Ingemar D a h l s t r a n d

DATA PROCESSING IN THE JUBA PROJECT

TABLE OF CONTENTS

Slavica Lundensia 9, 1983, pp. 107-123

1. INTRODUCTION

The data processing for the JUBA project started in the summer of 1981 with three simple programs LIST, FREQ and KONK. These programs produced a printout of a child's text, a frequency count of its words, and a concordance, respectively. In September 1981 we had a week's course at the Institute for Computer Linguistics in Uppsala, led by its director, assoc. prof. Anna-Lena Sågvall. This gave us very valuable background and ideas for coming work.

In the course of the winter of 1981/82 central ideas for grammatical analysis were formed. A complete dictionary of endings in the Serbo-Croat language was built and so was the beginning of a stem dictionary. Programs were made for building entries in the stem dictionary and for carrying out the analysis of an arbitrary Serbo-Croat word. In the meantime the stock of readied children's texts grew gradually. By early summer 1982 we were ready to carry out lemmatization of text with manual resolution of homonyms. First experiments were made with analysis spanning more than one word, i.e. identification of verb forms consisting of the L form of the base verb (e.g. "pisao", "pisala") and various tenses of "biti" (to be).

The linguists of the department had now started to use the programs in daily work. This gave us new experience and forced a development of easy-to-use interfaces and more safety measures in the programs. In the course of the summer and autumn the stem dictionary grew from a few hundred to a thousand words and the number of texts to 130. Programs to aid the entry of social data were made, and when the social data had been entered, they were analyzed using the statistical program package SPSS.

2. ORGANIZATION OF DICTIONARIES AND TEXTS

2.1. Organization of the Dictionaries

The stem and ending dictionaries (STLX and ENLX) and the programs working on these are the centerpiece of the JUBA software and essential for the understanding of the whole work. Their organization and function will therefore now be described in some detail.

Serbo-Croat is a language with a comparatively rich morphology. Especially the verbs have many tenses; each of these in turn is conjugated by number and either person or gender. In simple cases

these endings or groups of endings are attached to the same stem, but even in regular verbs with vowel stems there is an interaction between ending and stem that gives rise to variations. Many verbs with consonant stems have more or less systematic stem alternations. Nouns have fewer endings but on the other hand there is a lot of duplication: many nouns allow two different forms of the vocative or instrumental and two plurals.

After intense discussions in the project group, a scheme was developed that would cover all this variety. We worked through the system of endings and identified ending groups. An ending group consists of endings that appear together, and are attached to the same stem. For instance, the following endings represent one of the four main ways of forming the present tense:

-im	singular	1. person
-iš	singular	2. person
-i	singular	3. person
-imo	plural	1. person
-ite	plural	2. person
-e	plural	3. person

Attached to the stem "rad-", they form the present tense of "raditi" (to work): "radim, radiš, ... rade"..

A certain ending may well belong to more than one ending group. Thus, the ending "-e" belongs to several ending groups in the present and the aorist. It is, for instance, part of the ending group:

-em	sg.1
-eš	sg.2
-e	sg.3
-emo	pl.1
-ete	pl.2
-u	pl.3

These endings may be attached to the stem "rast-" of the verb "rasti" (to grow), giving: "rastem, rasteš, raste", etc. As we see, the "-e" happens to have a different meaning in this group, forming the 3. person singular, rather than 3. person plural as it did with "rad-".

Primarily, we try to form ending groups that form one complete

tense. In some cases, it is however more natural to form smaller groups that can be combined with different stems to give a complete tense. For instance, the six endings "-em, -eš, ... -u" just mentioned do also appear as two subgroups, the one consisting of the five endings "-em, -eš, ... -ete", and the other subgroup consisting of "-u" alone. When conjugating the verb "reći" (to speak), the first subgroup is attached to the stem "reč-", and the second to the stem "rek-", giving the complete present tense: "rečem, rečeš, reče, rečemo, rečete, reku".

