in several MS-DOS versions since 1985 (last release: 4.6); a first MS-Windows version was released in 1996 (latest release: 5.2β; cf. `http://www.wordcruncher.com`).

with respect to the requirements as summarized above, taking texts from the TITUS text data base[3] as examples.

Preindexation is indeed the most striking feature of the WordCruncher system if compared with text analysis programs that are based on sequential searching. Using preindexation, WordCruncher is able to analyze texts with practically no size limits: The largest file that has been indexed within the TITUS project is the corpus of Old English texts as prepared for the Dictionary of Old English project at the University of Toronto. As a plain text file, it has about 35 MB; after its preindexation, all occurrences of any word as appearing in it will be found within less than a second[4].

To use this feature, the text has to be prepared in a structured way, meeting the requirements of uniqueness as indicated above. This holds true, first of all, for the division of text elements that will be used for reference. In the present (Windows) version of the WordCruncher program, up to 10 levels can be assigned for this purpose, matching, e.g., pages and lines of a book edition or chapters, paragraphs and sentences of a book. Cf. Tbl. 3 where four such level markings consisting of the code | plus a certain letter (|A for „documents“, |b for „books“, |P for „pages“, |l for „lines“) are represented in one of the texts from the Šatberdi codex; Fig. 1 shows the dialog box for the definition of levels. The number of levels has been significantly increased, by the way, as against former (MS-DOS) versions of the program where only three levels could be assigned.

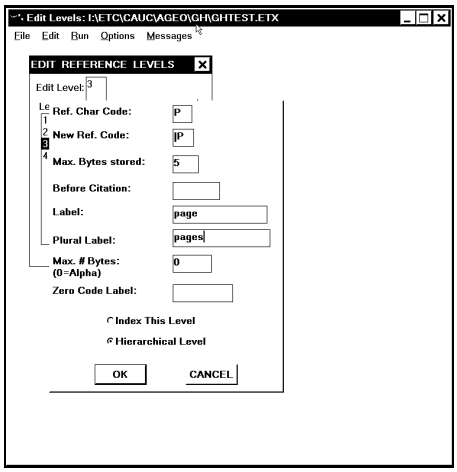


Fig. 1: Definition of text levels

[3] The TITUS project („Thesaurus Indogermanischer Text- und Sprachmaterialien“) which I have been running since 1986 aims at establishing a data base of all texts that are relevant for Indo-European studies, i.e. texts from ancient Indo-European languages (Old Indic, Hittite, Italic etc.) as well as neighbouring languages (e.g., Old Georgian). For its present state cf. `http://titus.uni-frankfurt.de/texte/texte.htm`.

[4] Practically, the size limit of a text file is 2 GB, depending on the limits of the DOS/Windows operating system. The maximum number of unique word forms in a text is limited to 2^{24} = 16,777,216.

```
SIF=\etc\tituscx.sif

▸Pcenter◂▸Ttitle◂Textus homiletici et exegetici▸Tn16◂
|ACod.Satb.
▸Tsubtitle◂e Codice Satberdiensi▸Tn16◂

▸Twl22◂Gregorius Nyssenus, De hominis opificio▸Tn16◂


▸Pnormal◂|bGreg.Nyss.Buneb.
|P67
|l1   ▸Tcgei16◂tkumuli ċmidisa da neṫarisa mamisa [!] čuenisa Grigoli▸Tn16◂
|l2   ▸Tcgei16◂Nosel ebiskoṗosisay ḳacisa šesakmisatws, romeli▸Tn16◂
|l3   ▸Tcgei16◂miuċera ʒmasa twssa Ṗeṫres ebiskoṗossa sabaṡielsa*▸Tn16◂
|l4   ▸Tcgei16◂(1) uḳuetu-mca ǯer-iq̇o didebay satnovebis-momgebeltay ṗaṫivita▸Tn16◂
|l5   ▸Tcgei16◂monagebtayta, iṗovnes-mca q̇ovelni sikadulni sapasetani undo, ražams-mca▸Tn16◂
|l6   ▸Tcgei16◂ševaṫq̇ut satnovebata šenta tkumulisa misebr Solomon▸Tn16◂
|l7   ▸Tcgei16◂brʒnisa, rametu uzeštaes ṗaṫivisa mis sapasetaysa ars madli igi ġirsi,▸Tn16◂
|l8   ▸Tcgei16◂šenda. da aċ maċuves čuen dġesasċauli. ċmidisa da didebulisa aġvsebisay▸Tn16◂
|l9   ▸Tcgei16◂čueulebisaebr ṗirvelisa moċq̇alebasa megobrebay šeni da čuen▸Tn16◂
|l10  ▸Tcgei16◂ċina-uq̇opt didebulebasa šensa. ō ḳaco ġmrtisao, ʒġuensa ḳninsa da▸Tn16◂
|l11  ▸Tcgei16◂vitar-igi šenda ġirs-ars, xolo ara tu uḳninessa ʒalisa čuenisa[sa]. da▸Tn16◂
|l12  ▸Tcgei16◂ese ʒġueni tkumul ars vitarca samoseli šeuracxi moksovili gonebisagan▸Tn16◂
|l13  ▸Tcgei16◂glaxaḳisa šromita. da mizezsa amis tkumulisasa vhgoneb tu▸Tn16◂
```

