An Exploration of Tool Support for Categorical Coding

Anjo Anjewierden and Hannie Gijlers, Department of Instructional Technology, Faculty of Behavioural Sciences, University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands
Email: a.a.anjewierden@utwente.nl, a.h.gijlers@utwente.nl

Abstract: In this paper we explore tool support for categorical coding of verbal and chat data. We consider tool support for manual coding, automatic coding by learning algorithms, and derive at a socio-technical approach in which human coders and learning algorithms together address the coding task. Given that a literature study suggests researchers devise, adapt and refine a wide variety of coding schemes, a categorization support system should handle and accommodate user defined coding schemes. Based on these ideas a prototype of the ChatIC tool was developed and evaluated with three coding schemes. For two coding schemes a sufficient inter-rater agreement between a human coder and the learning algorithms was reached.

Introduction

This paper explores the opportunities of tool support for categorical coding of verbal data and messages obtained from collaborative chats. Our main motivation is of a practical nature. More and more interaction protocols obtained from computer mediated communication (CMC) become available, from various (on-line) sources and on various domains. Behavioral scientists who want to study this data often devise and apply their own coding schemes, and given the lack of special purpose coding tools they are often faced with the time-consuming task of manual coding with general tools like Excel.

Recently, several papers have made the learning research community aware of the possibility of applying text analysis technology to semi-automatically code verbal data (e.g. Dönmez et al., 2005). These papers study the coding problem from a language technology perspective, mainly focusing on the machine learning algorithms used. We propose a slightly broader view by considering that going from manual coding to the application of fully automated coding technology is perhaps to big a step to take in one go. The principal point is that a behavioral scientist will generally define a coding system that is deemed suitable for the data and study at hand, and it is not at all clear that the categorical distinctions made in an arbitrary coding scheme can be picked up by existing language technology. The approach used by automatic coding tools is to first manually code part of the data (this is called the training phase) and then to determine the inter-rater reliability between the human and the algorithm (the test phase). The training stage is necessary as all algorithms learn from a set of coded data. For automatic coding to be feasible it is suggested to develop techniques that predict when the training set is large enough and it can be determined statistically that the remainder of the data can be coded automatically (Forsyth et al., 2007).

In the subsequent sections we look at categorization in practice, review existing tool support for categorization and finally describe ChatIC our socio-technical tool for categorization support.

Categorization in Practice

Analysis of interaction protocols is one of the main methods researchers use to gain understanding of the processes that contribute to meaningful and effective collaborative learning (Naidu & Jarvela, 2006). The analysis of chat data often starts with the choice for, or development of a coding scheme (Strijbos & Stahl, 2007). Next to the choice of analysis scheme, the segmentation of the discourse is an important aspect of the categorization process. Segmentation refers to the process that divides the text into units. Coding schemes frequently use utterances as their unit of analysis, which can be defined as individual message units, that can be distinguished from other units by a pause, comma, or stop (van Boxtel, van der Linden, & Kanselaar, 2000). One can argue that in chat communication utterances are partly defined by the learners themselves. However, learners' chat messages might contain inadequate punctuation, or even distinct messages within one chat line. Categorization on the utterance level is not sufficient to capture learners' knowledge construction-process or cognitive processing (van Boxtel, van der Linden, & Kanselaar, 2000; Weinberger & Fischer, 2006). In order to capture these processes a coarser grained unit of analysis is needed.

