Artificial Intelligence in Education
C.-K. Looi et al. (Eds.)
IOS Press, 2005
© 2005 The authors. All rights reserved.

Automatic and Semi-Automatic Skill Coding With a View Towards Supporting On-Line Assessment

Carolyn ROSÉ, Pinar DONMEZ, Gahgene GWEON, Andrea KNIGHT, Brian JUNKER, William COHEN, Kenneth KOEDINGER
Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh PA, 15213

Neil HEFFERNAN
Worcester Polytechnic Institute
100 Institute Road, Worcester MA, 01609-5357

Abstract. This paper explores the problem of automatic and semi-automatic coding of on-line test items with a skill coding that allows the assessment to occur at a level that is both indicative of overall test performance and useful for providing teachers with information about specific knowledge gaps that students are struggling with. In service of this goal, we evaluate a novel text classification approach for improving performance on skewed data sets that exploits the hierarchical nature of the coding scheme used. We also address methodological concerns related to semi-automatic coding.

1. Introduction

The goal of the TagHelper project [5] is to develop text classification technology to address concerns specific to classifying sentences using coding schemes developed in support of educational research an other behavioral research fileds. A wide range of behavioral researchers including social scientists, psychologists, learning scientists, and education researchers collect, code, and analyze large quantities of natural language corpus data as an important part of their research. Currently there are a wide range of corpus analysis tools used to support corpus analysis work either at a very low level (e.g., word frequency statistics, collocational analyses, etc.) or at a high level (e.g., exploratory sequential data analysis once a corpus has been coded with a categorical coding scheme), but no widely available tools to partly or fully automate the time consuming process of doing the categorical behavioral coding or content analysis. In this paper, we address both technical and methodological concerns in developing technology for streamlining the categorical type of protocol analysis.

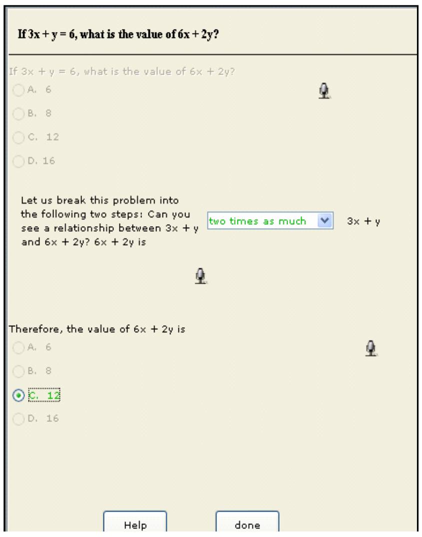
As an additional focus, in this paper we explore the potential of supporting on-line assessment with technology for automatic and semi-automatic skill coding of assessment items based on predictions from the text of the problem statements. On this level, the work reported in this paper is part of a larger effor towards addressing the "Assessment Dilema", which is a fundamental dilemma teachers face in trying to use assessment to guide instruction. Specifically, assessment takes time away from instruction and teachers cannot be sure the time spent assessing will improve instruction enough to justify the cost of lost instructional time. We are addressing this dilemma by building and experimentally evaluating the effectiveness of a web-based "Assistment" system for middle school math in Massachusetts. On-line testing systems that grade students and provide reports reduce the demands on the teacher. However, they do not fundamentally address the assessment dilemma. In contrast to previous approaches, the Assistment system aims to 1) quickly predict student scores on standards-based tests, 2) provide feedback to teachers about how they can specifically adapt their instruction to address student knowledge gaps, and 3) provide an opportunity for students to get intelligent tutoring assistance as assessment data is being collected. Assistments provide more focused instruction than the feedback that is typically given by on-line multiple-choice systems. A skill coding of assessment items is meant to facilitate assessment of student knowledge on individual skills. The resulting model of student mastery can then be used for predicting total scores on standards based tests as well as mastery on individual standards. A detailed assessment of student knowledge is meant to keep teachers informed about the individual needs of their students in order to support them in their task of preparing their students for the tests.

