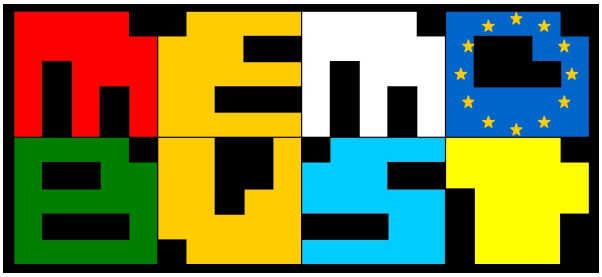

This module is part of the

Memobust Handbook

on Methodology of Modern Business Statistics

26 March 2014

Theme: Coding – Main Module

Contents

General section

1. Summary

The main theme of this document concerns the methods for automatic or semi-automatic (interactive) coding of answers to open questions. These are short descriptions (typically less than 10 words) in a respondent's own words formulated about the person's occupation, education followed, work performed, goods and services produced, etc. The code that is assigned to a description (if successful) originates from a classification. The classification itself is too complicated for respondents to directly search it for an answer. It is easier to let the respondent answer in his or her own words, and then to try to interpret this answer. Nowadays, this interpretation usually employs a computer if the material is delivered electronically, as we will assume. In the past, this coding was done completely 'manually'. That manual coding process is expensive, slow and non-transparent. Nowadays, the goal is to have the bulk of the coding work done by computer running special coding software. The remaining 'difficult cases' are then resolved more or less 'manually' as in the past.

2. General description

2.1 Introduction

Coding is an activity in the statistical process. It can be considered as a special type of derivation, and a rather difficult one. The purpose of coding is to match a code derived from a classification to textual information. The goal in this process is to reduce the large variety of answers to a convenient number, and to organise these answers (the classification used offers this option by means of its structure).

We view this matching as an interpretation of a description (the textual information) in the light of the classification concerned. An example is a description of an economic activity (in a respondent's own words) that is interpreted taking into account the NACE. Other examples concern descriptions of goods, descriptions of education that people have, illnesses suffered by people, and causes of death.

Coding is also very similar to a doctor's diagnosis of patients who present him or her with various complaints and symptoms. The task of a doctor is to diagnose an illness or abnormality based on a number of observations, answers from the patient and possibly additional tests (blood tests, for instance).

The main reason why variables with open answers are used is that this is convenient for the respondent. Also, there is far less influence on the answer. For such variables, the person can answer with a personally formulated text. If the respondent were required to give an answer in the form ultimately needed by NSIs to create statistics, then he would have to know the classification that serves as the basis for such a variable, such as the Standard Industrial Classification (SIC). However, this is much too difficult, and from a practical point of view, impossible to expect from a non-specialist.

In the past, coding these open-text answers invariably was done by human coders, specialists in coding of occupations, education, business activities, etc. The problem with this 'manual coding' is that it is time-consuming, expensive and not standardised. Consequently, over time, computers have been used increasingly to assist in the coding. This ranges from computer-supported applications, where the computer is used to provide search facilities in a file with codes and their descriptions, to a fully

automatic data processing application ('automatic coding'). To date, however, automatically coding all answers correctly has not been feasible, and the question is whether it ever will be. But it is not a requirement that coding should be fully automated. Partially processing such information can already result in substantial efficiency gains. Another benefit is that automatic coding is bound to increase, for cases that are not too difficult, consistency of the answers (codes), without a loss of quality and possibly even with an improvement of quality; for computer-aided coding, an audit trail can render the process well-defined. Obviously, special efforts are required to make the coding software suitable for this purpose.

In automatic coding, there are two big problems that must be dealt with:

1. To interpret the natural language descriptions, and
2. To link these descriptions to the classification that is used.

What is meant in the first point is primarily that the text is alphanumeric, not so much that it could be handwritten if a paper questionnaire is used . In fact, it is preferable that the text is not handwritten, as this is an additional complicating factor. A computer program must choose which code best fits a description. The problem with coding open text is that many complications can arise, such as:

- Spelling problems
- Grammatical problems (relationships between words, syntax)
- Semantic problems (meaning of words, concepts, sentence fragments, a single sentence, several sentences)
- Interpretation problems (which code from the classification best fits a description).