Collaborative learning is a multi-faceted activity, learners engage in communicative activities as well as domain and task-related activities, and also have to regulate and monitor their progress. Differences in the learning context and the questions guiding the research resulted in the development of a wide variety of coding schemes (De Wever, Schellens, Valcke, & Van Keer, 2006). Coding schemes may focus on one aspect or the collaborative learning process, combine different aspects of the collaborative process in one coding scheme, or distinguish between different dimensions that each focus on a specific aspect of the collaborative learning process (Strijbos & Stahl, 2007). Jonassen and Kwon (2001) compared the problem solving behavior of learners working in a face-to-face setting with the problem solving behavior of learners communicating through a CMC tool. They used a coding scheme based on the functional category system, developed by Poole and Holmes (2005) to study interaction in problem solving contexts and distinguish categories that are related to phases in the problem solving process. A similar learning process oriented approach was followed by Saab et al. (2005) who developed a coding scheme that included a number of categories that were based on inquiry learning processes. The math moves dimension in the analytical framework developed by Strijbos and Stahl (2007) is an example of a domain specific dimension, which distinguishes mathematical procedures like counting, using geometric expressions, and using algebraic functions. Krystyniak and Heikkinen (2007) developed a coding scheme for the analysis of verbal interactions of learners' completing an inquiry assignment in a chemistry laboratory. Their coding scheme distinguishes domain and task specific categories like the use of chemistry concepts and laboratory equipment.

The widely used, and often adapted, coding scheme for the classification of interaction developed by Bales (1970) is an example of a coding scheme focussing on communicative functions. The main categories include positive and mixed actions, attempted answers, questions and negative and mixed actions. Each main category consists of a number of sub-categories. Kumpulainen and Mutanen (1999) included a coding scheme for language functions in their analytical framework. The language functions dimension defined by Kumpulainen and Mutanen includes categories like reasoning and evaluation that are not available in the analytical framework of Bales. Arvaja (2007) further adjusted the coding scheme of Kumpulainen and Mutanen and included a code for exemplification. The coding scheme as adjusted by Arvaja is specifically suited to study communicative activities in a knowledge construction context. Recently, more and more coding schemes use a multi-dimensional approach (Saab, Van Joolingen, & Van Hout-Wolters, 2005; Weinberger & Fischer, 2006) that incorporate communicative functions as well as more cognitive, task or domain related categories.

This overview of coding practices shows that within the behavioural sciences a variety of coding schemes for interaction are used. Researchers adjust each others coding schemes, based on their own research questions, and combine schemes or dimensions from different coding schemes in their analytical framework (Strijbos & Stahl, 2007).

Categorization Support

Based on the problems identified in the introduction we argue that a categorization support system has to address at least the following: handle a wide range of data, support coding schemes defined by a researcher, and make the user aware of the opportunity of automatic coding when the combination of data and coding scheme allows this.

In an environment dedicated to categorical coding the user should expect functions like a simple interactive coding interface, browsing previous codes, and basic statistical information, as well as more advanced functions for example a query like "show me messages containing this term or syntactic pattern" and the ability to code all matching messages using a single directive. Parallel to active support for manual coding an automated component can learn from the codes assigned through the interactive interface. It can suggest a category for the next message, and provide indicators, for example, the kappa between itself and the coder. Such an environment is typical of what is commonly referred to as a socio-technical tool: user and system together solve the task of categorical coding.

The environment sketched above deviates from machine learning approaches in a fundamental way as it removes the distinction between the separated stages of training and testing. The user is continuously and unobtrusively informed about what the system learns. In order to determine whether the environment is realistic we scanned the literature for papers on tool support for categorical coding with the objective of finding ideas and techniques that could be usefully integrated.

MEPA (Erkens, 2002), developed at Utrecht University, is a tool for the analysis of sequences after categorical coding has been applied. MEPA does not address the coding problem in general, although it provides some automatic support for a particular coding scheme and semi-automatic support for segmentation through a large set of handcrafted rules (Erkens et al., 2005). We consider segmentation a pre-processing step and MEPA might play a role there. Segmentation, despite its importance, is not further discussed in the remainder of this paper.