Tbl. 3: Preparation of text file for WordCruncher (Windows)

Except for the level marking, the texts to be prepared for preindexation require certain additional informations as to structuring and formatting. While the basic structure of the texts is a plain DOS text format (i.e., only line feed codes, spaces and tabulators appear besides plain characters), several formatting prescriptions can be entered as „tags“, consisting of a begin and an end character, a letter defining the class of the tag, and the name of the tag. In the given example (Tbl. 3), various tags of the type ▸X(name)◂ are represented; ▸Pcenter◂, e.g., marks the beginning of a paragraph („P“) structure with line centering, and ▸Ttitle◂ marks the beginning of a text passage that uses a „text style“ („T“) adaptable to „titles“. The meaning of each tag has to be defined before preindexation, and the actual definitions are not only responsible for the indexing but also for the appearance of the text on the screen during retrieval; cp. Fig. 2 and Fig. 3 where the same text is displayed in two ways, depending on different assignments of fonts to the text structure tags. The tag definitions are stored in a so-called „SIF file“ („Style include file“) which must be referred to in the first line of the text to

be indexed; cf. Tbl. 4 showing the structure of the definitions mentioned above as appearing in the SIF file.

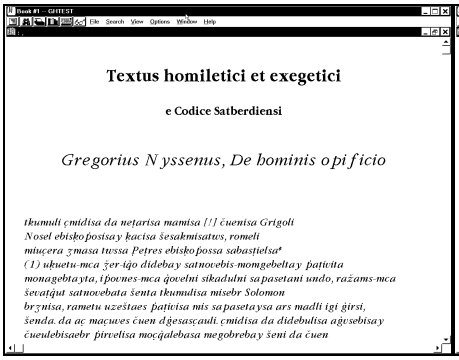


Fig. 2: Main text window

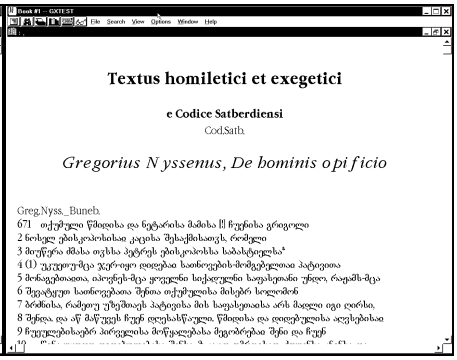


Fig. 3: Same, with *mxedruli* font

Another important element of the preparation consists of establishing sorting sequences for the linguistic elements. This can either mean standardized alphabetic orders or special applications by which certain character or delimiter functions are assigned to the letters in a given font. Using the dialog box illustrated in Fig. 4, characters can be assigned an equal value (e.g., *A* and *a* and *ä*; this is true even for pairs of characters such as *ae* vs. *ä*), parentheses can be marked as

```
BC=‣
EC=◂
DOCUMENT_STYLE
  LINE_LENGTH inches 11.0
  LEFT_MARGIN inches 0.1
  RIGHT_MARGIN inches 0.1
  INDEX_OFF superscript subscript
  END
PARAGRAPH_STYLE normal
  RULER normal
  JUSTIFICATION left
  END
PARAGRAPH_STYLE center
  RULER center
  JUSTIFICATION center
  END
...
```

```
...
TEXT_STYLE title
  FONT orient
  CHARACTER_STYLE bold
  FONT_POINTSIZE 24
  TEXT_COLOR [0 0 128][255 255 255]
  END
TEXT_STYLE cgei16
  FONT georgian-trs.
  CHARACTER_STYLE italic
  INDEX_FLAG on
  FONT_POINTSIZE 15
  TEXT_COLOR [255 0 0][255 255 255]
  END
...
```

Tbl. 4: Definition of formatting elements in a WordCruncher „SIF“ file

ignorable so that, e.g., Old Georgian abbreviated რ[ომელმა]ნ *r[omelma]n* „which (rel.pron.)“ and non-abbreviated რომელმან *romelman* are treated as equivalents, and so on.

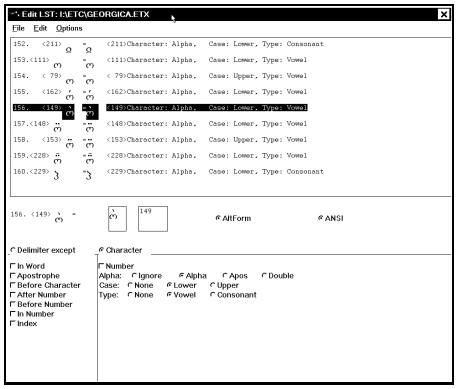


Fig. 4: Definition of character sequence

When a text has been successfully preindexed, it is ready for retrieval with the so-called „search engine“ which searches not the text but its index. Its central element is a so-called „word wheel“ which is nothing but an alphabetical list of the words contained in the text. It can be used for searches of single word forms such as Old Georgian არს *ars* „(he) is“ (cf. ?) or combined searches of word forms in certain contextual environments such as თქუმულ *tkumul* „spoken“ + არს *ars* „is“ forming an „exact phrase“ sequence (cf. ?); it can also be used for so-called „wild card“ and „substring“ searches by filtering the word wheel with respect to certain phonotactic or morphotactic conditions (cf. Fig. 7 and Fig. 8 where word forms containing *eb* inside and *ay* at the end are searched for). For the sake of lexical investigations, word forms that belong to one lemma can be grouped together (cf. Fig. 9 where the case forms of კაცი *ḳaci* „man“ are collected), and such lists can be saved for further usage (cf. Fig. 10). It goes without saying that using the word wheel for a language such as Georgian requires a keyboard interface that matches the font the linguistic elements are represented with.

