Journal of Official Statistics
Vol. 10. No. 4, 1994, pp. 449–463
© Statistics Sweden

Miscellanea

Under the heading Miscellanea, essays will be published dealing with topics considered to be of general interest to the readers. All contributions will be refereed for their compatibility with this criterion.

Automated Coding of Census Data

Damir Kalpić[1]

Abstract: In an attempt to decrease costs, increase timeliness, and improve the quality of the 1991 census data processing in Croatia and Bosnia-Herzegovina, intelligent optical readers were purchased. Besides the fields containing coded numerical values, fourteen free-text variables were also included in the census. This article deals with the automated coding of these optically read free-texts. The Croatian language is highly inflected and nouns and adjectives change significantly with changes in case. This characteristic of the Croatian language has made the automated coding of census data more difficult. This paper discusses the coding algorithm and the results.

Key words: Character recognition; coding algorithm; intelligent optic readers.

1. Introduction

This article reports the development work conducted to process the 1991 Croatian population census with optic reading of census forms and automated coding of fourteen variables originally recorded in free-text, written by hand in capital letters. Much of this work is based on the international experience with automated coding as reported in, for instance, Lyberg (1981), Lorigny (1988), and Wenzowski (1988). The method described is being used by another republic in the former Yugoslavia, Bosnia-Herzegovina. There is a high degree of similarity between the languages in Croatia and Bosnia-Herzegovina, which enables the system to be applied in Bosnia-Herzegovina.

Intelligent optic readers were purchased from a private company in Germany. The accuracy of the optical reading is controllable at the character level. If the level of confidence for letter recognition is unsatisfactory, the original text is shown to an operator who can then choose the correct character manually. The system

[1] University of Zagreb, Faculty of Electrical Engineering, Applied Mathematics Department, Avenija Vukovar 39, 41000 Zagreb, Croatia.
Acknowledgements: The author is much indebted to colleagues Vedran Mornar and Spomenko Kenda whose creative programming led to the ultimate success of the algorithm. Jakov Binefeld from VIP contributed much organizational support and Marijan Gredelj, Dražen Spalatin and Srđan Dumičić were our main support at the Croatian Statistical Institute. In Bosnia-Herzegovina we owe thanks to Nermana Slatina.

offers the most probable character as the default value. The confidence level is set especially high for the first character in a word. Unintelligible characters are replaced by the character "#" since "#" is never used in Croatian texts. A discussion of optic reading and related problems are discussed in detail in Dumičić, Kecman, and Dumičić (1992).

Coding is a necessary step when processing censuses as textual data cannot be processed statistically until the texts have been translated to unique numerical codes.

The program is written in C and recognizes the optically read text (with varying levels of distortion) and searches for a match in the thesauri. Different matches have different probabilities. Automated coding is used on variables like occupation, religion, nationality, education, geographic location, and type of activity.

For each variable, the matching algorithm searches through the thesaurus of phrases that are associated with a code. Each word from the thesaurus is assigned a weight depending on its frequency of appearance in nonsynonym phrases. The higher the frequency, the lower the weight. Each input word is analyzed and a number of best matching words from the thesaurus are selected. Missing characters, similar characters, and unintelligible characters are also taken into account, but some penalty is introduced. In this way, a measure of similarity between the input word and the thesaurus word is obtained. Candidate words are sorted in descending order in accordance with a score derived from their similarity multiplied by the word weight.

In a second pass, the candidate words are combined to find the best matching phrases from the thesaurus. A list of matches, sorted in descending order from most likely match to less likely match, is formed. Depending on the similarity score and the distance to the second nonsynonym candidate phrase, the code is accepted or is left for human arbitration. These arbitrations are recorded and used to improve the thesaurus further and to fine tune the parameters. The levels of the parameters determine important aspects of the way the program runs.

For variables like occupation, nationality, etc., the thesauri of expected phrases were prepared empirically, using updated data from previous censuses. In the thesaurus, each code is associated with more than one phrase. For example, for the variable occupation, a systematic list of occupations was used. In the list, each code has as affiliate, a number of textual descriptions that describe the same occupation, for example, engine driver, engineer, train engineer, etc.

Our program was developed to transform the textual descriptions into codes with a given level of confidence or to display the list of candidates in descending order of their match probabilities. Human arbitration then chooses the correct match from the candidates displayed.

To monitor the production process, a multiuser arbitration mechanism with a transaction log was developed. The information derived from the monitoring was used as part of a process quality control whose ultimate purpose is to increase the automated coding rate.