The goal of text categorization is to classify a document and assign it a theme or topic (Manning & Schütze, 1999). Superficially, text categorization or document classification, and categorization of chats appear similar tasks: given a document assign it a label or code. Automatic text categorization is successfully applied to self-contained documents, for example newspaper articles, web pages and weblog posts, with a high degree of accuracy. Self-contained documents can be described as being on a particular topic, often evidenced by the nouns they contain, and because of the length (several hundred words or more) machine learning algorithms have sufficient information to learn to classify correctly. This contrasts with messages from chats or verbal data. These messages are short (often just a few words) and topic classification is less relevant. Text categorization algorithms view a document as a set of features and classification is performed on the features rather than the original text. The most common features are word or character n-grams (a sequence of n consecutive words or characters), and patterns involving information about the syntactic structure (e.g. a personal pronoun followed by a verb). Based on the features, a statistical or machine learning method (e.g. Naive Bayes, Markov models) is applied to derive the probability that a certain document (reduced to a set of features) belongs to one of the possible categorizations. Algorithms derive the relevance of the features from a manually coded training set. This general framework, including the underlying algorithms, has also been implemented for the categorization of verbal data (CodeLearner; Forsyth et al., 2007, Ainsworth et al., 2007), educational chat messages (Anjewierden, Kollöffel & Hulshof, 2007) and content-rich sentences (TagHelper; Dönmez et al., 2005).

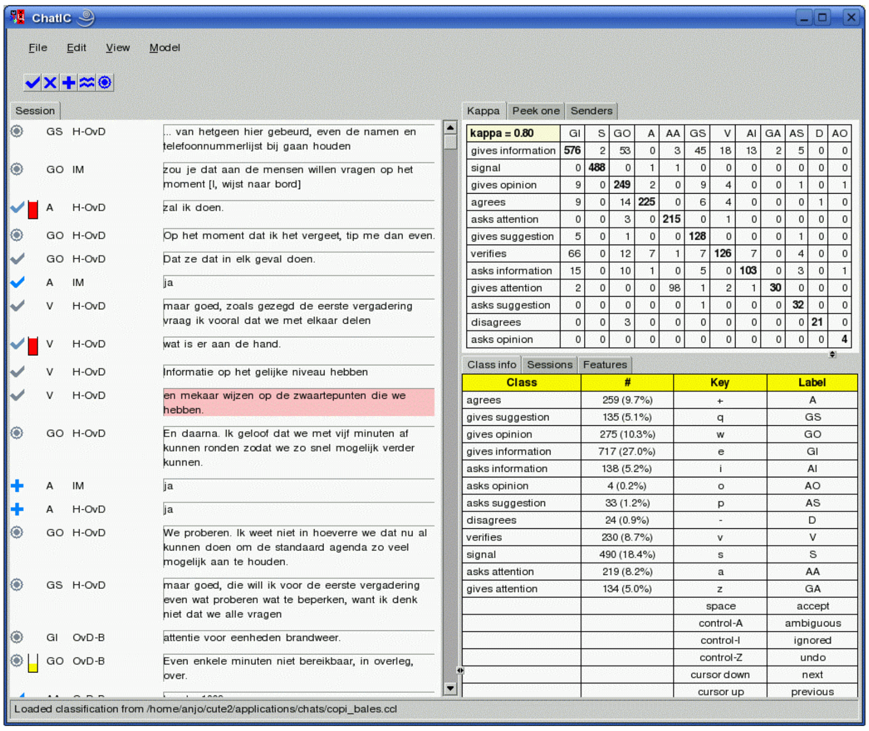
TagHelper (Rosé, 2007), developed at Carnegie Mellon, is the most visible tool for semi-automatic coding support. The website provides download, extensive documentation and an on-line course. As input, TagHelper takes a training set in Excel. The tool implements a large number of algorithms for text categorization and has flexible mechanisms to select which features to use, e.g. words, bigrams, and part-of-speech (POS) bigrams. We conducted a number of experiments using TagHelper with our coded data. On one of our test sets, and experimenting with algorithms and settings, TagHelper obtained a reasonable kappa of 0.58 on Dutch chats (TagHelper only supports English and German). TagHelper reports kappa, and for each feature, the probability it belongs to any given category. Both the kappa and the feature probability are useful in the context of our environment. TagHelper is founded on traditional machine learning practices, including extensive cross validation. Running it can take extremely long. An important aspect we learned from the experiments with TagHelper is that the environment should prefer learning algorithms that can be computed incrementally to obtain interactive performance.

