USING EXPERT SYSTEMS TO MODEL AND IMPROVE SURVEY CLASSIFICATION PROCESSES

Frederick Conrad[1]
U. S. Bureau of Labor Statistics

In Lyberg, L., Biemer, P., Collins, M. DeLeeuw, E., Dippo, C. Schwarz, N. & Trewin, D. (Eds.)(1997), *Survey Measurement and Process Quality*. New York: John Wiley & Sons.

1. Introduction

Classification is ubiquitous throughout the survey process. Most analyses require that respondents' experiences, opinions or knowledge be assigned to categories in order to be treated as data. Sometimes this is done by the respondents themselves. For example, when counting their expenditures for a particular product category -- perhaps "meals in restaurants" -- respondents must first determine if individual purchases involve members of that product category -- should "going out for dessert" be figured in the response? On other occasions, data are classified by the interviewer. For example, the interviewer might need to indicate whether the respondent lives in an "apartment building" or "private home." On still other occasions, classification is carried out by classification specialists, sometimes called "coders," who assign open-ended statements, for example about work or educational background, to categories in a formal classification scheme, such as an occupational or field of study classification system.

By assigning open-ended responses to a category, the researcher is essentially signaling that they share certain critical features. This allows researchers to aggregate the responses and produce summary statistics, even if the responses differ from one another in some of their features. In principle, a response can be uniquely associated with a single category, though, in practice, the ambiguous mappings between responses and categories is one of the factors which makes the classification process challenging. Reliability between coders is not perfect, and the agreement rates can vary from one category to the next (Cantor & Esposito, 1992).

When responses are relatively short -- just a few words in length -- they are typically classified by coding clerks, some time after the data have been collected. The clerks either do this manually or with computer support, such as on-line coding manuals or software that suggests possible categories (codes) from which the clerk must select one. In some survey operations, the computer actually assigns codes to responses, though in even the

[1] The views expressed here are those of the author and do not reflect the position of the Bureau of Labor Statistics. I wish to thank the following colleagues for their invaluable contributions to the activities and ideas reported in this chapter: Marty Appel, Jelke Bethlehem, Paul Carney, Dan Gillman, Brad Hesse, Rick Kamalich, Dan Kasprzyk, Brian Kojetin, Leda Kydoniefs, Lars Lyberg, Jean Martin and Dan Pratt. Of course, any errors or misrepresentations in the text are entirely my responsibility.

best of these systems, some responses still need to be coded clerically. Computer support for classification tasks has been justified on the grounds that, relative to manual coding, it makes the process more reliable, makes additional resources available to the coder, and does so without increasing costs (*e.g.* Andersson & Lyberg, 1983). Whether they are coded manually, automatically or by some combination of the two, simple, open-ended responses are usually classified by looking up the response in a dictionary where it is associated with a small number of codes -- possibly a single code.

Some surveys involve more complex classification which is based on large numbers of variables, possibly elicited through unscripted dialogue with a respondent. These tasks are performed by professionals who have acquired expertise in the relevant subject area. They investigate hunches (classification hypotheses) by looking for relevant evidence much as a detective would. Because many factors are potentially relevant, they focus on just those factors that, experience tells them, promise to help in making the decision. The kinds of data required for these decisions may vary between categories. For example, the difference between a “material handler” and a “fork lift operator” may depend on the equipment they operate but the relevant criterion for distinguishing an “optometrist” from an “ophthalmologist” is more likely to be their education and certification, though equipment could be pertinent as well. The current chapter is concerned with this kind of classification expertise and a type of software that can support such tasks, expert systems.

We have developed two prototype expert systems that model classification tasks of this sort performed at the Bureau of Labor Statistics (BLS) and will use them throughout the chapter to illustrate general concepts about expert system technology in the survey process. In the next section (two), we review recent research and development in automated coding. By and large this work has been applied to the classification of short, open-ended responses and so contrasts with our work using expert systems. The following section (three) presents some of the basic concepts of expert system technology and discusses why expert systems are appropriate for a more complex classification. Then, in the fourth section, we describe the prototype expert systems. The section is intended to characterize the kinds of tasks for which this technology is appropriate. In the fifth section, we present representative results from an evaluation study and discuss some of the practical considerations of developing this kind of software in survey organizations.

2. AUTOMATED AND COMPUTER-ASSISTED CODING

Since the late 1970s, computers have been used to code simple, open-ended responses to survey questions. These responses usually consist of a few words each, for example, “I work in a cannery -- at the loading dock.” The response is classified on the basis of its content and known statistical relationships between the content words and categories in the classification scheme. The task does not require knowledge of the subject area. This contrasts with more complex classification where subject matter expertise is required and can be provided through expert systems. To provide a backdrop for our discussion of expert systems, the current section reviews some of the major developments in software for coding simple responses. The general idea involves matching the response string to an entry (or entries) in a dictionary in which each entry is associated with a category or numerical code. The innovation in this research has occurred in developing algorithms for constructing and searching dictionaries of this sort.

This approach has been applied primarily to industrial and occupational coding (*e.g*. Andersson & Lyberg, 1983; Appel & Hellerman, 1983; Appel & Scopp, 1987; van Bastelaar, Hofman & Jonker, 1987; Campanelli, Thomson, Moon & Staples, this volume), but has also been used to classify descriptions of products (Andersson & Lyberg, 1983), field of study (Bobbitt & Carroll, 1993; Ciok, 1993; Pratt & Burkheimer, 1994; Pratt & Mays, 1989), library book loans (Andersson & Lyberg, 1983) and demographic characteristics such as ethnic origin and language (Ciok, 1993). This effort has been carried out throughout the world, primarily in government statistical agencies (for reviews see Bethke & Pratt, 1989, Gillman & Appel, 1994, and Lyberg & Dean, 1990).

The various efforts to automate the coding process can be divided into “autocoding” (or “batch”) processing -- applications that entirely automate the coding process -- and “computer-assisted” (or “interactive”) coding systems -- software that provides human coders with pertinent information which the coders then incorporate into their classification decision making. Some coding systems can be run in both batch and interactive modes (*e.g.* ACTR, the system developed by Statistics Canada). Sometimes the two are used in conjunction: A large data set is submitted to batch processing in a first pass; those cases that cannot be coded automatically are classified interactively in a second pass. Unlike manual coding of these data, which usually occurs after the data have been collected, automated coding (in all of the above forms) can take place during the interview in both CATI (Computer Assisted Telephone Interviewing) and CAPI (Computer Assisted Personal Interviewing) environments (Pratt & Burkheimer, 1994).

Dictionaries, which associate possible response with codes, can be derived from coding manuals or previously coded data sets. There is sometimes a one-to-one mapping between content words (or phrases consisting of such words) and codes, though on other occasions a word or phrase is associated with multiple candidate codes. The process of mapping words to codes is sometimes carried out manually (*e.g.* Pratt & Mays, 1989), but for large classification systems it is usually automated. The U.S. Census Bureau’s Automated Industry and Occupation Coding System or AIOCS (Appel & Hellerman, 1983; Appel & Scopp, 1987) relies on an automatically generated dictionary (O’Reagan, 1972). The words from all of the phrases in the coding manual are converted to a standard form (basically the root word stripped of affixes). A dictionary is created from all of these standardized words. The dictionary includes weights for each word determined by the number of different codes with which it is associated. Links are created between the words in the dictionary and the phrases containing those words. The links and the phrases themselves form the knowledge on which the classification is based. In general, no subject matter knowledge is involved.

