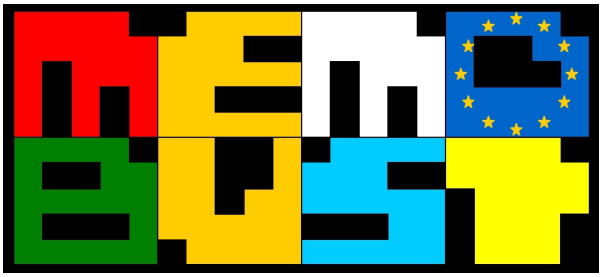

This module is part of the

Memobust Handbook

on Methodology of Modern Business Statistics

26 March 2014

Method: Automatic Coding Based on Pre-coded Datasets

Contents

General section

1. Summary

For a number of variables in questionnaires, one wants the answer in closed form, e.g., “city”; this is a relatively simple classifying task. Sometimes this task is much harder, e.g., when trying the get a code for occupation. One approach is to ask an open question (“what is your occupation”) and then try and code this text at the statistical office. For the sake of efficiency, that coding process will start by an automatic step.

Here we will describe the coding of open text answers based on existing sets of correctly coded answers. We will briefly look at some existing techniques and then focus on one method in more detail as an example.

2. General description of the method

We will first discuss briefly the general approach for (short) text classification in the literature, as described more extensively in Sebastiani (2001) and Joachims (2002).

The literature describes several techniques to classify text if a corpus is available, that is, previously coded (and verified) descriptions. Most of the literature concerns numerical rather than textual data, and is known as ‘pattern recognition’, ‘data mining’ or ‘business intelligence’. Recently, the application of these techniques and new ones to text has received much more attention due to the data explosion taking place at the internet; the terms used are ‘text-mining’, ‘web-mining’, etc. A number of these have been described in Sebastiani (2001). These are techniques based on data mining techniques, such as *K-Means*, *Naïve Bayes* and *Support Vector Machines*. For most of these techniques a description is represented by a very large sparse vector, where a 1 means that the word is present in the input and 0 that it is not; for this model, the order of the words is of no importance (the so called bag-of-words assumption). In order to reduce the size of these vectors, extra pre-processing techniques are used, such as Latent Semantic Indexing (Sebastiani, 2001). These classification problems are close to but not the same as the coding problem considered in the present document. The descriptions used in coding usually do not contain more than 10 words.

At statistical offices, automated coding based on pre-coded datasets (so called coding dictionaries) has been implemented using different matching algorithms. In the late sixties, the US Census Bureau realised different coding systems, based on “dictionary algorithms”, that build the dictionary on the base of a large sample of verbal responses manually coded by experts (Lyberg and Dean, 1992). The simplest algorithms searched for an exact match, while other ones were based on a classifier, that is to say, a word or a set of words corresponding to a specific code and whose occurrence is not lower than a defined level. Other coding systems use the so-called *‘weighting algorithms’*, which are based on a measure of similarity between the text to be coded and those of the coding dictionary. In this way, these methods consider not only *‘perfect matches’*, but also ‘*partial matches’* between input texts and texts of the dictionary[1]; this approach is comparable to *K-Means.*

[1] For this type of (so called fuzzy) string matching, see Hall and Dowling (1980) and Navarro (2001).

To illustrate the coding process based on pre-coded data, we will look at a method as applied at Statistics Netherlands, which belongs to the *'weighting algorithms'*.

As mentioned, this method is a nearest-neighbour technique, combined with a choice of a specific distance measure between two descriptions: first, each word (or combination of two words) is assigned a weight that indicates how specific that word is in the training set. This can be illustrated using Figure 1.

Figure 1. Examples of conditional distributions over (sorted) occupation classes given the key words 'lawyer' (a) and 'employee' (b).

Figure 1 shows histograms of $P(Code_i|Word)$ (the probability of $Code_i$, given that *Word* is in the description). Subfigure (a) of Figure 1 shows the probability distribution (sorted by frequency) for *Word* = 'lawyer', and (b) depicts the histogram for *Word* = 'employee'. The asymmetry of the distribution indicates how specific a word is. Following Chen et al. (1993), this specificity is quantified as[2]:

$$F(W) = \frac{\sqrt{\sum_{i=1}^{n} P(C_i \mid W)^2}}{n}, \qquad (1)$$

where n is the number of codes C_i where *W* occurs in the description.

Based on this definition, a word such as 'lawyer' is assigned a higher weight than a word such as 'employee', when comparing two descriptions. Note that, in this way, words with little meaning such

[2] Other measures for this are: entropy and the skewness of the distribution.

as ‘and’, ‘the’, etc. (stop words) naturally have a minimal effect, because they are given a low weight if they had not been filtered out earlier in the pre-treatments. The pre-processing step in which stop words are removed could, in principle, be omitted.