In the remainder of the paper we discuss in greater depth how a skill coding of assessment items can be used to facilitate on-line assessment. We then discuss alternative coding schemes we have been exploring. Next we discuss recent success in fully automatic skill coding using the 39 Massachussetts state standards for math at the 8th grade level (MCAS39). We also present results from an empirical evaluation of a coding interface that demonstrates the impact of automatic predictions on coding speed, reliability, and validity for semi-automatic skill coding. We conclude with discussion of current directions.

2. Motivation for Skill Coding for Assessment

The purpose of coding math problems with required skills is to eventually allow us to compute predictions about performance on state exams based on a limited number of interactions with the Assistments system (e.g., approx. 20 minutes per week). This is still work in progress. One of our planned approachs is to track each student's progress on the multiple skills and other cognitive components needed to do well on state tests, through a fully multidimensional IRT model or Bayesian inference network (e.g., 13) based on Assistment data. From this, one can predict the student's performance on a set of test questions tapping a distribution of skills similar to that seen in past state assessments. However, state tests are largely still developed using unidimensional IRT as a scaling tool [e.g. 10,8], which tends to force most individual differences to be driven by total test score. While there have been some successes developing multidimensional diagnostic reports for national tests such as the PSAT/NMSQT [4], our preliminary work with MCAS historical data suggests that fine-grained individual differences are swamped by gross number-correct groupings of students on high-stakes state tests, making multidimensional prediction problematic.

We are developing a cognitively-based, state-independent representation for encoding mathematical competency. This representation will be used to code state learning objectives, state test items, whole Assistment items and individual Assistment scaffolds. This coding then serves multiple functions within the proposed infrastructure. First, it allows us to draw correspondences between state standards and those of other states as well as the NCTM standards from which they are derived. As a byproduct, it allows us to match individual Assistment items to the corresponding NCTM standards as well as individual state standards. The proposed representation is finer grained than typical state standards. Thus, we argue that it is more suited to the task of predicting item difficulty because it explicitly represents the factors that make an item either difficult or easy for students.


If 3x + y = 6, what is the value of 6x + 2y?

If 3x + y = 6, what is the value of 6x + 2y?

A. 6

B. 8

C. 12

D. 16

Let us break this problem into the following two steps: Can you see a relationship between 3x + y and 6x + 2y? 6x + 2y is [two times as much] 3x + y

Therefore, the value of 6x + 2y is

A. 6

B. 8

[✓ C. 12]

D. 16

Figure 1 – Sample Assistment Item

	Overlap	**Unique Non-Overlap**
NCTM	"Use symbolic algebra to represent situations and to solve problems, especially those that involve linear relationships."	
MCAS P.7	"Set up and solve linear equations and inequalities with one or two variables, using algebraic methods, models, and/or graphs."	- specifies number of variables - models and graphs in addition to algebraic expressions
PSSA 2.8.8.E **(algebra strand)**	"Select and use a strategy to solve an equation or inequality, explain the solution and check the solution for accuracy."	- explaining the solution - checking the answer

Figure 2 Non-overlap of individual state learning objectives

While the state standards for mathematics nationwide are all based on the NCTM standards for mathematics, the example problem in Figure 1 illustrates why a state-independent component representation of mathematical knowledge is required for generalizing across state standards. Figure 2 displays the non-overlap between the relevant NCTM standard and the relevant learning objectives for Massachusetts (MCAS) and that of Pennsylvania (PSSA) for that problem. Because of the lack of direct correspondence between individual standards for different states as well as between NCTM standards and state specific standards, a more basic and fine grained representation is needed to demonstrate the precise connection between these different but very strongly related systems of standards.