The actual coding process involves matching response words and phrases to a dictionary entry. This is done through either dictionary algorithms or weighted algorithms

(Andersson & Lyberg, 1983). A dictionary algorithm requires a response string (a word or phrase in standard form) to match a single dictionary entry (sometimes independent of word order). While the match need not always be exact (*e.g.* Pratt & Mays, 1989) it must be close enough to an entry so that the match is unambiguous. If such a match is found, the response string is assigned the associated code. This is an effective strategy when responses are simple; as the response strings become longer, perhaps spanning the answers to several questions, they are more likely to match multiple entries.

Weighted algorithms make it possible to code such ambiguous matches with acceptable accuracy. In AIOCS, the words in the dictionary are weighted according to how well they predict a particular code: A word that is associated with only a single code receives a large weight while the weight is smaller for a word associated with many codes. The algorithm selects candidate phrases from the dictionary based on a score which is computed from the number, percent, and weight of matching words (Appel and Scopp, 1987). If it identifies a single phrase then the code associated with that phrase is assigned to the response string. If the algorithm selects multiple phrases, then the one with the highest score is chosen, assuming it exceeds the score of the closest competitor by a specified ratio, and the response is classified accordingly. It is possible that the algorithm will not identify any phrases with an acceptable score, in which case, the response is coded manually.

Certain systems include a type of knowledge that enables them to circumvent the algorithmic matching process. AIOCS performs a “logical analysis” in which rules map particular keywords directly to codes. If such keywords are present in the response field, they trigger the appropriate rule and the code is assigned without the usual lexical search. The number of keywords is kept small to assure that the approach is efficient. If the set of keywords were to become large, then complex matching issues would compromise its value. Clearly, the approach is beneficial: By one recent evaluation, AIOCS assigned codes using logical analysis at a much lower error rate than was found overall (Gillman and Appel, 1994).

A coding system for classifying field of study that uses a comparable approach is described by Pratt & Mays (1989). The abbreviated names of academic departments offering particular courses were mapped to field of study categories by rules that were general across institutions (“CS” => computer science) unless an institution used unconventional abbreviations in which case rules were developed that were specific to an institution (at institution 12345 “CS” => classical studies). These researchers were able to tune the system’s effectiveness by manually re-ordering the rules until they found an optimal sequence.

The logical analysis in AIOCS and mapping of abbreviated department names in the Pratt & Mays (1989) system, are both examples of a “rule-based” approach, in that knowledge is encapsulated in a set of rules which are invoked under precisely specified circumstances. They both exploit relationships between responses and codes that experts know to be the case, but which are not the result of executing an algorithm. In addition, they improve performance by avoiding computationally intensive matching procedures. This type of decision making is a version of the reasoning that drives expert systems, discussed in the remaining sections.

A range of performance has been reported for the numerous automated and computer assisted coding systems (Bethke & Pratt, 1989). The two measures that are jointly used to evaluate a system are (1) coding or production rate and (2) agreement rate (or its inverse, error rate). The first of these refers to the proportion of cases that are coded (regardless of accuracy) and the second is an accuracy measure, usually derived by comparing computer generated codes to “truth codes,” that is, the judgments of coding experts for some test set of responses. Without updating the dictionary, one can increase the coding rate by relaxing certain matching requirements, although this comes at the expense of lower agreement rates. It is hard to compare performance of different coding systems because the criteria for acceptable levels depend on the particular circumstances of the survey. In general, however, rates are considered acceptable if they are comparable to or better than the results of manual coding, and the amortized cost of developing and maintaining the systems is lower than personnel costs for the same task.

One new technology that has performed well in pilot, coding studies is neural nets (*e.g.* Bechtel & Abrahamson, 1991; Rummelhart & McClelland, 1986). Neural net technology is a type of classification software that “learns” the associations between a set of inputs and a set of outputs. In survey coding tasks, an input would be the words in an open-ended response and the associated output would be a code. Each association is represented as a pattern of activation in a network, where every node is either an input, output or mediating (“hidden”) unit. A node becomes active in response to external input or input from another node. When a node becomes active, it affects the activation level of all other nodes in the network; eventually these interactions subside and the entire network becomes stable. The resulting pattern is “remembered” by the network as the association between the input and output.

The network is “trained” on a set of many example cases where both input and output are known (*i.e.* the responses have been manually coded by experienced personnel). A learning algorithm repeatedly adjusts the weights that governs the influence of one node on another until the net produces an output pattern close to the known output. Once the network has been trained, it can be used for data processing, which, in the case of open-ended survey responses, means it can automatically code cases that have not previously been encountered. A system based on neural net technology (Raud & Fallig, 1993) was tested with Census Bureau industry and occupation data and produced about the same overall coding rate as AIOCS (48%) though its agreement rate was slightly lower (87% as opposed to about 90% for AIOCS) (Gillman & Appel, 1994).

Another technology that has performed well for simple coding tasks is Memory-based Reasoning (MBR) (Stanfil & Waltz, 1989). MBR computes the similarity of each unclassified case to every case in a database of classified cases. Similarity between any two cases is usually implemented as a measure of feature overlap -- usually some variant of the “nearest neighbor” metric -- where each case consists of multiple features. In a survey coding application, the features might include the content of open-ended responses, demographic variables like age and gender, as well as combinations of all those features. Although the similarity metric itself is computationally simple, the number of such comparisons can become astronomical for large databases. Because of this, MBR is intended to run on parallel computers where each case is a separate (physical) processing unit, making the number of comparisons far less relevant to performance time than it is on serial hardware.

MBR technology was tested on Census Bureau industry and occupation data (Creecy, Masand, Brij, Smith & Waltz, 1992). The MBR approach was more productive than AIOCS (61% coding rate with an overall agreement rate of about 88%) and was constructed in a fraction of the person months required to build AIOCS. However, the associated costs have led to some skepticism about its practicality (Gillman & Appel, 1994). MBR, like neural net techniques, requires a training set for any new domain; the clerical expense involved in certifying that the codes are accurate can be substantial. In addition, MBR must run on a massively parallel computer; one was borrowed for the test by Creecy, *et. al.*, (1992), but currently, they cost over a million dollars.

3. CONCEPTS BEHIND EXPERT SYSTEMS

The automation of simple coding tasks has been well served by the technological approaches described in the previous chapter. For the most part, the applications described above take advantage of the relationship between the words used by respondents and the way those words are usually coded. There is another category of survey classification task which relies on larger amounts of information, some of which may not be explicitly stated by respondents. In light of this extra complexity, it is generally not possible to create a dictionary of mappings between responses and codes. In these more complex cases, the classification specialist may infer the unstated information based on extensive knowledge of the subject area. Often the process is driven by a working hypothesis about the likely classification; the specialist decides which data to investigate based on their relevance to the hypothesis. This approach, which has many of the hallmarks of expert problem solving (*e.g.* Glaser & Chi, 1988), mirrors the approach that has been embodied in expert system techniques (*e.g.* Kahn, 1988), suggesting that there is a good fit between the task and the technology.