Based on formula (1), defined *per word (*or *word combination),* we can define a measure for the similarity of two *descriptions* D_1 and D_2 (after removing the words that occur multiple times in both descriptions):

$$Similarity(D_1, D_2) = \sum_{x_i \in D_1 \cap D_2} F(x_i)$$

where

$$D_1 = \{a_1, ..., a_n\} \quad \text{and} \quad D_2 = \{b_1, \ldots, b_m\} \ .$$

In other words, the similarity between two descriptions is determined by adding up the weights $F(x_i)$ of all shared words[3]. A new description D is compared with all the descriptions present in the training set, and the best N descriptions are retained. The code that occurs most often among the codes associated with the N best fitting descriptions is either selected (provided that it occurs frequently enough) or rejected, i.e., the algorithm cannot assign a code and the description will be presented to an expert.

Any matching algorithm, not just the example algorithm described above will return a list of possible codes along with some score. For automatic coding to work, this list must be reduced to either

- zero, i.e., even the top score code doesn’t have enough “confidence”;
- one, i.e., the description gets classified.

The first choice is important for practical implementations: a coding algorithm cannot classify every description, so some fraction of the set of descriptions must be passed on coding experts. This means that the automatic coding algorithm must do two things:

- try to classify the description;
- try to assign a measure of confidence in the assigned classification.

In the current example above the selection is done as follows. Let C be the most frequently occurring class among the N descriptions. C is selected unconditionally if $\#C \geq f_{GOOD} * N$, where $\#C$ is the frequency score of C among the scores associated with the N descriptions. If it is true for #C that ($f_{BAD} * N \leq \#C \leq f_{GOOD} * N$), then the selection of C is doubtful, and is presented to a specialist coder on this topic, who must then decide whether or not to assign C. If $\#C < f_{BAD} * N$, the selection of C is rejected: it simply does not occur frequently enough. The choices of f_{GOOD} and f_{BAD} are empirically determined using the data; for a higher quality (but less coded answers), these can be higher than for a lesser quality (but more coded answers).

[3] The use of synonyms, hyponyms and hypernyms could further increase the returns of the matchings; this has not yet been studied.

3. Preparatory phase

To prepare this method, one needs a sufficiently large pre-coded dataset. Not only the number of records is important, but one also has to check if each code has a sufficient amount of records; otherwise the classification becomes rather unreliable for those codes.

4. Examples – not tool specific

5. Examples – tool specific

To our knowledge, there is only one general coding system, based on pre-coded texts, that is currently available: ACTR (Wenzowski, 1988); this system allows the coding of texts based on data-mining techniques and has many pre- and post-processing options to make it suitable for any given classification (ACTR has been enhanced and is currently called GCODE). In addition, there are two other systems that are designed generically, but are only used at the statistical office where they were created (Hacking and Janssen-Jansen, 2009; Hacking and Willenborg, 2012; Rivière, 1994).

6. Glossary

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

7. References

Creecy, R. H., Masand, B. M, Smith, S. J., and Waltz, D. L. (1992), Trading MIPS and memory for Knowledge Engineering. *CACM* **35**, 48–63.

Chen, B., Creecy, R., and Appel, M. (1993), Error control of automated industry and occupation coding. *Journal of Official Statistics* **9**, 729–745.

Hacking, W. J. G. and Janssen-Jansen, S. (2009), The coding of economic activity based on spreading activation. Report, Statistics Netherlands, Heerlen.

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.

Hall, P. V. and Dowling, G. R. (1980), Approximate string matching. *Computing Surveys* **12**, 381–402.

Joachims, T. (2002), *Learning to classify text using support vector machines*. Kluwer.

Lyberg, L. and Dean, P. (1992), Automated Coding of Survey Responses: an international review. Conference of European Statisticians, Work session on Statistical Data Editing, Washington DC.

Navarro, G. (2001), A guided tour to approximate string matching. *ACM Computing Surveys* **33**, 31–88.

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

Sebastiani, F. (2001), Machine learning in automated text categorization. *ACM Computing Surveys* **34**, 1–47.

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

Specific section

8. Purpose of the method

The automatic coding step takes place just after the data have been collected, in most cases data from interviews. These interviews contain a few fields that are input for the coding step, e.g., “production of wooden crates” as input for the coding of economic activity. For simple classifications, e.g., “nation of birth”, a closed question can suffice in the interview; for more complex classifications such as education or occupation, a closed question will lead to long lists and the quality of the response will decrease rapidly. For that reason, it is often more logical to use an open question in the interview and code this answer at the statistical office; for reasons of efficiency, it is better to start coding automatically followed by a manual or a computer-assisted coding step (texts not coded automatically can be analysed by expert coders manually or with the computer support)

The method described here can be used to do the automatic coding step.

