Discussion Paper

Semantic networks for automatic coding (v2)

Léon Willenborg

December 15, 2021

In this paper, a methodology for automatic coding is suggested which is fairly general. An application in the area of business applications has served as the inspiration for the approach is described. Characteristic for this approach is that each code (say for a business activity) is characterized by one or more combinations of code words, or C-words. These combinations of C-words can be seen as definitions of the various codes that are used. It is assumed that the order of the C-words is irrelevant to describe a code. Because it is unlikely that people will use exactly those C-words when describing a business activity, synonyms, hyponyms and hyperonyms for the C-words are needed as well. These words connect the codes used in the classification with the descriptions provided by the respondents. A semantic network is used to link the descriptions to the codes.

The paper describes the approach, and also how to produce such a semantic network, and which software tools are handy to support this activity.

This paper is an updated version of [7]. As the original version of this paper was written in an older version of Word, it was decided to convert to LaTeX for the new version, for which all figures were redrawn and in some cases replaced by improved versions.

Key words: text interpretation, natural language processing, computational linguistics, derivation, code assignment, classification, knowledge representation, semantic network, ontology.

1 Introduction

1.1 Background

This paper[1)] is an updated and expanded version of [7]. The text of this paper (written in an older version of Word) was the starting point for the present document. It was first translated by Pandoc into a crude LaTeX-document. This was improved by additional ‘manual labour’. Also new pictures were drawn and included. The text was gradually updated and adapted to a new application (see below), however without forgetting the original one.

The first edition of the present paper describes a methodology for automatic coding that has its origins in a project at Statistics Netherlands. The aim was the development of an interactive tool that could be used at the Chambers of Commerce in The Netherlands to elicit detailed information about businesses and their activities. The input for this tool was supposed to be descriptions of the activities of businesses. Businesses were supposed to provide the descriptions of their (intended) activities, phrasing it in their own words. A desk clerk should input the information into the system, using this tool. It was not assumed that the respondents had any knowledge of the business classification that was used to code the information provided.

1) The views expressed in this paper are those of the author and do not necessarily reflect the policies of Statistics Netherlands. The paper benefited from the help of two colleagues. Sander Scholtus reviewed a recent draft version of the paper and came up with a list of possible improvements of the text. Arnout van Delden asked me a series of very specific questions about the method proposed in this paper. With an increased understanding of this method he would be able to help improve software that had been developed at CBS to classify business activities. From those questions I learned to sharpen some points in the text. The author is grateful to both of them.

The project delivered a prototype tool that was later upgraded to a coding tool that was adopted and used by the Chambers of Commerce.

The current paper describes ideas that have been used in the coding tool. The description, however, has been generalized, in the sense that it does not only apply to business activities. It also contains ideas that have evolved and matured since the original project was finished. In this second version of the paper, material has been added from a more recent project on cyber crime, that is, loosely speaking, computer crime. The goal of this project is to detect such crimes in reports to the police by victims (in their own words) or crime reports from the police in case of arrests.

The initial idea was that the approach in [7] could be applied here as well, and so it was tried. In any case, it was an interesting case to test the original approach to a new application, and see to which extent it works straight away, or requires some extra effort to make it work, or fails altogether. In the paper the original application serves as the source of inspiration for the approach and the application to cyber crime is a test case. Because the cyber crime application is of current importance it is emphasized in the present paper. The earlier work is there as preparation and motivation, and as a convenience to the reader of the present paper: there is no need to consult the earlier version.

1.2 Coding

Coding in official statistics is a process of 'deducing (the value of) a variable'. A *deduced variable* is a variable whose value for a sample entity is a function of the values of some other, auxiliary, variables that have been observed. Typically, the auxiliary variables are categorical variables. What sets coding apart is that at least some of the auxiliary variables are text variables. What makes coding difficult is the fact that the auxiliary variables contain free text, that is, phrases chosen freely by the respondents. So to deduce the value of the coded variable (usually an element of a classification) there is the problem of understanding 'written text' or 'written language'. The derived variable typically takes values in a classification, which is usually representable by a tree.

So coding is the process of interpreting descriptions given in writing (of a business activity, an occupation, a disease, etc) with respect to a suitable classification. The information is usually provided by respondents who are not supposed to have any knowledge about the classification that is used to interpret their response. This response is typically about the education one received, the job title, the goods being produced, repaired, transported, etc, the services provided, the business activity of a company, diseases that persons can be suffering from, the causes of death of persons, etc. These variables have in common that the corresponding domains are big, and that a rather complicated tree structure in the form of a classification tree is defined on them. It requires expertise of the various classifications, and in particular the classifying principles underlying them, to provide the correct answer for each respondent. A typical respondent cannot be expected to possess such knowledge. So the solution is that the respondent provides an answer to such a question in his own words, and this answer is interpreted by either a coding expert or a coding program, or both.

Coding can also be seen as a translation problem, translating descriptions into codes. In automatic coding this translation is carried out by computer, completely or partially. If it is done completely, the user information is coded without asking for feedback information from the person who provided the information. If it is done partially, the coding system needs this feedback to make certain decisions that lead to a code. It can also decide that no such code can be generated with confidence on the basis of the information provided. Fully automatic coding systems are currently an illusion, and probably always will be. The key problem is that the descriptions that are fed into the coding system may contain insufficient information to lead unambiguously to a single code. In practice, some form of feedback may be needed to clarify incomprehensible, ambiguous or inconsistent information. If such feedback is not possible one can switch to manual coding and try the find a code, where a coding expert looks at the information available.

Manual coding was in fact the first form of coding. It is rather labour intensive. Since computers are being used at statistical offices, they have been used in various forms to assist with coding. Between manual and automatic coding there also is semi-automatic coding, where the coding decisions are still being taken by coders, but this time they are supported by a simple information system, e.g. an electronic filing system with auxiliary information.

In the current paper we focus on automatic coding using a semantic network.

1.3 Automatic coding

Coding can also be seen as a classification problem (with a twist), as studied in areas such as data mining. D-words (as they are called in this paper) in the descriptions provided by respondents are used as predictors of the unknown codes. What distinguishes the automatic coding problem from the numerical prediction problem is that the prediction variables are alphanumeric rather than numeric. The D-words in a description are drawn from some specialized vocabulary, geared at a particular application that can be fairly extensive. Because the predictor variable consists of words rather than of numerical values, one is faced here with some language problems, which are absent from standard classification applications. These problems are due to incorrect spelling of words (problems with orthography), the use of specialized vocabulary, jargon, dialect, etc. (which requires to build a reasonably complete vocabulary), dealing with synonyms, hyperonyms and hyponyms (which relates to a thesaurus function on top of the vocabulary). To cater for all these aspects may require quite some effort.

The goal of automatic coding, in particular as considered in the present paper, is to classify a description, which is a set of D-words, into a code, which is described using a specific vocabulary, consisting of so-called C-words. The link between the D-words and the codes is provided by a semantic network, which embodies various relevant semantic relationships. A semantic relationship is based on the meaning of words or concepts.

1.4 Applications

Automatic coding of answers to open questions in a questionnaire is the typical application of coding in the official statistics setting. However, there are other applications in that area that require descriptions to be interpreted, or classified. We consider two of them.

The first application we want to highlight is about using information of web shops to compile price statistics. As an example consider web shops selling clothes. For collecting price information from such websites they should be ‘spidered’ regularly. This means that information on the clothes on offer in such web shops at that time, and in particular their current prices, is copied and processed to filter out the relevant information to compile price statistics. In order to do so one should be able to recognize clothes items that are found on these websites, and categorize them properly according to some clothes classification.

The detailed descriptions of the clothes items are like the descriptions in the standard coding setting. Furthermore the groups of articles considered are categories in the clothes classification deployed. The situation here is comparable to the standard coding setting: the descriptions chosen for the clothes items are entirely determined by the web shop. Different web shops use their own descriptions. Compared to the standard coding applications, the descriptions are more standardized, contain less writing errors, and use a fairly fixed jargon. Because the information is available digitally, and in bulk form, automatic coding is the type of coding applied. Feedback for clarifying ambiguous, etc. information concerning the items of interest (clothes) is typically not available (nor usually necessary).

The second application we want to put forward that is close to automatic coding, is about searching in large sets of tables published by a statistical office. The user of such tables is allowed to specify a description in his own words about the table or tables he is searching for. Each table is characterized by a short description, stating what the table is about, what are the spanning variables and what information the cells of the table contains, what units are used, etc. The search strings of the users can be compared to the descriptions in the coding setting, and the tables being searched for play a similar role as the codes in the coding setting.

In this case, interaction with the user is possible, in case the search information he provided is ambiguous or otherwise unclear.