An expert system is a piece of software that reasons about a problem much as a human expert would. For example, several classic expert systems were developed to diagnose medical problems based on the knowledge and diagnostic strategies of physicians (*e.g.* Buchanan & Shortliffe, 1984). Expert systems are the oldest and most stable technology to emerge from research in Artificial Intelligence (AI). In one survey of AI practitioners (Hayes-Roth & Jacobstein, 1994) expert systems were deployed in 27 broad industrial categories, with about 30% of the applications in Finance and Manufacturing. There are several well known success stories in which expert systems have increased productivity, improved quality and timeliness of products, and saved money well beyond their development costs (see, for example, Leonard-Barton, 1987).

The general kinds of problems that have been addressed by this technology include diagnosis, classification, scheduling, configuration, monitoring and design (Kelly, 1991). While several expert systems have been developed for various statistical applications (see Gale, Hand & Kelly, 1993, for a review), the technology has seen relatively little use in survey research. The one application of which we are aware, other than our own, is in the area of questionnaire design (Halfpenny, Parthemore, Taylor & Wilson, 1992; Taylor, Parthemore, Wilson & Halfpenny, 1992).

What distinguishes these problems from those to which more conventional techniques have been applied is that (1) there may not be an optimal solution -- or at least it may be hard to evaluate optimality -- but there may be many acceptable solutions, and (2) solutions require intimate familiarity with the problem domain rather than abstract formulae. This situation-specific knowledge is often referred to as heuristics.

3.1 Heuristic Knowledge

Heuristics are rules of thumb that lead to acceptable solutions, although the exact rationale for their success may be unknown to the problem solver. They just work. They are not guaranteed to produce the right answer but usually do, and do so in relatively few steps. For example, a seasoned taxi driver might know that it is wise to avoid a particular tunnel at certain hours of the day and certainly knows this without having constructed a quantitative model of traffic flow.

Heuristics are often compared to algorithms, which are precisely specified computational recipes, guaranteed to produce the expected outcome. If one bakes a cake by following a recipe to the letter, one is guaranteed to produce the expected cake. Algorithms are derived from “first principles,” that is, from what is known about the fundamental properties of the problem. Heuristics, in contrast, are based on experience -- they have been shown to work on many previous occasions, though their success cannot necessarily be explained on theoretical grounds. Algorithms tend to be general purpose. They are tailored to a particular situation when specific values are passed to them. Heuristics tend to be narrower in scope than algorithms, defined in terms of particular conditions.

Consider the traffic example above. A heuristic might consist of a simple set of rules such as “If you are in a hurry to reach the airport and the current time is between three and six o’clock PM, do not use the tunnel to reach the expressway. Instead, cross the river on a bridge and take Carson Street to the West End where there is an entrance to the expressway.” To reach the same conclusion algorithmically, one would need to consider a vast number of parameters (typical traffic volume, tunnel capacity, likelihood of breakdowns, presence of road construction, *etc.*) and determine how these factors interact with one another in general. The algorithm could then be applied to specific values, such as times of the day, in order to produce a recommendation on when to use the tunnel. In this problem, familiarity with particular city streets leads to a far simpler solution than would an algorithm.

The time to solve certain problems seems to grow exponentially as a function of their size. These are known as NP-complete problems (see, for example, Poundstone, 1988, Chapter 9). One goal of AI, and in particular, expert systems, has been to reduce the speed of this growth, even if it is exponential, and the key has proven to be informal knowledge about the domain, that is, heuristics (Nilson, 1980). Again, the solutions may not be optimal but they are serviceable. Many of the methods developed for solving problems of this sort have proven useful for other applications, even if the problems are not NP-complete. Heuristic methods simplify the problem by focusing on just those variables that, experience dictates, are relevant to the solution.

Because heuristics are informal and learned through experience, they are often not documented. One benefit of building expert systems is that the process creates a permanent record of this knowledge -- even if the software is never used. Of course, running the expert system assures that the knowledge is applied systematically and reliably.

3.2 Components of an Expert System

The knowledge in expert systems is generally represented as IF-THEN rules: IF certain conditions are present, THEN take a particular action or reach a particular conclusion. In this approach, small packets of knowledge are invoked when particular circumstances of the problem occur. The collection of rules required to solve problems in the particular content area is known as the knowledge base or rule base, hence the technique for developing expert systems is called “rule-based” programming.

Because rules are specific to a particular problem area, a knowledge base for automotive trouble shooting would share virtually nothing with a knowledge base for occupational classification. However, expert systems in these two different areas could have in common the reasoning mechanism that determines which rules to apply and the particular sequence in which to apply them. This mechanism is known as the inference engine. When the inference engine applies or “fires” a rule, that changes the current state of the problem -- hopefully to a state closer to a solution. The inference engine then searches for other rules in the knowledge base that might be applied under the new circumstances and continues this cycle until the problem is solved, or until it is determined that the system cannot solve the problem.

Inference engines come in two varieties, backward chaining and forward chaining. Different types of problems are better suited for one or the other of these. A backward chaining inference engine begins with a hypothesis about the solution to the problem and the reasoning process seeks evidence to support the hypothesis. They are appropriate for problems where there is a small set of possible hypotheses, such as diagnosis or classification tasks where the categories correspond to hypotheses. In contrast, forward chaining inference engines construct solutions in the absence of hypotheses. Strictly on the basis of the data describing the problem state, rule actions manipulate and create data that eventually comprise a solution. Appropriate problems for forward chaining are scheduling or configuration tasks where there are many possible solutions, none of which is known in advance.

In addition to the knowledge base and inference engine, an expert system requires data on which to operate and some mechanism to store the data elements. Data describe the problem state and determine which rules are eligible to be fired. This is because the condition elements of a rule (the IF part) are matched by the inference engine to the data elements present at that time. As rules are applied they create more data, changing the problem state, enabling other rules to become eligible. Data become available by being read from a file, entered by a user or created through the actions of rules. They are stored in a temporary memory, referred to by such terms as “working memory” or “session context.” In most contemporary expert system tools, the data are object-oriented (*cf.* Cox & Novobilski, 1991).

One thing an expert system does not have is an explicit control structure. The inference engine determines which rules to apply on the basis of the circumstances at any moment. This contrasts with conventional approaches, where the programmer uses such techniques as looping, subroutines, and “Go To” statements to explicitly control the program’s flow. The lack of an explicit control structure is one reason why expert systems are well suited for use in unstructured tasks, such as those for which we’ve developed our prototypes. This idea is illustrated in section 4.2.

In principle, one could develop an expert system from scratch, building the tools to encode rules and data as well as building an inference engine. However, most expert system programmers do their work in development environments or “shells” that include rule editors, object editors, an inference engine (or sometimes both forward and backward engines) and other tools. These are specialized for expert system development and help produce an application that performs efficiently. They are derived from significant research in computer science (see, for example, Forgy, 1982, or Cooper & Wogrin, 1988, Chapter 10, for a discussion of one procedure to optimize an inference engine). Organizations that are considering this technology are advised to use an expert system shell instead of a traditional computer language to avoid reinventing the wheel. A relatively recent review of available development products can be found in Stylianou, Madey, & Smith (1992).