A key characteristic of our cognitively-based knowledge representation is that it is composed of a vector of learning factors that distinguish problems from one another and predict item difficulty based on scientific findings from prior research and available state test results. An example of a learning factor is that students are known to have more trouble with scatter plots than line graphs partly because they are less common [1]. However, even important distinctions do not apply to all types of problems. For example, the graph type factor only applies to problem types that include graphs. In order to limit the number of judgments required to assign values to the representation vector for a specific item by human coders, we have designed a two-level representation in which first order learning factors identify the problem type (e.g., graph interpretation problems, simple algebraic simplification problems, or linear equality problems), and second-order learning factors make more fine grained distinctions (e.g., which type of graph, complexity of symbolic representation, or number of variables involved). Once the first-order factors have been specified, only a subset of second-order factors are relevant, and the others can be assigned a default value automatically.

3. Explorations of Fully Automatic Skill Coding

As we have been developing our cognitively based coding scheme, we have been exploring automatic coding with existing skill codings such as the MCAS39 as a proof-of-concept. The data we have consists of multi-class labels. There are 154 instances and 39 codes where each instance can be assigned a subset of these 39 codes. These codes are formed by 5 general categories; G, N, M, P, and D. Each of these categories has sub-level categories; for instance D-category is regarded as D.1, D.2, D.3, and D.4.

Applying a categorical coding scheme can be thought of as a text classification problem where a computer decides which code to assign to a text based on a model that it has built from examining "training examples" that were coded by hand and provided to it. A number of such statistical classification and machine learning techniques have been applied to text categorization, including regression models [12], nearest neighbor classifiers [12], decision trees [9], Bayesian classifiers [6], Support Vector Machines [7], or rule learning algorithms [2]. While these approaches are different in many technical respects that are beyond the scope of this paper to describe, they are all applied the same way. A wide range of such machine learning algorithms are available in the Minorthird text-learning toolkit [3], which we use as a resource for the work reported here.

One challenge in applying text classification technology to word problems is that the text of word problems contain many superficial features that make texts appear similar when they are very different at a deep level, or conversely, different when they are very similar at a deep level. These features include numbers, fractions, monetary values, percentages, dates, and so on. Thus, we replaced all the occurrences of features mentioned above with some pre-defined meta-labels, such as number, fraction, date, etc. A wide range of simple replacements can be made easily using search-and-replace facilities provided by the MinorThird toolkit. Other more complicated features must be tagged by hand and then trained using text classification technology.

As a baseline for our evaluation we explored training a binary classifier for each code using 4 standard text classification algorithms; namely SVM, DecisionTree, NaiveBayes, and VotedPerceptronLearner. In particular, SVM and VotedPerceptron classifiers are known to perform well on skewed data sets such as ours. We compared their performance using a

10-fold cross-validation methodology. SVM was the best performing approach. Nevertheless, although the performance was high in terms of percent correct, agreement with the gold stadard measured in terms of Kappa was very low, frequently 0, and in some cases negative.

The novel text classification approach we explore in this paper, which is our primary technological contribution, exploits the hierarchical nature of the MCAS coding scheme. The basic idea involves dividing the whole corpus into clusters according to the general categories, and then training and testing a binary classifier within each cluster separately. The hypothesis behind this approach is that if we can obtain relatively homogeneous clusters by exploiting each general category, then it will be simpler to train classifiers to operate within clusters because there will be fewer distinctions to make. Furthermore, since the texts within a cluster will be similar to each other, the trained classifiers can hone in on the fine distinctions that separate the lowest level classes.