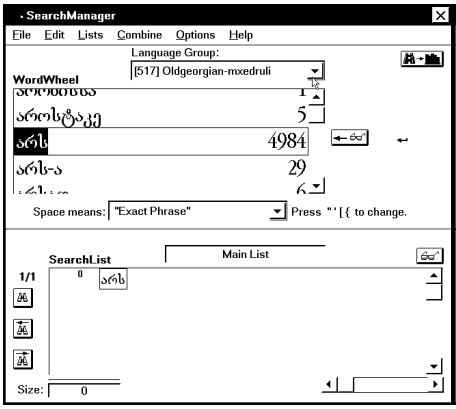
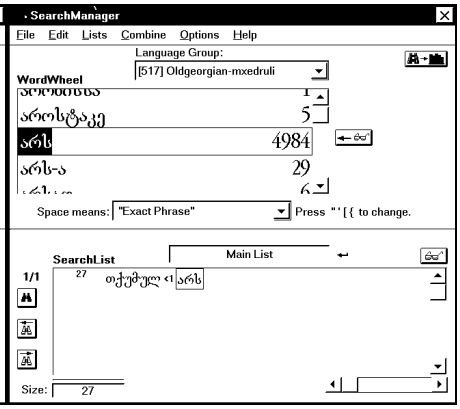


Fig. 5: „Wild card“ search



> **Fig. 6**: Filtered „word wheel“

This can be easily defined by integrating a „character map“ referring to the Windows-ANSI standard in the SIF file (cf. Tbl. 5 showing the definition table and Fig. 11 showing the help screen representation of the keyboard assignment).

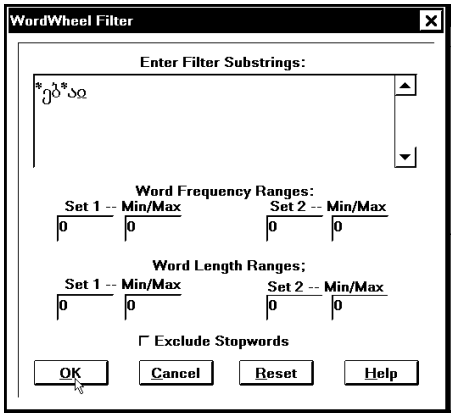
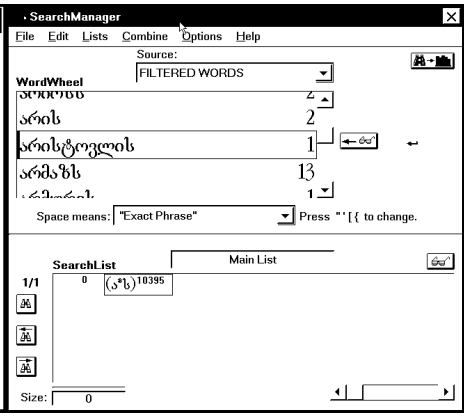