CodeLearner is a tool under development at the University of Nottingham. The objectives and approach are very similar to TagHelper. In Forsyth et al. (2007) a technique is developed "that [CodeLearner] be able to predict from a relatively small training set what its accuracy is likely to be on a much larger testing set." Given that CodeLearner is also an off-line tool, and uses time consuming cross validation for the prediction of future coding accuracy, an interactive environment can only incorporate the idea of predicting future accuracy and not the implementation.

ChatIC

In this section we describe our implementation of a dedicated socio-technical categorical coding environment called ChatIC. Figure 1 provides an overview of the user interface. The UI is modeled around the idea of a cockpit or Intensive Care (IC) unit. On the left is a window with the messages, with small indicators in front of each individual message. On the right are two windows with aggregation indicators that provide an overview of the overall state of the coding process. Coding itself is very simple. The next message to be coded is highlighted (middle of the left window) and the user types a single character. After entering the code, all indicators visible on the screen are updated, and the next uncoded message is highlighted.

ChatIC takes two inputs: a specification of the coding scheme and the messages to be coded. The latter is a file (Excel or XML) with information about the sender and the text of the message, optionally a time-stamp, recipient and channel over which the message was conveyed. Output is a file with the coded messages that can be read by, for example, TagHelper. The coding scheme is also specified in either Excel or XML and lists the possible categories (or classes), the keystroke and a mnemonic for each category. In Figure 1, the lower right window shows the coding scheme, in this case derived from (Bales, 1970), and the special keyboard short cuts.



●		GS	H-OvD	... van hetgeen hier gebeurd, even de namen en telefoonnummerlijst bij gaan houden
●		GO	IM	zou je dat aan de mensen willen vragen op het moment [I, wijst naar bord]
✓	🟥	A	H-OvD	zal ik doen.
●		GO	H-OvD	Op het moment dat ik het vergeet, tip me dan even.
✓		GO	H-OvD	Dat ze dat in elk geval doen.
✓		A	IM	ja
✓		V	H-OvD	maar goed, zoals gezegd de eerste vergadering vraag ik vooral dat we met elkaar delen
✓	🟥	V	H-OvD	wat is er aan de hand.
✓		V	H-OvD	Informatie op het gelijke niveau hebben
✓		V	H-OvD	en mekaar wijzen op de zwaartepunten die we hebben.
●		GO	H-OvD	En daarna. Ik geloof dat we met vijf minuten af kunnen ronden zodat we zo snel mogelijk verder kunnen.
+		A	IM	ja
+		A	H-OvD	ja
●		GO	H-OvD	We proberen. Ik weet niet in hoeverre we dat nu al kunnen doen om de standaard agenda zo veel mogelijk aan te houden.
●		GS	H-OvD	maar goed, die will ik voor de eerste vergadering even wat proberen wat te beperken, want ik denk niet dat we alle vragen
●		GI	OvD-B	attentie voor eenheden brandweer.
●	🟨	GO	OvD-B	Even enkele minuten niet bereikbaar, in overleg, over.

kappa = 0.80	GI	S	GO	A	AA	GS	V	AI	GA	AS	D	AO
gives information	**576**	2	53	0	3	45	18	13	2	5	0	0
signal	0	**488**	0	1	1	0	0	0	0	0	0	0
gives opinion	9	0	**249**	2	0	9	4	0	0	1	0	1
agrees	9	0	14	**225**	0	6	4	0	0	0	1	0
asks attention	0	0	3	0	**215**	0	1	0	0	0	0	0
gives suggestion	5	0	1	0	0	**128**	0	0	0	1	0	0
verifies	66	0	12	7	1	7	**126**	7	0	4	0	0
asks information	15	0	10	1	0	5	0	**103**	0	3	0	1
gives attention	2	0	0	0	98	1	2	1	**30**	0	0	0
asks suggestion	0	0	0	0	0	1	0	0	0	**32**	0	0
disagrees	0	0	3	0	0	0	0	0	0	0	**21**	0
asks opinion	0	0	0	0	0	0	0	0	0	0	0	**4**