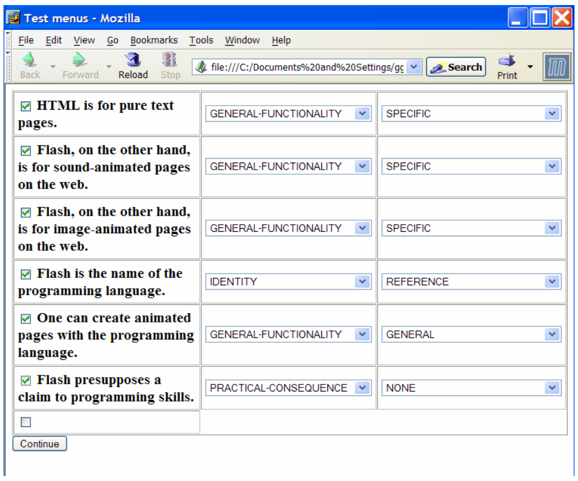
We used a 10-fold cross-validation methodology to train classifiers for splitting the data into clusters. For example, on each itteration, we train a classifier for each of the 5 general categories over 9/10 of the data. We then use the trained classifier to split the 10th segment into 5 separate clusters, one for each general category. We do this 10 times and then combine all of the separate clusters that belong to the same general category.Separation into clusters using the trained classifiers was not perfect. Nevertheless, the similarity between texts within clusters was still higher than over the whole corpus, and fewer separate low level classes were in each cluster than were in the whole set. We then used 10-fold cross-validation within clusters to obtain an accuracy for binary classifiers within clusters. We combined the results from individual clusters in order to obtain an agreement score for each of the MCAS39 labels across clusters using cluster specific classifiers.

On average the new classifiers performed significantly better than the baseline classifiers both in terms of percent agreement and Kappa ($p < .05$). Out of 29 classes that we had at least 2 instances of in our data, we were able to train classifiers to detect 13 of them at the .7 Kappa level or better. An additional 5 were between the .65 and .7 Kappa level, just missing an acceptable performance. An additional 5 showed significant improvement but did not reach the .7 level. For 4 out of the 29 classes, we were not able to achieve a substantial improvement over the baseline. In order to achieve an acceptable level of agreement while saving time over coding by hand, it is possible to allow the classifiers that have an acceptable performance be applied to the data and simply check the data over for places where additional codes from the remaining classifiers must be added. The first level classification of the data into rough clusters effectively narrows down the number of categories that must be considered for any single problem. Thus, we have determined that on average, using the information provided by the automatic predictions, a human coder would only need to consider 8 potential codes on average rather than 39 in order to achieve a complete coding of the data with human level agreement.

4. Issues Related to Semi-Automatic Skill Coding

While these explorations of automatic coding technology are promising, they leave open the question of what is the best course of action for dimensions of coding schemes where an acceptable level of agreement with a reliable gold standard cannot be achieved with a fully automatic approach. This is typically the case where there is a shortage of hand coded examples to use for training, or there are many categories that are very subtly differentiated, or there are many infrequently occurring categories. For example, the amount of hand coded data we had access to for the MCAS coding experiment described above was relatively small (only 150 instances). And several categories only occurred one or two times in the whole set. The question is whether it is better in cases where automatic coding cannot be done with an acceptable level of reliability to make automatic predictions, which will then be checked and corrected, or simply to code a portion of the data with no support of automatic predictions. To this end, we conducted a small formal study to measure the impact of automatic predictions on speed, validity, and reliability of human judgment when applying a categorical coding scheme.

Materials. For this study we use a coding scheme developed in connection with a net based communication project focusing on usage of technical terms in expert-layperson communication described in [11]. Materials for the experiment include (1) a 6 page coding manual that describes the definitions of a coding scheme with 14 separate codes and gives several examples of each; (2) a training exercise consisting of 28 example sentences; and finally, (3) 76 sentences for the experimental manipulation. Two expert analysts worked together to develop a "Gold Standard" of coding for the explanations used in the training exercises as well as the examples for the experimental manipulation that indicates the assigned correct code for each sentence.