3.3 Knowledge Engineering

One thing that expert systems shells are not equipped with is the knowledge from any particular content area. That must be extracted from the mind of the expert(s). This extraction process is usually carried out by a knowledge engineer who interviews the experts about their practices in particular circumstances or observes experts at work and encodes those practices as rules. Because experts’ procedures have become second nature over the years, it can be difficult for them to articulate what they do. The knowledge engineer must help the expert to specify his or her knowledge at a sufficiently detailed level so that it can be modeled as rules that successfully solve problems. Knowledge engineering is an iterative process of modeling the expert's knowledge and refining the model based on the expert's feedback. It is considered the primary bottleneck in developing expert systems (Hoffman, 1987).

4. CASE STUDIES

When it is possible, most surveys use standardized procedures to collect and process data. The best example of this approach is the structured questionnaire. However, there remain essential activities -- particularly classification tasks -- for which it is difficult to develop a scripted procedure. In such situations, professional data collectors and processors have license to determine what data elements to investigate, how to word any questions they ask, what order to ask them in, how to keep their inquiry to a minimum, *etc*. as long as they can produce reliable results. These people have amassed expertise of a particular problem area from years of work and study, and their knowledge has a heuristic character. Much of their knowledge is undocumented and is communicated orally from senior to junior staff.

We have chosen two classification activities at the Bureau of Labor Statistics, both of which rely on extensive heuristic knowledge, to explore the applicability of expert system techniques. The first of these is a data review activity in which an analyst evaluates decisions made by data collectors about the comparability of products. It is a classification task in the sense that the reviewer assigns each such decision to one of a small number of categories to indicate the adequacy of the decisions, and therefore, the usefulness of the data. Both the task and the expert system, which is referred to as COMPASS, are described in section 4.1. The second task for which we have explored expert systems is a type of occupational classification done primarily during the interview. Unlike the occupational classification tasks described in section two, for which a simple dictionary look-up is possible, this task requires many subtle distinctions and a flexible dialogue between the interviewer and respondent. The task is described in section 4.2 along with the expert system which is known as MatchMaker.

4.1 COMPASS

The Consumer Price Index (CPI) is conducted monthly by BLS to measure the average change in price for a fixed set of goods and services. The index is updated each month by comparing the price of commodities in the current month to the price previously collected. Because of marketplace changes, a product priced in an earlier month may no longer be available. In order to maintain a continuous sample under these conditions, a comparable product may sometimes be substituted for the unavailable product. Commodity Analysts (CAs) review each product substitution to determine its comparability to the original product. If the CA deems the substitute and original products to be comparable then their prices can be compared in computing the CPI.

In addition to price data, the Field Representatives collect data about numerous specifications of a product -- for example “screen size” and “sound quality” for televisions. The specifications are enumerated on a check list for each major product area, and a particular collection of specification values describes a particular product. The CAs rely on specification values in making comparability judgments. The CA must compare all of the relevant specification values for the original and substitute products and evaluate the pattern of specification data from the perspective of comparability. By judging the original and substitute products to be comparable, the CA has essentially assigned them to the same narrow, product category, for example, the category of 27 inch, color televisions with “surround sound” and “picture-in-picture” capability; in this sense, the CA is performing a classification task.

CAs’ knowledge is deep and specialized. Their knowledge in one product area is detailed and extensive, but unrelated to knowledge of other product areas. For example, a CA who specializes in apparel might consider linen and cotton to be equivalent fabrics when comparing certain garments because their effect on quality is equivalent, yet distinguish between linen and nylon because this difference affects quality. Such knowledge is meaningful in only this particular apparel area and would not be useful in judging comparability of products in other areas, for example insurance or fruit. However, there is

general, analysts tend to look for evidence that products are not comparable, and only after failing to find such evidence do they accept the substitution as comparable. Such a process is efficient because each checklist includes some specifications or groups of specifications for which a mismatch allows the analyst to immediately reject the substitution. In contrast, a match between specification values does not allow any conclusions until other specifications -- possibly all -- are compared.

At certain points in the monthly production cycle, comparability decisions can constitute a substantial portion of the CA’s duties. Of the 78,000 products priced each month for the CPI, about 2700 are substitutes and must be reviewed by a staff of about 25 CAs. In addition to being labor intensive, reviewing substitutions is prone to visual search errors as analysts locate information in the product descriptions, comprehension errors as they extract the content of those descriptions, and consistency lapses if they should irregularly apply a piece of knowledge. Rare as these performance complications may be, they are a potential source of error for the CPI, one that might be reduced through computer support.

The Expert System

COMPASS (Comparability Assistant) is an experimental, expert system designed to check CAs’ comparability judgments about product substitutions (Conrad, Kamalich, Longacre & Barry, 1992)[2]. After a CA has reviewed a month of substitution data, COMPASS is

[2] This expert system should not be confused with another expert system, also called COMPASS, described by Prerau, Gunderson, Reinke & Goyal (1985).

used to make comparability decisions about those same substitutions. The value in the procedure lies in the discrepancies that might arise between CA decisions and those made by COMPASS: Because such differences are potentially due to errors by the CA, resolving them should improve data quality. In principle, a tool such as COMPASS could be used in data collection, or, for the routine cases, in place of the CA. The current architecture and knowledge engineering procedures would not change if COMPASS' functionality were extended in these directions. COMPASS has been implemented as a quality check because this allows us to explore the feasibility of expert system technology in substitution review without impacting concurrent production activities.

To date, the COMPASS experiment has been conducted for 13 of the roughly 350 product categories surveyed for the CPI. The thought is that knowledge bases will eventually be developed for all product areas following the procedure used thus far. In order to evaluate the generality of the procedure, the test categories were sampled from throughout the universe of products. Four categories were chosen from the food area (Fruit, Cereal, Fish, and Dinner), three from the apparel family (Women's Swimsuits, Women's Pants and Shorts, and Women's Nightwear), three from services (Theater Admission, Automobile Finance Charges, and Hospital and Patient Services), and three that were unrelated to each other (TV, Automotive Body Work, and Ski Equipment).

Knowledge Engineering. Expert systems are sometimes criticized because after being developed for test cases, they are not easily scaled up for production use (*e.g.* Creecy, Masaland, Smith & Waltz, 1992). Large knowledge bases can be unwieldy and hard to maintain (Bachant & Soloway, 1989). Scalability should not be a problem for COMPASS because the knowledge engineering process has been distributed: A separate, relatively small knowledge base is constructed for each product category. However, the task of building 350 individual knowledge bases is still likely to be enormous, even if spread among many CAs. To address this, the COMPASS methodology places the knowledge engineering task into the hands of the experts themselves -- the CAs. The basic idea is for each CA to develop a knowledge base for the product area(s) in which he or she is expert. By giving the CAs direct responsibility for encoding their own expertise, COMPASS' decisions receive extra credibility: If COMPASS and a CA reach different decisions because COMPASS applied a rule and the CA did not, then either the CA invoked "special case" knowledge to which COMPASS was not privy, or the CA forgot to apply the rule; in either case, it cannot be argued that COMPASS was provided with the wrong rule because it was the CA who entered the rule in the first place.