The automated coding program includes the following functional modules:

- single word recognition,
- recognition of phrases consisting of more than one word,
- user interface and application administration.

Although conceptually straightforward, developing the user interface proved an arduous task because of the limitations of the computer platform (IBM 4381, CICS). Human arbitration is necessary when the automated coding produces a match whose confidence level is too low. It is through the user interface that the human coder interacts with the coding program. A list with up to nine candidate phrases is displayed on the screen. These candidates are ordered in descending sequence of their estimated similarities to the optically read input phrase. To illustrate with a fictitious example involving the English city of Bath:

Input optically read phrase:
SYN#PSI## #F STMULATION IN BATH

Candidate phrases:
1. SYNOPSIS FOR STIMULATION IN THE BATH
2. SYMPOSIUM OF SIMULATION IN BATH
3. ...

The human coder can request to view the other input variables on the screen and determine the correct meaning given the context.

2. Algorithm for Word Recognition

2.1. Basic assumptions

The algorithm for single word recognition is based on the following assumptions:

2.1.1. The input word is similar to some other word from a known set of expected words

Most often, an input phrase consists of one or several words that describe a specific variable. Synonyms on the variable level have the same code assignment. For example, the pair "housewife" and "housekeeper" are synonyms as are "engine driver" and "engineer in train." Their expected values have been stored in the initial thesaurus of occupations.

Due to the linguistic characteristics of the Croatian language where words change in accordance with seven cases, two different plural forms (1 train = 1 vlak; 2–4 trains = 2–4 vlaka; 5 trains = 5 vlakova) and three genders, it cannot be expected that the thesaurus contains all possible forms of a word. To illustrate the problem, here are some possible variations of the occupation "engine driver":

Vozač vlaka	(Engine driver of train)
Vozač u vlaku	(Engine driver in train)
Vozač vlakova	(Engine driver of trains)
Vozač u vlakovima	(Engine driver in trains)

Note how the Croatian noun "vlak" changes with case. Five letters "ovima" are added to the four letters of the root "vlak," to denote case. In multiword phrases, the possibilities grow; since case is expressed by a suffix, word order is left relatively free. Almost all word orders are grammatically correct and a change in word order suggests a change in stress or stylistic difference. So, the algorithm for automated coding of Croatian texts has to accept similar words with more tolerance than would be appropriate for English texts.

2.1.2. Input word might not be correct

There are a number of error sources that can affect a word while being processed: for example, respondent error in formulation, interviewer error, and optic reading error. A posterior analysis has shown (Dumičić, Kecman, and Dumičić 1992) that each letter has a probability of misinterpretation, most often mistaken as a similar letter. The problem of content dependent similarity estimation is discussed later.

2.2. Creation of the thesaurus

A separate thesaurus is required for each of the textual variables such as occupation, type of activity, etc. Each thesaurus contains a number of expected phrases and their associated codes. Each word is stored separately, and phrases containing the word are tagged. During processing, particularly in early stages, the thesauri are complemented with phrases that were overlooked or omitted from the start. Once satisfactory coding rates are achieved, the thesauri are not updated further. There is a point at which further improvements to the thesauri do not result in significant improvements in coding rates.

2.3. *Word weight calculation*

The Wenzowski (1988) algorithm which calculates the weights of single words was not applicable to our automated coding program because we did not have access to previously calculated phrase frequencies from earlier surveys. Instead of using weight 1 for all words, a measure of the discrimination power of a word was estimated. A word that appears at least once in every set of synonymous phrases would have the weight 0 because this word contributes no relevant information for deciding the code. A word that appears only in phrases related to a single code has the maximum weight. The basic formula is:

$$W(i) = (n - f_i)/n$$

where

$n =$ total count of unique codes
$f_i =$ count of codes related to phrases containing single or multiple appearance of the word i
$W(i) =$ weight of the word i

Obviously, for $f_i = 1$, $\lim_{n \to \infty} W(i) = 1$ suggesting that maximum weight is given to the word which uniquely identifies a phrase.

For practical reasons, to achieve higher processing speed, integer arithmetics were used. So the weights are rounded off by the function INTEGER to nearest integer

$$W(i) = INTEGER \{[(n - f_i) \times probability\_one/n] + 0.5\} \quad (1)$$

where *probability_one* is a parameter representing the value for probability of a 100% certain event (which is otherwise normally equal to 1) for purposes of integer arithmetics.

Due to $n \gg 1$, the weight can vary from 0 to *probability_one*.