Test menus - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///C:/Documents%20and%20Settings/gc
Search
Print
Continue

☑ HTML is for pure text pages.	GENERAL-FUNCTIONALITY	SPECIFIC
☑ Flash, on the other hand, is for sound-animated pages on the web.	GENERAL-FUNCTIONALITY	SPECIFIC
☑ Flash, on the other hand, is for image-animated pages on the web.	GENERAL-FUNCTIONALITY	SPECIFIC
☑ Flash is the name of the programming language.	IDENTITY	REFERENCE
☑ One can create animated pages with the programming language.	GENERAL-FUNCTIONALITY	GENERAL
☑ Flash presupposes a claim to programming skills.	PRACTICAL-CONSEQUENCE	NONE
☐		

Figure 3 Prototype TagHelper interface used in study

Coding interface. Participants coded the example sentences for the experimental manipulation using a menu-based coding interface displayed in Figure 3. For the standard coding interface used in the control condition, the example sentences were arranged in a vertical list on a web page. Next to each sentence was a menu containing the complete list of 14 codes, from which the analyst could select the desired code. No code was selected as a default. In contrast, a minimally adaptive version was used in the experimental condition. The only difference between the adaptive version and the standard version was that in the adaptive version a predicted code was selected by default for each sentence. That predicted code appeared as the initial element of the menu list and was always visible to the analyst. The other elements of the list in each menu were identical to that used in the standard version, so correcting incorrect predictions was simple.

Participants. The participants in our study were Carnegie Mellon University and University of Pittsburgh students and staff. 20 participants were randomly assigned to two conditions. In the control condition, participants worked with the standard coding interface described above. In the experimental condition, participants worked with the minimally adaptive coding interface described above that displays predicted codes for each sentence in the corpus set up in such a way that 50% of the sentences were randomly selected to agree with the Gold Standard codes, and the other 50% were randomly assigned. We randomly selected which sentences to make incorrect predictions about so that the distribution of correct versus incorrect predictions would not be biased by the difficulty of the judgment based on the nature of the sentence.

Experimental procedure. Participants first spent 20 minutes reading the coding manual. They then spent 20 minutes working through the training exercise using the coding manual. As they worked through the 28 example sentences, they were instructed to think aloud about their decision making process. They received coaching from an experimenter to help them understand the intent behind the codes. After working though the training exercise, participants were given a Gold Standard set of codes for the training sentence to compare with their own. Altogether training took 45 minutes. After the training phase, participants were given a five minute break. They then spent up to 90 minutes working through 76 sentences, coding each sentence.

First we evaluated the reliability of coding between conditions. Average pairwise Kappa measures were significantly higher in the experimental condition ($p < .05$). Mean pairwise Kappa in the control condition was .39, whereas it was .48 in the experimental condition. As a measure of the best we could do with novice analysts and 50% correct predicted codes, we also analyzed the pairwise Kappa measures of the 3 participants in each condition who's judgments were the most similar to each other. With this carefully chosen subset of each population, we achieved an average pairwise Kappa of .54 in the control condition and .71 in the experimental condition. This difference was significant ($p < .01$). The average agreement between these analysts' codes from the experimental condition and the Gold Standard was also high, an average Kappa of .70. Thus, the analysts who agreed most with each other also produced valid codes in the sense that they agreed with the Gold Standard. Next we evaluated more stringently the validity of coding. We found that analysts in the experimental condition were significantly more likely to agree with the prediction when it was correct (74% of the time) than when it was incorrect (16% of the time). This difference was significant using a binary logistic regression with 760 data points, one for each sentence coded in the experimental condition ($p<.001$). Average percent agreement with the gold standard across the entire population was significantly higher ($p < .05$), and average Kappa agreement was marginally higher in the experimental condition than in the control condition ($p=.1$). Average agreement in the unsupported condition was a Kappa measure of .48. In the experimental condition, average agreement with the gold standard was a Kappa measure of .56. Thus, we conclude that analysts were not harmfully biased by incorrect codes. Coding time did not differ significantly between conditions, thus providing some confirmation of the estimate that 50% correct predictions is a reasonable break even point for coding speed. Average coding time in the control condition was 67 minutes and 36 seconds. In the experimental condition average coding time was 66 minutes and 10 seconds. On average, time saved by checking rather than selecting a code was roughly equivalent to time lost by correcting a prediction after checking and disagreeing with a prediction.