9. Recommended use of the method

1. Recommendations on the use of the different methods for coding (automatic or assisted) are given in the module “Coding – Different Coding Strategies”: the decision about which is the most suitable coding approach to be adopted in a survey depends on different correlated factors. If it has been decided to use automatic coding, one needs a training set of coded descriptions that is available in electronic form, and a correct code (after verification) that is assigned to each description.

10. Possible disadvantages of the method

1. The main disadvantage occurs when the classifications changes (which is not uncommon). Especially, if it is a large change, many coded records need to be recoded and often there does not exist a 1:1 mapping between old and new codes. As a result, a large portion of the pre-coded material needs to be recoded. If one uses a semantic network, the resulting amount of rework is much less; of course this depends on the way how the network is constructed.

11. Variants of the method

1.

12. Input data

1. During the coding phase, the input is quite simple: a textual description, in most cases no more than 10 words. During the construction of the “coding machine” the input consists of pre-coded datasets that are used to train the coding algorithm, i.e., a set of records containing a description and a correct code.

13. Logical preconditions

1. Missing values

    1.

2. Erroneous values

    1.

3. Other quality related preconditions

    1.

4. Other types of preconditions

    1. The input text to be coded should not be too large; in general, this will result in many possible codes for this input text by the method.

14. Tuning parameters

1. In general, all practical automatic coding algorithms need a *score cut-off value*, to make a selection which descriptions are coded and which need manual or assisted coding. In our experience this parameter is rather robust.

15. Recommended use of the individual variants of the method

1.

16. Output data

1. Per description the following is derived by the method:

    - a score;
    - a classification code.

17. Properties of the output data

1.

18. Unit of input data suitable for the method

Incremental processing

19. User interaction - not tool specific

1. None

20. Logging indicators

1. To monitor the quality of the coding process, all relevant parameters influencing the classification must be stored for later analysis, i.e., comparing (a subset of) the automatically coded texts with correctly coded.

21. Quality indicators of the output data

1. The method is tested by splitting a pre-coded data set into two parts: 10% test set and 90% learning set. After training the method with the learning set, the method is tested by feeding it the descriptions from the test set; after coding, both set of codes (N) from the algorithm and the test set are compared.

A small part of the test set will be rejected ($N_{rejected}$) by the algorithm (e.g., because it's too vague), and the non-rejected matching part ($N_{Coded} = N\text{-}N_{rejected}$) is used for the comparison; the number of descriptions that were coded correctly is $N_{CorrectlyCoded}$. As described in the module "Coding – Measuring Coding Quality" we can use two measures to quantify the quality of the automatic coding method:

$coding\_rate = \frac{N_{Coded}}{N}$, i.e., what fraction was coded;

$precision\_rate = \frac{N_{CorrectlyCoded}}{N_{Coded}}$, i.e., what fraction was coded correctly.

22. Actual use of the method

1. This method was used by the US Census Bureau already in the nineties (Creecy et al., 1992). It is also used at INSEE (the SICORE system; Rivière, 1994), at Statistics Canada and ISTAT (ACTR, now G-CODE; Wenzowski, 1988). This method is also used at Statistics Netherlands (Hacking and Willenborg, 2012).

Interconnections with other modules

23. Themes that refer explicitly to this module

1. Coding – Main Module
2. Coding – Different Coding Strategies
3. Coding – Measuring Coding Quality

24. Related methods described in other modules

1. Coding – Manual coding
2. Coding – Automatic Coding Based on Semantic Networks
3. Coding – Computer-Assisted Coding

25. Mathematical techniques used by the method described in this module

1.

26. GSBPM phases where the method described in this module is used

1. 5.2 Classify and code

27. Tools that implement the method described in this module

1. G-CODE: see Wenzowski (1988)
2. SICORE: see Rivière (1994)

28. Process step performed by the method

Coding

Administrative section

29. Module code

Coding-M-Automatic Coding Based on Pre-coded Datasets

30. Version history

Version	Date	Description of changes	Author	Institute
0.1	02-04-2013	first version	Wim Hacking	CBS
0.2	20-01-2014	following review by Stefania Macchia	Wim Hacking	CBS
0.3	30-01-2014	following review from EB	Wim Hacking	CBS
0.3.1	30-01-2014	preliminary release		
1.0	26-03-2014	final version within the Memobust project		

31. Template version and print date

Template version used	1.0 p 4 d.d. 22-11-2012
Print date	21-3-2014 18:06