1.5 Overview

The current paper describes a semantic network for automatic coding and its deployment. The architecture of such a semantic network is independent of a particular area of application. This implies that a general shell could be used to handle descriptions (spell checking & correcting, parsing, and other purely language related tasks), to make inferences using the semantic network to find codes corresponding with descriptions, and to handle the feedback from a respondent in case a single code cannot be obtained. The semantic network is the core of such a system, and the idea is that it is also separate from the ‘handling’ software and can easily be replaced by another such semantic network for another application. The semantic network is as independent data for the handling software.

We briefly describe how the automatic coding process based on a semantic network is supposed to operate. We start with the basic input, which is a description, provided by a respondent. This is read and parsed by the system and the individual tokens (words) are identified. Possibly the description is first spell checked and corrected.

The words obtained from the description are then matched with the set of D-words in the semantic network. If all words in the description are recognized as being part of this set, we are ready for the next phase. If not, efforts have to be developed to handle the unrecognized words in the description. They have to be added to the semantic network. Coding experts have to look at these words and decide to add them to the semantic network, or rather to incorporate them. This is a relatively laborious job, but an essential one, to let the semantic network ‘learn’. After the semantic network has been extended, the words that were originally not recognized (as they were not in the set of D-words) now should be recognized.

The next step seeks to replace the D-words by so-called C-words. They form, so to speak, a subset of the D-words and they are used to describe the codes in the classification system that the coding system uses. This reduction to C-words is via synonyms and hyperonyms, semantic relations that denote semantic equivalence or generalization, respectively. The codes in the classification are described in terms of combinations of concepts. Each concept is a set of C-words. In the ideal case one code is now found in the coding system, at the desired level of detail. This code is assigned to the description. In case no such code is found, the information in the description is imperfect: either no code is found or more than one code is found (at the right level of detail). In both cases feedback from the respondent is needed in order to arrive at a single code. The description may contain too much information (unnecessary detail) or too little information to distinguish between two codes. The information provided by a respondent may even be inconsistent or incoherent. All this requires a setting where the feedback by the respondent can indeed be given.

1.6 Structure of the paper

The remainder of the paper is structured as follows. In successive chapters various ingredients of a semantic network for coding are discussed. We start in Chapter 2 with discussing words, divided into two groups: C-words and D-words. In Chapter 3 various constructs made from specific types of words are considered, namely concepts, codes and classifications. Then, in Chapter 4, the various kinds of relations that exist between words, concepts and codes (from classifications) are discussed. Chapter 5 is entirely devoted to illustrating aspects of the approach set out in previous chapters by presenting a few examples. In Chapter 6 we discuss briefly how an automatic coding process works using a semantic network could work. We start with the ‘happy flow’, which deals with the ideal case. This gives an insight in the coding process under ideal conditions. In practice many exceptions can occur to the ideal case. We then continue discussing these in brief. Full treatment would require quite a lot of extra space, possibly at book length. In Chapter 7 a summary and discussion of the main points of the approach taken in this paper are given.

Remark ‘□’ marks the end of a **Remark**. □

2 Words

The basic building blocks for a semantic network for coding are words. We consider two classes of words: C-words that are used to define concepts in a classification. These concepts in turn are used to define codes, or, equivalently, nodes in a classification. They form the target part of a coding activity. At the other end, we have the D-words, that is, words that actually appear in descriptions given by respondents. The D-words and C-words may be related through semantic relationships. D-words not connected to C-words either are not useful, as they carry no or little information, or they are new and have not been incorporated into the semantic network.

2.1 C-words: building blocks for concepts

C-words are the building blocks of the coding methodology described in the present paper. What can be built with them are what in the present paper are called concepts. Concepts are used to define codes. A *concept* is represented as a finite set of C-words. This implies that the order of the C-words used to describe a concept is unimportant. It should be stressed that this is an assumption, not an established fact.

The C-words have to be considered as formal words used to define the codes in a formal way. As a matter of fact, it is not even important from which language(s) the C-words are taken. Their semantic content is important, not the (natural) language(s) from which they are taken. Of course, the idea is that one language should be chosen (and specified) and all C-words are supposed to be taken from this language. However, it would be possible to take C-words from a language different from that in which the descriptions are formulated. This is convenient in countries with more than one official language, as it would allow to define the code structure in one language and add appropriate D-words in the other language, as synonyms, hyponyms or hyperonyms, at the appropriate places.

The C-words should be chosen in such a way that they can describe all codes in the classification, including the most detailed ones that will be used, adequately, via concepts. We assume that the most detailed level defines the desired level of coding. It is therefore not necessary to consider more detailed codes. Less detailed codes however cannot be discarded since each higher level code consists of two or more codes. So we exclude the possibility that a node that is not a terminal node has exactly one child-node. In such a case there would be no branching, and such a node could be left out, without losing any information.

This does not imply that respondents cannot give more detailed answers than the coding level requests. But through the use of suitable semantic relationships this information can be channelled to the desired level.

For a given classification the set of C-words to describe its concepts can be viewed as a fixed set. If the classification changes, it may be necessary to modify the corresponding set of C-words accordingly.

2.2 D-words: building blocks for descriptions

D-words are the words that are actually found in descriptions provided by respondents for the topic of the classification. In principle there are two options:

1. Correctly and incorrectly spelled words. In this case the raw input of the descriptions is directly used, without any modifications.
2. Only correctly spelled words. In this case it is assumed that all descriptions are first spell-checked and corrected.

In the first case the set of D-words is bigger than in the second case. The advantage of this set is that it can be fed with the words as they appear in strings. This can occur any time new descriptions are received. But these 'raw' words, if not already in the currently existing set of D-words, have to be interpreted and classified (synonyms, etc have to be specified). In the second case the polishing step (the spell checking and correcting) needed to obtain the words may potentially destroy useful information, assuming that it is done automatically. If it is done interactively, it may take a similar effort as in the first case when dealing with new D-words, not already in the current set of D-words. Some experimentation is probably needed to find out which option to choose for the set of D-words.

Contrary to the set of C-words, the set of D-words is to be considered open-ended, as new words may pop up as long as new empirical material (i.e. descriptions) is collected. In case this happens, such a word should be integrated into the semantic network. This means that it should be connected to the C-words via semantic relationships. These have to be defined by workers maintaining the semantic network used. This can be considered as a learning activity of the semantic network.

At any time the set of C-words is a subset of the set of corresponding D-words. In fact, the C-words associated with a classification can be considered as the starting point for the set of D-words, even without any empirical material (i.e. descriptions). The descriptions are the sources of additional D-words. Descriptions are parsed into words and any word that is not on the list, i.e. in the current set of D-words, is added.

3 Concepts, codes and classifications

In this chapter we consider fundamental building blocks for a semantic network used for automatic coding, such as concepts, codes and classifications.

3.1 Concepts: building blocks for codes

As we have indicated before concepts are defined in terms of C-words. More precisely we assume that each concept is represented by a finite set of C-words. This implies that the order of the C-words comprising a concept is irrelevant. This is an assumption, which probably holds for the majority of concepts in the majority applications. It remains to be seen if this leads to problems in some cases.

In order to be consistent there are some constraints applicable to the set of concepts, such as:

- If A and B are concepts characterizing the same code c, neither of them should be strictly contained in the other.
- A concept can correspond to only one code.
- Synonymous concepts should belong to the same code.
- With each code at least one concept should be associated.

It can a priori not be excluded that C-words are synonyms, but it should be avoided that the concepts such words are involved in are different. Say a and b are C-words and synonyms, and c is another C-word. Then the concepts $\{a, c\}$ and $\{b, c\}$ cannot be associated with different codes. If d is another C-word and not synonymous to a, b or c then $\{a, c\}$ and $\{b, d\}$ can be associated with different codes, as $\{a, c\}$ and $\{a, d\}$ are not synonymous.

All conditions combined imply that there exists a map $\kappa : \mathcal{K} \to \mathcal{C}$ from the set $\mathcal{K}$ of concepts (for a particular application) to the set $\mathcal{C}$ of codes (in the classification for that application), and this map κ is surjective. It induces an equivalence relation '$\sim$' on $\mathcal{K}$: For $K_1, K_2 \in \mathcal{K}$ we have $K_1 \sim K_2$ if (and only if) $\kappa(K_1) = \kappa(K_2)$. The equivalence classes consist of descriptions that correspond to the same code. The set of equivalence classes of $\mathcal{K}$ induced by κ is denoted by $\mathcal{K}/\kappa$. In plain words it simply means that the set of concepts can be partitioned in such a way that each part contains all the concepts used to characterize a particular code.

3.2 Codes: the building blocks for a classification

We assume that for our coding problem a classification is given. This can be about goods, business activities, occupation, education, diseases, causes of death, etc. We assume this classification to be a directed tree, where the direction of each arc is from special to more general. Then it holds that any two nodes are connected by a unique shortest path. Usually there is also a unique root node, which can be viewed as the origin of the classification. See Figure 3.1.