2.4. *Word recognition*

When analyzing an input word, the program initially searches the thesaurus and selects all words whose first letter matches the first letter of the input word. Following the process described above, the optical reader should not allow errors in the first character.

The measure of similarity between the input and the considered word is calculated on the basis of:

- difference in word lengths, usually due to the presence of a prefix,
- matching of equal or similar characters from the beginning to the end,
- matching of equal or similar characters from the end to the beginning, if words differ in length,
- matching after shift,
- length of continuous matching strings.

The general parameter values, used later on, were initially set heuristically supplemented with intuition, and eventually determined after considerable testing:

$$probability\_one = 1000$$

The value for scaling the probabilities with *probability_one* was chosen to indicate that the precision of three significant digits is appropriate.

The importance of character matching decreases from the first letter to the last. The importance of character i is calculated as $a/(b \times i)$ where i is length of word and $i = 1, \ldots, q$.

The parameter values have been set to:

$a = 1$, and $b = 1$.

This type of weighting is motivated by the case and gender changes in Croatian which cause major suffix changes. The values for a and b are selected mostly for reasons of simplicity. A more sophisticated procedure could, for example, use a value for b which depends on word length. In English, where words do not change with case, this procedure would make little sense. Probably, one should instead test the plural form to determine whether "s" or "es" had been added to the word. Words with irregular plurals, e.g., children, men, etc., could be included in the thesauri. For languages that form complex compound words, like German, this procedure would hardly be applicable.

When calculating the match, equal characters are weighted by the parameter *equ_char* and similar characters (except for the first character) are weighted by the estimated expected frequency of such a deviation between characters, multiplied by *equ_char*.

An unintelligible character, replaced by "#," does not imply the breakdown of a continuous string. Unintelligible characters are weighted by *unint_char*. The empirical values are:

$equ\_char = 10$

$unint\_char = 5.$

The empirical values above tell us that when an unintelligible character is encountered, a 50% probability is assigned that the character is the one expected.

The resulting similarity score is approximately normalized by dividing a given word's assigned similarity score by the highest possible unnormalized similarity score for the word. A perfect match results in *probability_one*.

A period (full stop) "." in an input phrase implies that an abbreviation was used when the questionnaire was completed. The period represents any string of characters with equal probability. However, a penalty value which expresses the possibility of expanding the word incorrectly, *pen_abb*, is introduced to differentiate the expanded abbreviation from the full word.

The best a possible match can be is *probability_one*. In the case where the matched words differ and if after the normalization the similarity score is *probability_one* (due to integer arithmetics) the penalty *pen_abb* is subtracted from the similarity score, as is done with abbreviations. It has been empirically determined that the value for *pen_abb* should be variable-dependent, as abbreviations are more common in some variables than in others.

Words with a similarity score higher than a threshold value, *min_sim*, are considered candidates for a match. The threshold value is defined individually for each variable.

The list of candidate words appears in descending order of similarity score, i.e., the candidates with the highest similarity scores first and those with the lowest last. To create a subset of (most likely) candidate words from those words at the top of the list, a maximum cardinality parameter, *max_cand_wds*, is defined. Only this ordered subset is then processed further. The parameter value is set as:

$max\_cand\_wds = 20.$

This parameter was decided empirically as a compromise between accuracy and processing speed. Larger parameter values usually, but not necessarily, yield a better match. But, due to the frequent occurrence of long phrases, the processing speed of the current hardware prohibits larger parameter values.

Obviously, a word recognition procedure that searches only for candidates having the same first letter as the input word will have a certain failure rate. For example, if the first letter of the input word is misinterpreted by the optic character recognition (OCR), the set of candidate words may be empty or may contain (usually improper) words with low similarity scores. In such cases, a longer and more complete procedure is invoked. It differs from the quick one mentioned above by initially selecting all words in the thesaurus regardless of their first letters. The expected slow down in processing time can be of a magnitude of 27, as there are 27 single letters in the Croatian alphabet.

To deal with optically misread letters, a table of estimated expectancies of misinterpreted Croatian letters as a "similar" letter has been established. Table 1 shows the expectancies, multiplied by 10 to avoid the floating point arithmetics. Columns represent optically read characters and rows the characters from candidate words.

