TRADING MIPS AND MEMORY FOR Knowledge Engineering

Robert H. Creecy
Brij M. Masand
Stephen J. Smith
David L. Waltz

As part of its mission to profile the people and economy of the U.S., the Census Bureau collects industry and occupation data for individuals in the labor force. For the 1990 Decennial Census, each of an estimated 22 million natural language responses to questions on the census long form had to be classified into one of 232 industry categories and 504 occupation categories. If done fully by hand the cost of this task would be on the order of $15 million.

This article presents a new automatic classification system—the Parallel Automated Coding Expert (PACE)—that takes advantage of a massively parallel supercomputer. The system is based on an empirical learning model called Memory-based Reasoning (MBR) [28]. Following the MBR model, the PACE system uses a training database of 132,000 previously classified returns to classify new census returns not contained in the database. This contrasts with the Automated Industry and Occupation Coding System (AIOCS), an automated system developed by the Census Bureau for the 1990 Census, which is essentially an expert system driven by knowledge extracted from human experts and tested via the same preclassified database. The thesis of this article is that the MBR paradigm provides a more accurate, more robust and simpler solution that is directly reflected in a much reduced software development effort. Case in point, the building of PACE required four person-months while the expert system required 192 person-months. Moreover, PACE exhibits higher performance; it can process approximately 60% of the returns accurately while AIOCS can process approximately 47%. MBR is well-matched to data parallel computer hardware and programming models, such as those of the Connection Machine, so that the final system operates rapidly, though the computational requirements are substantially higher than those required to run the expert system.

The Census Classification Task

The industry and occupation data the Census Bureau collects consists of free text and multiple choice responses from over 22 million U.S. citizens. The actual questions and a typical response look like those in Table 1.

Before 1990, industry and occupation coding was performed using extremely expensive and time consuming clerical methods. The clerks used bulky procedure manuals and dictionaries of phrases and codes to classify cases. A section of the alphabetical index to industries coding manual that a clerk might use to classify the previous example is given in Table 2.

Many approaches to automatic coding have been tried and are summarized in [19]. One such system, the AIOCS [3, 4] was developed over the past eight years and used in the 1990 Decennial Census. To minimize cost and maximize accuracy the AIOCS system was augmented by the existing clerical coding system. Text from the long forms was keyed at seven processing offices across the country, and then sent electronically to Census Bureau headquarters. There AIOCS attempted to assign an industry and occupation code to each case. Those cases that could not be reliably coded by AIOCS were sent to a staff of over five-hundred clerks working at the Kansas City processing office. They used a computer-assisted method that allowed easy searching of a computerized version of the clerical coding manuals to complete the coding process. Final clerically assigned codes were transmitted back to headquarters to be included in the Decennial Census publications and computer data products.

AIOCS is essentially an expert system that uses a lexicon based upon the phrases that appear in the clerical coding manuals along with a pattern matching and a numerical weighting scheme based on an entropy measure. This basic approach is supplemented by lexical techniques that attempt to find misspellings and synonyms and by logical analysis that directly recognizes and processes special cases. The development of AIOCS required intensive interaction between subject matter experts, who had intimate knowledge of the clerical coding methods, and computer professionals who developed the expert system. This knowledge acquisition phase of the project represented, as with other expert systems, a large part of the total effort and expense of the project.

Other statistical approaches to the coding problem have been explored at the Census Bureau in the past [10, 20], but have never been implemented, partly because a sufficiently large, representative sample of computer readable responses was not available. In 1986, a sample of 132,247 responses to the industry and occupation questions from the 1980 Census was triply coded by clerks and reviewed by experts in order to provide a valid data set for evaluating the AIOCS system. This data set was also the basis of a more recent statistical approach [12]. Creecy's work was founded on two potential advantages that a sample-based classifier offered over AIOCS: 1) classification is based upon a database of the text from actual responses rather than the text from the clerical coding manuals, which may not match the wording respondents actually use; and 2) the sample contains information about the frequency of occurrence of codes and words in the population of responses. Although the classifier resulting from this work was not as good as AIOCS, it suggested that if a powerful enough computer were available, the data set could successfully be used as a training set for a new classifier. This idea led directly to the Connection Machine implementation of PACE described in this article.

Main Results

Following the MBR methodology, PACE uses the 132,247 example database as a training database and a disjoint set of examples is used as a test database. PACE matches each new census return with the entire training database and assigns a code based on the codes of the (previously classified) nearest matches. The 132,247 example database is also required for the development of the expert system as it was needed as a testing set to accurately evaluate the performance of the system. In Table 3 we compare the performance of the two systems and their development times. The reported development time reflects the amount of effort needed to build each system assuming no existing development tools other than standard programming languages. It can be directly compared to the effort needed to develop each system exclusive of the time required by both systems to develop the training database. The performance numbers indicate the percentage of the database that can be classified automatically, while controlling accuracy to meet or exceed that of human coders (about 10% error rate on industry codes and 14% error rate on occupation codes).