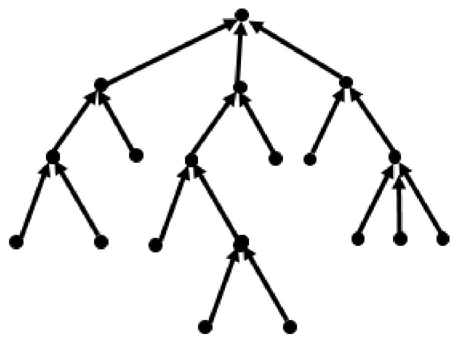

Figure 3.1 A classification as a directed tree.

With each node of this directed tree structure we assume that a code has been associated. Such a code describes a particular good, business activity, occupation, type of education, cause of death, etc. that is associated with the corresponding node.

In the previous section it was indicated that each code is characterized by a set of concepts, and each concept is defined by a set of C-words. Just above we stated that a code is associated with a classification. It should fit into the hierarchy defined by the classification. A code is so to speak part of two structures, one derived from language (through the C-words and the semantic structure defined on this set) and the other from the classification being used, which we consider as given.

In the next section these relationships are considered in more detail. These issues are illustrated in Figures 3.2 and 3.3, respectively.

In Figure 3.2 the nodes are labeled in a way consistent with the hierarchy. Singled out are the nodes that are associated with the target codes. The target codes for the automatic coding exercise define the set of codes that should result from the coding process, if all goes according to plan. They do not necessarily correspond to the most detailed categories, as is indicated in the

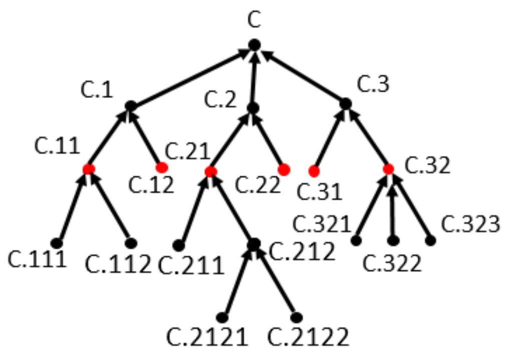

Figure 3.2 Classification tree, codes and target codes (red nodes).

case of Figure 3.2. In Figure 3.3 code C is shown and four nodes corresponding to concepts that define this code. Each concept is defined in terms of C-words. Figure 3.3 should be read as follows

$$C = (a \wedge b \wedge c) \vee (a \wedge e) \vee (b \wedge c \wedge e) \vee (d \wedge e). \quad (1)$$

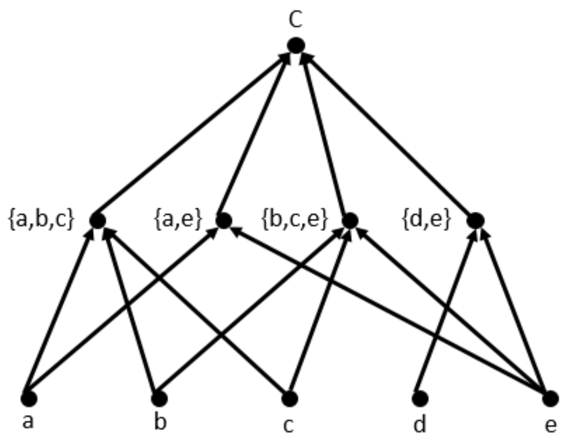

Figure 3.3 A code defined in terms of concepts, which in turn are defined using C-words.

This means that C is the generated code if the concepts $\{a, b, c\}$, $\{a, e\}$, $\{b, c, e\}$ or $\{d, e\}$ are activated. A concept is activated if and only if the C-words of which it consists are activated. So the concept $\{a, b, c\}$ is activated if and only if the C-words a, b and c are activated. Likewise for the other three concepts. Note the peculiarity of the interpretation of Figure 3.3 (and similar such digraphs). For the concepts we need conjunctions (∧) and for the code description we need disjunctions (∨). So the idea is that code C is generated if one of the defining concepts is activated. This happens when all the C-words of this concept are activated by D-words generated from in a description (1). Viewed as a Boolean expression, it is said to be in disjunctive normal form.

Remark Defining codes in terms of one or more combinations of concepts would not go beyond the framework sketched. The reason is because any Boolean expression can be rewritten in disjunctive normal form. The conjunctive terms should be defined as concepts. This would also yield a code defined as a disjunction of concepts. □

So once all codes in the classification have been described in terms of concepts, the next thing to do is to use the structure of the classification as a highway to connect the various codes, each defined in terms of concepts. It should be checked if, by using semantic relations (i.c. hyponymy) and by removing C-words from concepts, one can go from any node / code in the classification (except the top node) to its (unique) parent code / node.

We now briefly consider special codes that are often used in practice, because they are so convenient. Such codes define a remainder category, applying to all the cases that do not fall under the heading of the alternative possibilities. The label for such a category could formally be something like 'other' or 'none of the above' (e.g. in a questionnaire, where it is the last-mentioned category of a question). Of course, as a description of the category for our purposes it is not useful, although this code is well-defined. It is a negative definition of a code. See Figure 3.4 as an example of this.

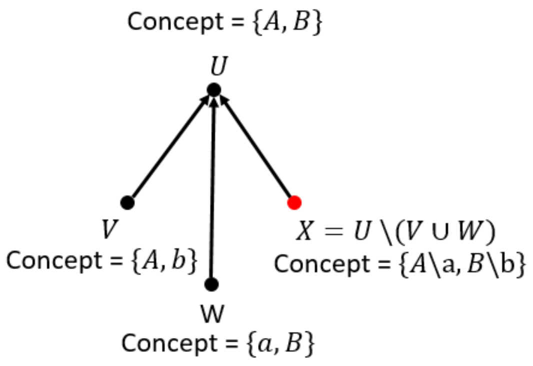
Concept = $\{A, B\}$
U
V
Concept = $\{A, b\}$
$X = U \setminus (V \cup W)$
Concept = $\{A\setminus a, B\setminus b\}$
W
Concept = $\{a, B\}$

Figure 3.4 Exception code (red node).

Remark Codes with exceptions in their characterization need to be handled differently in the coding process from codes without exceptions. It is a somewhat specialized topic so it is not discussed here (See, however, [7]). □

From practical experience (e.g. earlier held surveys) it is known that certain codes occur more frequently in a population than others. This kind of knowledge can be incorporated into the data by using weights for each code, reflecting the appearance of the codes in practice. Such a weight is (proportional to) an observed frequency.

These weights (or frequencies) may help to decide to choose a code when a description leads to various alternatives.

The methodology is partly language-independent and partly language-dependent, as we remarked already. In the present paper we do not go into these language-dependent issues (see however, [7]).

Remark We assume that the domain of the variable being coded is the set consisting of all its codes. Usually there is a hierarchy defined on this domain (which is a digraph), which signifies a relationship like hyperonymy/hyponymy for words. But the exact nature of this relationship we can take for granted. This structure is usually a directed tree structure, yielding a partial order by taking the transitive closure of the hierarchy.[2] □

When a classification is used in a coding application, a desired level of coding should be specified. Or if the classification is not organized in this way, individual nodes should be marked as at the desired level of detail. This means that more detailed results are replaced by less detailed ones, at the corresponding desired level of detail. This can be done automatically by following the arcs in the classification tree until the node is reached at the right level. In case a description is not detailed enough and there are more nodes at the desired level that are more detailed, the answer provided is not precise enough. Additional information is needed to single out one of these nodes at the target level.

Remark When building the semantic network it seems natural to start with the codes and describe these via concepts using C-words. This set of C-words can also be taken as the nucleus of the set of D-words. One can then continue with looking at the semantic structure at the set of C-words. Then the set of C-words can be expanded by adding words taken from descriptions, and from one's knowledge of the language in which the descriptions are stated. These new D-words also need to be put in the semantic framework started with the C-words. So it is a reversed approach, from finish to start. □

4 Relations and constraints

In the approach to coding described in this paper, a semantic network is the centrepiece. For general background information on knowledge representation and semantic networks see [3], [4], [5] and [6].

A semantic network establishes the link between descriptions on the one hand and the codes in the classification on the other. The link is established through connections among and between D- and C-words, concepts and codes. Various relations that exist among and between the fundamental objects (words, concepts, codes, classification) discussed in the previous section, are detailed in the present one.

The semantic relations that exist at the word level imply relations at the concept and code level. The codes are part of a classification which is a tree structure. This structure should be compatible with the relations implied by those at code level via the relations that hold at the concept level.

The aim of this section is to indicate the various relationships that hold between entities in a semantic network for coding. And also to point out that for a semantic network certain constraints need to hold in order for it to be consistent. These are not automatically satisfied and have to be explicitly checked for such a semantic network. This checking needs to be repeated every time it is updated as a result of learning.

[2] This is obtained by adding derived relationships: if (i, j) and (j, k) are arcs so is (i, k).

4.1 Semantic relations among words

As we have explained before, the building blocks for concepts (and therefore, indirectly, also for codes) are C-words. To make the coding possible we need to link D-words to C-words, using semantic relationships. Important semantic relationships that will be used for this are presented in Table 4.1.