To illustrate the way Table 1 works, let us examine the first row. If the letter "A" is expected in the word from the thesaurus, and the letter "G" is optically read instead, Table 1 (A, G) states that there is a 20% probability that "A" is the character in the original input word, and that the candidate word is the correct choice. In 80% of the cases, "A" was not truly the character and the candidate word from the thesaurus should be penalized for letter mismatch. The table values were deduced by analyzing the outcome of optical reading of a few hundred sample input forms. In theory, the sample size can be estimated empirically, i.e., until increases in the sample size introduce no significant changes. These empirical values derive not only for the optical similarity of letters, but also from the language morphology and habitual syntactic, handwriting and other errors. For instance, people often confuse Croatian Letters "Ć" and "Č," or they think that due to computer processing the letter "C" should be filled-in instead. Thus, in 90% of the cases, "C" can be accepted as "Ć," but also in 90% of the cases it can be accepted as "Č" if that letter was expected in the word. In 10% of the cases, it was really "C" or some other letter, different from "Ć" or "Č." This suggests that the incorrect candidate word was chosen for comparison.

Using the data from Table 1, we provide the following English language example.

The right word is "EVEN."

a. From the OCR machine came "EVEA."

Candidate words from the thesaurus are:

Table 1 [N,A] = 6 → EVEN – 60% probability that "A" should be replaced by "N."

Table 1 [R,A] = 6 → EVER – 60% probability that "A" should be replaced by "R."

Both words are equally probable.

b. From the OCR machine came "EVEM."

Table 1 [N,M] = 6 → EVEN – 60% chance that "M" should be substituted by "N."

Table 1 [R,M] = 0 → EVER – no chance that "M" should be substituted by "R."

Candidate word "even" receives a higher similarity score.

It is interesting to note that the matrix is not symmetrical.

Tests have shown that the values for some parameters should be specific for each variable. On the other hand, which parameters should be variable-specific and the values such parameters should assume were not the result of research and it cannot be proved that they are optimal. An initial intuitive guess based on sample input data was necessary. A description of the parameters follows:

Bonus for no shift (*bon_no_shift*). The similarity score is increased by this amount if the words fit without a left or right character shift.

Minimum string length when the strings are shifted (*min_sub*). When matching by shift is attempted, this is the minimum overlapping substring length before the shift stops.

Bonus for continuous string match (*bon_cont*). The similarity score is increased by this amount for each contiguous character match.

Minimum character similarity for continuous string match (*min_ch_sim*). If the similarity between the input and candidate character is equal or higher than this value, a contiguous character match is assumed.

Penalty for mismatch (*pen_abb*). For the words that do not match perfectly, this is the minimum penalty that must be subtracted from *probability_one*. This is also the minimum penalty when an abbreviation is encountered.

Penalty for the difference in word lengths (*pen_len*). The similarity score is decreased by this amount multiplied by the difference in length between the words compared.

Measure of similarity for mandatory choice of a word (*word_mand*). If the similarity of a word to the input word is greater than this parameter, it is mandatory that the considered word becomes candidate.

Minimum measure of similarity (*min_sim*). Words that have a similarity score lower than this value cannot become candidates.

Maximum similarity difference (*max_sim_diff*). A word that has a similarity score so much worse than the score of the worst already found candidate word cannot become a candidate itself.

The threshold for entering the complete procedure (*thresh_complt*). If the similarity score for the best matched word is lower than this parametrical value, the complete word recognition procedure is invoked.

Specific parameter values, determined after testing, for the variable *Occupation*:

bon_no_shift = 7
min_sub = 5
bon_cont = 3
min_ch_sim = 6
pen_abb = 5
pen_len = 5
probability_one = 1000
word_mand = 900
min_sim = 500
max_sim_diff = 300
thresh_complt = 749

To illustrate more exactly the word similarity computation, a simplified pseudo-code is presented.

Let **I** be the ordered set of cardinality n_i containing letters i_j, $j = 1, \ldots, n_i$, which form the input word.

Let **A** be the set of all the words from the thesaurus.

Let **T** be the ordered set of cardinality n_t containing letters t_j, $j = 1, \ldots, n_t$, which form the word selected from the thesaurus.

Let **W** be an ordered set of cardinality n_w containing candidate words, together with their already computed similarity values

Table 1. Expectancies of letter misinterpretations

	A	B	C	Č	Ć	D	Đ	E	F	G	H	I	J	K	L	M	N	O	P	R	S	Š	T	U	V	Z	Ž
A										2	6									4							
B							4													4							
C				9	9													6									
Č			9		9																						
Ć			9	9																							
D							9											6	4								
Đ		4				9																					
E			4						4	6		4		2													
F								5																			
G			6												4			6									
H	4													6			7										
I								6					6														
J						4						6						4			4		6		4		
K	6							4									6			6				4			
L			8							4																4	
M											6						6										
N	6										9					6								5			
O			6			8		6																			
P						6																					
R	6	4						4		2				6												4	
S		7								7												9					
Š																					9						
T									6																		
U											6						6								9		
V																								9			
Z											3												4				7
Ž																										9	

$S_j, j = 1, \ldots n_w$, to the input word, sorted by decreasing similarity.