A complication that can arise in conjunction with this last point is that, viewed from the classification perspective, a description may be incomplete, or that it may relate to two or more different codes. These complications may be due to the fact that a respondent is not likely to be familiar with the classification used, and therefore can provide ambiguous or irrelevant information, or information that lacks detail or is too detailed. Furthermore, it is possible that the classification has been set up purely from a theoretical perspective, without taking into account how to map descriptions to these codes.

In this document, *coding* refers to the activity with the goal of converting descriptions (which are represented as strings of symbols) to a code, originating from a classification. *Coding* often refers to coding by a specialist coder. In this document, this is called *interactive coding*. Coding with the help of a computer program is referred to as *automatic coding* (if the decisions about individual records are not taken by a person) and *computer-supported coding* (if, in a large part of the cases, the computer/an algorithm does not make any coding decisions but only presents suggestions to a human coder, or acts as an electronic reference file or index). *Coder* refers to a person that concentrates on coding according to one or several classifications. This could be a full-time coder at the statistical office, or *specially trained* interviewer in the field.

2.2 Elaboration

Coding an open-text question is a process of interpreting an answer in terms of a predefined set of possible answers. This choice is sometimes made by respondents, during an interview or when filling in a questionnaire, possibly with an interviewer's assistance. However, this choice can also be made

afterwards by coders, at the statistical office, lacking the feedback of the respondent. Because this manual coding is a rather time (and money)-consuming process, automating the process is extremely worthwhile. This is known as automatic coding. In this process, descriptions (answers from respondents in their own words) are the input, and the output is a set of codes related to a certain classification.

When respondents are permitted to give an answer in their own words, this gives them a lot of freedom. In addition, this prevents a situation where the respondents have to know the classification (which often requires specialist knowledge to be understood and used) or where respondents do not agree with the answer selection provided. A disadvantage, however, is that this must be followed by a rather expensive, time-consuming and error-prone coding process in order to code these answers. For that matter, it is highly questionable whether the answers provided always contain the precision and details that are desired or needed in order to code according to a given classification. To sidestep this problem, it is also possible to attempt to use a number of simple closed questions, and then to arrive at a desired code using a derivation scheme. As a result, it is possible to exert influence on the desired type of information and the detail level of the answers (see, for example, Hacking et al., 2006).

Before answers to questions can be used to produce statistical results, coding is indispensable. As a matter of fact a kind of coding is also applied if a closed question is used, but in that case, it is the respondent who does the coding, and has to decide which answer is best among the possible ones. As a rule, coding can be done at different places in the data collection or throughput process steps, as indicated in Table 1.

In practice, combinations of the four options provided in Table 1 are generally always used. The selection of the options is often based on shifting the effort involved and the difficulties of the coding. The 'most convenient' approach depends on a large number of preconditions, such as:

- The *domain* or *area of application* of the question (including the 'hardness' / 'softness' of the question). 'Gender' concerns a harder piece of data than 'opinion about the government'. The first is more stable than the second and, furthermore, generally easier to indicate;
- The expertise of the respondent or the interviewer;
- The structure and complexity of the classification;
- The desired stability of the coding, i.e., how much or how often does the classification change over time?
- The number of respondents (or the net sample size);
- The input medium;
- The form of the source material: separate words, statements, short sentences, paragraphs;
- The desired balance between quality, output level and efficiency of the coding method;
- The desired speed ('throughput time') of the processing;
- The available budget;
- The desired detail of the coding results;
- The desire to make the coding process reproducible and transparent.

Table 1. Possible places to code and by whom/what?