The numbers in Table 3 show that PACE exhibits a 54% improvement over the expert system in coverage of occupation codes and a 10% improvement for industry codes. If AIOCS had been used in the 1980 Census processing, the expert system would have resulted in a 47% reduction in clerical workload and if PACE had been used, it would have resulted in a 60% reduction in clerical workload. These improvements are substantial given the size of the coding task for the 1990 Census (approximately 22 million returns) and its cost (approximately $15 million).

Table 1.

For whom did this person work?	Essex Electric
What kind of business or industry was this?	Photography-Battery Div.
Is this mainly:	0 (Manufacturing)
0. Manufacturing	
1. Wholesale	
2. Retail	
3. Other?	
What kind of work is this person doing?	Apprentice Electrician
What are this person's most important activities or duties?	Wiring Machinery
Was this person employed by:	0 (Private Company)
0. private company	
1. federal government	
2. state government	
3. local government	
4. self employed	
5. working without pay?	
What is this person's age?	25

This response should be classified into the industry category "Photographic Equipment and Supplies" (code 380) and the occupation category "Electrician Apprentices" (code 576)

Table 2.

Phrase	Code	Industry Category
Photographic Apparatus	380	Photographic Equipment and Supplies
Photographic Cameras and Supplies	651	Sporting goods
Photographic Control Systems —electronic	341	Radio TV and Communication Equipment
Batteries, automotive, secondhand	682	Miscellaneous Retail Stores
Battery manufacturing	342	Electrical Machinery, equipment
Battery—retail	620	Auto and Home Supply Stores
Battery—wholesale	500	Motor Vehicles and Equipment

Table 3.

	% of Industry Codes Assigned	% of Occupation Codes Assigned	Person-months Development Time Required
AIOCS (expert based)	57	37	192
PACE (memory based)	63	57	4

Table 4.

		Number of Exact Word Matches
Test Example:	"The Computer Industry"	
Training Example 1:	"Retail Computer Sales"	1
Training Example 2:	"The Automobile Industry"	2

The MBR Approach

MBR was introduced by Stanfill and Waltz in 1986 [28] and recent interest has been growing [1, 6]. Ideas related to MBR, however, have a long history. MBR comprises a series of variations on nearest neighbor classification schemes, with the addition of other statistical techniques [29]. The simplest nearest neighbor technique consists of assigning a given example to the same category as the preclassified example most similar to it. For example, a very simple version of MBR uses a Hamming distance metric where the nearest neighbor is the example with the highest number of matching fields. For an excellent review of nearest neighbor techniques, see [13].

Although this idea is conceptually simple, it seems difficult to implement efficiently. MBR potentially requires comparing a given example to every training example in the database; and computing the similarity between examples and cases generally requires more sophisticated measures than Hamming distance, since the likelihood of exactly matching free text strings is small. The larger the database of preclassified examples, the more likely the nearest neighbor algorithm will arrive at the correct classification since an exact or nearly exact match becomes more likely. Storing and matching against these preclassified examples becomes, however, more and more expensive as the database increases in size.

In general, classification accuracy increases slowly (less than linearly) as the database size is increased, while the computational requirements of the larger database grow linearly. The processing time can be reduced by using better algorithms. In fact, nearest neighbor algorithms are in some ways very close to information retrieval algorithms used in text search engines. The information retrieval problem is usually solved on serial machines by use of a hash table or a precompiled inverted index or concordance. (Text algorithms are particularly applicable to the census classification task since much of the relevant information in each example is contained in the free text fields of the example [18].) Another approach to this O(N) match problem is to obtain a speedup by using parallel computing hardware where the number of processors (P) equals the number of preclassified examples (N), giving a nearly constant time solution. PACE uses a variant of this method where each processor holds 16 examples.

MBR augments a simple nearest neighbor match by weighting relevant data in the training examples more heavily than other data. For instance, consider the case where the industry field of a training example includes the following phrase: "The computer industry." Of the three words "computer" is clearly the most important. It could even be argued that "the" and "industry" occur so often in the database and in so many different categories that they should have little or no bearing in forming the match, as illustrated in Table 4.

If the Hamming distance metric were used, then the second training example would be chosen as the nearest neighbor, when it is obvious that the first example is a better match. We later introduce several feature-weighting and evidence-accumulation methods that can correct this error. These weighting algorithms for MBR systems make use of information metrics similar to Shannon's [22, 24] or statistically derived metrics. Similar ideas on modified metrics have appeared in the nearest neighbor classification literature [15, 25] but have not often been used in practice.

Distinguishing Fields from Features

A second difference between MBR and simple nearest neighbor classification involves the nature of features. The terms "fields" and "feature" are often used interchangeably to describe pieces of the data structure that are useful for matching. We would like to distinguish them in the following way:

Field: A division of the data within an individual example that is supplied a priori by the data structure. Age and Industry Type are examples of fields in the census data structure.

Feature: An element or grouping of the data that may or may not be provided a priori, but must be determined according to the increase in classification accuracy it provides in the partial match. Features might be conjunctions of fields, or they may be portions of a field or combinations of portions of fields.

MBR can create features from fields by grouping. For instance, the conjunction of the Age and Industry Type fields may produce a feature that is more useful at classification than either field by itself. In PACE, features are no more complex than two-element conjunctions. In principle, feature construction could be extended to n-element conjunctions.