Procedure SOLVEWORD to form the set W

```
for complete := 0 to 1
| if complete = 0
| | select from A all the words having the first
    letter = $i_1$ into the set B
| else
| | B := A /* select all the words regardless to
the first letter */
| $n_w$ := 0 /* set W is empty */
| for next := 1 to card (B)
| | select next word from B
| | the letters of the selected word form the
    set T.
| | $S_{max}$ := 0
| | S := 0
| | for j := 1 to min {$n_j, n_t$}
| | | /* maximum unscaled similarity */
| | | $S_{max} := S_{max} + equ\_char \times a/(b \times j) +$
      $bon\_cont$
| | | if ($i_j = t_j$)
| | | | /* characters are equal */
| | | | $S := S + equ\_char \times a/(b \times j) +$
          $bon\_cont$
| | | else if $i_j$ = "."
| | | | /* penalty for abbreviation, characters
          assumed equal */
| | | | $S := S - pen\_abb$
| | | | for jj := j to $n_t$
| | | | | $S := S + equ\_char \times a/(b \times jj) +$
            $bon\_cont$
| | | | | $S_{max} := S_{max} + equ\_char \times a/(b \times jj)$
            $+ bon\_cont$
```

```
| | | | $j := n_t$
| | | else if $i_j$ = "#"
| | | | /* unintelligible character in input word */
| | | | $S := S + unint\_char \times a/(b \times j) + bon\_cont$
| | | else
| | | | /* similar characters */
| | | | $S := S + Table\_1\ [t_j, i_j] \times a/(b \times j)$
| | | | if $Table\_1\ [t_j, i_j] > min\_ch\_sim$
| | | | | /* if similar enough, the continuous string is assumed */
| | | | | $S := S + bon\_cont$
| | if $n_i \neq n_t$
| | | /* words differ in length, penalty for different word lengths */
| | | $S := S - pen\_len \times \mathrm{ABS}(n_i - n_t)$
| | || Similarity is computed also backwards,
| | || and with left and right shifts until over-
| | || lapped strings of the input and the
| | || thesaurus word are not less than
| | || $min\_sub$. $S$ and $S_{max}$ are accumulated in
| | || this process
| | else
| | | /* bonus for matching without shift */
| | | $S := S + bon\_no\_shift$
| | | $S_{max} := S_{max} + bon\_no\_shift$
| | /* scaling with maximum possible similarity */
| | $S := probability\_one \times S/S_{max}$
| | if $S > min\_sim$
| | | /* the word from thesaurus is similar enough to the input word */
| | | if $(n_w := 0) \vee (S > word\_mand) \vee [(n_w > 0)\ \&\ (S_{n_w} - S) < max\_sim\_diff]$
| | | | /* if the set of candidate words is empty, or the similarity is high for mandatory choice or the difference in similarity to the worst match is not too high */
| | | | insert the word and its similarity value to ordered set $\mathbf{W}$
| | | | $n_w := n_w + 1$
| | | | if $(n_w > max\_cand\_wds)$
| | | | | /* drop the worst word */
| | | | | $\mathbf{W} := \mathbf{W} \backslash w_{n_w}$
| | | | | $n_w := n_w - 1$
| /* until all the words from B have been examined */
| if $(n_w > 0)\ \&\ (S_1 > thresh\_complt)$
| | /* if words were found and the most similar word is over threshold */
| | exit SOLVEWORD
| /* continue with complete search */
exit SOLVEWORD
```

2.5. Phrase recognition

Phrases that determine the variable's code can contain a single word or multiple words. Multiple phrases that lead to the same code are considered synonyms. First, sets **W** of candidates for each of the input phrase words are formed, then phrase recognition can proceed. For each candidate word, all phrases that contain that word are tagged. The algorithm then proceeds by forming the ordered set **P** as a list of candidate phrases to the input phrase sorted by their descending similarities S_p. The algorithm is controlled by empirically determined parameter values, specific for each variable.

