AUTOMATIC CODING OF TRANSCRIPT DATA FOR A SURVEY OF RECENT COLLEGE GRADUATES

Daniel J. Pratt and Jennifer W. Mays, Research Triangle Institute
Daniel J. Pratt, P. O. Box 12194, Research Triangle Park, NC 27709

KEY WORDS: Classification, natural language, computer-assigned codes

ABSTRACT

Research Triangle Institute collected transcripts from participating schools as part of a survey of recent college graduates for the U.S. Department of Education Center for Statistics. The task included classifying nearly one million course records according to subject matter. An automatic coding system (autocoder) was developed specifically for this task, but with the thought of expanding the general method to other tasks. The system featured a two-step coding process: first the course was assigned to a generic department, and then it was given a code within that department. An "expert" user developed the coding rules (dictionaries) for use with the system. The autocoder then used the rules to process the coded file; if not, it was set aside for manual coding. For those records which remained uncoded, the autocoder provided computer-assisted coding tools. As a final step, the course records were reprocessed against the coding dictionaires to provide summary statistics and guarantee consistency of coding. This project has given its developers insight into the nature of automatic coding of free-form text and the assurance that a more general and flexible automatic coding system could be efficient and cost-effective for future projects.

I. INTRODUCTION

Determining efficient methods to interpret free-form text responses for analysis has been an interesting issue for survey and computing professionals for years. One solution has been to hire personnel to classify the responses manually, with or without computer assistance, using a well-defined set of rules for coding the responses. A recent alternative approach has been to develop a dictionary of possible responses and their codes, and use this dictionary as the basis for an automated coding system. Research Triangle Institute (RTI) had to choose between these approaches to classify college course titles as part of a survey of recent college graduates (RCG).

RTI contracted with the U.S. Department of Education Center for Statistics to provide a public use file of transcripts for students who received bachelor's degrees between July 1, 1985 and June 30, 1986. The eligible sample size was 16,977 students. Transcripts were collected from degree-granting institutions and other schools from which students had transferred. The public access transcript file was to include a classification for each course title in the file, such that the course level data could be used in analysis. RTI used a three digit numerical coding scheme to distinguish various types of courses. The first two digits mapped directly to the two digit Classification of Instructional Programs (CIP) system developed by G. S. Malitz (1987). The third digit was used, as needed, for further specificity. There were nearly 1 million course level records.

In selecting the best approach to coding the transcript data, two alternatives were considered: computer-assisted manual coding done by trained staff members, or an automated coding system using a rules dictionary developed and updated by an expert or a pool of experts. RTI had the opportunity to review the transcript coding methods used in a prior RCG study before reaching a decision. National Opinion Research Center (NORC) developed a microcomputer-based system called the Computer Assisted Data Entry System (CADE) to combine the tasks of keying the transcript data and classifying course titles. There were twenty transcript coder/keyers trained for the task, and each transcript was handled as an entity to provide internal consistency and to reduce the likelihood of lost data. The major features of this coding system included extensive training in the use of the CIP system to code courses properly and the development of an on-line version of these coding rules to provide further assistance (C. Jones et al. 1986). Because a human coder had to make a decision as to how to classify each course title, the CADE implementation was a computer-assisted manual model.

RTI personnel were interested in evaluating ways of solving the generic, free-form text problem and decided that this project offered an opportunity to develop an automated coding system. The development and implementation efforts were viewed as a test to see if an automated coding system could handle the transcript coding task efficiently. If the results were favorable, RTI would consider developing a generalized automatic coding software system for the classification of free-form text responses.

II. INITIAL DESIGN FEATURES

The RCG autocoder was developed using the VAX C programming language in a Digital Equipment Corporation VAX/VMS environment at RTI. The initial design involved using standard VMS file structures. The features of the RCG autocoder included: processing course records in batch mode; updating the coding dictionaries; handling uncoded records through a menu of interactive options; and reprocessing the entire set or a specified subset of the records which had been processed up to that point.