The ending groups are each given a short, unique name or identification number. In the case of the verbs these are numbers of two or three digits. The first digit identifies the tense or mode. The second digit is the number of the ending group within that tense. The third digit, if present, is the number of the subgroup. Thus,. the ending groups shown above for the present tense of "raditi" and "rasti" are called 42 and 41, respectively. The subgroups used to conjugate "reći" are called 411 and 412.

In the case of nouns, most ending groups have only one member. These have been named systematically. A typical name of a noun ending group is "vse", meaning 'vocative, singular, ending "-e"'.

Having established these concepts, we are now ready to understand the exact contents of the dictionaries.

2.1.1. The ending dictionary (ENLX)

Each occurrence of an ending in an ending group gives rise to one entry in the ending dictionary. The entry consists of the following fields:

- The ending itself
- The number of the analysis class, period
- The name of the group
- The meaning of the ending, when occurring in this group, in an abbreviated notation

(An analysis class corresponds roughly to the traditional word class of grammar, but is based on more formal criteria. For our work, we divided Serbo-Croat words into the following analysis classes: (1) nouns, (2) adjectives, (3) pronouns, (4) numerals, (7) verbs, (8) inflexible words.)

Here is a listing of part of the ending dictionary, namely all verb endings beginning with an "e". I have marked with an asterisk

those endings discussed in the examples above.

e	7.32	ao.sg.2	
e	7.322	ao.sg.2	
e	7.32	ao.sg.3	
e	7.322	ao.sg.3	
e	7.42	ps.pl.3	*
e	7.44	ps.pl.3	
e	7.41	ps.sg.3	*
e	7.411	ps.sg.3	*
e	7.44	ps.sg.3	
em	7.41	ps.sg.1	*
em	7.411	ps.sg.1	*
emo	7.41	ps.pl.1	*
emo	7.411	ps.pl.1	*
emo	7.44	ps.pl.1	
en	7.81	pf.part.pass	
ete	7.41	ps.pl.2	*
ete	7.411	ps.pl.2	*
ete	7.44	ps.pl.2	
eš	7.41	ps.sg.2	*
eš	7.411	ps.sg.2	*
eš	7.44	ps.sg.2	
ći	7.62	ps.ger	

2.1.2. The stem dictionary (STLX)

Each entry in a stem dictionary consists of the following fields:
- The stem allomorph itself
- The analysis class of the word
- Two characters used to identify special properties of the word. For nouns: gender and animation, for verbs: aspect and reflexivity, for inflexible words: grammatical word class.
- The lemma to which the stem belongs (for verbs: the infinitive)
- A list of names of ending groups that may be attached to this allomorph. If the list gets too long, it may be split over two entries.

We have tried to define the ending groups in such a manner that most lemmas will only need one entry (allomorph) in the stem dictionary. Several entries are needed if there are allomorphs of the stem, if the flexion is irregular, if there are multiple flexions to the lemma, etc. These multiple entries will be held together by the lemma they have in common. They do not have to be adjacent in the dictionary, which is organized strictly alphabeti-

cally by stems.

Here is an example from the stem dictionary, listing those verbs beginning with an "r", that we have discussed above. You will find the group names 42, 41, 411, and 412 in the list after their respective allomorphs, again marked with asterisks. In addition, you will note that "reći" has a duplicate present tense of type 41, formed from the stem "rekn-": "reknem", "rekneš", "rekne", etc.

```
ra          7 i0 rasti      12                                  92
rad         7 i0 raditi                  42* 52 62
radi        7 i0 raditi     11  21  31              71      91
ras         7 i0 rasti          212
rast        7 i0 rasti          221 32  41* 52 61      81      q1 q2
rad         7 i0 raditi                                81      q1
re          7 p0 reći       13
rec         7 p0 reći                       52
rek         7 p0 reći       221 212 321 412*    61 72
rekn        7 p0 reći                   41
reč         7 p0 reći               322 411*           81      q1
```