Fig. 7: Substring search



> **Fig. 8**: Filtered word wheel

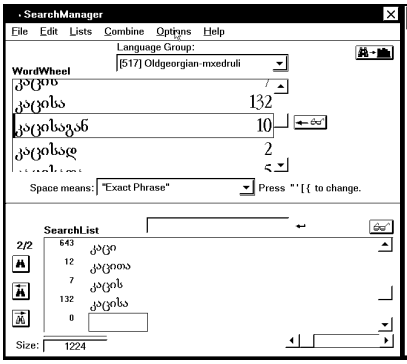


Fig. 9: Related word forms collected

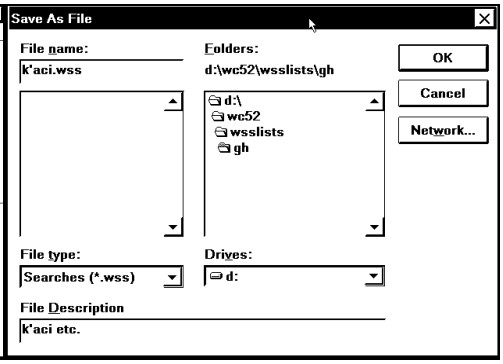


Fig. 10: Save list function

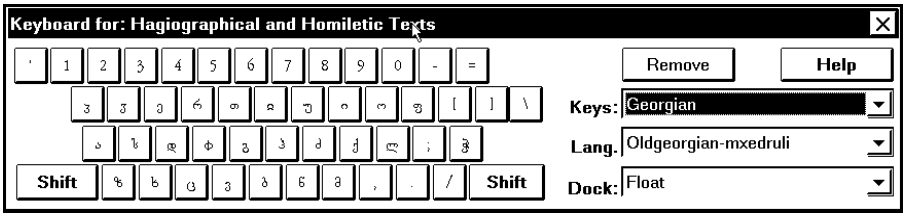


Fig. 11: Keyboard definition as usable in the search engine

```
CHARACTER_MAP Georgian
  MAPS 61=61 63=63 96=96 65=192 83=234 68=68 70=70 71=204 72=208 74=74 75=214
  MAPS 76=217 80=229 220=246 42=42 89=89 88=88 67=194 86=86 66=66 78=78 77=77
  MAPS 59=59 58=58 95=95 94=196 35=35 124=197 39=39 97=97 115=115 100=100 102=102
  MAPS 103=103 104=104 99=99 118=118 98=98 110=110 109=109 44=44 46=46 45=45
  MAPS 123=92 91=123 93=125 106=106 107=107 108=108 246=148 228=132 113=113
  MAPS 119=119 101=101 114=114 116=116 122=122 117=117 105=105 111=111 112=112
  MAPS 252=129 43=43 121=121 120=120 125=182 92=184 64=231 126=93 181=219 124=91
  MAPS 131=159 225=160 237=161 243=162 228=132 224=133 229=134 231=135 234=136
  MAPS 235=137 232=138 239=139 238=140
  END
```

Tbl. 5: Character map table definition within „SIF“-file (extract)

Using the word wheel for searching, the contexts of the word form(s) searched for will immediately be presented in a reference list showing the occurrences with their contexts grouped for scrolling (cf. Fig. 12 for four occurrences of case forms of კაცი *kaci* „man“). For single word forms, the reference list can also be invoked using the internal hyperlinking structures of the preindexed text file, viz. by double clicking on a certain word form in the main text window. The reference list can further be sorted according to the contextual structures of the occurrences (cf. Fig. 12), and a „text out“ function is available for storing its contents (cf. Fig. 13), e.g. with a view to preparing a printed index.

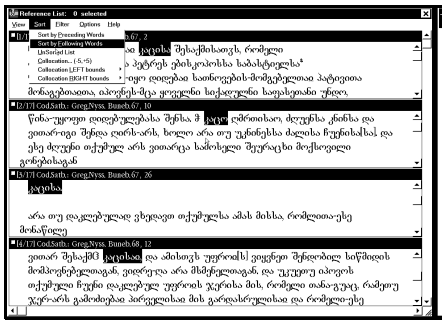


Fig. 12: Sorting of reference list

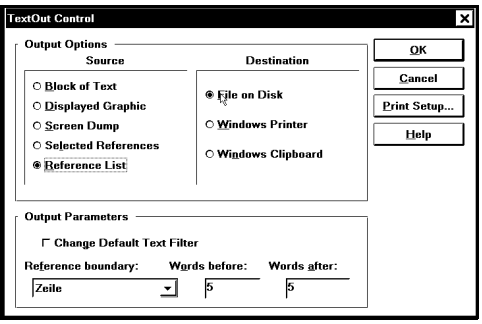


Fig. 13: Text-out function

Another feature of the program that is worth mentioning consists in the embedding of graphics. It can be foreseen that a new epoch of philological work will constitute itself soon when digitized images of manuscripts are readily available together with the philologist’s interpretation in one computer screen, as in Fig. 14 showing a page from the so-called „Sinai lectionary“ (VIIth century) as an

example. With the WordCruncher system, hyperlinking of this type is not restricted to graphics, though, but can also be applied to parallel texts etc.

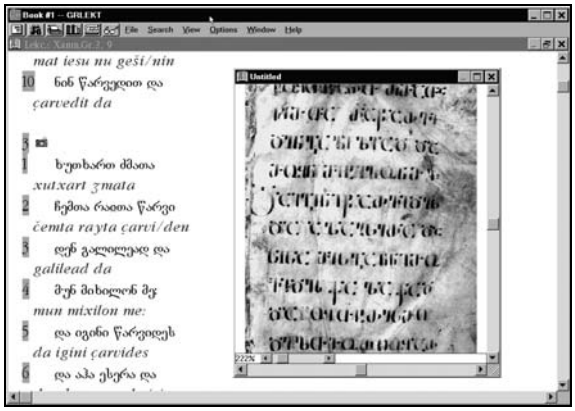


Fig. 14: Text with graphics hyperlink

All in all, the WordCruncher system in its present state of development seems to match the requirements of a scholarly text retrieval to a large extent. This is why it has been chosen as the primary retrieval tool for the TITUS project[5]. There are still some shortcomings, however, that require further improvement: For the time being, morphological tagging is not an intrinsic feature of the program, and it can only provisionally be used by entering the necessary informations as normal text elements as can be seen in Fig. 15 where the taggings are marked by a black background. And with the present version, the generation of a complete „KWIC“ concordance of a given text is not possible albeit this was an outstanding feature of earlier (MS-DOS) releases[6].

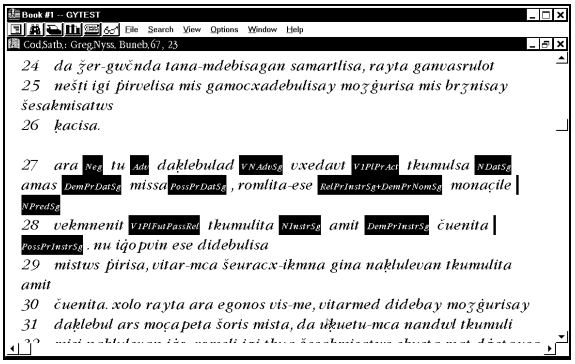


Fig. 15: Morphological tagging (provisional method)

[5] A WordCruncher server has meanwhile been installed which allows for immediate internet retrieval of many of the texts that are incorporated in the TITUS data base; cf. `http://titus.uni-frankfurt.de/texte/tituswc.htm` for details.