A third addition to the simplest nearest neighbor methods is the use of k nearest neighbors for some decisions, rather than only using a single nearest neighbor. Our MBR model includes methods for combining information from the various weighted fields within an example, as well as algorithms for combining the information from the k nearest neighbors.

Modifications to MBR to Extract Fields in Free Text

Classification methods such as nearest neighbor [11], bayesian statistics, clustering [2], and empirical learning systems [22] are commonly designed for data fields that take on scalar numerical values. For example, consider a database where each example contains only the single field "Age:"

	Examples			
	x0	x1	x2	x3
Age:	10	33	19	67
Class:	A	B	A	C

The "Age" field in this case is comparable across examples and is a field such that is easy to tell whether one example is "close" to another. Consider, however, a database that is more similar to the form of the census domain:

	x0	x1	x2	x3
Word:	car	big	retail	factory
Class:	A	B	A	C

In this case there is no order that can be placed on the examples that makes sense in terms of classification. In fact, this case which appears to have a single field "word," and can be more truthfully represented by four binary fields:

	x0	x1	x2	x3
car	1	0	0	0
big	0	1	0	0
retail	0	0	1	0
factory	0	0	0	1
Class:	A	B	A	C

This second format is the one that is preferred by standard methods, and it brings home the true dimensionality of the database. In the case of the census database, there are over 50,000 different words, potentially corresponding to a 50,000-dimensional, binary-valued space.

One problem with performing nearest neighbor classification with free text is that the textual data contained in the fields is not easily comparable. There are so many different ways of expressing occupations and industries as phrases that exact match cannot work (e.g., the duties field: "Serving food and drink" does not match a phrase with only a slight modification: "Serving food and drinks"). With numeric fields, a distance or degree of match can be computed even in the absence of exact match (i.e., we know 100 does not equal 101, but we know that, in general, it is a better match than that between 100 and 3). We need to use an analogous softer match on the free text. This can be partially accomplished by matching specific words in text fields rather than entire text fields. Thus in our previous example we could note that the two text fields: "Serving food and drink," and "Serving food and drinks" match exactly on three out of four words—not a perfect match, but nonetheless much better than the match to a phrase such as "Driving trucks cross country."

Given this example it is tempting to go even further and to enhance features by coercing words into canonical forms—thus the two phrases could both be coerced into: "Serve food and drink." It seems plausible that the use of canonical-stemmed forms could improve classification accuracy. Empirically, however, stemming is not always useful, and is often harmful [16]. Consider the two words "attorney" and "attorneys." It might be thought that both these words have basically the same meaning and could be used interchangeably. This is not the case in the census database. All but 2% of the occurrences of the word "attorneys" belong to the "Legal Services" industry category, while 32% of the occurrences of the singular form "attorney" belong to classes other than Legal Services. What this means is if a query containing the word "attorneys" is classified into the legal services category, then this will be the correct classification 98 times out of 100. If the query contains the word "attorney" it will be correctly classified only 68 times out of 100, a tremendous difference in classification accuracy between the singular and plural forms of a single word.

Thus, we see from the "attorney"/"attorneys" example that blindly stripping prefixes and suffixes from words to arrive at a canonical stem must be used prudently, if at all. It is our belief, and the belief of some other researchers who use data-driven approaches like MBR that the data rather than human intuition should drive the design of the application [8, 9].

Extracting Features

The census database used for this work contains three numeric fields and four free text fields (corresponding to the responses to the questions about company name, industry, occupation and daily duties). Each free text field is broken up into word fields. Each word field is tagged with the value of its source free text field, e.g., the word "attorney" that occurs in the Industry free text field is distinguishable from the word "attorney" occurring in the Occupation free text field. The distinction between words is represented by appending a ".i" for industry a ".o" for occupation, a ".c" for company name or a ".d" for duties ("attorney.i", "attorney.o", etc.). Keeping these words distinct can be important. In the census database, for example, when the word "attorney" occurs in the Occupation text field the occupation classification is most likely to be that of a lawyer. When the word "attorney" appears in the Industry field, however, it is more likely to be describing an attorney's office and the Occupation classification is more often "clerk" or "secretary" than "lawyer."

PACE features consist of single word fields along with all pairwise conjunctions between them. For example, the responses to the census return presented earlier consists of nine word fields: "essex.c", "electric.c", "photographic.i", "battery.i", "div.i", "apprentice.o", "electrician.o", "wiring.d", "machinery.d"; and three numeric fields: Industry Type = 0, Company Class = 0, Age = 25.

Features can be formed by combining these word and numeric fields into a set of all pairwise conjunctions. In this example, there are 12 fields corresponding to 144 ordered conjunctions or 72 combinations since the order of the fields in the conjunction is ignored.

The difference in predictive importance between a single field feature and a conjunctive feature can be substantial. In an example from the census database the fields "shop.i" and "machinist.o" occur as single field features. The probabilities of an example being classified in the general machinery industry category given either of these two fields are 0.24 and 0.29 respectively. When a conjunctive feature is formed, "shop.i" & "machinist.o" the conditional probability jumps to 0.93, i.e., if the fields occur in a test example the chance for correct classification using them separately is 24% or 29%, but when used together correct classification can be accomplished 93% of the time.