The coding rules set (dictionary) is the most important feature of the coding system because it is the chief determinant of how each record gets coded. The quality of coding is directly related to the quality of the rules used in coding. This set of rules may be developed using: prior coding efforts; an already established standard set of rules for the given

task; or the use of an expert or pool of experts to develop appropriate rules and their codes. The last approach was taken by RTI for the RCG study. One individual expert was given the responsibility for determining meaningful rules based on a review of sample course records and a thorough understanding of the classification scheme. As mentioned earlier, the primary reference for coding rules was the CIP system.

The coding of course records was handled as a two step coding exercise, and thus there were two coding dictionaries. First, the original school department was standardized using a department rules set. Then, within each standardized department, there was a set of rules which provided the final code for the course. For each course, the listed department provided meaningful categorical information and the text field provided specificity within department. This two-dictionary structure helped reduce the CPU time spent searching for a rule to code the course.

Table 1. The Order of Autocoding Processing Steps

1. A batch of course records was processed overnight.
2. The expert user reviewed uncoded records using interactive tools.
3. The expert user updated rules base to enable coding of previously uncoded and yet-to-be-coded records.
4. The remaining uncoded cases were reprocessed in batch mode.

Transcripts were handled separately according to whether they were from the degree-granting institution or from another (transfer) institution. The content of the two types of transcripts differed significantly, and it was felt that the data entry component would be better managed if they were handled separately.

The transcript data was defined hierarchically. There were transcript level records, term level records, and course level records, all of which were keyed by RTI data entry keyers. The course level records comprised the input for the autocoder. Each record was of fixed length and contained a course title and department as well as ancillary information not needed for coding. The system itself was divided into two basic components: the batch processing component and the interactive component.

A. Batch Processing Steps

Batch processing handled the overwhelming majority of the course records which were coded for the recent college graduate study. Batch processing of each course record involved three steps. The text was parsed into a standardized structure. Then the school department was translated into a uniform department. Finally, the parsed text was compared with rules for that department.

During parsing the input record was read and reformatted into a fixed length output structure. The original department and course number, which had been treated as one field for keying, were split into two fields and the course number was dropped. Next the 40 character course description was divided into distinct words.

Certain rules governed this division into words. The recognized character set was the alphabet. All other characters were dropped and viewed as word delimiters. Lower case letters were converted to upper case for ease in comparing the words. In addition, a word was defined as having seven or fewer letters since it was felt that this was enough to determine word content. The resident expert defined a small set of trivial words to ignore during processing. This set included words such as "INTRO" and "FRESHMAN", which added no meaning to a course title and were thus deemed insignificant. The number of parsed words was limited to the first six non-trivial words.

The example below illustrates the parsing strategy employed. The input department was read. All numbers were dropped. The course title was separated into words by the word delimiter, in this case "/". "INTRO" appeared in the list of words to ignore so it was discarded. Finally, "PROGRAMMING" was stripped down to the first seven letters for the resultant parsed text of "PROGRAM".

Table 2. Parsing the Input Course Record

Initial Course Description

School ID Number	Department and Number	Course Title
12345	C S 101S	INTRO/PROGRAMMING

Parsed Text Output

School ID Number	Department	Parsed Course Title
12345	C S	PROGRAM

The next step was to standardize the department. The department and school identifier were compared with the department dictionary, which included the original department, a school identifier (as appropriate), and a standard department code. If a school specific conversion existed for a given department, it was used. Otherwise, the file was checked for a translation without reference to any school. Many schools used the same department abbreviation conventions so general rules were applicable. However, some schools had their own abbreviations which were often in conflict with general rules. An example is the school department CS. This meant Computer Science at most schools, but it referred to Classical Studies at one school so a

special rule was necessary for this school. If no translation was found, the department was set to NONE. Refer to Table 3 to follow the example course through the department translation step. Note that because the school ID did not match the school specific rule ID, the general translation to COMP was applied, indicating a Computer Science offering.

Table 3. Translating the Department

Input Fields		Translation Rules		Output Dept
Sch. ID	Dept	Orig Dept	Sch. ID	Standard Dept.
12345	C S	CPS		COMP
		C S	32109	CLAS
		C S		**COMP**

The final step was coding the course. Once a standard department was established, the parsed text was compared with the set of rules for that department. Each rule included the rule number within department, the rule words (one or two), and the associated code. The parsed text had to include each word in a given rule for a match to occur. Each word of the parsed text was compared with each word in the rule. If the number of matched words was less than the number of rule words, that rule would fail and the next rule for the department was read. The rules dictionary constituted a list of rules ranked in order of descending importance. If no rule matched the parsed text, the record was left uncoded.