[6] The largest index I produced with WordCruncher for DOS was a complete word concordance of the works of the Greek medical author, Galenos, consisting of a text file of 17 MB. This has recently been published in an abridged form (Index Galenicus, compiled by Jost Gippert, Verlag J.H.Röll, Dettelbach 1997; cf. also `http://titus.uni-frankfurt.de/lexica/galeninx.htm`).

Let us now turn to the specific aims and requirements of a multilingual text retrieval. When speaking of multilingual texts, there are at least two cases that have to be distinguished. One is texts that are constituted by multilingual elements in that they contain quotations from foreign languages within them; cp., e.g., Fig. 16 which shows an Old Maldivian document containing Arabic words. The special task of a retrieval software to be applied to such texts consists of the separation and proper administration of the elements belonging to each language. This requires a means of encoding language boundaries within the texts (i.e., tags or delimiters) as well as handling of different scripts, script directions, etc. (note that in the given example, the Brahmi-type Old-Maldivian script runs from left to right while the embedded Arabic passages are written from right to left).

Fig. 16: Maldivian document containing Arabic words

The second case to be dealt with when speaking about multilingual text retrieval concerns texts in different languages that are interrelated with each other in a certain way, e.g. in that one of them is translated from the other (or both represent translations from a third one etc.); this case is typically represented by Bible translations. Another case to be mentioned here is texts that refer to the same contents (e.g., historical data), but more or less independently from another; this is typical, e.g., for the relationship between chronicles and eyewitness reports. In all these cases, a special task of text retrieval consists in establishing the interdependencies that may exist between certain linguistic elements such as names, translational word pairs, syntactical units and the like[7]. For this purpose, a special method of marking will be required that allows for a „synchronous“ administration of textual levels (e.g., chapter structure, sentence structure) as well as single words or word forms. This can either be done in a unified text structure (e.g. with „parallel“ texts formatted in columns) or with separate text „windows“ synchronized externally.

[7] For a thorough discussion of the problems involved cf. J. Gippert, Towards an automatical analysis of a translated text and its original: The Persian epic of *Vīs u Rāmīn* and the Georgian *Visramiani*, in: Studia Iranica, Mesopotamica et Anatolica 1, 1994 [1995], pp. 21-60.

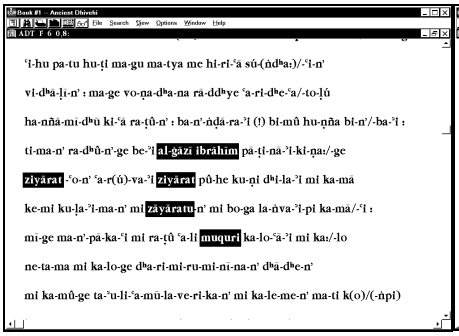


Fig. 17: Maldivian/Arabic mixed text

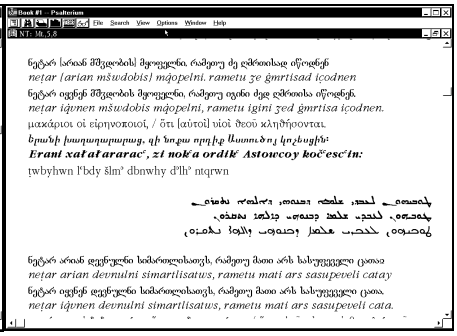


Fig. 18: Multilingual NT arrangement

Evaluating the WordCruncher system with a view to these requirements, we may state again that it offers a good deal of practicable solutions. The most important feature consists in the „text style“ function as referred to above which allows not only for a differentiation of fonts, i.e., scripts, but also of languages to be kept separate during preindexation and retrieval. This extends both to the usage of different transcription systems (cf. Fig. 17 where the Old Maldivian document treated above is handled in this way) and to the adaptation of original scripts (cf. Fig. 18 where several Bible versions are contrasted, including Syriac written from right to left as well as additional transcriptions[8]).

The language differentiation thus produced is reflected both in the hyperlinking structure of the text (so that double clicking of a word form will invoke its „search“ in the appropriate environment only, cf. Fig. 19) and in the search engine. Here, the user can choose the appropriate language

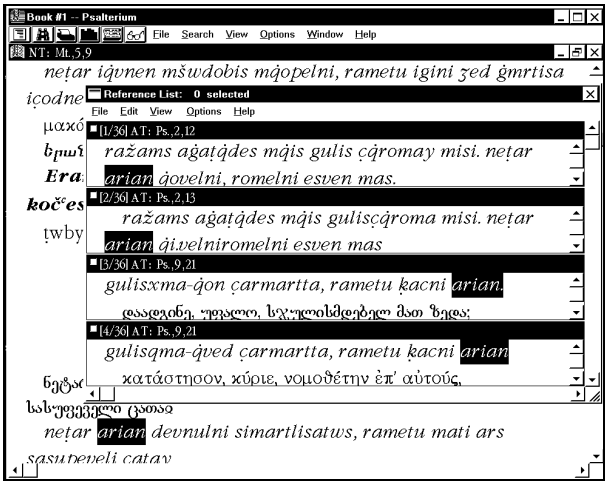


Fig. 19: Language-specific search

[8] For testing purposes only, Syriac has been represented in three scripts, Estrangelo, Nestorian, and Serto, in the file.