Coder?	Where?	Type of survey	Advantages	Disadvantages
Respondent	Field	CAWI	• direct feedback	• no knowledge of the classification
Interviewer	Field, NSI	CAPI (field), or CATI (NSI)	• direct feedback	• superficial knowledge of the classification[1]
Coding expert	NSI	PAPI, CAWI, CAPI, CATI	• expert knowledge of the classification • can also use extra information that was included	• direct feedback not always possible (sometimes possible for businesses) • feedback is very time-consuming • coding may be inconsistent • not (always) transparent
Automatic coding tool	NSI	PAPI, CAWI, CAPI, CATI	• fast, consistent coding • coding knowledge is specified in a system and is therefore transferable • can be made transparent (audit trail) • can operate day and night	• no direct feedback • only the relatively simple cases are coded (but that is often the bulk)

When descriptions are being coded, errors can be made, either by the coders or by the coding program used. Insight into this can be gained through experiments (double blind coding), possibly depending on the detail level of the classification used.

In coding, both interactive and automatic, an optimum must continually be found between maximising the yield (the coding percentage) and maximising the quality (that is, minimising the number of errors). There is also a third maximisation to consider: the smallest possible effort (from the employer's perspective, to control costs, etc.). An important means of preventing incorrect coding is by establishing a *doubt category*. Traditionally, human coders were not permitted to have a doubt category (or only a very small one)[2], but this is allowed for an automatic coding program. The records that are rejected by such a program because of difficulties encountered, are subsequently presented to human coders for coding. In addition, using an interactive coding module (based on an informative

[1] The amount of knowledge of the classification that an interviewer must have depends very much on the interactive coding tools as used in the CAPI/CATI tool. More knowledge may increase coding accuracy and/or rate, but may also increase costs as the interviewers must be (re)trained.

[2] A lot of classifications contain a code "other …" at many places in the classification tree, which allows the human coder to "code" not sufficiently specified answers.

base) during CAPI, CAWI or CATI allows an escape from the conflict between yield and quality: such a module can give feedback during the interview to help and reduce ambiguities or vague answers (for example, see Hacking, 2006).

Experience has shown that nearly every source and every coding contains a large fraction of easy records to code, and a smaller fraction of difficult records to code (this situation is often referred to by the 80% / 20% rule, but these percentages should not be taken too literally). Automatic coding focuses mainly on the easier fraction of records to code, which represents the bulk of the material to be coded.

The automatic classification techniques can generally be divided into two groups:

1. ***Language-based***: Here, we really look at the meaning of the words, and make use of language-specific attributes, such as grammar and the relationships between words and concepts (such as synonyms, hyponyms, hyperonyms, etc.)
2. ***Statistical***: Here, descriptions are only viewed as a collection of words, which are often described by a sparse vector $Z = \{w_1, \cdots, w_n\}$, where n is equal to the number of words occurring in the vocabulary, and w_i the frequency of word i in the description. As a rule, the word order is not included as input for the classification. We could view this approach as classifying a house by first breaking it down and then looking at the stones in the pile of rubble. The assumption used here is known as the *bag-of-words*[3] assumption.

These two approaches – language-based and statistical – are extremes. It is very well possible that, in practice, a mixed form will be selected. This could involve, for example, an approach with some 'light grammatical pre-processing', followed by automatic coding based on statistical techniques.

2.3 Comments about classifications and coding

Here we want to take a moment to examine classifications in this section. A classification provides the codes that should be associated with the descriptions provided by respondents, (if this is possible, which is not guaranteed). Some examples of large hierarchical classifications are:

- NACE – Standard Industrial Classification
- NSTR, PRODCOM – classifications of goods

Mostly, classifications must be considered as given, only to be changed by special committees responsible for their maintenance.

In coding, use can be made of classifying principles that form the basis for a classification, such as the different dimensions that could play a role. Often, these dimensions can be mapped onto the different hierarchies in a classification tree: in figure 1, level 2 (codes 1.1 and 1.2) may relate to the concept inside or foreign trade, e.g. It would be a good idea to explicitly describe these classifying principles with a classification. Unfortunately, in practice, these kinds of principles are not always explicitly formulated, which means that one has to make guesses about them. It is also possible that a classification is set up based on clear principles, but that the practical situation forces compromises to

[3] The assumption that, for a description, only the separate words that occur play a role, and not the order and the combinations of these words in the description.

be made, or even forces some principles to be violated. These inconsistencies in the classification will hinder the coding of text towards itself.