So far, we have discussed mainly verbs, and from the examples you might believe the stem and ending dictionaries contained verbs only. In fact there are separate stem and ending dictionaries for verbs, nouns and adjectives, all built on the same principles. For grammatical analysis of an arbitrary word, these are merged into two common dictionaries. The common ending dictionary has 255 entries: 55, 43 and 157 entries for nouns, adjectives and verbs, respectively. The stem dictionary by now has over 1000 entries and shall keep on growing as far as necessary for an exhaustive analysis of the texts.

2.1.3. The word form dictionary (OFLX)

Words from the remaining analysis classes are stored in the word form dictionary, for reasons to be discussed below. Here each possible word form is enumerated with an entry consisting of:

- The word form
- The occurrence number of this form (to distinguish homonyms)
- The meaning of the word form in this lemma, abbreviated in the same way as in the ending dictionary
- The analysis class

- The two special information characters
- The lemma

Multiple meanings give rise to different entries.
Here is an excerpt, including three homonyms of "je", which can be the 3. person singular present of "biti" ("is"), or the genitive or accusative of "ona" ("her"):

```
ja          1 nom         3 .. ja
jao         1 -           8 ij jao
je          1 ps.sg.3     7 i0 biti
je          2 gen         3 .. ona(f)
je          3 ack         3 .. ona(f)
jedanput    1 -           8 .. jedanput
jer         1 -           8 kj jer
jesam       1 ps.sg.1     7 i0 biti
jesi        1 ps.sg.2     7 i0 biti
jesmo       1 ps.pl.1     7 i0 biti
jest        1 ps.sg.3     7 i0 biti
jeste       1 ps.sg.3     7 i0 biti
jeste       2 ps.pl.2     7 i0 biti
jesu        1 ps.pl.3     7 i0 biti
joj         1 dat         3 .. ona(f)
joj         2 -           8 .. joj
još         1 -           8 .. još
ju          1 ack         3 .. ona(f)
```

The word form dictionary includes non-congruent pronouns, conjunctions, interjections, numerals, etc. Also, completely irregular verb forms like the present tense of "biti" are included here, rather than introducing them through stem and endings.

You will realize that the placing of a word in this dictionary rather than in the stem and ending dictionaries is a matter of practical judgment based on knowledge of the language. So is the parsing of words in stem and ending; it can often be done in two ways. When we have a choice, we try to keep the ending dictionary as small as possible, because it has to be kept whole in the fast memory during processing. By contrast, the word form dictionary may grow very large without unduly increasing processing time. The stem dictionary falls in between these two in sensitivity to size; it is desirable to be able to keep all stems beginning with any one letter in memory at one time, as you may infer from the description to come of the program for grammatical analysis.

2.2. The program for grammatical analysis

To analyze a word, the program parses it in all manners possible: thus "reknem" will be parsed: "r-eknem, re-knem, rek-nem, rekn-em, rekne-m", and "reknem" (there exist zero endings, so the program must take this into account). Each of the potential stems is looked up in the stem dictionary, and each ending in the ending dictionary. If an ending or the corresponding stem does not exist, this parsing gives no result. If at least one stem and one ending exist, analysis is continued; the group name of each ending will be searched for in the list of allowable ending groups for each stem. In our example the following will happen.
The ending "-em" will be found in the dictionary, and found to belong to groups 41 and 411.
The corresponding stem "rekn-" will be found in the dictionary and it is seen to take the ending 41.
Consequently, we have a hit: "reknem" is present tense, singular, 1. person of "reći".