from a scrolling box, thus opening the specific word wheel with its keyboard settings etc. (cf. Fig. 20 where Greek is used as an example). This function can further be applied to combined searches across language boundaries as illustrated in Fig. 21 which shows the words meaning „gold“, Georgian ოქროსა *okrosa* (dat.), Greek χρυσόν and Syriac *dhb*, searched for as appearing within the same verse.

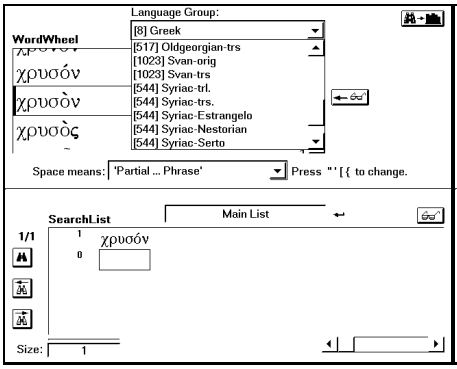


Fig. 20: Language choice in search engine

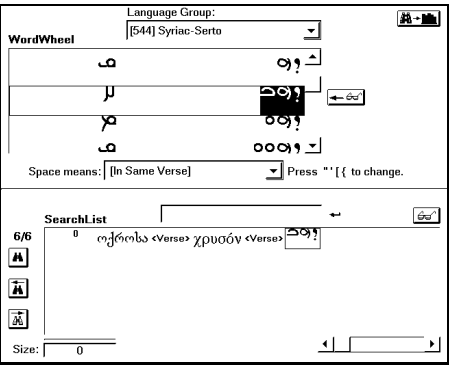


Fig. 21: Multilanguage search

A search of the indicated type requires a unified text structure, i.e., the several versions have to be arranged in a verse by verse alignment within one text file. It should be noted then that the same structure can also be applied to a text existing in several versions in one and the same language. This is demonstrated in Fig. 22 where various witnesses of the legend about the conversion of Georgia to christianity by St. Nino are contrasted. Here, the list of „languages“ covers not only Armenian (as represented by the *Patmowtʿiwn hayocʿ*, a 13th century translation of the Georgian chronicle) but also abbreviations such as

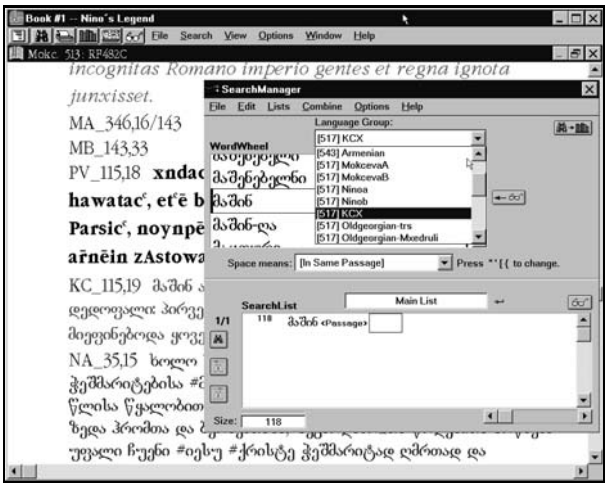


Fig. 22: Various sources of St. Nino's legend

KCX or MokcevaA referring to different Old Georgian texts (*Kartlis cxovreba*, i.e. the Georgian chronicle, and the older redaction of *Mokcevay kartlisay*, i.e., the Šaṭberdi version of the legend).