Semantic relationship	Explication	Examples
Hyperonymy / hyponymy	A is generalization of B	Cars, trucks, ships are vehicles
Synonymy	A denotes the same as B	A spud is a potato
Homonymy	A has several meanings	A plant is an organism as well as a production unit

Table 4.1 Semantic relationships used in a semantic network

Synonymy expresses semantic equivalence. It should be stressed that synonyms in this paper are defined in a more general way than usual. They do not distinguish between verbs and nouns; nor between different tenses of verbs, singular and plural forms, cases of nouns (nominative, genitive, dative, etc.)

The C-words and D-words belong to different worlds, in a sense. The C-words are used to describe codes (via concepts) and the D-words belong to the world of descriptions (possibly after having been spell-checked and -corrected; see Section 6). To connect the two 'worlds', use is made of semantic relationships, i.c. synonymy and hyperonymy. There are also homonyms. They belong to two or more groups of synonyms.

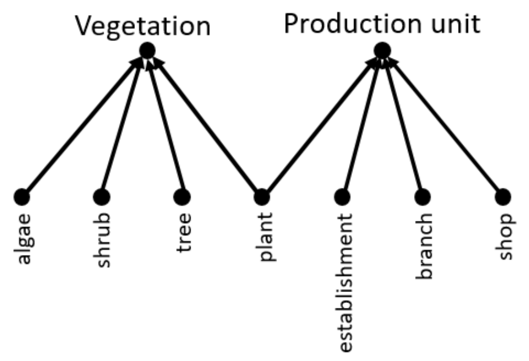

Figure 4.1 Synonyms and homonyms. The homonym 'plant'.

The following example is about the concept of homonym, which is relevant in a coding context concerning business activities.

Example The word 'plant' has several different meanings. It can mean an economic entity: a place where things are produced, a 'factory', a 'mill', etc., or it can refer to a biological entity, which has its own set of synonyms (together with 'flower', 'tree', etc.). In Figure 4.1 the word (= a string of characters) 'plant' has two meanings indicated by the arrows pointing to 'Vegetation' and 'Production unit'. Although 'plant' belongs to both sets of synonyms they are not joined into one set; they are different due to different semantic relationships in each of these sets. There are two different concepts involved here, which happen to be represented by the same string of characters. □

We assume that the set of C-words form a subset of the set of D-words for a particular coding problem. This is simply realized by letting each C-word also be a D-word. Of course, this makes sense only in case these words belong to the same language, although it would not harm if they do not. Usually the set of C-words will be much smaller than the corresponding set of D-words. In practice, not all D-words will be connected to C-words. Some D-words contain no or little specific information, and are of no use (directly or indirectly) by defining C-words, and in the coding process. These words are called stop words. We will keep them in the collection of D-words as it is useful to know the stop words in a particular application (see Section 6).

Hyperonymy is useful in coding, as it is an embodiment of generalization. In case a respondent gives an answer that is too detailed for the coding purposes, it allows deducing an answer at the target level for the coding process. One should be prepared for the situation that a respondent might give an answer that is more detailed than the level required, unaware of the classification or the level of detail he should answer. But it is somewhat silly to ask again for an answer if the required result can be deduced by the coding system.

Suppose that a respondent in a survey about business activities answers that he ‘grows apples and pears’, but that this is too detailed for the purpose of the coding and that the required level is ‘growing fruit’. The answer given implies that, and we would like the coding system to deduce this from the answer provided. Given that the semantic network contains ‘fruit’ as a hyperonym for ‘apples’, ‘pears’, ‘prunes’, etc., this would be possible. Even a more detailed level should be contemplated to be part of the semantic network. Another respondent may reply that he ‘grows Braeburn and Granny Smith’. Ideally the coding system should recognize ‘Braeburn’ and ‘Granny Smith’ as cultivars of apples. A cultivar, in this particular case—a hyponym—of ‘apple’. See also Figure 4.2.

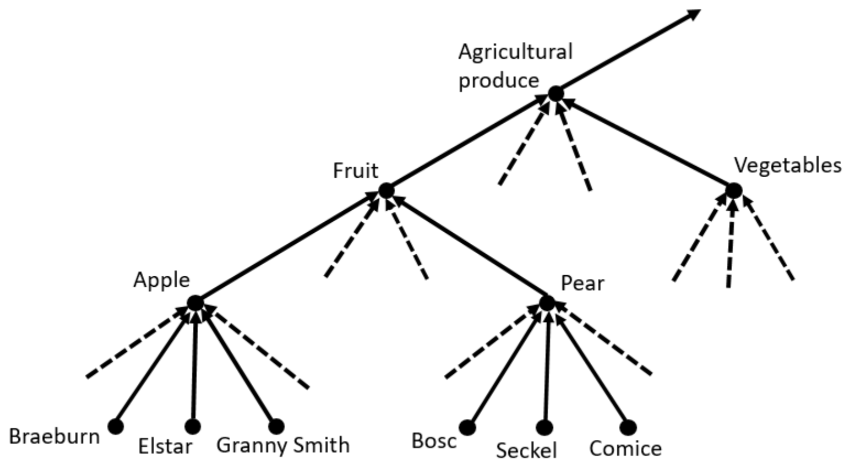

Figure 4.2 Agricultural produce as part of a products classification.

To illustrate a different kind of relationship that can exist between words, consider the following example about synonyms.

Example 4.2. The words ‘produce’, ‘make’, ‘fabricate’, ‘manufacture’ and ‘create’ as well as related words, could be considered synonyms.[3] Some of these 'related words' are given below. The words in each entry have the same stem in common. We have the following lists of words related to the keyword at the beginning of each item.

'produce' : 'produces', 'produced', 'producing' , 'producer', 'producers', 'production', 'product', 'products';
'make' : 'makes' , 'made', 'making', 'maker', 'makers';
'fabricate' : 'fabricates', 'fabricated', 'fabricating', 'fabricator', 'fabricators', 'fabrication', fabrications';
'manufacture' : 'manufactures' , 'manufactured', 'manufacturing', 'manufacturer', 'manufacturers', 'manufactory', 'manufactories', 'factory', 'factories';
'create' : 'creates', 'created', 'creating', 'creator', 'creators', 'creation', 'creations'.

□

All these words can be considered members of a set of synonyms (in a generalized sense). Words like 'factory' and 'plant' (for production purposes, not as vegetation)[4] can also be defined as members of the same set of synonyms.

In case of synonymy we are dealing with a relationship that is symmetric. In case of an asymmetric semantic relationship (like hyponymy/ hyperonymy), we represent it in such a way that the direction is from the more specific to the more general. In that case, if a word is connected to another word, and if the specific word is triggered, the hyperonym will be triggered as well.

It is possible that a group of words have a common hyperonym but this hyperonym is not a C-word. In that case one can invent a D-word to represent this set. This hyperonym should then be used as a hyponym in another relation, linking it ultimately to a C-word. Otherwise there is no point in storing that such relationships exist (i.e. finding hyperonyms for stop words is a superfluous activity).

Remark There are more semantic relationships than Table 4.1 indicates. For instance there are antonyms (opposites), meronyms (indicating that something is a part of something bigger) and holonyms (the dual of meronyms). However, such relations are not required for the approach described in the present paper. But they may have potential for an extension of this approach. Antonyms may be of use in dealing with exceptions. Meronyms / holonyms may be used as an alternative way to generalize notions, apart from the hyponym / hyperonym relationship that is used in the present paper. If a company produces 'spark plugs for cars' than one cannot conclude that this company 'manufactures cars' on the basis of the fact that a spark plug is part of a car. One can only conclude that it produces a car part. But then one uses the fact that a 'spark plug for a car' is a 'car part', using hyperonymy. □

For C-words we can ask what relations can or may exist between them. First of all it can be avoided that they are synonyms, because the sets of synonyms could then be combined. Also it can be avoided that C-words are homonyms, by choosing them all differently. C-words can be

[3] To which could be added: 'build', 'design', 'construct', which all refer to producing something, out of nothing (so to speak) or from rough materials like stone, wood, metal, etc.
[4] See Figure 4.1.

hyperonyms or hyponyms. In fact this relationship and its inverse are important to change level of detail.

4.2 Derived relations for concepts and codes

In this section we apply the hyperonym / hyponym relationship at the C-word level to create relations at the concepts level.