Though creating conjunctive features and their weights is beneficial to system accuracy, there is an associated computational and storage cost. Some 65,000 training examples produce over 4.5 million features. Despite the magnitude of this number, the conditional probabilities used to weight the features can be calculated quickly on the Connection Machine system. This can be done either at run time where calculating only the features relevant to the current test example can be accomplished within 200 milliseconds; or the weights can be precomputed and stored. Precomputing and storing the conditional probabilities of all 4.5 million features can be accomplished in less than 10 minutes on a 32k processor, CM-2 Connection Machine.

Calculating Feature Importance

Once features have been created it is necessary to weight the features according to their ability to classify the examples. In PACE, weights are calculated by methods similar to the sum of squared probabilities metric described in [28]. Stanfill and Waltz adopted the sum of squared probabilities metric in order to assign higher weights to features that occur in fewer different categories. In the following examples we see this for three different features.

Consider a domain with only 10 categories (Figure 1). To determine the importance of two different features, a histogram across the categories is constructed for each feature. If the sum of the distribution is normalized to one, each bar of the histogram represents the probability of the given category, given that the feature occurs in the example [P(C|F)]. From the histograms in Figure 1 it appears that feature 1 is equally likely to occur in each category and thus is not very useful (this might correspond to a feature such as the word "the"). The histogram of feature 2 indicates that if this feature is present, the test example must belong to one of only two possible categories. Thus, we need a weight metric that reflects this difference. The sum of the squared conditional probabilities results in a weight of 0.1 for f1 and 0.5 for f2. The highest importance weight of 1.0 corresponds to a feature that is perfectly correlated with a single category.

$$\text{Weight }(f_k) = \sum_i P(C_i|f_k)^2$$

$$\text{Weight }(f_1) = 0.1^2 + 0.1^2 + 0.1^2 + 0.1^2 + 0.1^2 + 0.1^2 + 0.1^2 + 0.1^2 + 0.1^2 + 0.1^2 = 0.1$$
$$\text{Weight }(f_2) = 0.5^2 + 0.5^2 = 0.5$$
$$\text{Weight }(f_3) = 1.0^2 = 1.0$$

It should be noted that the sum of squared probabilities varies from a maximum of 1 when a feature appears only in one class and is of maximum usefulness, to a minimum of $\frac{1}{N}$ when the feature appears uniformly across all classes $\frac{1}{N}$ (N = the number of different classes).

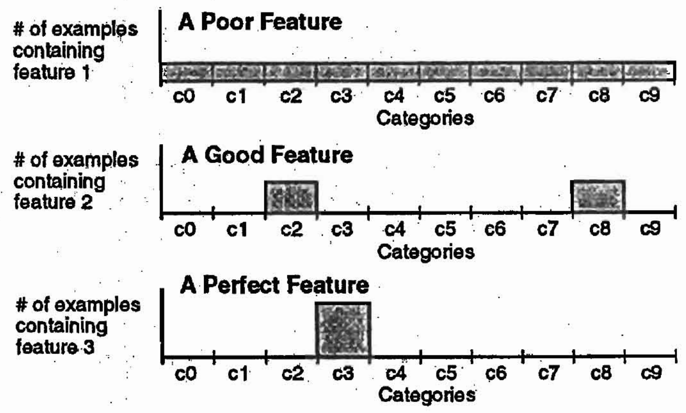
Per Category Feature Importance

In PACE, the sum of squares importance weighting in conjunction with the k-nearest neighbor metric, provides the best classification accuracy for occupation coding. A different method provides the best classification accuracy for industry coding. This second method contrasts with the sum of squared probabilities method by assigning different weights for the same feature depending on the category the feature is found in. For this reason, we refer to this method as "per category" feature importance in contrast to the sum of squares metric which is identical for a feature across all categories (we will distinguish the two as "per category" and "cross category"). To see the benefits from per category importance, consider a test example that contains only the single feature "weaver." In the census database the word "weaver" occurs 84% of the time in the occupation category for cloth knitters and weavers. It also occurs in four other categories for basket and various types of wire and metal weaving, but much less frequently. If cross category feature weighting were used then the feature would have the rather high weight of 0.71 $(0.71 = (.84)^2 + (.07)^2 + (.03)^2 + (.03)^2)$ and matching training examples would have the same high score for each of the five weaving categories. Therefore, the metric does a good job of distinguishing these five categories from the rest of the possibilities with high confidence (0.7), but it can make no distinction between these categories. Clearly valuable information has been lost since the probability of the cloth weaver category conditional to the word "weaver" is 12 times higher than the next most probable category.

This problem was resolved in PACE by calculating *per category feature importance.* This method calculates the conditional probability [P(C|F)] for every feature in every category and stores this number along with the feature in each example. In this method each feature of the query contributes a different importance to the match depending on the category in which each example occurs.