One reason that professional knowledge engineers are usually involved in this type of activity is because they have the skills to elicit knowledge from experts -- even when experts are unaware of that knowledge (Hoffman, 1987; Kelly, 1991). The substitution review task makes knowledge engineering by the CAs possible because CAs can often reach a decision on the basis of a single inference (rule), and rarely more than two. Such inferences are easier for CAs to articulate than are the more complex inference sequences typical of expert problem solving. The brevity of these inferences seems to be related to the use of checklists. They make explicit the criteria that analysts believe determine a product's value, and reduce the need to infer one criterion from another. Consider another apparel example. The checklist for Women’s Pants and Shorts includes a specification for fiber content and a specification for washability. The latter might be inferred from the former -- most silk, for example, cannot be machine washed -- but because data are collected specifically for these criteria, there is no need for the CA to make such inferences.

Even if experts can articulate their own knowledge, they are usually unable to write a rule that a computer could process. This is relatively easy in COMPASS for several reasons. First, the expert system shell in which it is implemented uses simple, English-like syntax in its rule language. This enables CAs, who are typically not programmers, to easily understand and encode such rules. Second, CAs are provided with a set of rule templates which present the structure of common comparability rules but are written without reference to particular product areas or specifications. In many cases, by adding only a few pieces of information to a rule template, the CA can construct a rule for his or her particular product area from a template. Finally, CAs are given the basic computer code to run the system, but without any domain-specific rules. This “skeletal knowledge base” is a starting point to which they can add any rules they create. It enables CAs to concentrate on entering their knowledge rather than thinking about how the program will run.

In addition to software, COMPASS entails a methodology for developing a knowledge base. Each CA first writes in English the criteria they believe to be relevant in determining product comparability. In addition they indicate the criteria required for a product description to be considered complete and internally consistent. Regardless of how successful the remainder of the exercise, this step documents a body of knowledge that was previously unwritten. Next, the CA encodes these criteria as rules and adds them to the skeletal knowledge base. An example rule is presented in Figure 2.   It is taken from the knowledge base for Televisions (“E31011” is a code for TVs, well known to the CAs in this area) and the A spec (“new_a” is the value for the substitute product and “old_a” is the value for the original) describes a product’s color capability, where a value of 1 indicates black and white, a value of 2 indicates color. The rule concludes the products are not comparable if the two products have different values for this spec (“<>“ indicates “does not equal”), that is, if one TV set is black and white and the other is color.

```
RULE for comparability A spec
IF new_a OF dB3 E31011R 1 <> old_a OF dB3 E31011R 1
THEN comparability OF quote IS no
```

Figure 1. Example Rule from Television Knowledge Base.

This initial knowledge base is executed on 3 months of substitution data consisting of the keyed specification data collected in the field as well as the CA’s own comparability decisions (or possibly decisions by another CA). Any discrepancies between CA judgments and those of COMPASS are then resolved by either (1) revising the knowledge base, (2) accepting the discrepancy as the result of knowledge too specialized to warrant a specific rule, or (3) accepting it as the result of knowledge available to a CA but unavailable to COMPASS, for example, hand written messages by the field representative. If the CA determines that a rule needs revising, then the effort is usually confined to that rule; this modularity is direct result of the way knowledge is represented in expert systems. After the CA is satisfied with the performance of the knowledge base, the procedure is repeated three times on new data sets, each consisting of one month's substitutions. Some data on the effectiveness of this approach are presented in the next section.

One challenge in developing COMPASS knowledge bases is data in the form of open-ended or "free" text. Most of the specifications take fixed values (one out of a small set of acceptable numbers) but some are assigned unrestricted values. For example, the specification for the brand names of certain apparel items takes the brand name itself as its value. When COMPASS encounters such free text, it compares it to a dictionary of previously encountered free text entries which the CA has judged to be equivalent and associated with a single classification. For example, the CA might have created a list of all brands judged to be "high quality," consisting, in the apparel example, of several designer brands, including spelling variations. If the current free text entry is found in the list, COMPASS can reason as if brand quality were a fixed value specification. If the entry is not already listed, the CA must classify the free text entry and add it to the appropriate list. Our experience has been that in certain product areas, such as apparel, one should expect a steady, but small, percent of new brand names each month, implying a corresponding amount of knowledge base maintenance each month.

Architecture. The skeletal knowledge base includes control knowledge that orders the rule application in a way that resembles CA decision making. CAs tend to gather evidence to support one of a small set of hypotheses, for example, "not comparable," rather than constructing hypotheses from the patterns of specification values. This is mirrored in the system by its primary use of a backward chaining (*i.e.* hypothesis driven) inference engine. In addition, CAs search for evidence of non-comparabilities prior to evidence for comparability. COMPASS' decision making is structured so that it attempts to eliminate comparability in seven different ways before concluding two products are comparable. This is shown in Table 1.

```
1. completeness OF quote IS incomplete
 2. completeness OF quote IS complete
   2.1 consistency OF quote IS inconsistent
   2.2 consistency OF quote IS consistent
     2.2.1 comparability OF quote IS no
     2.2.2 comparability OF quote IS refer
     2.2.3 comparability OF quote IS adjust
     2.2.4 comparability OF quote IS yes
```

Figure 2. Goal Hierarchy (Level5 Object Agenda)

COMPASS initially evaluates the completeness of the product descriptions, terminating if either is incomplete. It then checks their consistency, terminating if either contains logical anomalies. Finally it determines comparability. There are four goals associated with comparability. COMPASS first tries to prove that the substitution is *not comparable*. Barring that outcome, it seeks evidence that the substitution requires review by the analyst and if so, it concludes *refer*. This resembles the flagging of an item for manual coding in traditional automated coding systems and gives it the character of computer-assisted automated coding (see Section 4). If COMPASS cannot prove that the quote should be referred, it then tries to establish that the substitution can be compared if it is *quality adjusted*, that is, if its price is statistically adjusted to account for quality differences (Armknecht & Weyback, 1989). It does not execute the adjustment but signals it may be applicable. If there is no evidence for that, COMPASS concludes that the products are comparable.

4.2 MatchMaker

The second task that we have modeled with expert system techniques is a type of occupational classification known as *job matching*. The Occupational Compensation Survey Program (OCSP) publishes national and regional rates of pay for each of about 80 job categories. The job categories are those found in the U.S. Federal Government and are used in part to set the wages of Federal employees. (They are also considered by many private sector employers in establishing their rates of pay.) The data are collected by means of face to face, establishment interviews, in which a Field Economist (FE) collects information from a personnel or payroll representative at the establishment. Based on information from the respondent, the FE attempts to match establishment positions to government job definitions. These definitions range in length from several paragraphs to several pages, and describe the job at up to eight levels of sophistication, for example Engineer I, Engineer II, *etc.*. Once a job has been matched -- and many are not matched -- the FE can record the wages paid by the establishment to employees holding that job. The FE therefore, is both the interviewer and the “coder.”

The FE must account for each position in the establishment, by either matching or explaining why it cannot be matched. The FE must document each decision and the criteria used in sufficient detail so that a supervisor can review their decision making. Job matching requires the FE to be (1) intimately familiar with the government descriptions, (2) skilled at interpreting the respondents’ descriptions of establishment jobs, and (3) able to judge the similarity of one to the other. This presupposes that the FE is able to elicit informative statements from the respondent. To do this, the FE must formulate questions during the course of the interview, tailoring them to the particular respondent in the current interview situation.