In a classification based on a tree structure, it is possible to assign codes to the nodes (or: vertices) such that they reflect this structure.

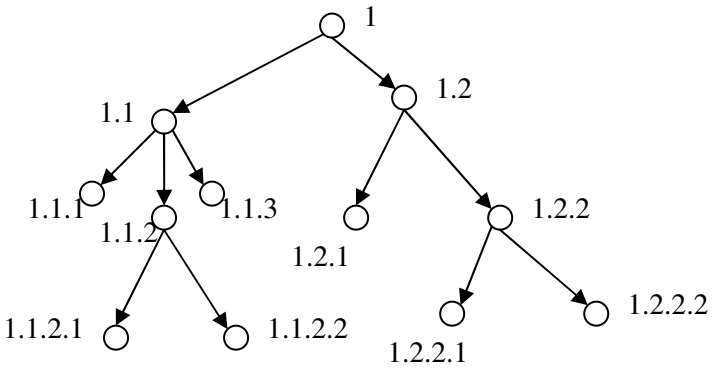

Figure 1. Example of a directed tree with labels for the nodes

Classifications consist of a set of categories, which also have a relationship among themselves. This relationship moves from general to more specific (i.e., in the direction of the arrows).

To clarify the difficulties that may arise from inconsistent classifications we will describe a few peculiarities that can occur in classifications. In the current Dutch standard industrial classification, we have a category 'clothing' that can be split in different ways, depending on the context. The 'manufacture of clothing' is split into the 'manufacture of outerwear' and 'manufacture of underwear'. However, in the clothing retail trade, this category is split into: 'retail trade of women's clothing', 'retail trade of men's clothing', 'retail trade of children's clothing' (and, perhaps, also the 'retail trade of baby clothing').

In the Dutch standard industrial classification (SBI-93), different splits of clothing are possible:

- According to age: clothing for babies, toddlers, children, teenagers and adults.
- According to gender: women's and men's clothing.
- According to how it is worn: underwear and outerwear.
- According to use: work clothing (including uniforms), leisure clothing, clothing for going out, clothing for formal events (for weddings, for academic events, such as receiving a PhD, for a fancy ball, receiving a medal, etc.), and everyday clothing.

Depending on the industry sector, the above splits may or may not apply.

Another example, also from the Dutch standard industrial classification (SBI-93), concerns agricultural products. The activity associated with these products determines the splitting level of these agricultural goods:

- *Cultivation* of vegetables. (Additional detail about these vegetables is not necessary.)

- *Wholesale trade* in potatoes for seed and potatoes for the retail market. (A split into the type of potato is necessary in order to code the type of wholesale trade.)
- *Processing* of potatoes. In other cases, different splits can occur.

Another problem arising from the classification definition is the following: how far apart, conceptually, are the categories? If there are two categories that are rather close together, then, in practice, it will be difficult to make a distinction between the two, based on descriptions. In this case, the descriptions must be quite precise. This can be difficult for a respondent who is not considered to be familiar with the classification, because this person will not be aware of a – probably quite subtle – difference between the two categories.

Finally, coding problems may arise from the lack of examples for certain codes: such a lack makes it difficult to construct an informative base or train a machine-learning approach for these codes.

These different problems require different solutions that, however, are not always available. After all, there are more issues that play an important role in official classifications than these methodology-related matters. Usually, a classification was already officially established at an earlier date. This must be viewed as given for the coding process. Changes to a classification generally are made with regards to the subject matter itself and not with observation / measurement in mind nor the coding problems that the classification poses when used in practice. It would be preferable if a classification was set up also addressing these issues. Experiences could then be used to adapt a classification and make it useful and applicable. There is little sense in retaining a theoretically ideal classification that cannot be used in practice due to observational or coding problems.

2.4 Misconceptions about coding

Here we want to point out several widespread (and persistent) misconceptions.