In the "weaver" example above the matching training examples would now have different scores based on their categories. Any

Figure 1. Comparing feature classification value by category distribution



training example occurring in the "cloth weaver" category would have a score (P([cloth weaver]|"weaver")) equal to 0.84 whereas other categories would have scores of 0.07 or 0.03. Thus the score now correctly reflects the high probability of the "cloth weaver" category and the test example is more likely to be correctly classified.

Accumulating Evidence

It has been assumed to this point that the nearness between a test and a training example is proportional to the sum of the per category or cross category weights of each feature contained in the test example:

$$\text{Nearness}(i, j) = \sum_{k,l} w(k)\, f(k,l)$$

where
i indexes test examples
j indexes training examples
k indexes the features in test example i
l indexes the features in training example j
$f(k,l) = 1$ if feature l = feature k
$= 0$ otherwise.
$w(k)$ is the weight of feature k

This is similar to the techniques that have proved effective in previous work [28]. It has also been used successfully in the census task and will be called the SUM metric. One difficulty with the sum metric is that examples of many words tend to have higher match scores than examples with few words.

Maximization Metric

Another approach considers all features in the test example and chooses the training example with the greatest P(C|F), i.e., the greatest probability for classifying the example into Category C given the feature F. This is an appealing strategy since it is founded on basic probability argument, and avoids preferring long examples. For instance, if the maximum $P(C|F) = 0.9$ for $C = C1$ and $F = F1$ then any query containing feature F1 will be classified correctly 90% of the time (assuming the database is large enough to eliminate sampling errors). This method has the additional advantage of a list of features and their weights that can be precomputed and stored for each category and the original data need not be retained. This can considerably reduce storage and computation costs. A disadvantage of the method (see Tables 5 and 6) is that the MAX metric does not take multiple sources of evidence from different features into consideration. The advantages and disadvantages balance out, however, to allow MAX to achieve performance comparable to the SUM metric on this database.

Error Minimization Metric

The Error Minimization Metric attempts to retain the best of both the MAX and SUM metrics. It is of interest since it outperforms both the SUM and MAX metrics when applied to the census Industry coding task. The general idea behind it is that if there is a single feature with a high P(C|F) in a training example, then that feature should have a large effect on the cumulative match score of the example. If, however, two examples representing two different categories each have large numbers of moderately weighted features, then the nearest neighbor should be the one with the largest cumulative weight. In essence, the ERROR metric behaves like the MAX metric in certain situations and like the SUM metric in other situations. It gives strong weight to any outstandingly high P(C|F) value but still takes into consideration additional information from other features.

More precisely:

$$\text{Nearness}(i,j) = 1 - \prod_{k,l}(1 - w(k)\, f(k,l)) \text{ where } w(k) = P(C|F)$$

Here the error of a feature is equal to $(1 - P(C|F))$, or the probability that an error is made in assigning the category based on this feature. This metric has the following interesting properties: 1) if any feature contained in the example has a $P(C|F) = 1$ the error is 0 independent of the other features; 2) each feature with a nonzero weight contributes to increasing the certainty of the match score; and 3) the metric varies between 0 and 1.

To illustrate why the ERROR metric might be more successful than the SUM or MAX metrics, consider the two cases depicted in Tables 5 and 6 where a query example is matched against two training examples. In the first case the two training examples differ by one having a single highly weighted feature and by in the other having several features with low weights. In this case the SUM metric gives a higher score to example 2, leading to an incorrect classification. The ERROR and the MAX metric give the correct result.

Table 6 illustrates that the MAX metric can also cause mistakes. Here, one of the two matching training examples has a large number of low weight features, whereas the second has only a single matching feature of slightly higher weight. In this case the MAX metric performs an incorrect classification. The SUM and ERROR metric give the correct result.

K Nearest Neighbors

In a large database, such as the census database, there can be many matches other than the nearest match that can potentially contribute to the accuracy and the confidence of the assigned category. For example, if a large number of near neighbor examples are of the same category, a higher confidence score can be assigned to the classification than if the nearest neighbors were of many different categories. To explore this idea classification experiments were conducted using k nearest neighbors (typically 10-15), using the following algorithm.

After the match step, the k matches having the best scores are identified. Cross-category weights are used for this step. The individual scores are then aggregated by

Table 5.

	Example 1* P(C1|F)	Example 2 P(C2|F)
F1	0.9	0.1
F2	0.0	0.5
F3	0.0	0.5
Max metric	0.9	0.5
Sum metric	0.9	1.1
Error metric	0.9	0.77

*denotes true nearest neighbor

Table 6.

	Example 1* P(C1|F)	Example 2 P(C2|F)
F1	0.4	0.41
F2	0.4	0.0
F3	0.4	0.0
Max metric	0.4	0.41
Sum metric	1.2	0.41
Error metric	0.78	0.41

*denotes true nearest neighbor

Table 7.