FEs usually concentrate their questions on the duties, skills and credentials required by the position but there is no scripted questionnaire for the interview. This is because organizations vary so much that a single instrument would likely be inapplicable in many establishments and because the respondents, whose participation is voluntary, usually lack the time to answer any but the most pertinent questions. To reduce the length of the interview, the FE must ask primarily about those jobs that will plausibly be found in the establishment -- a kind of knowledge that is not explicitly documented -- and ask only those questions necessary to confirm a match. This requires more flexibility than can be accommodated through a structured instrument.

The job matching process is vulnerable to inconsistency. FEs develop different approaches from one another, potentially leading to different conclusions. In addition, being human, individual FEs may be inconsistent from one interview to the next. The risk of incomplete information is also inherent in the job matching procedure. FEs may forget to collect all of the information on which to base their decisions because of time pressure and the unscripted nature of the interview. Both of these potential sources of error would be reduced through computer support for the job matching process. The problem is that current CAPI tools do not provide the kind of flexibility to model an interview like this which is essentially improvised. Traditional CAPI tools primarily implement structured paper and pencil instruments, where the skip pattern among questions is specified in advance and is invariant. In principle an expert system could enable FEs to tailor the sequence of their questions to the particular interview situation while assuring they collect all information about relevant job criteria using essentially the same approach.

The Expert System

MatchMaker is a prototype expert system to support job matching. It questions the FE about characteristics of an establishment job that must be present in order to match the job. The FE responds to MatchMaker on the basis of information from the respondent, so the FE must craft an appropriate question to the respondent in order to reply to MatchMaker. By using MatchMaker, any two FEs will ask respondents about the same features of a job. MatchMaker has been designed for use on a notebook computer in the interview, but even if it is not ultimately used in that situation, it can be used (1) to prepare for an interview, (2) to create a data report after an interview, and (3) as an interviewer training tool. Having encoded the relevant job knowledge, we could deploy the same knowledge base in these various situations.

In our research efforts to date, we have encoded knowledge for six of about 80 OCSP job definitions: Accountant, Accounting Clerk, Computer Programmer, Engineer, Engineering Technician and Personnel Specialist. These were chosen to span the range of job matching complexity. Accountant is considered one of the more straightforward jobs to match; Personnel Specialist is viewed as one of the most challenging and is associated with a high rate of supervisory intervention. In addition to knowledge for individual jobs, we have encoded knowledge about the jobs that are likely to be found in particular industrial sectors in establishments of various sizes. This is intended to help FEs generate hypotheses about likely jobs and confirm them by questioning the respondent.

Knowledge Engineering. The expertise in MatchMaker was encoded by analyzing publications, by interviewing and observing FEs, and by searching pertinent data bases. By analyzing how the FEs do their work (a task analysis), we recognized that before they engage in job matching *per se*, they learn as much as they can about the current establishment, for example its industrial sector, its employment size, and its organizational structure. Based on this, FEs adjust their expectations for likely matches. A facility was developed to rank order job match hypotheses for different combinations of these factors based on the experience of experts and historical data. The knowledge to support job matching itself was derived largely from the OCSP definitions (published in a manual for FEs) as well as supplementary documents (collections of questions and answers) and technical memoranda (responses to questions about specific field situations).

The task analysis revealed that, in practice, most matches hinge on the job's duties, its required skills and education, the supervisory relations (who reports to whom) and the absence of exclusionary evidence[3]. As a result, the knowledge engineering staff[4] formed rules primarily about this information. Once the knowledge was implemented as functioning software, FEs reviewed it and the knowledge base was refined. While there is a single knowledge base for all of the jobs in MatchMaker, in practice the knowledge for each job is modular. As in COMPASS, this is intended to simplify the development and maintenance of the knowledge as the full set of job definitions are encoded.

Architecture. Much like the FEs, MatchMaker seeks evidence to confirm each job match hypothesis. As a result, it is built around a backward chaining inference engine. Each hypothesis is supplied by the FE, possibly by selecting it from the rank ordered list mentioned above. The inference engine then queries the FE for the conditions to apply each rule. It first seeks to confirm the overall match, for example, an Accountant, and then it pinpoints the level, for example Accountant IV. Under some conditions, when a match is not confirmed, MatchMaker can propose alternative jobs. To date this has been implemented when two jobs differ only on one criterion. For example, educational requirements (degree versus no degree) can be the primary distinction between accountants and accounting clerks. While MatchMaker may suggest an alternative, the FE must decide whether or not it should be investigated.

One advantage to the expert system approach in the case of job matching is that the inference engine determines the branching sequence (skip pattern in a conventional interview). This is helpful because the numerous job attributes under consideration combine into a complex branching structure (see Figure 3) that would be hard to encode

---

[3] Each job definition includes a list of criteria that specifically rule out a match even if other criteria would confirm the match -- "exclusions." For example, a company position that otherwise matches the Computer Programmer definition but requires a bachelor's degree in a scientific field other than computer science, is explicitly excluded.

[4] The development of MatchMaker was a collaboration between BLS and colleagues at the American Institutes for Research (AIR). AIR staff were involved in most aspects of the design process and were entirely responsible for implementing the executable code.

explicitly for the full set of jobs, and that would make an instrument difficult to maintain. Rather than focusing on the precise sequence in which rules should be applied, the knowledge engineer can focus on the massive quantity of content knowledge that needs to be captured in rules. The inference engine determines which pieces of information to query based on what information has (and has not) been collected at any point.

To illustrate this, consider the logic depicted in Figure 3. If the duties (upper left) match those in the definition of Personnel Specialist, but none of the education requirements are satisfied (middle left), the inference engine applies a rule that redirects control to an alternative, less skilled job -- Personnel Assistant (far right). There is nothing in the rule that indicates its sequence, simply the knowledge that an alternative job shares some of the duties of Personnel Specialist and requires less education.

Insert Figure 3 about here

The FE can initiate several computational tasks as needed. In particular, MatchMaker includes a help feature that provides increasingly detailed (hypertext) descriptions of the jobs and a notepad feature to document the circumstances surrounding any decisions. In addition, each activity of the inference engine is written to this notepad, providing an annotated reasoning history that could be examined by a supervisor.

By building flexibility into the design of MatchMaker and providing tools to support various component tasks, we intended to design a software system that would fit into the FEs work. This seems to have been reasonably successful. In a questionnaire about usability, 22 FEs were asked “How well does the system model the steps involved in job matching?”. On average, they rated it 6.1 out of 9 where 1 was “not at all” and 9 was

5. Evaluation and Resource Issues

This section addresses two issues about the value of expert systems in survey organizations: “Do they work?” and “What resources are required?.” To address the first of these, we present performance data from a COMPASS knowledge base. This type evaluation compares the decisions of the software to those of CAs for the same data. The second issue is hard to address with specific monetary figures at this point because the research and design character of our work is so different from the production environment in which systems like ours will ultimately be deployed. Instead we focus on the types of human resources that are needed and potential impact on the expert system on the organization.

in a given cell represents the proportion of decisions (for that iteration) due to that combination of COMPASS and CA decisions. For example, the upper most left cell in the first iteration corresponds indicates that both COMPASS and the CA reached a “Yes” decision 71% of the time.

[5] Completeness and consistency decisions will not be discussed here though some data are provided in Conrad, *et. al.*(1993).