Suppose that $\{a_1. \dots, a_n\}$ is a concept, where the a_i are C-words for $i = 1, \dots, n$. Further we let '$\leq$' denote a binary relation on the set of C-words, such that $a \leq A$ means that A is a hyperonym of a, or equivalently, that a is a hyponym of A. Then if $a_i \leq A_i$, for $i = 1, \dots, n$, we have

$$\{a_1, \dots, a_i, \dots, a_n\} \leq \{a_1, \dots, A_i, \dots, a_n\}. \tag{2}$$

If also $a_j \leq A_j$ for some $j \neq i$, we have

$$\{a_1, \dots, a_i, \dots, a_j, \dots, a_n\} \leq \{a_1, \dots, A_i, \dots, a_j, \dots, a_n\} \leq \{a_1, \dots, A_i, \dots, A_j, \dots, a_n\} \tag{3}$$

and also

$$\{a_1, \dots, a_i, \dots, a_j, \dots, a_n\} \leq \{a_1, \dots, a_i, \dots, A_j, \dots, a_n\} \leq \{a_1, \dots, A_i, \dots, A_j, \dots, a_n\}. \tag{4}$$

See Figure 4.3, which shows a Hasse(-like) diagram for this situation.

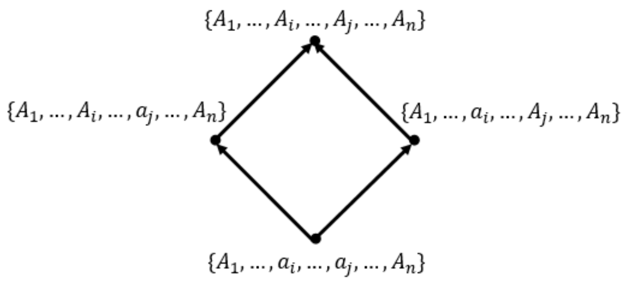
$\{A_1, \dots, A_i, \dots, A_j, \dots, A_n\}$
$\{A_1, \dots, A_i, \dots, a_j, \dots, A_n\}$
$\{A_1, \dots, a_i, \dots, A_j, \dots, A_n\}$
$\{A_1, \dots, a_i, \dots, a_j, \dots, A_n\}$

Figure 4.3 Generalization of concepts.

Also, deletion of a C-word from a concept, yields a more general concept. So for instance

$$\{a_1, \dots, a_i, \dots, a_n\} \leq \{a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n\}. \tag{5}$$

This latter kind of generalization can be seen as an extreme form of the one introduced in (2). The generalizations can be applied repeatedly to build a chain. Applying transitivity yields single step generalizations consisting of several smaller step generalizations. In this way the set of concepts and their generalizations form a partially ordered set.

It is clear that in this way for a given concept many generalizations can be found. But a lot of these are not very useful, as they do not correspond to any code. In order to get the useful ones, we have to look at the codes in the classification and the way in which they are defined in terms of concepts.

As we have seen before, the codes in a classification are usually associated with nodes in a (directed) tree structure. With the codes we also have associated concepts that define them, and the concepts are represented in terms of C-words. See Figure 4.4. In this figure negation of C-words used in concepts is indicated by the use of colors for the arcs connecting C-words and Concepts. If the arc is black it means that the C-word to which the arc is attached is affirmed. If the arc is colored orange it means the associated C-word is negated, implicating exclusion.

Figure 4.4 contains additional information, namely concerning D-words and descriptions. The basis for the D-words are descriptions in the form of written texts. They can be viewed as a selection from words in these descriptions that are useful for classifying the written text from they are drawn, literally, in raw form.

Remark In Figure 4.4 a layer containing constraints (see Section 4.3) has been left out on purpose. The picture is intended to focus on the several layers between texts at the bottom and codes at the top, consisting of words and word combinations. □

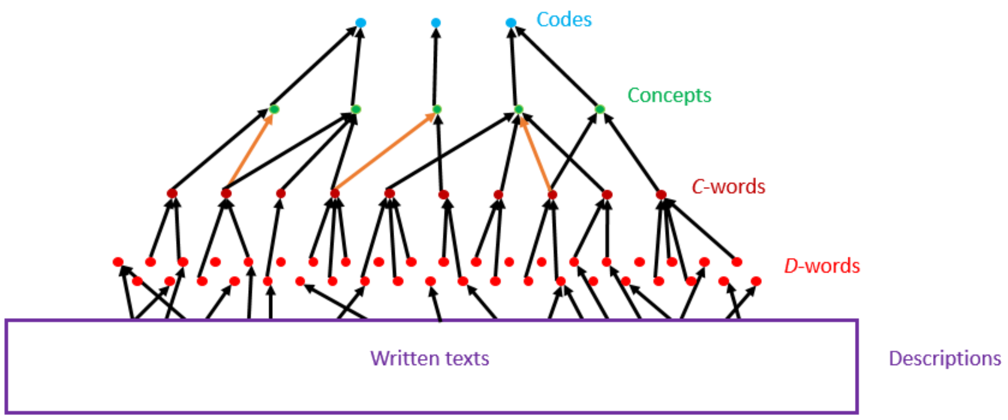

Figure 4.4 Hierarchy of codes, concepts, C-words and D-words. The arrows between C-words and Concepts are either black or orange. A black arrow means affirmation of the attached C-word, an orange arrow means negation (exclusion).

The next task is to use the (directed tree) structure of the classification to link the concepts associated with the nodes that are directly linked in the tree structure using the semantic relationships hyperonymy/hyponymy. See Figure 4.5 for an example.

In Figures 4.5 and 4.6 we have hyperonym / hyponym relationships between certain C-words, where:

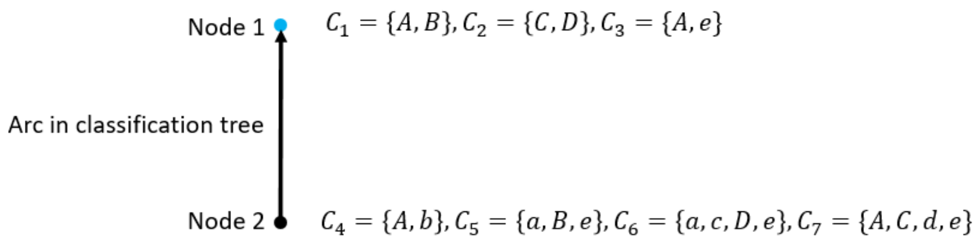
Node 1
$C_1 = \{A, B\}, C_2 = \{C, D\}, C_3 = \{A, e\}$
Arc in classification tree
Node 2
$C_4 = \{A, b\}, C_5 = \{a, B, e\}, C_6 = \{a, c, D, e\}, C_7 = \{A, C, d, e\}$

Figure 4.5 Concepts associated with parent-child nodes.

$$a \leq A, , b \leq B, c \leq C, d \leq D. \tag{6}$$

So in this case, A is less detailed than a , or put in another way, A is a hyperonym (superordinate) of a. Similarly, for the other pairs of C-words appearing in (6).

Once concepts associated with the various nodes in the classification have been linked together in this manner, we have finished a semantic network for this classification. In practice, such a network is likely to be in a state of permanent adaptation. New descriptions of codes in terms of concepts can be defined, and these have to be tied into the existing semantic network. Also new D-words will be found and have to be added to the network. It will also be a never (formally) ending task to check the correctness of the semantic network, initially and after any change that has been made to its structure or its content.

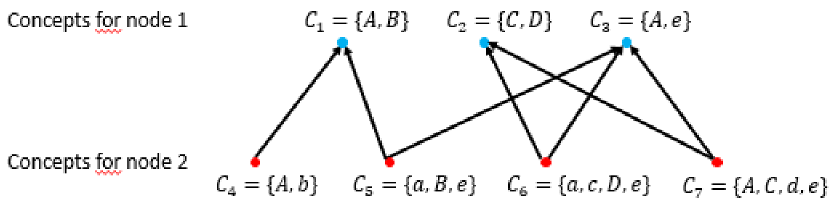

Figure 4.6 Semantically linked concepts associated with parent-child nodes.

4.3 Constraints

This section is to make the reader aware that certain constraints exist for a semantic network used for coding, in order to be consistent. It should be admitted that the intention of this section is rather modest. It tries to call for attention to a specific problem, but it does not try to solve it at all. So instead of discussing a complete set of such constraints, we only offer some examples of constraints to hold. A thorough discussion of this issue is better left to a special paper that is entirely devoted to this topic.

So we only contend ourselves with providing some examples of the constraints we have in mind for semantic networks for coding. In Section 3.1 some of such constraints are mentioned.

The semantic relationships defined for the set of concepts should comply with the hierarchical structure of the classification used. To illustrate this, consider Figure 4.7. It shows two codes, code 1 and code 2, and two concepts, concept 1 and concept 2, characterizing code 1 and code 2, respectively. There are three maps / functions κ, λ and σ, where σ is as defined above. The map $\lambda : V_{cod} \to V_{cod}$ is defined by the classification, for each $c \in V_{cod} \backslash \{root\}$, is the code associated with the (unique) parent node. We assume that κ and λ are given. In case a semantic relation σ (as discussed in the previous section) has been defined, linking concept 1 and concept 2, we require that the diagram in Figure 4.7 is commutative. This means that $\kappa \circ \sigma = \lambda \circ \kappa$. In case no semantic relation has been defined, it is advisable to find out if this is still possible. It may require finding concepts defining code 1 and concepts defining code 2 that can be linked semantically. Or it could require that a new concept 1 for code 1 or a new concept 2 for code 2 needs to be added, so that a semantic link can be established. In case such a link σ is found, it should satisfy the commutativity property shown in Figure 4.7.