1. '*Low quality input versus high quality output*'. This misconception conflicts with the truism: 'garbage in, garbage out'. A source of the trouble may be that the classification distinguishes codes/subjects that seem the same to a 'naive' respondent. In this case input data are obtained that do not offer adequate information for correct and sufficiently detailed coding. The remedy for this could be as follows (while interviewing): in the event of vague / ambiguous texts, one can permit an appropriate code (a 'doubt category', or a less detailed code) or multiple (detailed) codes in instead of a single code, possibly with probabilities assigned to the possible codes. Other solutions are better wording of the questions to elicit more precise answers or the use of interactive coding modules (when using CAPI, CATI or CAWI) to refine initial answers through feedback using further questions.
2. '*Less detailed codes and therefore a higher yield*'. This may not be true in case the classification used is skewed at various levels, in the sense that the distribution of its scores in the population is skewed. An extreme example is shown in Figure 2, which is skewed on all hierarchical levels. In general, the link between a description and a less detailed code is not necessarily less ambiguous: if we would code all the occupations in the government by a code 'occupation in the government', this would not necessarily simplify the coding. For the practical situation, such a skewed distribution may imply that we can obtain a reasonable coding result with relatively little effort (i.e., by coding the most frequently occurring codes),

while a relatively large amount of effort must be put into the remaining part. If the coded corpus has a skewed distribution, then, typically, there are classes in the tail of the distribution for which too few examples are known to make a reliable and complete classification or classification model. Coding less detailed, i.e., at a higher level in de classification tree, doesn't present a solution, since this skewedness is often present on multiple levels. In Figure 2, for example, the skewedness occurs at both lowest levels. This skewedness may be due to classifications that are designed in a rather unbalanced way. It is also possible that they become more skewed due to changes in the population: certain NACE codes gradually disappear, while others start to occur more often.

3. '*80% automatic coding is attainable'*. The general opinion is that coding is an easy task. However, in our experience a yield of 40% of automatically coded records is a more realistic figure than the 80% claimed[4]. The yield strongly depends on the complexity of the coding problem. This also involves a difference in definition: if the classification literature for example refers to, for example, a percentage of 70% coded records, then this means that, of the 1000 texts, some 700 were correctly coded automatically. However, when the system must not only code, but also 'guarantee' some quality level of coding, the coding yield drops significantly. Another frequently occurring reason for overly optimistic estimations in the literature is that experiments have only been performed with codable descriptions, and that the non-codable descriptions have been ignored. It also plays a role in the validation of a coding system.

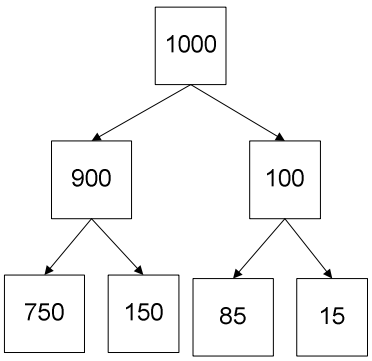

Figure 2. Example of an asymmetrically distributed corpus at all levels

3. Design issues

The approach to a new coding problem is driven by the following aspects:

- *The material that is available*: Is there already coded material in electronic form? Methods based on already coded material (as described in "Coding – Automatic Coding Based on Pre-coded Datasets") use this to train their machine-learning model. Methods that are applicable when there is no coded material (as described in "Coding –

[4] The coding rate of 40% applies to the more challenging coding problems, e.g., when coding occupation with more than 1000 codes. If the respondents are experts at the classification, this rate may increase. In addition, coding very simple classifications (Municipalities or Country) may result in coding rates over 95%.

Automatic Coding Based on Semantic Networks") construct an informative base to guide the coding of texts; having coded answers may help the construction of such informative bases.