[6] Analyses for all knowledge bases are available from the author.

The entries in the main diagonal cells are agreement rates and the entries in off-diagonal cells are disagreement rates. A high degree of correspondence between COMPASS and the CA would appear as large numbers along the main diagonal and numbers close to zero in the off-diagonal cells. Iterations 1, 3 and 4 approach this pattern, but the row for analyst *No* judgments diverges, showing small numbers in all cells, with the smallest in the diagonal cell. For the second iteration, there are simply no observations in this row. Our ability to evaluate the system's accuracy for *No* judgments is compromised by their infrequent occurrence in this analysis,[7] although for those *No* decisions available, the number of discrepancies is relatively high.

Insert Table 1 about here

Because COMPASS is run as a quality check, not all discrepancies indicate inaccuracy. For example, the system might be applying the CA's rules consistently even when the CA has failed to do this. In such a case, the expert system would improve data quality by performing the classification task as the CA would perform it under ideal conditions. An analysis of the discrepancies in the first two iterations confirmed this interpretation. In the first iteration, 6 of the 17 discrepancies were due to some combination of analyst error, possible field messages to the analyst which were not available to the system, and peculiar price changes that were considered too rare to warrant special rules. All of these discrepancies involved NO decisions by the analyst, helping to explain the apparent inaccuracy of the system for those items. In the second data set, 15 out of 19 discrepancies were attributed to analyst error, and one to a field message that the system could not accommodate. All of these discrepancies concerned analyst Yes judgments.

The discrepancies signal that, by and large, COMPASS was doing what it was designed to do, namely checking CA decisions and noting inconsistencies. Based on knowledge entered by the CAs, COMPASS reached the expected conclusions, and when these diverged from the analysts' decisions, they were interpretable. If COMPASS is used in a production context, the CA can revise discrepant decisions of this type.

5.2 Resources and Organizational Issues

In order to develop expert systems for production use, a survey organization must be prepared to allocate certain human resources and to manage the consequences of deploying the software. In addition, the chances that the project will be successful are increased if there is an advocate or champion in the organization.

Human Resources

[7]About 30% of Analyst *No* decisions produced *Refer* decisions by COMPASS and so are not included here.

Most programmers in the data processing or MIS group in survey organizations are not skilled in rule-based programming or in eliciting knowledge from experts. Rule-based programming is considerably different from the conventional, “procedural” programming that these employees are used to and, in our experience, has been difficult to teach to them. Knowledge elicitation skills require background in cognitive science and even the humanities (Kelly, 1991). Because professional knowledge engineers are skilled in both, an organization embarking on such a project should seriously consider retaining such personnel. In the COMPASS project, we created a situation in which the CAs could do their own knowledge engineering, but this was only possible because project members understood the technology well enough to develop tools (rule templates, skeletal knowledge base) for the CAs to use that reduced the programming burden on them.

Of course, one or more experts are needed to supply the knowledge that will become the substance of the expert system. This can involve a substantial amount of the experts’ time, so their managers must make the time available by actually reducing other obligations. A certain amount of communication and coordination is also required to enable knowledge elicitation to occur. The expert and knowledge engineer probably work in different offices or groups, sometimes even in different regions of the country, so their collaboration will need to be supported by the appropriate managers. Where travel is required, additional funds will need to be made available.

Consequences

Introducing expert systems into the survey process produces certain social consequences. For example, in developing COMPASS and MatchMaker, we needed to reassure FEs and CAs that the knowledge elicitation process was not done to evaluate their knowledge or performance; presumably other experts will have similar concerns. Additionally, using the expert system is likley to change the nature of the expert’s work. One frequently articulated goal for expert systems is to handle the routine cases and refer the more subtle and complex cases to the expert. This is the approach adopted in COMPASS. It seems as though it should reduce boredom, reserving the expert for tasks that are stimulating. However, a possible danger in this approach is that it may increase the amount of stress on the expert by requiring only difficult judgments; this, in turn, can reduce morale (Hauser & Hebert, 1992).

Among the technical consequences of introducing expert systems is the need for system integration. The new software must be made to dovetail with the rest of the data processing system. For example, MatchMaker is intended to be used for data collection so it must write the data to the appropriate data base format and a data communication procedure must be developed. Similarly, if COMPASS is to be used in production, it will need to be invoked and supplied with data by other programs, possibly running on different computers. All of this requires coordination and high level planning.

Additionally, the knowledge in expert systems needs to be frequently updated. In the case of MatchMaker, job descriptions change and this needs to be reflected in the knowledge base. In COMPASS, free text items, like brand names, are in continual flux, and these changes need to be encoded in the knowledge base. Over the lifespan of an expert system, the most frequent and, therefore, costly activity may be maintaining the knowledge base (Bachant & Soloway, 1989).

Advocacy

The complexity of large organizations, such as government statistical agencies, can severely impede the development of an expert system. Numerous parties, all of whom work in different offices, may need to be involved in making decisions which can slow the process. In addition, reluctance to change current, familiar practices can lower the priority of the project for some of its participants, leading it to fall between the organizational cracks. Alternatively, it is possible for an expert system to be technically successful but, ultimately, unused (Hayes-Roth & Jacobstein, 1994). To overcome organizational inertia and to promote the use of the expert system, the project needs an advocate within the organization -- possibly the knowledge engineer (Kelly, 1991). Whoever the advocate is, this person must be able to communicate the vision of the project to the skeptics and to navigate around numerous organizational obstacles.

6. Conclusions

Based on our experience, expert systems are a promising technology for systematizing the classification of certain survey data. Rather than widespread use of stand alone expert systems we may see increased embedding of the technology in other software systems. The autocoding applications discussed in Section 2 have begun to move in this direction. When considered in this light, expert system technology has a particularly bright future in the statistical community because there are numerous data processing tasks currently executed by large software systems that are sometimes simplified by introducing executable, expert knowledge.

There is an additional organizational benefit from building expert systems. The documentation that arises from this activity can facilitate training and technology transfer. By creating a tutoring interface to an expert system’s knowledge base, an organization can make this knowledge available to its members -- even if the expert from whom it is derived is unavailable.

Finally, the early vision of applying AI to replace workers with smart software seems to have evolved into one in which employees are supported in their decision making by this software. That is the route we have taken with COMPASS and MatchMaker, and where the value of the approach seems to be greatest. The goal is to improve the quality of the decision making -- not necessarily the quantity of decisions made for a fixed cost. Expert systems are highly specialized in their knowledge and generally incapable of the common sense reasoning at which people are so good. However, they do not experience memory lapses and do not get tired, ill or hungry. By simultaneously exploiting the strengths of the software and the strengths of the users, the quality of these important statistics can only be improved.

References

Andersson, R. & Lyberg, L. (1983). Automated coding at Statistics Sweden. *Proceedings of the Joint Statistical Meetings, Section on Survey Methods,* 41 - 50. Alexandria, VA: American Statistical Association.

Appel, M. V & Hellerman, E. (1983). Census bureau experience with automated industry and occupational coding. *Proceedings of the Joint Statistical Meetings: Section on Survey Research,* 32 - 40. Alexandria, VA: American Statistical Association..