QUERY (with correct code 579):

Feature	Weight
"Apprentice"	9
"Painter"	36

RESULTS (using k = 15, nearest matches):

Code	Name	Score	Matches (nearest 15)
579	Painters, Construction	261	7
789	Hand Painting, Coating	72	2
759	Painting & Paint Spraying	72	2
556	Supervisors, Painters	72	2
569	Carpenter Apprentices	45	1
558	Supervisors, Nec.	36	1

MATCHES:

Rank	Score	Code	Text
1	45	569	"Apprentice Carpenter Painter"
2	45	579	"Apprentice Painter"
3	36	579	"Painter"
4	36	579	"Painter"
5	36	556	"Painter-Foreman"
6	36	556	"Painter"
7	36	579	"Painter"
8	36	789	"Sign Painter"
9	36	579	"Painter"
10	36	579	"Painter"
11	36	558	"Painter-Foreman"
12	36	789	"Sign Painter"
13	36	579	"Painter"
14	36	759	"Furniture Painter"
15	36	759	"Painter"

categories (among the best k matches). Finally the category with the highest aggregate is used to classify the test example.

In cases where there is a single clear winner, all k-nearest matches will belong to the same category, and the k-nearest neighbor method performs the same as a single nearest neighbor match. If there are a number of near matches, this method can protect against overvaluing a single spurious near match. In the real example in Table 7, the nearest neighbor method can incorrectly predict occupation category 569 (which has one match in the first 15 matches), while the k-nearest neighbor method correctly predicts category 579 (which has 7 of the first 15 matches).

Optimizing K

A further improvement in the algorithm can be gained by optimizing the value of k. A range of values from 3 to 25 were tried. (Results are shown in Table 8.) Only variations in the overall accuracy are reported. All of the experiments were run on the same set of 1,000 randomly selected examples.

Assigning a Confidence Score

Apart from overall accuracy, a robust confidence score is crucial for increasing the percentage of the test cases that can be processed at human levels of accuracy. For the same overall accuracy, a confidence score that better separates good matches from poor ones will result in a higher percentage of processed examples. Initially, the match score itself was used as a confidence score. However, this did not take into account matches from other categories that had scores close to the best score. In order to include the effect of having closely competing categories, we compared the score of the best predicted category with the next best predicted category, replacing the score of the best category with an adjusted confidence score using a formula A/(A + B) where A is the score of the best category and B is the score of the next best category. This formula reduces the confidence when the scores of the best and the next best category are close. The use of adjusted scores for the k-nearest neighbors metric increased the number of examples processed by 3%.

Evaluating Performance

Two measures are relevant in evaluating the performance of these classification systems; we will call them *accuracy* and *coverage*. Coverage measures the percentage of all returns/forms that are attempted by the system and not referred to human coders. Accuracy measures the percentage of attempted classifications that are correct. The ideal system would thus have high accuracy (few mistakes) and high coverage (few referrals). In general, both cannot be simultaneously maximized, and compromises must be made (Figure 2a). In the case of the census system, we were constrained by the problem definition to produce a system that was at least as accurate as human coders. Thus, this compromise required a decrease in the coverage in order to increase the accuracy to levels of 86% for occupation coding and 90% for industry coding. This tradeoff was accomplished by setting "referral thresholds" for each class, such that if the confidence score of the class of the best match does not exceed the threshold, then the form is referred [7].

Referral thresholds are calculated automatically. This is done by first collecting a list of test examples that have all been classified into the same category and then sorting them by confidence score. A referral threshold is then chosen as the match score of each example and coverage and accuracy are evaluated at that point. A threshold for the required accuracy can then be chosen and the coverage determined (Figure 2b). The choice of a given threshold divides the list of examples into two sublists (Figure 2c): those that are covered by the system and those that are referred to human coders (in Figure 2c the size of these two lists are noted as C and R, respectively). Thus for any given threshold, accuracy can be expressed as C′/C (where C′ denotes the number of examples in C that were correctly classified), and coverage can be expressed as C/(C + R).

Validation Methodology and Best Classification Metrics

Cross validation was used to test and compare the different classification metrics. Both two-way cross validation (where the database was randomly split into a test set and a training set) and n-way cross validation (where only a single test example was drawn from the database at a time) were used. In both cases the test and training set were kept completely isolated and the relevant statistics for feature weights were calculated solely from the training set. For the results that we cite, random test sets of at least 5,000 examples were used. This is large enough to ensure the statistical significance of the estimates of overall accuracy and coverage.

While most of the MBR scoring metrics presented so far were very successful on the classification task, we found techniques that are superior to the others for both the industry and the occupation coding problems. Interestingly, different techniques were best for each of the two tasks.

For industry coding, the best MBR method is based on per-category weighting of conjunctive features using the error minimization metric. PACE was able to process 63% of all returns at the same level of accuracy as a human coder, which is significantly better than the 57% of returns that are handled by the expert-based AIOCS techniques (Figure 3).

The best occupation coding MBR method is based on cross-category weighting of single field features using the summation metric and k nearest neighbors. In this case an optimal k was found to be 12, which resulted in PACE being able to process 57% of the returns at the accuracy level of a human coder. This contrasts very favorably with the 37% attained by AIOCS (Figure 3).