- *The available software*: this point is strongly related to the previous point. Depending on the available software, one may start to construct an informative base or code a representative set of descriptions to feed a machine-learning method.
- *The coding method used*: manual, interactive or automatic / in batch. These methods can be combined into a strategy, e.g., start with interactive coding in the field followed by automatic coding at the statistical office (see "Coding – Different Coding Strategies"). The different individual approaches have been described in "Coding – Manual Coding", "Coding – Computer-Assisted Coding", "Coding – Automatic Coding Based on Pre-coded Datasets" and "Coding – Automatic Coding Based on Semantic Networks".
- *The intended quality of the coding*: is it important that a lot of descriptions are coded, and that errors are accepted in some cases? Or does one take a more cautious attitude and should every code be correct with high probability? Besides measuring the quality, this may also give input for the enhancement/extension of the informative base or retraining of the machine-learning model. For more information see "Coding – Measuring Coding Quality".
- *Maintenance*: How well can a coding strategy be kept up-to-date? The classification may alter its form year to year, new answers may be used to enrich the informative base. How to construct and to maintain such an informative base is described in "Coding – How to Build the Informative Base".

4. Available software tools

Many of the methods described in the literature and their implementations are still in an academic phase, making their real-world application not (yet) really feasible. The following generic coding tools have been developed at several statistical offices or companies:

- Blaise: The Blaise suite contains several possibilities to search through classification(tree)s.
- SICORE from INSEE (see Rivière, 1994): This is based on decision trees.
- GCODE (successor of ACTR) from Statistics Canada (see Wenzowski, 1988): This is based on a kind of Nearest Neighbour technique.
- StafS from SPSS.
- Cascot from the Warwick Institute for Employment Research of the University of Warwick, UK. (See http://www2.warwick.ac.uk/fac/soc/ier/software/cascot/.)

In addition to these there is an abundance of specific coding tools, geared at a particular application. Every NSI has probably a few of them. They are usually not supported and are of limited use outside the NSI where they are used.

5. Decision tree of methods

6. Glossary

For definitions of terms used in this module, please refer to the separate "Glossary" provided as part of the handbook.

7. References

Hacking, W. J. G, Michiels, J., and Janssen-Jansen, S. (2006), Computer Assisted Coding by Interviewers. Blaise Users Conference 2006.

Hacking, W. and Willenborg, L. (2012), *Coding – interpreting short descriptions using a classification*. Contribution to the CBS Methods Series, Statistics Netherlands, The Hague and Heerlen.

Rivière, P. (1994), The SICORE automatic coding system. Working Paper, Conference of European Statisticians, Cork.

Wenzowski, M. J. (1988), ACTR – A Generalised Automated Coding System. *Survey Methodology* **14**, 299–308.

Interconnections with other modules

8. Related themes described in other modules

1. Micro-Fusion – Object Matching (Record Linkage)
2. Coding – How to Build the Informative Base
3. Coding – Different Coding Strategies
4. Coding – Measuring Coding Quality
5. Derivation of Statistical Units – Derivation of Statistical Units

9. Methods explicitly referred to in this module

1. Coding – Manual Coding
2. Coding – Automatic Coding Based on Pre-coded Datasets
3. Coding – Automatic Coding Based on Semantic Networks
4. Coding – Computer-Assisted Coding

10. Mathematical techniques explicitly referred to in this module

1.

11. GSBPM phases explicitly referred to in this module

1. 5.2 Classify and code
2. 5.5 Derive new variables and statistical units

12. Tools explicitly referred to in this module

1. Blaise
2. ACTR / GCODE
3. Cascot
4. SICORE
5. StafS

13. Process steps explicitly referred to in this module

1. Input
2. Throughput

Administrative section

14. Module code

Coding-T-Main Module

15. Version history

Version	Date	Description of changes	Author	Institute
0.1	03-06-2012	first version	Leon Willenborg	CBS
0.2	31-10-2013	revised version	Wim Hacking	CBS
0.3	24-01-2014	revised version	Wim Hacking	CBS
0.4	19-02-2014	revised version	Wim Hacking	CBS
0.4.1	20-02-2014	preliminary release		
1.0	26-03-2014	final version within the Memobust project		

16. Template version and print date

Template version used	1.0 p 4 d.d. 22-11-2012
Print date	21-3-2014 18:05