Appel, M. V. & Scopp, T. S. (1987). Automated industry and occupational coding. Paper presented at Development of Statistical Tools Seminar on Development of Statistical Expert Systems, Luxembourg.

Armknecht, P. & Weyback, D. (1989). Adjustments for quality change in the U.S. Consumer Price Index. *Journal of Official Statistics, 5,* 107 - 123.

Bachant, J. & Soloway, E. (1989). The engineering of XCON. *Communications of the ACM*, **32**, 311 - 317.

van Bastelaer, A. M. L., Hofman, L. M. P. B. & Jonker, K. J. (1987). Computer-assisted coding of occupation. *Automation in Survey Processing.* The Netherlands Central Bureau of Statistics, 77 - 86.

Bechtel, W. & Abrahamson, A. (1991). *Connectionism and the Mind: An Introduction to Parallel Processing in Networks*. Cambridge, MA: Blackwell Publishers.

Bethke, A. D. & Pratt, D. J. (1989). Automatic coding methods and practices. Unpublished manuscript. Research Triangle Institute. Research Triangle Park, NC.

Bobbitt, L. G. & Carroll, C. D. (1993). Coding major field of study. *Proceedings of the Joint Statistical Meetings, Section on Survey Research*, 177 - 182. Alexandria, VA: American Statistical Association.

Buchanan, B. G. & Shortliffe, E. S. (1984). *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project.* Reading, MA: Addison-Wesley.

Cantor, D. & Esposito, J. L. (1992). Evaluating interviewer style for collecting industry and occupation information. *Proceedings of the Joint Statistical Meetings: Section on Survey Research*, 661 - 665. Alexandria, VA: American Statistical Association..

Ciok, R. (1993) The results of automated coding in the 1991 Canadian census of population. *Proceedings of the U.S. Bureau of the Census 1993 Annual Research Conference*, 747 - 765. Washington, DC: U.S. Department of Commerce.

Conrad, F., Kamalich, R., Longacre, J. & Barry, D. (1993). An expert system for reviewing commodity substitutions in the Consumer Price Index. *Proceedings of the Ninth Conference on Artificial Intelligence for Applications*, 299 - 305. New York: IEEE Computer Society.

Cooper, T. A. and Wogrin, N (1988). *Rule-based Programming with OPS5*. San Mateo, CA: Morgan-Kaufmann.

Cox, B. J. & Novobilski, A. J. (1991). *Object-Oriented Programming: An Evolutionary Approach, Second Edition*. Reading Massachusetts: Addison-Wesley.

Creecy, R. H., Masand, B. M., Smith, S. and Waltz, D. (1992). Trading MIPS and Memory for Knowledge Engineering. *Communications of the ACM*, **35**, 48 - 63.

Forgy, C. L. (1982) RETE: A fast algorithm for the many pattern/many object pattern matching problem. *Artificial Intelligence*, **19**, 17 - 37.

Gale, W. A., Hand, D. J. & Kelly, A. E. (1993). Statistical applications of artificial intelligence. In Rao, R. (Ed.) *Handbook of Statistics*, **9**. Amsterdam: Elsevier Science Publishers.

Gillman, D. & Appel, M (1994). Automated coding research at the Census Bureau. *Statistical Research Report Series No. RR94/04*.

Glaser, R. & Chi, M. T. H. (1988). Overview. In M. T. H. Chi, Glaser, R. and M. J. Farr (Eds.). *The Nature of Expertise*, xv - xxviii. Hillsdale, NJ.: Lawrence Erlbaum, Inc.

Halfpenny, P, Parthemore, J. Taylor, J. & Wilson, I. (1992). A knowledge based system to provide intelligent support for writing questionnaires, in A. Westlake, (Ed.) *Survey and Statistical Computing*. Amsterdam: Elsevier Science Publishers.

Hauser, R. D. & Hebert, F. J. (1992, Winter). Managerial issues in expert system implementation. *SAM Advanced Management Journal*, 10 - 15.

Hayes-Roth, F. & Jacobstein, N. (1994). The state of knowledge-based systems. *Communications of the ACM*, **37,** 27 - 39.

Hoffman, R. R. (1987). The problem of extracting the knowledge of experts from the perspective of experimental psychology. *The AI Magazine*, **8,** 53 - 64.

Kahn, G. (1988). MORE: From observing knowledge engineers to automating knowledge acquisition. In S. Marcus (Ed.) *Automating Knowledge Acquisition for Expert Systems*, 7 - 36. Boston: Kluwer Academic Publishers.

Kelly, R. V. (1991). *Practical Knowledge Engineering.* Bedford, MA: Digital Press.

Leonard-Barton, D. (1987). The case for integrative innovation: An expert system at Digital. *Sloan Management Review,* 7 - 19.

Lyberg, L. & Dean, P. (1990). International review of approaches to automated coding. Paper prepared for the Conference on Advanced Computing for the Social Sciences, Williamsburg, VA.

Nilson, Nils (1980). *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga Publishing Company.

O'Reagan, R. T. (1972). Computer assigned codes from verbal responses. *Communications of the ACM*, **15,** 455 - 459.

Poundstone, W. (1988). *Labyrinths of Reason: Paradox, Puzzles and the Frailty of Knowledge.* New York: Doubleday.

Pratt, D. J. & Burkheimer, G. J. (1994). On-line automatic coding in computer-assisted data collection. Manuscript submitted for publication.

Pratt, D. J. & Mays, J W. (1989). Automatic coding of transcript data for a survey of recent college graduates. *Proceedings of the Joint Statistical Meetings, Section on Survey Research*, 796 - 801. Alexandria, VA: American Statistical Association.

Prerau, D. S., Gunderson, A. S., Reinke, R. E. & Goyal, S. K. (1985). The COMPASS expert system: Verification, technology transfer, and expansion. *Proceedings of the Second Conference on Artificial Intelligence Applications*, 597 - 602. New York: IEEE Computer Society.

Raud, R. & Fallig, M. A. (1993, May). Automating the coding process with neural networks. *Quirk's Marketing Research Review*, 14 - 47.

Rummelhart, D. L., McClelland, J. L. and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* vol. 1: *Foundations,* Cambridge, MA: MIT Press.

Stanfil, C. & Waltz, D. L. (1986). Toward Memory-based reasoning. *Communications of the ACM*, **29**, 1213 - 1228.

Stylianou, A. C., Madey, G. R. and Smith, R. D. (1992). Selection criteria for expert system shells: A socio-technical framework. *Communications of the ACM,* **35,** 10, 30 - 48.

Taylor, J., Parthemore, J. Wilson, I. & Halfpenny, P. (1992). Computer aided questionnaire design. Paper presented at Conference on Computing in the Social Sciences, Ann Arbor, MI.

Table 1. Proportions of comparability decisions for all combinations of CA and COMPASS decisions.

		COMPASS Decision			
Iteration	CA Decision	Yes	No	Adjust	N
	Yes	.71	.01	.01	
1	No	.05	.01	.02	147
	Adjust	.02	0	.16	
	Yes	.84	.01	.12	
2	No	-	-	-	75
	Adjust	.01	0	.01	
	Yes	.73	.02	0	
3	No	.06	0	0	48
	Adjust	0	0	.19	
	Yes	.77	0	0	
4	No	.10	0	0	30
	Adjust	0	0	.13	