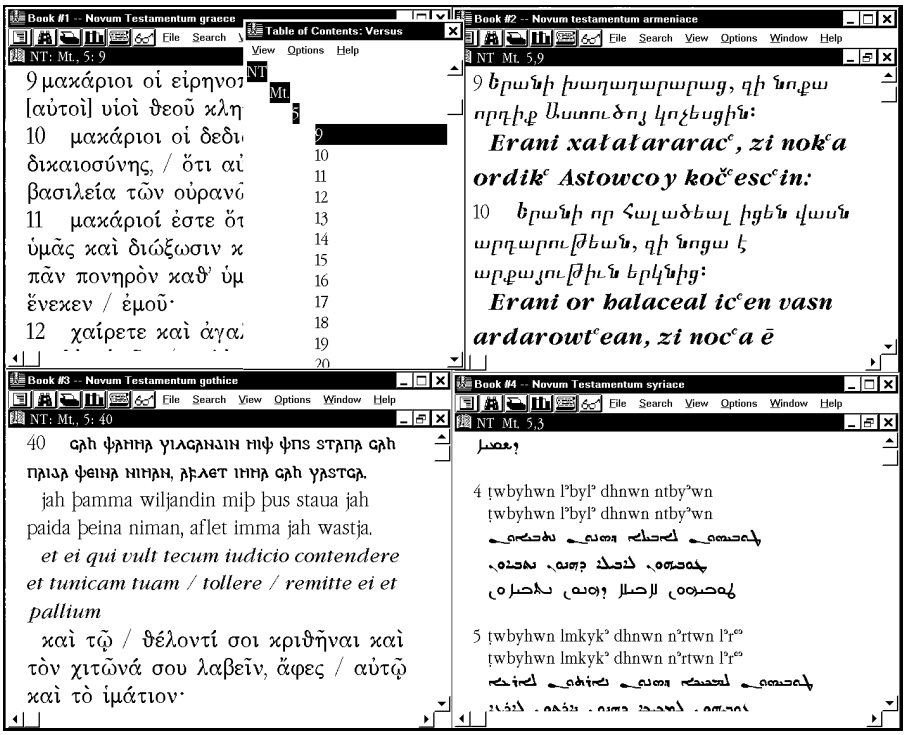


Fig. 23: Four Bible versions synchronized

Another way of establishing interdependencies between related texts consists in external „synchronizing“ which can be applied after indexation only; cf. Fig. 23 where a synchronized arrangement of four distinct Bible versions (Greek, Armenian, Gothic, and Syriac) is taken yas an example. This procedure has obvious advantages as far as the screen appearance is concerned; linkage remains limited, however, to the levels of text structure (in the given example, Mt. 5,9) and cannot be extended to linguistic elements.

As was stated above, the main feature which controls the language functions within the WordCruncher system is the text style management. In reality, there are three types of interdependent definitions that have to be provided in a SIF file when languages (or „quasi-languages“ such as KCX) are to be kept separately: the

TEXT_STYLE definition (cf. Tbl. 6) refers to a FONT declaration (cf. Tbl. 7) which, in its turn, is connected with a LANGUAGE definition (cf. Tbl. 8). Although this system has proved quite successful within the TITUS project (the amount of languages to be definable within one SIF file is 30 which will be enough for most cases), it has at least one major disadvantage: It does not enable a truly unique encoding of separate languages (and scripts to be applied to them) because it implies a so-called font mapping on an 8-bit encoding basis (DOS-ASCII or Windows-ANSI). This means that one and the same byte value may represent different characters in different contexts such as, e.g., a Latin *a*, a Greek α, a

```
TEXT_STYLE mxge12
  FONT georgian-mroveli
  INDEX_FLAG on
  FONT_POINTSIZE 12
  TEXT_COLOR [0 128 0][255 255 255]
  END
TEXT_STYLE mxge16
  FONT georgian-mxedruli
  INDEX_FLAG on
  FONT_POINTSIZE 16
  TEXT_COLOR [128 0 0][255 255 255]
  END
TEXT_STYLE mxge22
  FONT georgian-mxedruli
  INDEX_FLAG on
  FONT_POINTSIZE 24
  TEXT_COLOR [128 0 0][255 255 255]
  END
...
```

Tbl. 6: TEXT STYLE definitions >>>

```
FONT georgian-mxedruli
  FONT_NAME TITUS-Mxedruli
  FONT_FAMILY roman
  CHAR_SET ansi
  PITCH proportional
  DIRECTION left-to-right
  FONT_TYPE TrueType
  LANGUAGE Oldgeorgian-Mxedruli
  END
FONT georgian-trs.
  FONT_NAME TITUS-ChristianEast
  FONT_FAMILY roman
  CHAR_SET ansi
  PITCH proportional
  DIRECTION left-to-right
  FONT_TYPE TrueType
  LANGUAGE Oldgeorgian-trs
  END
...
```

Tbl. 7: FONT declarations >>>

```
LANGUAGE Oldgeorgian-Mxedruli
  LANGUAGE_ID GEORGIAN
  CHARACTER_MAP Georgian
  TEXT_STYLE mxge16
  LST_FILENAME\ETC\GEORGICA.ETX
  END
LANGUAGE Oldgeorgian-Xucuri
  LANGUAGE_ID GEORGIAN
  CHARACTER_MAP Georgian
  TEXT_STYLE axge16
  LST_FILENAME\ETC\GEORGICA.ETX
  END
LANGUAGE Oldgeorgian-trs
  LANGUAGE_ID GEORGIAN
  CHARACTER_MAP Georgian
  TEXT_STYLE cgei16
  LST_FILENAME\ETC\GEORGICA.ETX
  END
...
```

Tbl. 8: LANGUAGE definitions

▸**Pnormal**◂|p9
▸**Tdge16**◂ozia Ωva ioatam. ioatam Ωva akaz. akaz Ωva eze╔ia.▸**Tn16**◂
▸**Tcgei16**◂ozia Ωva ioatam. ioatam Ωva akaz. akaz Ωva eze╔ia.▸**Tn16**◂
▸**Tmge16**◂ozia Ωva ioatam; ioatam Ωva akaz; akaζ Ωva eze╔ia.▸**Tn16**◂
▸**Tcgei16**◂ozia Ωva ioatam; ioatam Ωva akaz; akaζ Ωva eze╔ia.▸**Tn16**◂
▸**Tgr16**◂äO▄í╫@ ┌è / ┴┴ éΣΣ▌ Θ▐Σ ΩòΣ äI∩╫ ▐π, äI∩╫ ▐àπ ┌è ┴┴ éΣΣ▌ Θ▐Σ ΩòΣ äAϕ▄, / äAϕà▄ ┌è ┴┴ éΣΣ▌ Θ▐Σ ΩòΣ ÄE▄▐αí╫Σ,▸**Tn16**◂
▸**Thy16**◂Ozia cnaw zyova±am: Yova±am cnaw za╫az: A╫az cnaw zezekiay:▸**Tn16**◂
▸**Thyti16**◂Ozia cnaw zyova±am: Yova±am cnaw za╫az: A╫az cnaw zezekiay:▸**Tn16**◂
▸**Tsk16**◂%wzy& &wld lywtm ywtm &wld l&╨z &╨z &wld l╨zqy&▸**Tn16**◂