Each word in the input phrase, containing n words, is matched with a word from the phrase in the thesaurus, which is also found in the set **W** (word order is irrelevant here). The phrase similarity score, S_p, is computed and inserted into set **P** if certain conditions are met

$$S_p = \frac{\sum_i^n S_w(i) \times W(i)}{\sum_i^n W(i)}. \tag{2}$$

Here, i denotes single words, from every set **W**, which were matched with corresponding words from the candidate phrase.

The $S_w(i)$ is the normalized similarity score between the input word and the matching word from **W**, as computed by the procedure SOLVEWORD and $W(i)$ is the weight of the word from **W** calculated by formula (1).

The similarity score tends towards *probability_one* as the single word similarities increase. This statement holds if each input word produces:

1. a non-empty set **W**, and
2. a match between a member of set **W** and a word from the candidate phrase.

If **W** does not include a word present in the candidate phrase, the denominator is increased by the weight of the unmatched word divided by an empirical parameter *input_lacks_word* (α).

If, on the contrary, the input phrase contains a word that cannot be found in the thesaurus, the denominator is increased by the average thesaurus word weight, divided by an empirical parameter *candidate_lacks_word* (β).

Generally, by expanding formula (2), the similarity score between the input phrase of n words and the candidate phrase becomes

$$S_p = \frac{\sum_i S_w(i) \times W(i)}{\sum_i W(i) + \sum_j W(j)/\alpha + k \times W_a/\beta}. \tag{3}$$

In the above:

i denotes matches (single words) between **W** and the candidate phrase;

j denotes words in the candidate phrase that were never matched to the input word;

k is the count of words from the input phrase which were not matched to the phrase from the thesaurus;

W_a is the average weight of words from the thesaurus.

The similarity score is first multiplied by the word weight of the best matched word in **W**. If the resulting score is greater than a certain value (*min_sin_qu*), only phrases introduced by words with $S_w(i) \times W(i) > min\_sin\_qu$ are considered. Otherwise, a more complete search for candidate phrases occurs. In this more complete search, the analysis of a candidate phrase is mandatory if it contains a word that meets the following criterion: $S_w(i) \times W(i) > thresh\_mand$, i.e., a similarity score multiplied by the word weight which is greater than the threshold value. In any case, a list of candidate phrases is formed, sorted by descending similarity.

The processing speed and quality of the results depend heavily on the number of candidate words assigned to each input word. Given that the phrase contains k words and n candidates have been assigned to every input word and that the average number of phrases containing a single candidate word is m, a complete search results in $k \times n \times m$ computations of the similarity scores of candidate phrases and insertions of these scores in the candidate phrase list. Candidate words bearing small weights introduce many candidate phrases ($m \gg$). On the other hand, it can be expected that candidate words having little similarity to the input word would introduce erroneous candidate phrases. So, the complete procedure is invoked only when no satisfactory solution is otherwise obtained.

All parameters that affect the processing speed and accuracy were initially set intuitively. Later, values of the parameters were varied in tests using sample data for different variables. Eventually, the most plausible values have been established. As an example, the parameters for the variable Occupation are presented.

Characteristics of the thesaurus for the variable Occupation:

Number of phrases = 21644
Number of distinct codes = 2922
Number of words in phrases = 72148
Number of distinct words = 13109

Table 2. Examples of word weights

Position in the list	Croatian word	English translation	Word weight
1.	Za	For	1
2.	Na	On	7
3.	I	And	50
4.	Radnik	Worker	99
5.	U	In	103
6.	Poslovođa	Chief	158
7.	Rukovalac	Manipulator	180
8.	Proizvođač	Producer	466
9.	Tehničar	Technician	525
...			
79.	Održavanje	Maintenance	883
...			
13057.	Pastir	Shepherd	1000

In Table 2, a few words are selected from the ascending list to illustrate the word weights, calculated by formula (1), with *probability_one* = 1000.

Parameter values for the variable Occupation:

input_lacks_word = 4
candidate_lacks_word = 6
min_sim_qu = 950
thresh_mand = 900.

The similarity score determines whether the code will be assigned by the coding program or be left for human coders. The program assigns a code when:

1. The similarity score is sufficiently high;
2. The difference in similarity score between the first ranked and the second ranked candidate phrase is large enough.