Class	#	Key	Label
agrees	259 (9.7%)	+	A
gives suggestion	135 (5.1%)	q	GS
gives opinion	275 (10.3%)	w	GO
gives information	717 (27.0%)	e	GI
asks information	138 (5.2%)	i	AI
asks opinion	4 (0.2%)	o	AO
asks suggestion	33 (1.2%)	p	AS
disagrees	24 (0.9%)	-	D
verifies	230 (8.7%)	v	V
signal	490 (18.4%)	s	S
asks attention	219 (8.2%)	a	AA
gives attention	134 (5.0%)	z	GA
		space	accept
		control-A	ambiguous
		control-I	ignored
		control-Z	undo
		cursor down	next
		cursor up	previous

Loaded classification from /home/anjo/cute2/applications/chats/copi_bales.ccl

Figure 1. Overview of the ChatIC interface.

For each chat there are five columns (left window of Figure 1). The first column is a colored (gray to blue) icon that indicates the state. The possible states are *coded* (a checkmark icon), *uncoded* (bullet), *message with same features as seen before* (plus), *ambiguous* (squiggle) and *ignore* (cross). The color scheme for the icon is a statistic about how certain the tool is that the coding is correct, the statistic is translated to a color ranging from gray (low confidence) to bright blue (high confidence). The second column is an alert indicator. If ChatIC would have assigned the same code the alert indicator is empty, if it disagrees the indicator is red. There are two examples of the alert indicator in Figure 1. The third column shows the assigned code. For coded messages this is the code assigned by the user, for uncoded chats this is the suggested code by the underlying algorithm. The fourth column displays the sender and the last column the text of the message.

Turbo Coding

Taking advantage of the functionality of the underlying infrastructure, a text analysis toolkit (Anjewierden et al., 2008), there are a number of "turbo" coding functions. One is based on entering a message and then listing all messages in the data set that contain the same features. For example, entering "temperature increases" shows all messages that contain the features "temperature" and "increase". The user can then code all resulting messages with a single action. Another turbo coding function is the application of a pattern search mechanism based on the natural language syntax of the messages. The POS-pattern "<pp> <vb>" finds all messages that contain a personal pronoun (pp) immediately followed by a verb (vb), e.g. "**I think** the answer is 4". These functions can speed up coding significantly if the user is able to relate her coding scheme to such natural language patterns.

Experience with Coding Schemes

We evaluated the relation between the coding scheme and the accuracy of automatic coding. For this evaluation we selected three coding schemes. The first coding scheme we selected is derived from the communicative functions of Bales. It was selected because coding schemes based on communicative functions are commonly used. Figure 1 shows both the coding scheme and the resulting kappa table, achieving a more than reasonable κ=0.80 using single words as features.

The second coding scheme makes a distinction between messages about the domain ("when the blinds are down the temperature decreases"), regulatory ("shall we try the next exercise"), technical ("could you move the window, I cannot read it"), and off-task or social messages ("well done partner"). Our experience is that humans mainly look at the terms to correctly apply this coding scheme. Learning algorithms should pick up the terms related to one of the four categories relatively easily.

The third coding scheme refines the domain-oriented messages into two categories: *observations* and *interpretations*. An observation is when the learner sees something in the learning environment and says: "the temperature increases”. An interpretation occurs when the learner draws a conclusion or presents a hypothesis: “when the blinds are down the temperature decreases”. This coding scheme should present a challenge for algorithms, because many of the terms (e.g. temperature) are as likely to occur in an observation as in an interpretation. In general, humans use the syntactic structure to make the distinction, for example “if ... then” is a strong pattern for an interpretation.

The messages for the second and third coding schemes were obtained by transcribing face-to-face

kappa = 0.76	REG	DOM	TEC	SOC
regulative	**12967**	912	867	22
domain	449	**4592**	13	0
technical	121	13	**1412**	4
social	443	5	40	**703**

Figure 2. Kappa table for the second coding scheme.

interaction between 16 groups of three learners trying to solve a learning task. A student, who had experience with coding similar protocols using Excel, coded all 22,536 messages manually with the first version of ChatIC. She reported that the user interface is far superior compared to Excel, and spontaneously stated that after a while the tool predicted how to code the next message. Figure 2 shows the inter-rater agreement as a kappa table between the human coder (rows) and a Naive Bayes classifier with single words as features (columns).