Table 4. Coding the Course

Input Fields		
Dept	Parsed Text	
COMP	PROGRAM	
Dept/Rule Number	**Rule 1**	**Rule 2**	**Code**
COMM999			090
COMP001	CLEP		000
•			
•			
COMP012	DATA		112
•			
COMP028	**PRO**		**111**
•			
COMP999			110

Result Code: 111

Table 4 demonstrates that the example parsed text, PROGRAM, was compared sequentially with each rule in the COMP department until a match was found. It is important to note that the applicable rule contains fewer letters than the parsed text. The minimum number of letters were used in a rule to allow for variant abbreviation conventions. The root rule words must match, but the parsed text may contain additional trailing letters for each rule word. However, the rule word may not be longer than the parsed word. Once the appropriate rule was identified, the code field was assigned the value of 111, and the course record was written to the coded data file.

Each of the above steps was followed for every one of the 923,037 records in the set of course level records. Uncoded records were written to one output file and coded records were written to another file. The major coding effort was devoted to the batch processing phase. However, the system provided additional tools for the expert user.

B. Interactive Tools

Each of the dictionaries was equipped with interactive tools to enable update and query access. Thus, the user was able to add, delete, or modify rules in each of the dictionaries. As the system processed records interactively, any new rules would be compared with previously uncoded records for a given department. If the new rule applied to a record, it would be coded and transferred to the coded data base. If records continued to be uncoded after reprocessing, the user could correct any miskeyed input text, or the record could be coded manually.

Access to the coding dictionaries enabled the user to modify and expand the rules sets over the course of the project. The user could access the dictionaries directly or via the processing of uncoded records. The three data dictionaries were: the list of words to ignore during parsing; the department standardizations; and the coding rules within each standard department.

Each dictionary was accessible for additions, deletions, updates, and queries. Every change required confirmation to avoid unintended updates. Each dictionary had an indexed file structure which was used for direct (random) access. As a result, searching for the appropriate entry was very efficient.

In addition to modifying the rules bases, the interactive coding menu provided other utilities for handling uncoded course titles. One option was to modify the text of the course title or department. This would be appropriate if a keying error prevented the course from being coded. Because this option allowed the user to change the input data, the old data record was archived.

In some cases the user needed to code a course title by manual assignment without modifying the rules bases. This circumstance arose if the course title was an isolated occurrence that did not merit an autocoding rule. Every course that was coded by this interactive, manual process was flagged as a hand-coded record and could thus be excluded from final reprocessing.

The interactive processing of uncoded records was done in one of two ways. The first option was to process each of the uncoded courses sequentially. The other method was to handle the uncoded courses for a specified department. This latter design would be preferable if there were different experts for each discipline performing the coding review tasks. The

recommended approach to resolving uncoded courses was to develop new rules, as appropriate, so that future batch coding would be more efficient.

III. RULES BASE DESIGN CONSIDERATIONS

The structure and content of the rules base dictated the efficiency and accuracy of the automated coding operation. As mentioned, the RCG course processing was implemented using two dictionaries. First, a standard department was assigned from the department rules base. Then, the parsed text was compared with the set of rules for that department. The content of the course-level coding dictionary changed significantly over time. This affected the level of specificity with which each record was coded.

Certain rules were set up as special default rules for departments. These rules contained no rule words and were applied only if no other specific rules matched the input text but the department matched. There were 79 such rules in the coding dictionary. In the original design of the system, records coded with the default rule were viewed as "semi-coded" because the system knew which department offered the course but the course description did not secure a code within the current rules base. Using this scheme, there were three types of records: coded records, in which a specific rule applied; semi-coded records, which matched only the default rule for the department; and uncoded records, which failed to match any rules.

During the processing of the data, efforts were made to accelerate the automated process. This included revising the rules in the coding dictionary. It was assumed that the default code for a department was correct in most cases, and specific rules for that code were thus unnecessary. It was necessary only to include rules in that department which mapped to different codes. Records which previously had been identified as coded, semi-coded, and uncoded were now considered exception, default, and uncoded records, respectively. This reduced the size of the coding dictionary, but it was no longer possible to distinguish between a "fully coded" record and a "semi-coded" record.