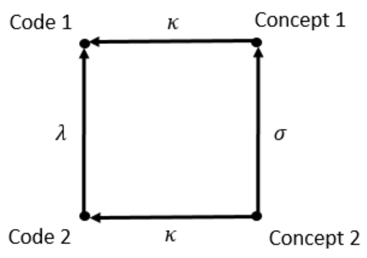
$$\begin{tikzcd}
\text{Code 1} & \text{Concept 1} \arrow[l, "\kappa"'] \\
\text{Code 2} \arrow[u, "\lambda"] & \text{Concept 2} \arrow[l, "\kappa"] \arrow[u, "\sigma"']
\end{tikzcd}$$

Figure 4.7 Commuting diagram for various maps concerning codes and associated concepts.

The advantage of such semantic links is that they can replace links derived from λ which are in a sense extraneous. The semantic links are internal and they allow inferences and generalizations ('spreading action') within the semantic network. In fact we ideally should have that all concepts associated with code 1 are linked semantically with those of code 2. In case a concept K associated with code 2 is not linked semantically to a concept associated with code 1, we could use λ to generalize from K but this does generally not yield a single concept, but a set of concepts, namely all those associated with code 1. So K may be linked in this way to a concept associated with code 1 to which it is not semantically related. This is to be avoided, if not to be forbidden.

In fact, there is even a stronger requirement concerning the concepts associated with the same code than that they should be different. This requirement asserts that no (nontrivial) semantic relations should exist between concepts associated with the same code.

Another kind of constraint involves concepts and the C-words from which they are constructed. It is required that there are no two concepts consisting of the same C-words. If we put this in another way, we require that, if denotes the set of C-words and denotes its power set, there is a map that is injective.

Another constraint that we mentioned before concerns the choice of C-words. They must be chosen in such a way that they are never homonymous. In this respect D-words and C-words are different. D-words may have homonyms, but C-words never have. If we leave out the homonyms from the D-words (that each link to two or more D-words) as well as the stop words (that link to none of the D-words) the remaining set of D-words is such that each of them link to exactly one C-word, by definition.

5 Some examples

It is possible that hyponyms or hyperonyms of a D-word are also D-words themselves. This is necessary when these D-words appear in concepts associated with codes at different levels of detail. An example of this situation occurs for instance in the coding of business activities.

Example Consider the following activities related to vegetables:

1. 'growing of vegetables' corresponds to a unique code A , no matter what the vegetables being grown are, potatoes, lettuce, beet roots, onions etc. [5)]
2. 'processing of potatoes' leads to a unique code B, which is different from 'processing of grain' or 'processing of beet roots'. [6)]
3. 'wholesale of seed potatoes' and 'wholesale of edible potatoes' correspond to different codes C and D. [7)] So 'wholesale of potatoes' would be ambiguous, in the sense that no unique code would correspond to this description. What is lacking is the type of potatoes concerned, i.e. seed potatoes or edible potatoes. The distinction between these forms of wholesale is made since they are directed at different markets, roughly farmers and retail.

In summary we can define concepts involved in the codes A, B, C and D, as follows:

Code A: {vegetables, growing},
Code B: {potatoes, processing},
Code C: {edible, potatoes, wholesale},
Code D: {seed, potatoes, wholesale}.

So we see that in the first example it is not necessary to specify what vegetable it is that is being grown. In the second example to mention that potatoes are processed, is just the right level of detail and a single code corresponds to this activity. In case of wholesale, it needs to be specified what kind of potatoes this activity concerns: seed potatoes or edible ones. It is possible (and useful) to define the concepts {edible, potatoes} and {seed, potatoes}. Both are hyponyms of potatoes. □

Example Careful attention should be devoted to the organization of D-words. The first idea is perhaps to simply organize them in disjoint sets, each of which is a set of synonyms. We could, for instance, have a group of words related to 'vehicle', to which such words as 'trucks' and

5) The reason is that the growing of vegetables is very similar, no matter the variety that is grown. For the application intended at least, these growing activities are considered the same. This is a choice.
6) The reason is that these processes are quite different, and important to make for the application intended.
7) The reason is that they operate on different markets, and for the application intended, this distinction is important, apparently.

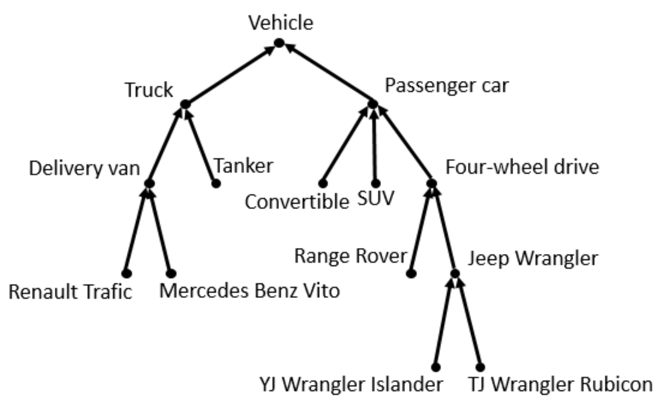

Figure 5.1 Vehicle tree.

‘passenger cars’ (among several others). See Figure 5.1.The ‘trucks’ can be divided into ‘delivery vans’ and ‘tankers’ (among several others). The ‘passenger cars’ consist of ‘convertibles’, ‘SUV’s and ‘Four-wheel drives’ (among others types). Several brands have ‘Delivery vans’, such as Renault (the ‘Trafic’) and Mercedes-Benz (the ‘Vito’). Among the four-wheel drives we have the ‘Range Rover’ and ‘Jeep Wrangler’. This latter one consists of various types, among them the ‘YJ Wrangler Islander’ and the ‘TJ Wrangler Rubicon’. If there is no need to have such an elaborate tree structure for vehicles, one could use a flat directed tree, with ‘vehicle’ as root, and all the categories mentioned as synonyms of this word. As representative word for the set of synonyms ‘vehicle’ could be used (suppose that this is a C-word). However, this supposes that in any combination with, say, activities such as ‘manufacturing’, ‘sales’, ‘repair’, etc, vehicles would have to be specified at the same level of detail. However, a tree structure as described and as depicted in Figure 5.1, would be easier to maintain. □

The following example is about clothes. This example illustrates a few interesting aspects, relevant for automatic coding in general.

Example Clothing is a collective term that comprises many different things. See Figure 5.2. The distinctions that are made depend on different criteria.

- The use of the clothes can be important (for professional use (uniforms for military personel, police men and women, doctors, guards) or civilian clothes for daily use),
- The gender (male, female) of the persons they are intended for,
- The age groups of persons (grown-ups, children, babies),
- Subcultures (goths, hiphop, etc.),
- The layer for the clothing (underclothing, outerwear),
- Ethnic, cultural or religious groups may have special clothing (Western, African, Indian, Arabic, etc. to mention some broad groups).

The example in Figure 5.2 is in fact restricted to human clothing. There is also animal clothing, e.g. for horses, dogs, cats, etc. This would be part of the ‘other clothes’ category, among other types of clothing not mentioned here such as ‘sports clothing’, ‘leisure wear’, etc.

These distinctions are important in view of the activities. For instance, in case of manufacturing, it is important to distinguish between ‘outerwear’ and ‘underclothing’, whereas for retail the

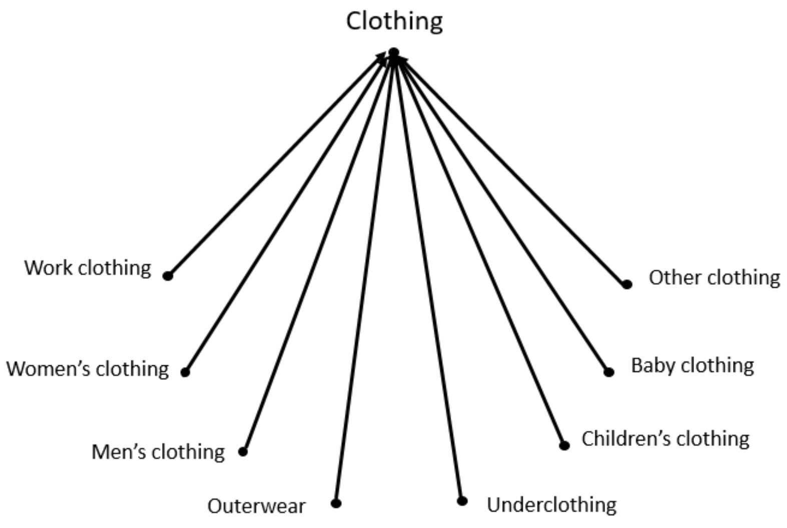

Figure 5.2 Clothing example.

distinction between 'men's clothing', 'women's clothing' and 'children's clothing' is relevant. This is comparable to the example concerning the business activities involving agricultural products.