Table 3 shows similarity scores calculated for *probability_one* = 1000. The columns

Table 3. Decision table for automated coding.
Difference in similarity between the first and the second rated candidate phrase, divided by 10

0	06	11	16	21	26	31	36	41	46	51	56	61	66	71	76	81	86	91	96	
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	96–100
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		91–95
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			86–90
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1				81–85
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1					76–80
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1						71–75
0	0	0	0	1	1	1	1	1	1	1	1	1	1							66–70
0	0	0	0	0	1	1	1	1	1	1	1	1								61–65
0	0	0	0	0	0	1	1	1	1	1	1									56–60
0	0	0	0	0	0	0	1	1	1	1										51–55
0	0	0	0	0	0	0	0	1	1											46–50
																				Similarity score of the first rated candidate phrase divided by 10

represent intervals of the difference in similarity to the input phrase between the first and second ranked candidates. The rows show the intervals of similarity scores to the input phrase of the first ranked candidate. The interval values were divided by 10 for practical reasons. A table value of 1 represents automated coding while 0 suggests that the case was deferred to human coders.

The values in the table were first determined intuitively and then adjusted after some testing. We looked at the outcomes of the first ranking candidate of cases referred to human coders (which variables, which similarity scores, the difference between the first and second ranking candidates, whether first ranking candidate was accepted). Information derived in this way was used to modify the coding program and maximize the automated coding rate. As the automated coding rates improved, less and less of this type of analysis was conducted.

The following illustrates how Table 3 is used; the example comes from the Introduction and the similarity scores are hypothetical.

Candidate phrases	*Similarity score*
1. SYNOPSIS FOR STIMULATION IN THE BATH	615
2. SYMPOSIUM OF SIMULATION IN BATH	498
3. ...	

Let us suppose that the similarity to the input phrase of the first and second ranking phrases are 615, and 498, respectively. This makes the difference 615 – 498 = 117. The decision value is taken from Table 3 [61–65, 11–15]. Similarity 615 indicates the interval 61–65 (row 8), and the difference 117 is from the interval 11–15 (column 3). The value from the table is 0, meaning that human arbitration is required.

3. Summary of Results

The following outcomes of each automated coding attempt are possible:

U0 – The input value is blank.

U1 – The input phrase matches a phrase from the thesaurus completely, letter by letter.

U2 – The input phrase is not identical to any phrase in the thesaurus but automated coding can be accepted.

U3 – The input phrase differs from every phrase in the thesaurus to the extent that human arbitration is necessary. Up to nine best rated candidate phrases will appear on the operator's screen, sorted by descending similarity to the input phrase.

U4 – The input phrase is unintelligible so that an expert is required.

U5 – The numerical code is directly present in the input phrase as result of local processing and the completed census form for some variables (e.g., the code for *type of activity* for all the employees of an institution).

The processed variables were:

1. Place of birth
2. Place of parents' habitation
3. Place of former habitation
4. Nationality
5. Language
6. Religion
7. Name of the school completed
8. Parent's occupation
9. Foreign country
10. Occupation
11. Head of household's (the breadwinner) occupation
12. Type of activity
13. Site of professional activity
14. Foreign country for former guest workers.

For the assessment of processing quality only the proportions of U2 and U3 were relevant. The quality expected for different variables was also different. They were grouped as follows:

a. Occupation (variables 8, 10 and 11)
b. Type of activity, site, place (variables 1, 2, 3, 12 and 13)
c. Other (variables 4, 5, 6, 7, 9 and 14).

For each group of variables upon the estimated effects of manual work reduction, a weight was provided. Prescribed minimum U2 to U3 proportions to achieve the three discrete quality levels were determined by the census organizer and are shown in Table 4.

The quality levels were defined as:

1 "fair"
2 "good"
3 "excellent."

Table 5 provides the results for the whole census data in Croatia.

From the Tables 4 and 5 it can be seen that the quality "excellent" (3) for groups A and B has been achieved, while for C the level is 1.5. Weighted average for the whole is 2.85, rather close to "excellent" (3).

There is always a trade-off between processing speed and quality. We found that we had to ease up on our quality requirement so to avoid a bottleneck in processing. The coding program is written in C and the developmental work used a PC/AT and a Cromemco DCS-1/320. Data were processed at the Croatian Statistical Institute where two PC/ATs with 8 MB i860 boards and MicroWay C were used. Some difficulties with the compiler code optimization were encountered.

After restructuring the code and departing somewhat from the optimization, the program worked fine. The findings from the automated coding development work were transferred to an IBM 4381/T91. A multiuser, interactive application using CICS was developed for cases deferred to human coders.

At the Statistical Institute of Bosnia-Herzegovina, automated coding was conducted using an IBM 4381/92E, which resulted in slow processing. The quality was similar to the results obtained in Croatia. Unfortunately due to the war, the official quality evaluation is not available.