kappa = 0.42	N	OBS
non_class	**3076**	75
observation	482	**266**

kappa = 0.24	N	INT
non_class	**1395**	1287
interpretation	30	**512**

Figure 3. a) Kappa table for observations versus other domain messages, b) Kappa table for interpretation versus other domain messages, using collocations

The messages classified as domain were thereafter manually coded following the idea of a multi-dimensional coding scheme as *observation, interpretation* and *other domain.* Figure 3a shows the kappa table when we take the binary classification *observation* vs. *non-observation* using the same algorithm as above. Although κ=0.42 might be reasonable in some contexts the large number of false positives (482) compared to the number of true positives (266) indicates the algorithm does not pick up the distinction between observations and other domain-oriented messages. The results for interpretations are similar (κ=0.27).

In order to test our hypothesis that *interpretations* can be recognized using syntactic patterns (if ... then ...) we implemented a feature extractor which defines a feature for each occurrence of [A, B] when word A occurs before word B in a message. The most frequent of these word collocations were selected. Figure 3b displays the kappa table and in this case, interestingly, the algorithm misses only 30 of the 542 interpretations found by the user, although it also incorrectly classifies a large number of messages as interpretations. The overall conclusion is that this coding scheme is too subtle for machine learning algorithms to classify correctly.

Advanced Indicators

The development of ChatIC as an interactive socio-technical tool provided two major challenges: defining useful indicators and developing algorithms to compute them in real-time. One of the most used indicators for behavioral researchers is Cohen's kappa and the underlying algorithms have been programmed such that they can compute kappa for Naive Bayes (unigram, bigram, POS-bigrams, and collocations) in about 0.2 seconds for our largest test set (22,500 messages).

Cross validation and extensive test/train splits are not possible in real-time and likewise are the techniques proposed by CodeLearner to predict future accuracy. ChatIC is interactive and the position on the screen the user is watching most is the next message to be coded. Given that ChatIC predicts the category of the next message a graph with the moving average of the accuracy of this prediction is a useful indicator of future automatic coding accuracy. We have designated this the peek-one indicator, an example of the resulting graph is shown in Figure 4 (this indicator replaces the kappa table when the user selects the Peek-one tab in the top right window of Figure 1).

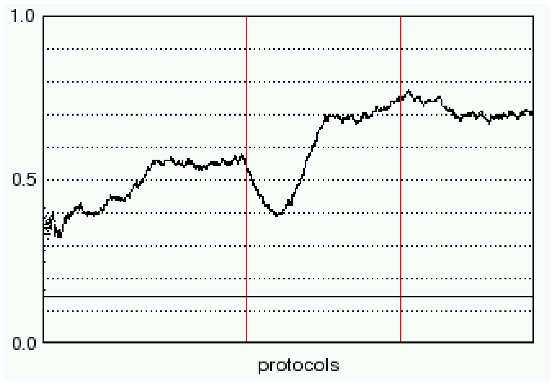


Figure 4. Peek-one statistic for the first coding scheme.

The peek-one indicator displays the accuracy on the y-axis (0...1). The solid horizontal line is the estimate, which is derived from the kappa table using Pearson's chi-square. On the far right of Figure 4 the peek-one indicator is about 0.7 which suggests that ChatIC correctly predicts 7 out of 10 codings. Although peek-one is not as rigorous as the approach of CodeLearner the idea of a leveling off of automated coding accuracy is still helpful. In this particular case, the indicator reveals an insight in the underlying data. The vertical lines in the figure demarcate the start and the end of protocols. Reading from left to right, the peek-one indicator goes up slowly to approximately 0.55 and once the second protocol is taken into account drops to about 0.40. After a while it goes up again, reaching its maximum peek-one accuracy of about 0.75. The explanation, provided by the coder, is that the first protocol is derived from management discussions about a certain task, while the second and third protocol are from field workers on the same task. Apparently, managers and field workers use different terminology, and it takes the learning algorithm a little while to adapt. The significance in the context of ChatIC as a socio-technical tool is that a researcher gets enough information to detect and, possibly, explain the behavior of the tool in relation to the data.