Table 8.

k	Industry % Correct	Occupation % Correct
3	71.4	52.9
5	72.5	54.7
8	73.7	54.9
10	73.3	55.5
12	73.2	57.1
15	73.3	57.1
18	73.5	56.2
25	72.4	55.8

It is clear that weighted feature metrics perform better than unweighted feature metrics. Although the error metric provides the best performance for industry classification, the other metrics are not far behind. For occupation classification, the k nearest neighbors method performed best, even with single words as features and cross-category weighting. These results suggest that, for the level of performance we achieved, a number of different methods and metrics lead to comparable results. Table 9 indicates how the main variants of the MBR metric compared.

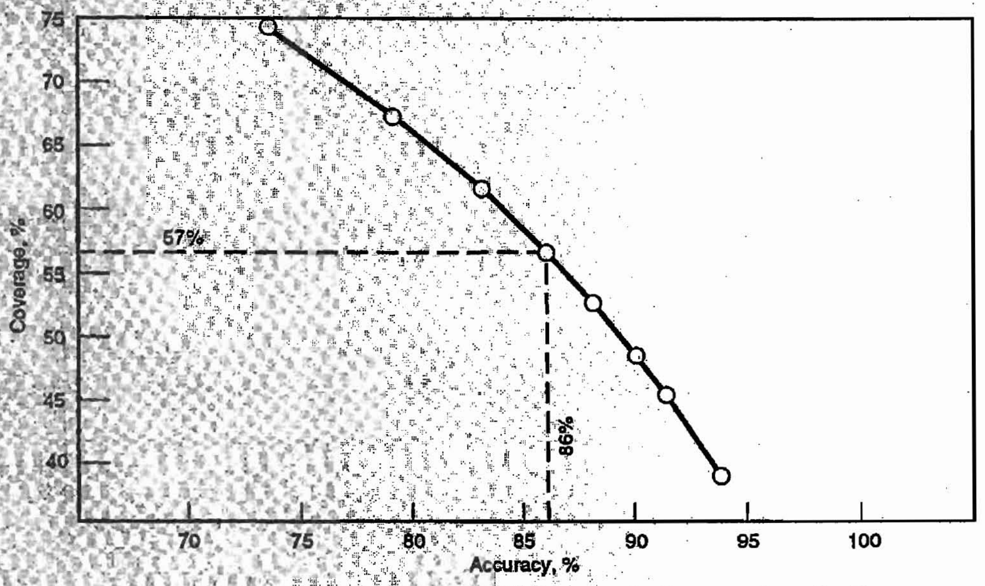
Sensitivity to Database Size

Since all of these MBR methods are dependent on the size of the training database, it is reasonable to ask how much performance improvement could be gained by having an even larger training database. We explored this question by constructing an experiment that compared the overall match-rate (accuracy at 100% coverage) at different database sizes. To conduct the experiment we performed a nearest k neighbors classification (k = 12) by randomly selecting increasingly larger subsets of the database. This experiment compared the accuracy of assigning industry codes by matching on the Industry and the Company Name fields with cross-category weighting. A random sample of 10,000 examples was used as a test set.

We found a difference of 12% in the overall match rate accuracy when the training set was varied in size from 10% to 100% of the full 132,000 example database (Figure 4). More importantly, doubling the database size seemed to yield an improvement of 3–4% over the range of the experiment. This suggests that doubling the database size to 264,000 examples would result in an accuracy of up to 75% at 100% coverage. Empirically we found that a 3–4% change in accuracy at 100% coverage led to a comparable change in coverage at human accuracy levels. A 3–4% change in coverage would result in significant financial savings for the Census Bureau.

We conclude that the perfor-

Figure 2. Threshold calculations for coverage and accuracy

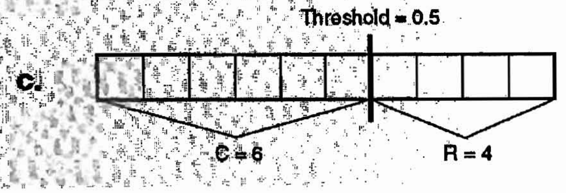


A. Trading Off Coverage for Accuracy in Occupation Coding

Predicted Category	343	343	343	343	343	343	343	343	343	343
Actual Category	343	343	343	512	343	343	762	343	512	765
Confidence Score	0.95	0.93	0.91	0.82	0.80	0.63	0.50	0.47	0.39	0.35
Correct Classification	+	+	+	−	+	+	−	+	−	−

Possible Thresholds										
Coverage	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Accuracy	100%	100%	100%	75%	80%	83%	71%	75%	66%	60%

B. Ten Examples for Category C_1



mance of PACE is sensitive to database size and that we could not have reduced the database size by 50% without seriously compromising performance. It is also clear that the current database size is smaller than the optimum, and that enlarging it could yield significant improvements.

The Data Parallel Implementation

One of the main advantages of an MBR approach to classification is it enables the use of massively parallel supercomputers by easily accommodating the programming model most naturally suited to massively parallel computers. This programming model is called "data parallel" computation and represents problems and implementations where many copies of a given data structure can be distributed to multiple processors and processed in parallel.

In the census application this data structure consisted of the fields and features of each training example, their corresponding weights (if they were precalculated and stored), and the class of the data item. This data structure was instantiated with the fields and category of each example and stored one per processor on an 8K processor CM-2 parallel computer (132K virtual processors are simulated). To compare a new example with the 132k examples stored in the CM-2, features from different fields of the example (typically words) are broadcast serially and the processors compare the broadcast features with the features of their respective examples in parallel, modifying their cumulative scores in case of a match. The scores are then compared across all examples to find the nearest match (this is accomplished using a global max operation).