Another point to note is that the various types of clothing mentioned in Figure 5.2 are not disjoint categories. For instance, babies are children. Children are boys or girls and hence male of female, etc.

The 'other clothing' category is an example of an exception category. In this case it is very useful, as it seems to be well neigh impossible to list all types of clothes that exist. Besides for the coding problem at hand it is possibly only of interest to make some distinction in clothing and lump the rest into a single remainder category and treat that in a separate way.

Figure 5.2 is presented as a very flat directed tree. But with some additional clothing types distinguished a more general structure is required. Because the nodes are not disjoint this is not a tree, but a directed acyclic graph (DAG). Figure 5.3 shows a (partial) elaboration of the clothing example in Figure 5.2. It elaborates some of the categories shown in this figure. This elaboration is far from complete. But it shows that the structure obtained is not a directed tree, but a DAG, as certain objects, such as 'gloves', belong to more than one category, in this case 'women's clothing' and 'men's clothing'. □

Example. It seems to be a good idea that the most detailed codes used in the classification at hand are atomic, that is, should pertain to a single business activity, product, service, etc. In practice this is, however, not always the case. One may, for instance, find a code in a classification for 'selling and repairing cars', because these combined activities occur in practice and are equally important for some businesses, in addition to the activities 'selling cars' (without repairing them) and 'repairing cars' (without selling them). If the main activity would have to be chosen, neither 'repairing cars' nor 'selling cars' would give an adequate picture of the activities of some businesses. But it must be admitted that the concept of 'atomic activity', 'atomic type of education' is not as clear-cut as it may seem at first sight. It also depends very much on the application. It is similar to the problem of distinguishing between atomic and composite entities. Here, as well, what is an atomic entity depends on the application. In conclusion, we can say that this aspect of the descriptions will always be somewhat problematic. □

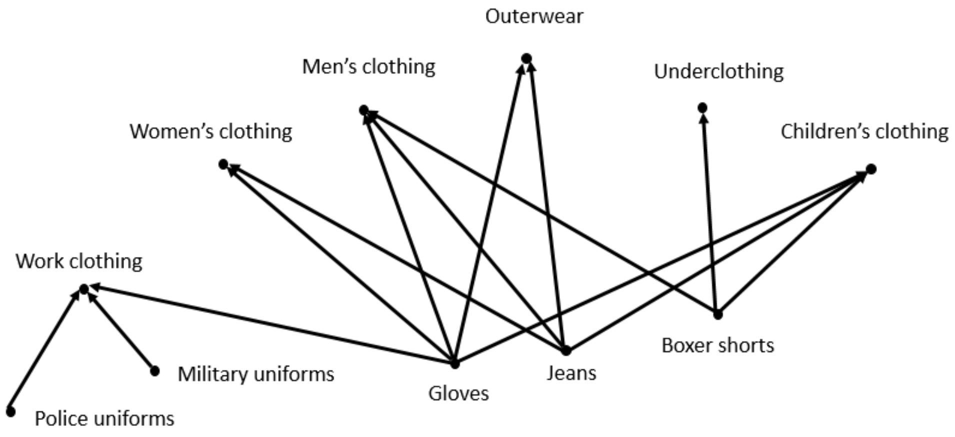

Figure 5.3 Partially elaborated structure for clothes (a DAG).

Example. It should be noted that sometimes, for descriptions collected for other purposes than statistical ones, there may be a bias. Or the description may deliberately be chosen more general. This is a more serious problem, and little can be done about it, except for asking for information from an alternative source, that is more neutral and provides more honest and realistic answers. □

6 Coding with a semantic network

In this section we give a brief description of how automatic coding system could work. The description is mainly to give the reader an idea of the various steps that are needed. The description focuses on the happy flow, which is the best case possible. In practice many exceptions can occur and a real system should deal with these. Some of these exceptional cases are discussed as well, to give the reader an idea of what to expect. The discussion of exceptions is not complete: not all exceptions are discussed nor is what is discussed complete.

In Section 6.1 we describe the so-called happy-flow, which deals with the situation when a description contains the information to find exactly one code (which then automatically is assumed to be the correct one). In fact, we consider a simplified situation here, namely when there are only positive descriptions, that is, without exceptions. They have been excluded here for the sake of brevity.

6.1 The `Happy flow'

The automatic coding process using a semantic network in case a description yields a single code in the classification used ('happy flow') is roughly as follows:

1. Start with a description S, a string of characters.
2. (*optional*) Put S through a spell-checker. Resulting string: S^*.

3. Segment S^* into tokens /words.
4. Check that each of these tokens / words in the string is a D-word.
5. Find the C-words that these D-words imply, using the semantic network.
6. Find the concepts that these C-words imply, using the semantic network.
7. Find the code K in the classification used that these concepts imply, using the semantic network. In case several codes are triggered, the most detailed one should be chosen.
8. Attach the code K to the string S.

Of course, this flow is achieved in the ideal case. In practice one is likely to encounter exceptions that require extra work. In Subsection 6.2 each of these steps is considered separately, but with an eye to exceptions that may occur.

6.2 Analysis of the process flow

This description in the previous section contains the 'happy flow' where each step is ideal. The input string S apparently contains the right information to lead to a single code. Although it is not guaranteed that this code is correct, it is usually accepted as the code that corresponds to S.

In practice, however, one usually finds exceptions to the ideal steps. A major reason for this is because respondents do not know (and are not expected to know) the underlying classification and consequently may be unaware as to what information to put into their descriptions and at what level of detail. Another, more general, reason is related to the language that respondents use, which may contain jargon, slang or words from dialects or even other spoken languages. Or they may have spelling errors in their answers. Respondents may also use exceptions in their answers, which add another level of complexity to the coding process.

We discuss each of the steps in the happy flow briefly and point at certain complications that might arise.

Step 1 Ideally the string should contain numbers, alphanumeric characters, interpunction (dots, commas, hyphens, etc). But maybe it contains characters outside this range as well. These should be handled as well, in order to avoid problems later on in the coding process. This step could be viewed as an ETL step, or a data cleansing step.

Step 2 This step is not exclusive for coding. It is a problem in the area of spell-checking and correcting. Tools[8] exist that can handle this problem.

Step 3 The segmentation process is called tokenization (see e.g. [2]). The aim is to recognize tokens, in our case words in the descriptions. If the descriptions are longer pieces of text, it becomes important to consider interpunction as well. Also the grammatical structure of descriptions could then be taken into account. In the present paper we assume that this is not necessary, as the descriptions are assumed to be short.
This tokenization problem transcends automatic coding. We shall not go into it any further here. We can also assume this problem to be solved, in the sense that adequate software exists for this step.
All sorts of errors are possible due to the incorrect spelling of words, the incorrect usage of interpunction, the incorrect splitting of words, the incorrect concatenation of words, etc. A problem in some languages, like Dutch or German, is that long words may appear in

8) An example of such a tool is Ginger (see `http://www.gingersoftware.com`)

descriptions (if correctly spelled). The problem is that these long words may pose spelling problems for many people. Under the influence of English they tend to break up long words into smaller components. The opposite also happens. If one does not pay special attention to this particular problem (in Dutch, say) the automatic coding system is bound to under-perform. To cope with these spelling problems one may have to introduce an extra spell-checking and spell-correcting step.

Step 4 This step checks if all tokens / words found in the previous step can be linked to D-words currently in the semantic network. There are two possibilities for each of these tokens.

1. All tokens / words can be recognized, and so the entire description is recognized.
2. At least one of the tokens / words is not recognized as a D-word. Those that are not, should be set aside and an expert has to add them to the semantic network. This is an opportunity where the semantic network can 'learn' from new input. The description should be put on a special list, and passed again to the coding system after it has been updated with the new tokens (now D-words).

In case of a happy flow, all tokens in the description are recognized as D-words. It is important to check that this is the case. A token not recognized as a D-word cannot be discarded, as it may very well be a new D-word. Such a word has to be incorporated into the semantic network, which includes its linking to the C-words in there (and hence to concepts and codes). In case not all D-words in a string are used, a situation arises that may lead to coding errors.

So tokens in the description that are not recognized should be dealt with first, that is, before the description is used for coding. This requires experts to look at these tokens / words and 'feed' them to the semantic network. It is possible, of course, that some of the unrecognized tokens turn out to be stop words, so are actually not used in finding codes.

After the semantic network has been updated the description needs to be parsed again. Now all tokens should be recognized as D-words.

Step 5 If the entire description is recognized, the D-words it consists of are linked to C-words (or: activate C-words), unless they are stop words. In the latter case they are simply discarded. Note that these words should also be recognized first as D-words, to make sure that all tokens / words in a description are used.

Step 6 From the activated C-words all concepts they activate should be found, which should be done automatically by the semantic network.