It is interesting to note that this analysis scheme is quite independent of the fact that Serbo-Croat is the language being analyzed. All information pertaining to Serbo-Croat is in the dictionaries; none in the program. The very same scheme and program can therefore be used for any language that has a flexion based on words consisting of stems and endings. That includes e.g. English, Russian, German, and Swedish. Of course, a general approach and program like this one could be considerably less efficient, in terms of storage or manual work, than one designed for the language in question. There would be difficulties with French, because the language is not clearly parsed into words ("a-t-il", "qu'on", etc). Work has recently started to build dictionaries for Polish, Macedonian and Swedish, and this is progressing well. Minimal changes had to be made to the programs to accomodate the transcriptions used and the longer words encountered.

2.3. Organization of the Texts

The texts containing what the children said during retelling of stories and other activities are stored as text elements in files (members of partitioned data sets, to use a common terminology). Each file is a year's class of children. The texts are written with small letters. They contain not only information about what the children actually said, but also - where there is a difference - what the child should have said, had it used the standard language. Further there is a division into syntactical units and

some information on pauses and the like. The promptings of the test leader are also included in the base text. Finally, Serbo-Croat contains five letters outside the English alphabet. The three Swedish special letters occasionally appear in the texts when the child has been at a loss for a Serbo-Croat word or referred to a Swedish name. In all, the following operators and letters appear over and above the English alphabet.

()	Test leader's comments
//	Cadence
/	Anticadence
+	Predication limit
...	Pause
¦	Nothing (corresponds to oval with a slash)
-	Word form fragment
*	<wrong pronunciation> * <right pronunciation>
&	<wrong form> & <right form>

If the word was said using both the wrong pronunciation and grammar, the order will be: <said word> * <said word with the right pronunciation > & <right word>

Serbo-Croat letters and their position in the ASCII schema:

ć	7/13
č	7/14
đ	7/12
š	7/11
ž	6/0

The letter "y", which does not exist as such in Serbo-Croat, is used after asterisk to transcribe different reflexes of the historical phoneme "jat". This device is necessary for representing the different standard and dialect forms by only one lemma in the dictionaries, e.g. "brijeg*bryg, breg*bryg, brig*bryg", etc. All analytical operations are performed, as always, on the forms after the asterisks. By different retrieval programs any of the reflexes can be found through *y without being contaminated with other ije's, je's, e's or i's.

The Swedish special letters "å", "ä" and "ö" are surrounded by hooks <> in transcription, to distinguish them from the Serbo-Croat letters "ć", "š" and "đ" occupying the corresponding positions in the ASCII schema.

3. THE PROGRAM LIBRARY

3.1. Overview of the Library We have already mentioned the existence of a program for grammatical analysis. We shall now list the complete program library with a one-line explanation of each program. Those programs that are of special interest will then be discussed in detail. Programs marked with an asterisk are machine dependent (the processsing is done on a Univac 1100/80 system). All the other programs are machine independent - or intended to be. They are written in Fortran 77 and can be run anywhere that language is available.

Build-up and handling of texts without analysis:

CTS*	General Univac processor for editing and job execution
BARNA*	Select children and year classes for processing
LIST	Make a nice printout of a text
FREQ	Make a frequency table
KONK	Make a concordance
BELAEGG	List instances of selected word forms from a text

Build-up and handling of dictionaries:

KOPIA	Copy the flexion of a word from a model dictionary
NOM-M	Build the dictionary entry for a masculine noun
GALLRA	Check a dictionary for format
SORTLX	Sort a dictionary
BYGGLX	Sort and merge the word class dictionaries into one

Grammatical analysis of texts:

READWC	Select part of a child's text and add context to each word
MERGE	Provide one or more analyses of each word of a text
SHOW	Allow the user to choose between homonyms
SELC	Select words belonging to a given analysis class

Phonetical analysis of texts:

COUNT	Counts correct and incorrect phonem realizations

Miscellaneous stand-alone programs:

ENTER	Enter social data from questionnaires
ENTEL	Enter data from dictionary test
GRALYS	Analyze a Serbo-Croat word (for demonstration and check)
MVREG	Compute mean, variance and regression line
FINTRYCK	Pretty printout of texts including correct alphabet