The majority of records were coded using default rules. Table 5 shows that 76.4 percent of the primary course records and 63.6 percent of the transfer course records were coded strictly based on the standard department because exception rules did not apply.

The disadvantage of this method was that the most commonly used rules no longer appeared first for a given department, so more processing time was used before a matching rule was found. All exception rules were prioritized above the default rule. This increased the average number of rules read per course processed. The average number of rules read was 45.7 for the primary school file and 38.9 for the transfer file.

However, the new method handled exceptions nicely because the expert could arrange the rules in order of descending importance. The default rule was applied only if all of the exception rules had failed to match with the parsed course title.

Table 5. The Impact of Using the Exception/Default Rule Organization.

	Primary File
# Total Records	824,291
# Coded Records	810,209
# of General Case Records	619,390
Percentage of General Records	76.4
Average Number of Rules Read Per Record	45.7
	Transfer File
# Total Records	98,746
# Coded Records	89,760
# of General Case Records	57,132
Percentage of General Records	63.6
Average Number of Rules Read Per Record	38.9

Even though the rules set was overhauled after the coding task was in progress, consistency was not a problem because the entire data file was reprocessed after the rules set was stabilized.

IV. CONSISTENCY

The issue of consistency in coding is one of the most persuasive reasons for selecting an automatic coding system over the computer-assisted manual coding alternative. Once a set of rules is established, the computer utilizes those rules the same way every time. This cannot be said for human coders, even the most highly trained and proficient ones. There are different ways to interpret rules, and inter-coder discrepancies invariably are part of any manual coding operation. In fact, the same individual may make two different decisions for a given input text field at two different times.

If a rules set does not change during a project, all automated coding will be consistent. However, the RCG autocoding approach allowed for a dynamic set of coding rules. Therefore, a course could be coded one way, the pertinent rule changed subsequently, and the same course coded differently for a different student. It is always possible in this environment to recode the entire set of records as a final step to provide complete consistency. Cost is a major factor in deciding whether or not this is feasible for a given study. If a rules base is stable for the duration of coding, the issue becomes irrelevant. For the RCG study, the rules set went through significant transformation during the coding phase. Final reprocessing guaranteed consistent coding.

The results showed that 33,778 of the 923,037 records, or 3.66 percent, had different result codes as compared with initial coding. A large percentage of these recoded records were coded early in the project, prior to the reorganization of the coding dictionary. The results indicate that a dynamic rules set may be subject to inconsistencies over time, and that it may be necessary to recode records as a final processing step.

V. COST AND EFFICIENCY CONSIDERATIONS

The comparative costs of different approaches to the task of coding free-form text is worthy of investigation. We can provide only some very rough cost comparisons for the RCG study based on some reasonable, albeit arbitrary, assumptions. Keying of the data was not a factor because it was required as part of the deliverable data file. It is assumed that keying costs are the same whether a manual or automated approach is used. The CADE system provided an efficient mechanism for reducing overhead associated with tracking records through data entry and coding by handling all tasks in one step (C. Jones et al. 1986). On the other hand, the automated approach requires the keyed text file only as input since codes are not manually assigned.

The development and maintenance of the rules base requires the major investment of human resources for the autocoder, whereas trained coding personnel are needed for the manual method. Assuming that it takes one minute, including invested training time, to assign a code by hand to each course, it would have taken approximately 7.6 years (given 167 hours per month) for one person-equivalent to code all 923,037 records. In comparison, the autocoder was implemented by one expert user and all records were processed in less than eight months of elapsed time. This included the time spent developing and maintaining the coding dictionaries.

Processing costs are another major consideration in evaluating coding systems. If the system is too expensive to operate, no matter how attractive otherwise, it will not be used. Since the RCG autocoder was developed specifically for this project, and was not significantly tested prior to its use, it went through some evolution as the project went on. The revisions of the techniques used dramatically increased the speed of reprocessing the data.