5. Current Directions

In this paper we have discussed the problem of automatic and semi-automatic coding of on-line test items both from the language technology and human-computer interaction angles. The specific application area we discussed was a skill coding of math assessment items, the purpose of which is to allow the assessment to occur at a level that is both indicative of overall test performance on state exams and useful for providing teachers with information about specific knowledge gaps that students are struggling with. We presented results from an evaluation that demonstrates that skill coding of math assessment items can be partially automated and a separate formal study that argues that even in cases where the predictions cannot be made with an adequate level of reliability, there are advantages to starting with automatic predictions and making corrections, in terms of reliability, validity, and speed of coding. One focus of our continued research is developing new text classification techniques that work well with heavily skewed data sets, such as our MCAS coded set of math problems.

6. Acknowledgements

This work was supported by the National Science Foundation grant number SBE0354420 and a grant from the US Department of Education, Institute for Education Sciences, Effective Mathematics Education Research grant number R305K030140.

References

[1] Baker R.S., Corbett A.T., Koedinger K.R., Schneider, M.P. (2003). A Formative Evaluation of a Tutor for Scatterplot Generation: Evidence on Difficulty Factors. *Proceedings of the Conference on Artificial Intelligence in Education*, 107-115.

[2] Cohen, W. and Singer, Y. (1996). Context-sentsitive learning methods for text categorization, In *SIGIR'96: Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 307-315.

[3] Cohen, W. (2004). *Minorthird: Methods for Identifying Names and Ontological Relations in Text using Heuristics for Inducing Regularities from Data*, http://minorthird.sourceforge.net.

[4] DiBello, L. and Crone, C. (2001, July). Enhanced Score Reporting on A National Standardized Test. *Paper presented at the International meeting of the Psychometric Society*, Osaka, Japan.

[5] Donmez, P., Rose, C. P., Stegmann, K., Weinberger, A., and Fischer, F. (to appear). Supporting CSCL with Automatic Corpus Analysis Technology, to appear in *the Proceedings of Computer Supported Collaborative Learning.*

[6] Dumais, S., Platt, J., Heckerman, D. and Sahami, M. (1998). *Inductive Learning Algorithms and Representations for Text Categorization*, Technical Report, Microsoft Research.

[7] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features, In *Proc. 10th European Conference on Machine Learning (ECML)*, Springer Verlag, 1998.

[8] Massachusetts Department of Education (2003). *2002 MCAS Technical Report*. Malden, MA: Author. Obtained August 2004 from http://www.doe.mass.edu/mcas/2003/news/02techrpt.pdf

[9] Lewis, D. and Ringuette, R. (1994). A Comparison of teo learning algorithms for text classification, In *Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 81-93.

[10] Mead, R., Smith, R. M. and Swandlund, A. (2003). *Technical analysis: Pennsylvania System of School Assessment, Mathematics and Reading.* Harrisburg, PA: Pennsylvania Department of Education. Obtained August 2004 from http://www.pde.state.pa.us/a_and_t/lib/a_and_t/TechManualCover.pdf.

[11] Wittwer, J., Nückles, M., Renkl, A. Can experts benefit from information about a layperson's knowledge for giving adaptive explanations?. In K. Forbus, D. Gentner, T. Regier (Eds.), *Proc. Twenty-Sixth Annual Conference of the Cognitive Science Society*, 2004. 1464-1469.

[12] Yang, Y. and Pedersen, J. (1997). Feature selection in statistical learning of text categorization, In *the 14th Int. Conf. on Machine Learning*, pp 412-420.

[13] Yan, D., Almond, R. and Mislevy, R. J. (2004). *A comparison of two models for cognitive diagnosis. Educational Testing Service research report #RR-04-02.* Obtained August 2004 from http://www.ets.org/research/dload/RIBRR-04-02.pdf