3.2. Build-up and handling of texts without analysis

CTS is a general Univac processor for editing programs and data and starting runs. Though the processor and its facilities are thus machine dependent, similar processors are nowadays found at any reasonable machine installation. The facilities used are the entering, storing, changing and printout of texts. Programs can be started from the CTS environment either interactively or as batch

runs. (Both types of runs have advantages. An interactive run is faster and allows the user to enter data at the keyboard during the run. A batch run is cheaper and gives better facilities for handling the results; they can be stored in a file, inspected, and selectively printed or reprocessed.)

When one tries to build a user environment on a system like CTS, one will soon note that it contains a number of weak points. These things a professional programmer will learn and handle as a matter of daily routine. The same points can become serious sources of errors and irritation when the system is used on and off by people whose primary interest and training is in linguistics (or any other subject outside data processing: medicine, sociology, etc.) A crucial operation in any editing system is the replacing of an old text version by a new one. A user-friendly editor really ought to compare the two versions and check, say, that the new version is longer than the old one and that they have a large part in common; otherwise it should ask the user if he really wants to do the replacing. Also, the system should not delete old text versions definitely, until the user says so. As it is, trivial errors may sometimes spoil half a day's work. On the whole, however, the text handling has been a success: all the children's texts and parts of the dictionaries have been built up by the people at the Department of Slavic Languages with occasional help and support from the Computing Centre.

The programs LIST, FREQ and KONK are conventional programs for printout, frequency tables, and concordances, respectively. (A concordance is a listing of occurrences of words in context.) They do contain a couple of interesting points. The Department early in the project bought a DEC VT-131 video terminal with a standard Serbo-Croat keyboard, including the special letters mentioned above. On the other hand, the printer (an IDS-560 "Paper Tiger") did not have these letters. The programs therefore had to include a subroutine to simulate the special letters on the printer. A text line is converted into two lines, one containing the diacritica, one containing the base letters. For instance, the word "plaćaš" would be split into two lines:

```
      ' "
   placas
```

which, when printed after each other, give a fair approximation of the real word.

Furthermore, even when struck on the terminal, these letters will not get their proper place in the Serbo-Croat alphabet. In much of our daily work, we ignore this fact. Words will be stored and retrieved in an order that differs from the true alphabetic order. For publishing purposes, however, we have the option of

sorting the words in their true order. The word is then converted into a sort key by substituting certain letters by character pairs. Thus, ć is changed into c', š into s", etc. Thus plaćaš becomes plac'as", when used as a sorting key. Any letter that can take a diacritic but did not, that is: <c, d, s, z>, and <e> (because of the transcription of jat) has a blank inserted after it.

A third point to note is the possibility of choice; the LIST, FREQ and KONK programs may be run for any of the forms separated by asterisk (*) and ampersand (&), i.e. for the said form, the standard form, or the phonetically correct but grammatically wrong form, or for any combination of these. Thus you may make one list of what the child really said, another of what it should have said, etc. As a special case it is possible to make frequency tables and concordances of "corruption patterns". A corruption pattern is defined as consisting of that letter string which changes when going from the corrupt to the standard form. Thus the form lisisa*lisica has the corruption pattern s*c (s instead of c), the forms reko*rekao and čito*čitao both have the corruption pattern !*a (dropped a), etc.

3.3. Build-up and handling of dictionaries

We have already noted that the dictionaries were partly built up by using the local editor. In certain cases it was found helpful to use special programs to speed up the process and make it safer.

The program NOM-M aids the user in building up the description of a masculine noun. It will put a series of questions to the linguist, some of which are dependent on previous answers. The user is first asked for the nominative and genitive singular of the noun, then whether it is animate or not, if there is any consonant stem change, long plural, etc. The complete flexion is displayed to the user who may then make corrections. Finally, the dictionary entries are written to a file, to be merged later with the existing dictionary.