Problems encountered in this step are at the heart of the semantic network used. It is very well possible that some semantic relationships are lacking and that others are wrong. It may also be the case that the semantic relationships that have been defined are too limited and should be expanded. The description may not be ideal: it may lead to more than one concept (belonging to different codes; if they all belong to a single code then there is no problem) and hence to more than one code; or it may correspond to no code at all.

Step 7 There are three possibilities that can occur when trying to find a code triggered by the description.

1. No code is triggered. This can occur for several reasons. It starts with the observation that there was no code for which a combination of defining concepts was triggered. This indicates that some vital C-words in some concepts are missing. This requires additional information from the respondent (if available) to arrive at a code
2. The description triggers exactly one code at the required level of detail. This is the ideal case, described in the happy flow. (It is actually a bit more complicated; see the comment below.)
3. The description triggers more than one code at the required level. Apparently the description is not specific enough. In this case additional information is also required to

arrive at a single code.

The problems with this step may be that either no code is found or more than one. In the latter case crucial information is possibly lacking from the description. Without any additional information (provided by the respondent, or after inspection of a coding expert) no single code can be found and the coding is inconclusive. Another possibility is that no code is found to match the description. This can happen if no combination of concepts is found that correspond to a code. This in turn can have various causes, ranging from the case that not a single C-word is found to correspond to the description, to the case that several concepts are found to match with the description, but together they do not define a code, nor does any sub-combination of the concepts. It is even possible that a single code is found, but on closer inspection this turns out to be wrong. In a sense this is the worst-case, as a lot of things may be wrong then. We will not delve into this further, as it takes us too far from our goal of this paper. These topic need to be addressed separately, and in all necessary detail.

Step 8 In case no (unique) code is found at this step, it should be decided what to do: to ask for feedback information from the respondents (if available) or have coding experts look into the problem. They may be able to detect vital information that may lead to another code; or cast doubt on the code that has been found in the first step. □

The comments above should also give some indication about the exceptions that may occur in the various steps. Handling them is a major task, but not for the present paper to describe.

7 Summary and discussion

The idea behind using a semantic network in automatic coding is to separate the data (i.c. the semantic network) from the software for navigating it. The navigation program is supposed to be independent of the particular contents of a semantic network. The semantic network itself is dependent on a particular application, in particular the classification used.

For the coding problem there is a 'source side' (descriptions in a natural language, D-words) and a 'target side' (C-words, concepts, codes, classification). The semantic network links the 'source side' to the 'target side'.

At the 'source side' the problem is to extract as much information from such a description as possible. In doing so one faces language problems due to imperfections in the descriptions (ranging from spelling errors, through the use of words that are not recognized by the system), but also problems due to the fact that the respondent is unfamiliar with the classification being used to code the descriptions. This may manifest itself through over-complete descriptions, that is, one that contains more information than is needed for a single code, or under-complete, that is, contains insufficient information to lead to a (unique) code.

It should be noted that the 'source side' of the coding problem is (natural) language dependent, in the sense that the choice of words and phrases is not limited, whereas the 'target side' is not, in the sense that it works with a limited vocabulary. Of course, the C-words used at the 'target side' are chosen from a language, but at the 'input side' one may have to deal with different languages in which the descriptions are given, e.g. in Dutch / Flemish, French and German in Belgium, in various languages in the territory of the European Union, in English, French and Inuit in Canada, etc. One can then choose one language from which to take the C-words to describe the concepts, and the concepts in turn to describe the codes acting in the classification that is being used. For each of the languages mentioned one has to arrange that descriptions in that language can be handled, i.e. translated, into the target language.

An assumption of the approach in the present paper is that the order of words defining a concept is irrelevant. This is certainly true in many cases, but there could be some exceptions. In that case the order of the D-words is important. It would be straightforward to modify the current model accordingly.

Another extension of the model presented here concerns concepts. It may be necessary that not only synonyms and hyperonyms / hyponyms of D-words should be taken into account, but also sets of concepts. This means that e.g. it should be possible to have concepts A, B, C, D, such that $\{A, B\}$ is a synonym of $\{C, D\}$ without A being a synonym of C (or D) and B being a synonym of D (or C). Or $\{A, B\} \leq \{C, D\}$ holds, without $A \leq C$ and $B \leq D$. Also this extension of the model would be quite feasible.

References

[1] D. Jurafsky & J. Martin (2000). *Speech and Language Processing*. Prentice-Hall.

[2] A. Mikheev (2004). Text segmentation, Chapter 10 in: *The Oxford Handbook of Computational Linguistics*, R. Mitkov (ed.), Oxford University Press.

[3] J. Sowa (1984). *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley.

[4] J. Sowa (1991). *Principles of Semantic Networks*, Morgan Kaufman.

[5] J. Sowa (1992). Semantic Networks, in: S.C. Shapiro (ed), Encyclopedia of Artificial Intelligence (2^{nd} ed.), Wiley.

[6] J. Sowa (2000). *Knowledge Representation*, Brooks/Cole.

[7] L. Willenborg (2012). Semantic Networks for Automatic Coding, Paper, CBS, The Hague. [first version of the present paper]

Appendix
A Producing a semantic network

To build a semantic network for an application involves some effort. The starting point is a set of descriptions that have been labeled by codes (of the coding system we are using for the problem at hand). This requires that the descriptions have first been coded by experts in the subject we are dealing with. Care should be taken to cover all codes of the coding system used. For each code a diverse set of descriptions is needed.

The effort is to elicit the D-words for each description. And also to group them, if necessary. Processing a reasonable number of such descriptions in a short term would be considerably enhanced by using a specialized tool, which is described in Section A.1.

If we have processed the (training) set of descriptions, we have achieved the following

1. A list of D-words.
2. Combinations of D-words that describe individual codes in the coding system.
3. A list of 'word junk', that is, words from the descriptions that have not been selected as useful.

The usefulness of the items in points 1 and 2 are obvious. However, the list of word junk mentioned in point 3 is also very useful. For a new description should first be scanned to see if there any D-words that have already been identified in the description and mark them. But it can also mark the word junk that has been encountered before and mark that too. This saves the expert who uses the tool, to check these words again. The more descriptions have been processed the more D-words and word junk has been found, the less new words remain for inspection, making the job easier and quicker.

A.1 Useful software tools

We describe several tools that are useful to construct a semantic network. In separate subsections we describe these tools.

A.1.1 Finding D-words

We describe a useful tool to process a considerable number of descriptions to generate input for the semantic network. We briefly describe how it should operate. The descriptions should be coupled with codes of the classification used. We can view these descriptions as examples of the various codes in the coding system used. They should cover all the codes.

The tool is an interactive tool, assisting the experts responsible for producing a semantic network for the given problem. Its basic aim is to help finding D-words in descriptions. The tool not only collects and organizes those (sorting and deduplicating such words in a cumulative list) but also organizes the words not selected (which it also organizes as the D-words in a separate list). The words not selected—the word junk—is useful material when searching for D-words in texts. The word junk is used in the process of finding D-words in a training set of descriptions. The word junk is only useful for the process looking for D-words, not for the actual coding process as this takes the C-words (and those linked to it) as a starting point and looks which of those appear in new descriptions.

The tool should also indicate which D-words were found together in the same description.[9)]

If a new text is read the tool checks which of the collected D-words can be found in it. If it finds a D-word it is colored in the display of the text, say in blue. The same happens with word junk: if this is encountered it is also colored but in a different color, say grey. Words that are not colored blue or gray have not been encountered before are displayed in a third color say black. The analyst needs to concentrate on the black words. He then selects new D-words among them (if any) by clicking on them with the cursor of his mouse. Then new candidate D-words are identified, which are colored red. If the description has been inspected the analyst indicates this by clicking a ready button. The new D-words are the added to the D-word list (deduplicated and sorted this is stored). The appropriate code is stored with the D-words pertaining to it. The non-selected are added to the word junk list (which is also deduplicated and sorted before it is stored).

A.1.2 Finding C-words and concepts

Once the D-words have found they should be further processed to find C-words implied by the C-words. These C-words are supposed to be of special form: nouns in the singular, verbs in the infinitive, etc. So a tool is handy that presents the collected D-words one-by-one so that the analyst can produce the required form. The tool stores the C-words making sure that no duplicates appear and that the list is sorted alphabetically.

A.1.3 C-words and associates

Once the C-words have been found it is necessary to associate semantically related words with them. This is to increase the chance that a C-word is identified in a (new) description, through using words semantically associated with them. Such words include synonyms, hyponyms, hyperonyms, conjugations of verbs, etc.

9) The analyst should indicate which D-words in a description belong together. It is possible that several combinations of D-words appear in a description. It is important to know which one belong together and which are independent.

Colophon

Publisher
Statistics Netherlands
Henri Faasdreef 312, 2492 JP The Hague
www.cbs.nl

Prepress
Statistics Netherlands, Grafimedia

Design
Edenspiekermann

Information
Telephone +31 88 570 70 70, fax +31 70 337 59 94
Via contact form: www.cbs.nl/information