The peek-one statistic for the second coding scheme is shown in Figure 5. It illustrates that the algorithm quickly reaches an accuracy of between 0.8 and 0.9 and does not improve with more messages being coded. The combination of a high enough kappa and a stable peek-one can be a reason for the researcher to stop coding and perhaps decide to automatically code the remainder of the messages.

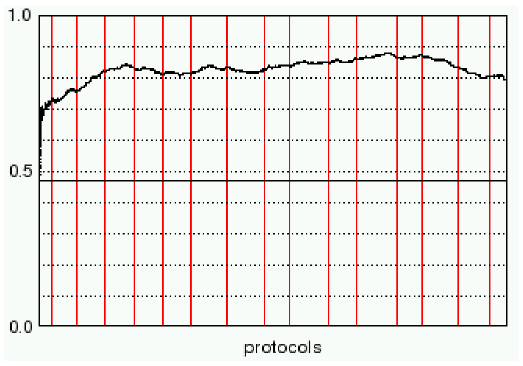
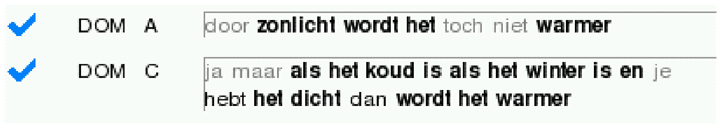


Figure 5. Peek-one graph for the second coding scheme.

Trust and Interactive Feature Feedback

An important aspect of a socio-technical tool is trust: the user should feel confident about how the system behaves. In ChatIC this is translated at the macro-level through the kappa and peek-one indicators. The user can see how these indicators change over time and whether the performance improves. In addition, the kappa table is clickable. If the user selects a cell, for example the cell with manual=domain and algorithm=regulative, ChatIC lists all corresponding messages. At the micro-level trust is supported by predicting the category for uncoded messages and through the alert indicators for discrepancies.



✓	DOM	A	door **zonlicht wordt het** toch niet **warmer**
✓	DOM	C	ja maar **als het koud is als het winter is en** je hebt **het dicht** dan **wordt het warmer**

Figure 6. Intra-message feature highlighting.

We are experimenting with a very detailed indicator of the relation between the features and coding. Figure 6 gives an example. All words in the message are highlighted as either bold, normal or gray. If a word is displayed as bold, the conditional probability of the word belonging to the assigned category is higher than the prior probability for the same category, and there is no other category for which this holds. If there are multiple categories for which the condition holds the word is in the normal typeface and if the condition does not hold it is gray. The idea is to emphasize the words that cause the algorithm to assign the code. The first line in Figure 6, "because of **sunlight it does** not **become warmer**", for example, suggests a semantic relation between the terms **sunlight** and **warmer** if we ignore the function words. We hope that techniques like this can be used to identify semantic patterns which go beyond n-gram statistics.

Conclusions

In this paper we have explored software tool support for categorical coding. Currently, tool support is minimal. TagHelper is about the only tool that is readily available and provides a useful, albeit time consuming, service to behavorial scientists. We developed the concept of viewing categorical coding as a socio-technical system in which the human researcher and learning algorithms together address the task of categorical coding. After developing this concept we implemented a first prototype of the idea as a tool called ChatIC, and evaluated it on coding schemes in common use by behavioral scientists. For some coding schemes, the inter-rater agreement between the human coder and learning algorithms was sufficient to consider the application of automatic coding, for another coding scheme the algorithm performed poorly. An important aspect of ChatIC is the notion of single message and aggregated indicators. We illustrated the code next message, feature highlight, the kappa table and peek-one indicators.

It is our feeling that a tool like ChatIC is of potential use to everybody involved in categorical coding and an open source version is available through http://edm.gw.utwente.nl. One possible scenario is to use ChatIC for initial coding and TagHelper, or other traditional machine learning tools, for off-line cross validation and experimentation with a variety of learning algorithms and settings if (part of) the data has to be categorized automatically.