▸**Pright**◂▸**Tse16**◂‘8zy" ^wjd j08TM y8TM ^wjd j"x7 ^x7 ^wjd j+7q0"▸**Tn16**◂
▸**Tsn16**◂‘8zy$ ^wjd j08TM y8TM ^wjd j$x7 ^x7 ^wjd j+7q0$▸**Tn16**◂
▸**Tss16**◂‘8zy" ^wjd j08TM y8TM ^wjd 1’7 ^’7 ^wjd j+7q0"▸**Tn16**◂

Tbl. 9: Passage from Matthew in DOS/ASCII representation

Georgian ა, an Armenian ա, etc., all being represented by a byte value of 97; this effect can be seen in Tbl. 9 where the Bible text passages as quoted above are rendered in plain DOS/ASCII format (text style markups represented in bold characters). The disadvantage will mainly be noted when multilingual texts thus produced are to be converted into different formats or when multilingual parts of a text are to be extracted: At present, the font information necessary for the distinction cannot be exported as such and has to be added manually again.

As this is a general problem of 8-bit based systems, a solution may be expected to emerge from the development and application of Unicode as the first OS-independent 16-bit encoding standard. Unfortunately, the WordCruncher system has no Unicode interface yet; the applicability of Unicode encoding to multilingual texts can, however, be demonstrated even now using a web browser such as the Netscape Communicator which (starting from release 4.0) allows for a handling of texts that are encoded in the so-called UTF-8 (8-bit Unicode translation) format. After installing an appropriate font, a multilingual text such as a synoptical arrangement of the various sources of St. Nino’s legend can well be displayed (and printed) on the basis of MS-Windows 95, if formatted as a web

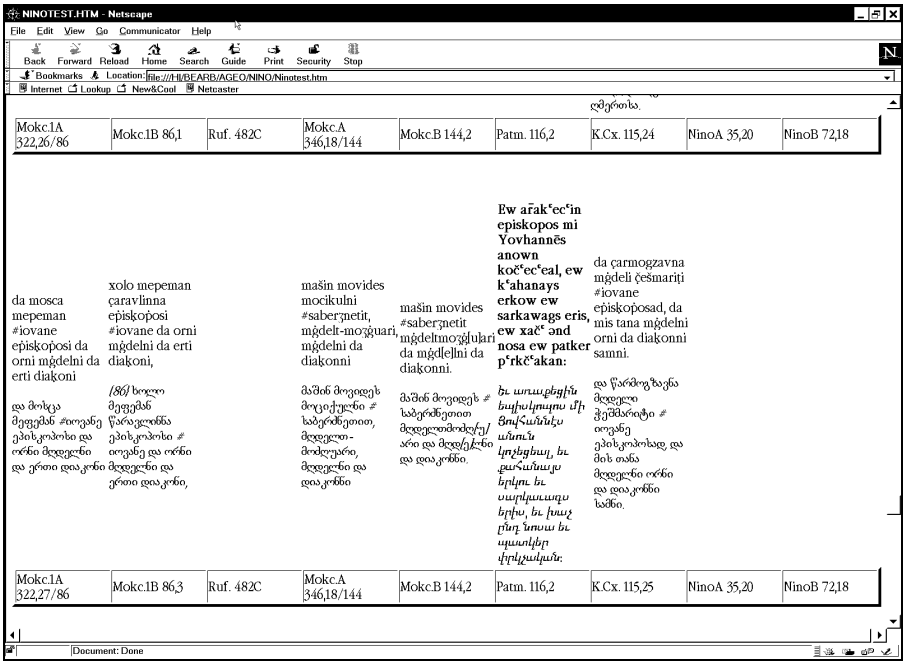


Fig. 24: Several sources of St. Nino's legend in Unicode (UTF-8) encoding

page; cf. Fig. 24 for an example[9]. It goes without saying, though, that this is hardly sufficient for a text retrieval of the type discussed here. And even with respect to future developments, it remains doubtful whether Unicode can be an adequate basis for this: As an encoding system, it is based on scripts, not on languages, and the problem of distinguishing homographs such as German *hier* „here" and French *hier* „yesterday" or French *haut* „high" vs. German *haut* „strikes" (vs. German *Haut* „skin") is not solved by it in any way. A tagging system for the distinction of languages will therefore remain necessary even when Unicode is applied.

[9] This page is available, together with other materials concerning Unicode, under `http://titus.uni-frankfurt.de/unicode/unitest.htm`. It may be interesting to note that using Unicode requires the „multilanguage" function of Windows 95 to be installed and that it will not work with a 80486 (or lower) processor.