4. Conclusions

The entire 1991 census data for Croatia and Bosnia-Herzegovina have been successfully processed. The predefined quality level was met for two groups out of three.

Speed and processing accuracy depend heavily on the parameter values. During

Table 4. Estimated quality of automated coding

Group		Quality Prescribed minimum proportions (%)			Weight
		1	2	3	
A	U2	31	38	54	0.55
	U3	69	62	46	
B	U2	38	54	77	0.35
	U3	62	46	23	
C	U2	58	83	95	0.10
	U3	42	17	5	

Table 5. Results of automated coding

	Outcomes of automated coding for 4902185 questionnaires								
Variable	U0	U1	U2	U3	U4	U5	Total	U2%	U3%
1.	35162	3391921	1167166	307216	720	0	4902185	79.16	20.84
2.	30725	3277544	1313310	280406	200	0	4902185	82.41	17.59
3.	2674398	1422465	622139	182910	273	0	4902185	77.28	22.72
4.	44984	4454591	370819	31745	46	0	4902185	92.11	7.89
5.	49165	4131540	693205	25624	2651	0	4902185	96.44	3.56
6.	66340	4092274	706887	35635	1049	0	4902185	95.20	4.80
7.	3132443	187556	628230	950694	3262	0	4902185	39.79	60.21
8.	176960	2573373	1293993	852405	5454	0	4902185	60.29	39.71
9.	4619653	208416	66661	5057	2398	0	4902185	92.95	7.05
10.	2739966	824099	628332	705564	4224	0	4902185	47.11	52.89
11.	3344018	587466	520190	447506	3005	0	4902185	53.76	46.24
12.	1584665	299982	223030	281880	10607	2502021	4902185	44.17	55.83
13.	4028938	637220	193997	41833	197	0	4902185	82.26	17.74
14.	4775659	95382	26430	2314	2400	0	4902185	91.95	8.05
All:	27303076	26183829	8454389	4150789	36486	2502021	68630590	67.07	32.93

	Outcomes of automated coding for 4902185 questionnaires groupwise								
Group	U0	U1	U2	U3	U4	U5	Total	U2%	U3%
A	6260944	3984938	2442515	2005475	12683	0	14706555	54.91	45.09
B	8353888	9029132	3519642	1094245	11997	2502021	24510925	76.28	23.72
C	12688244	13169759	2492232	1051069	11806	0	29413110	70.34	29.66
All:	27303076	26183829	8454389	4150789	36486	2502021	68630590	67.07	32.93

early testing, automated coding of a single occupation which consisted of a number of words took several hours! Due to algorithm refinement and proper parameter settings, the time has been reduced to fractions of a second; no processing bottlenecks are encountered if two i860 PCs are used. The processing bottlenecks remained with an IBM 4381/92E. The values of many of the parameters are linked to the language per se. For instance, a set of parameters that work well in English would have to be changed to accommodate idiosyncracies of Croatian.

Coding quality depends heavily on the thesaurus. For example, among 21,644 occupations, "housewife" (Croatian: domaćica) was absent from the thesaurus. Nevertheless, roughly a quarter of the population stated "housewife" as her occupation. "Hostess in aeroplane" (Croatian: domaćica u avionu) was the best candidate offered. By improving and enhancing the thesauri in the early processing stages, the coding rates were improved significantly. Yet, the coding rates for group C were mediocre. This is explained by many different schools having very similar names and that respondents were not aware of this fact. This proved surprising because we had expected high matching rates for group C. Further analysis of parameter values for decision U2/U3 might have entailed some improvements, but such analysis ended after the first satisfactory results had been achieved.

Since the processing of census data was so successful, we expect that word and phrase recognition will be tested on some other application.

5. References

Dumičić, S., Kecman, N., and Dumičić, K. (1992). An Implementation of Sampling Method on Optical Reading Control in the Census 1991. Proceedings of the 14th International Conference on Information Technology Interfaces, Pula, Croatia, 449–454.

Lyberg, L. (1981). Control of the Coding Operation in Statistical Investigations. Urval no. 13, Stockholm: Statistics Sweden.

Lorigny, J. (1988). QUID, A General Automatic Coding Method. Survey Methodology, 14, 289–298.

Wenzowski, M.J. (1988). ACTR, A Generalized Automated Coding System. Survey Methodology, 14, 299–307.

Received April 1992
Revised August 1994