Table 6. A Comparison of Estimated CPU Time, Elapsed Time Between Initial Coding and Final Reprocessing.*

	Initial Coding	Reprocessing
CPU	~ 30.5 hours	~ 7.5 hours
Elapsed	~ 600 hours	~ 25 hours

* All work was done using the RTI Ragland Computer Center VAXcluster. Elapsed time is affected by the number of other users on the cluster.

As may be seen above, the CPU time for reprocessing was reduced by a factor greater than 4. The elapsed time to process records was much more dramatic--reprocessing reduced time required by a factor greater than 20. There were a few factors responsible for this tremendous improvement. The original specifications called for direct access to each output record based on a unique key. This prevented the occurrence of duplicate records and allowed immediate query access. Since the key was 22 characters in length, the cost to maintain the key structure was not worth the great sacrifice in efficiency. Second, the coding dictionaries had been opened and closed for each course processed. This added a significant amount to the elapsed time for processing. During reprocessing, the output file was sequential rather than indexed and the dictionaries were opened and closed only once.

Even in the worst-case scenario, the cost of processing is not significant compared with the cost of human resources. An automatic coding implementation should be given serious consideration for a task such as transcript coding if there is a choice between two systems which have already been developed. Despite the problems related to slow turnaround time, the RCG autocoder compared favorably with a manual coding alternative.

VI. RECENT EXPERIENCE

Since the completion of the RCG study, we have learned more about general automated coding methods. We have researched various systems implemented with a number of variations on the methods used here.

Coding dictionaries have been established in a number of ways. The ordered rules approach is just one technique used in automatic coding. A different way to establish a coding dictionary is to code a representative sample manually, and this becomes the basis for later autocoding decisions. A third method is to use an on-line version of a standard coding manual to determine the coding rules.

Another distinguishing feature of various coding systems is the algorithm used to determine what constitutes a match between a given rule and a text field. Exact matching of every word in the text to every word in the rule is one technique. A prioritized set of required rule words, where some of the words in the text must match all of the words in the rule, as in the RCG project, is another. Others have successfully implemented a weighted scoring method in which the informational content of matching words, known as the heuristic weight, contributes to the overall ranking of possible solutions.

"Automatic Coding Methods and Practices", written by Albert Bethke and Daniel Pratt, provides a historical perspective on this subject and explains each technique in detail (Bethke and Pratt 1989). As an excerpt from the paper illustrates, most coding algorithms can be reduced to the following steps:

"For each response record, parse the response into words, standardize this set of words and match the set of words in the response against a 'coding dictionary' to find the best matching phrase. If the match is good enough, then assign that code to the response record, else leave the record uncoded -- it will be manually coded later."

The RCG coding system evolved significantly as the project progressed. Though it must evolve even further to be used for more general coding, it offered lessons which can be applied to future automatic coding efforts. It showed that

the structure of the rules base is the most significant factor in both storage and processing costs. In some cases, both can be kept to a minimum using the same technique, but sometimes storage costs must be weighed against the processing costs in designing a rules base. In this case, coding courses in two steps minimized both the storage space and the search time, keeping storage and processing costs down. Structuring the rules with most common ones first increased storage costs and reduced processing costs. In contrast, structuring the data base with exceptions first reduced storage costs significantly, but slightly increased the per-record processing time. RTI software developers are utilizing the experience gained from this project to implement a general, free-form text, autocoding system for survey research applications.

REFERENCES

MALITZ, G. S., A Classification of Instructional Programs. Washington, DC: Center for Education Statistics, US Department of Education, 1987.

JONES, CALVIN, REGINALD BAKER and ROBERT BORCHERS, High School and Beyond Postsecondary Education Transcript Study: Data File User's Manual. Chicago: National Opinion Research Center, 1986.

BETHKE, ALBERT and DANIEL PRATT, "Automatic Coding Methods and Practices", Draft Article. Research Triangle Park: Research Triangle Institute, 1989.

ACKNOWLEDGEMENTS

The authors would like to express their gratitude to certain individuals who contributed significantly to the success of this project: Barbara Elliott, the expert system implementer for the transcript coding application; John Riccobono and Graham Burkheimer, leaders of the recent college graduate study; and Randy Lucas, who provided software consultation and direction for the developers.