The other auxiliary program is the copying program KOPIA. This one allows the user to copy an existing model flexion for a new word. You enter <glasati morati> saying in effect: 'Please conjugate the new verb <glasati> like the model verb <morati>'. The program will copy the allomorphs of <morati> and systematically replace the letters <mor> by <glas>. In the process it will check that the substitution is possible, i.e. that all stems of <morati> in fact contain <mor>. (It would not be possible to copy <glasati> from <pisati>, because <pisati> has a stem <piš> that can not be converted to <glas>.) Like the whole dictionary scheme, the copy program is independent of the fact that the

Serbo-Croat language is being treated, whereas the masculine noun program NOM-M mentioned above is of course quite language dependent.

The program GALLRA will check that different fields in the dictionary entries are in the right columns, correct some errors of position, and weed out redundant lines.

The program SORTLX will sort a dictionary to which lines have been added, and BYGGLX will merge and sort all the partial stem and ending dictionaries into two whole dictionaries for the grammatical analysis to follow.

3.4. Grammatical analysis of texts

The program READWC reads a part or the whole of a child's text, provides each word with a context and sorts the words in alphabetical order.

The program MERGE takes this text file and grammatically analyzes the word forms against the dictionaries according to the scheme described above. It leaves the text file expanded with grammatcial identifications of each word and re-sorted in textual order. A word form may have zero, one or more grammatical identifications, according as the dictionary search turned out.

The file is now processed by SHOW, which displays the words one by one with their proposed identifications. The linguist may now, at the terminal, choose between identifications of homonyms, add identifications of unexplained words, and correct some identifications. The program will also find combinations of the L-form of a verb with some active form of <biti>, proposing these to be periphrastic forms. When a predication has been worked through in this way, it is again displayed, now as a whole, to the user for final correction and acceptance. After processing the whole text in this manner, we get it sorted alphabetically a second time, now on lemma as first key and word form as second key: we have now achieved a lemmatized frequency table and concordance.

The program SELC is a variation on MERGE, which allows the user to select all words belonging to a given analysis class. Since this is done before the lemmatization, it will at present select more than intended: all homonyms which could possibly belong to the given class will be selected, and all unexplained words as well.

A next step in the development of the software will have to be a scheme to feed back homonym resolutions, once performed, into the original text, so that the analysis need not be repeated.

3.5. Phonetical analysis of texts

The program COUNT counts correct and incorrect realizations of phonemes, taking special account of digraphs like <dj>, <nj>, and to how <jat> is realized. All corrupt words found are sorted according to corruption pattern. Development is now going on to classify corruption patterns into three classes:

1) Influence from Serbo-Croat dialects
2) Influence from Swedish
3) Individual variations, fragmented words, stammering, etc.

3.6. Miscellaneous stand-alone programs

ENTER and ENTEL aid the entry of social data from questionnaires and dictionary tests, respectively. The programs will prompt the person entering the data with short cues, check the entered data for length and contents as far as possible, and store them in the right place. Each line of data is displayed for possible correction and acknowledgement.

GRALYS analyzes a Serbo-Croat word for demonstration purposes, or to check the analysis in detail. Naturally, it uses the same subroutine as the program MERGE, but applies it only to one word at a time that is entered manually.

MVREG that computes the mean, variance and regression line for some manually entered data is an example of how to use the large computer and a terminal as a desk calculator.

FINTRYCK that produces a pretty printout of a text is again a program outside the JUBA software package. It is intended for editing and printout of text for publishing purposes; in fact, the present edition of Slavica Lundensia is produced in this way. The user may insert into his text several kinds of directives, controlling headings, margin alignment, letter size, underlining, and choice of alphabet. The printer allows three alphabets in the same text: American, Swedish and a third one of the user's choice, where the user may actually control the building up of up to 94 letters and other characters from dot matrices. This takes care of most applications. Economic saving compared to photo-setting is very considerable.