References

Ainsworth, S., Forsyth, R. S., Clarke, D. D., Robertson, L., & O' Malley, C. (2007). Automatic coding of learners' self-explanation when learning from diagrams., *European Association of Research on Learning and Instruction Conference.* Budapest.

Anjewierden, A. (2008). tOKo and Sigmund: Text analysis for ontology development and social research. http://www.toko-sigmund.org.

Anjewierden, A., Kollöffel, B.J., & Hulshof, C. (2007) "Towards educational data mining: Using data mining methods for automated chat analysis to understand and support inquiry learning processes. *International Workshop on Applying Data Mining in e-Learning, ADML 2007, Crete*, pp. 27-36, 2007

Arvaja, M. (2007). Contextual perspective in analysing collaborative knowledge construction of two small groups in web-based discussion. *International Journal of Computer-Supported Learning, 2,* 1556-1607.

Bales, R. F. (1970). Personality and interpersonal behavior. New York: Holt, Rhinehart and Winston.

De Wever, B., Schellens, T., Valcke, M., & Van Keer, H. (2006). Content analysis schemes to analyze transcripts of online asynchronous discussion groups: A review. *Computers & Education, 46*, 6-28.

Dönmez, P., Rosé, C., Stegmann, K., Weinberger, A., & Fischer, F. (2005). Supporting CSCL with automatic corpus analysis technology. Proceedings of Computer Supported Collaborative Learning (CSCL 2005), Taipei, Taiwan.

Erkens, G. (2002). MEPA (Multiple Episode Protocol Analysis), http://edugate.fss.uu.nl/mepa. Accessed Nov. 14, 2007.

Erkens, G., Jaspers, J., Prangsma, M., & Kanselaar, G. (2005). Coordination processes in computer supported collaborative writing. *Computers in Human Behaviour, 21,* 463-486.

Forsyth, R., Ainsworth, S., Clarke, D., & O'Malley, C. (2007). Semi-automatic categorical coding of verbal data in social science. *Unpublished.*

Jonassen, D., & Kwon, H. (2001). Communication patterns in computer mediated versus face-to-face group problem solving. *Educational Technology Research and Development, 49*, 35.

Krystyniak, R. A., & Heikkinen, H. W. (2007). Analysis of verbal interactions during an extended, open-inquiry general chemistry laboratory investigation. *Journal of Research in Science Teaching, 44*, 1160-1186.

Kumpulainen, K., & Mutanen, M. (1999). The situated dynamics of peer group interaction: An introduction to an analytic framework. *Learning & Instruction, 9*, 449-473.

Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. Cambridge, Massachusetts, MIT Press.

Naidu, S., & Jarvela, S. (2006). Analyzing CMC content for what? *Computers & Education, 46*, 96-103.

Poole, M. S., & Holmes, M. E. (1995). Decision development in computer-assisted group decision making. *Human Communications Research, 22*, 90-127.

Rosé, C. P. (2007). TagHelper. http://www.cs.cmu.edu/~cprose/TagHelper.html. Accessed Nov. 14, 2007.

Saab, N., Van Joolingen, W. R., & Van Hout-Wolters, B. H. A. M. (2005). Communication in collaborative discovery learning. *British Journal of Educational Psychology, 75*, 603-621.

Strijbos, J.-W., & Stahl, G. (2007). Methodological issues in developing a multi-dimensional coding procedure for small group chat communication. *Learning & Instruction, 17*, 394-404.

van Boxtel, C., van der Linden, J., & Kanselaar, G. (2000). Collaborative learning tasks and the elaboration of conceptual knowledge. *Learning & Instruction, 10*, 311-330.

Weinberger, A., & Fischer, F. (2006). A framework to analyze argumentative knowledge confrontation in computer-supported collaborative learning. *Computers & Education, 46*, 71-95.

Acknowledgments

The authors thank Guido Bruinsma who generously provided a data set, Carolyn Rosé who provided information about TagHelper, Natalia Szargiej for her assistance during the coding process, and three anonymous reviewers for their helpful comments.