Calculations of the cross-category and per-category weights can also be accomplished under the data parallel model. To accomplish these calculations efficiently, examples belonging to the same category are grouped together in adjacent processors in what is called a segment of processors (Figure 5). The feature whose weight is desired is then matched to each example in parallel and a 1 or a 0 is retained in each processor to signify the presence or absence of the feature in the particular example. It should be noted that the conditional probabilities required for both the per-category and cross-category weightings are simply the sum of these 1s and 0s within each segment divided by the sum across all segments. To accomplish these summations a powerful data parallel operation called *Scan* was used. This operation can perform the summation within each segment in parallel in slightly less time than one general communications step. The summed value of each segment is left in the processor at the end of each seg-

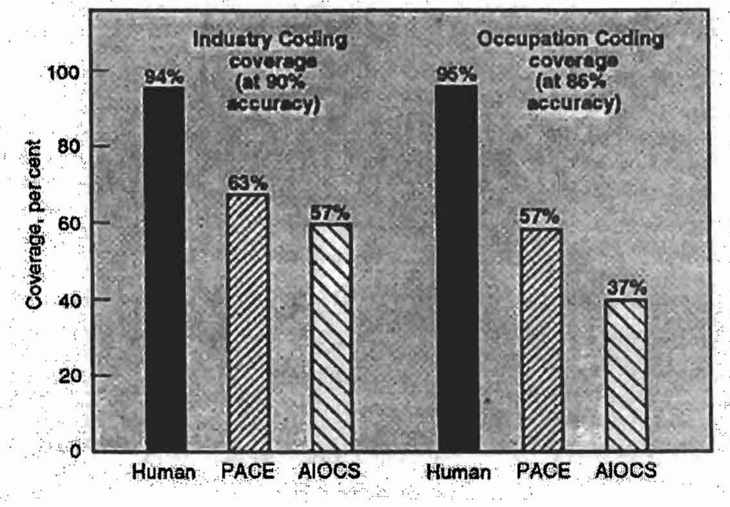


Figure 3. Relative coding performance of different systems (Note that Human coverage does not achieve 100% as some responses handled by human coders may be referred to experts, or may not be codable at all.)

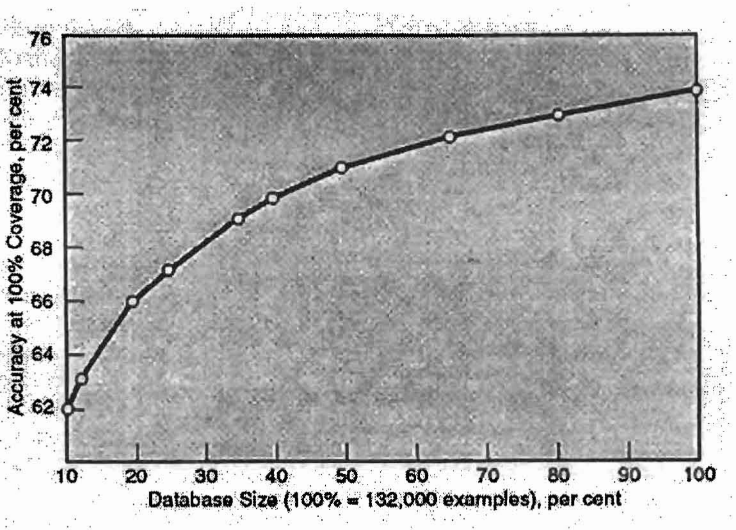


Figure 4. Sensitivity of industry coding to training database size

ment where it can be copied back to the other processors in the segment via another scan operation and then divided by the global sum to determine the conditional probability for each category for this particular feature.

This parallel weight calculation algorithm is sufficient for calculating feature weights dynamically when there are only a few (<100) features in a test example, but when all features of the entire database must be calculated and stored it is not as efficient as possible. Since each feature must be broadcast in turn the algorithm is serial in the number of different features whose weights need to be calculated. In the census application there are over 4 million pairwise conjunctive features. To calculate these weights a more parallel algorithm is used.

To calculate the weights of all 4.5 million features, a parallel data structure was constructed residing in 4.5 million virtual processors. This data structure is a composite made up of slots for the first field, the second field, the category and a return address (Figure 6). The processor containing each training example creates all possible pairs of fields and instantiates this new data structure. The intent of this computer data structure is to act as a palette where all 4.5 million features can be sorted such that identical features end up within the same segments and identical features with the same category will be near each other within the same subsegment. This can be accomplished with a single sort by viewing the field slots of the composite number as the most significant bits of the key and the category slot as the least significant. Once the features have been organized in this way segmented scans can be used in the same manner as previously (the global sum can now be replaced by a scan across the feature segments). Once the weights are calculated they can be sent back and stored in their originating training example via the return address retained in the composite data structure. Using this method the 4.5 million feature weights can be calculated and stored in only a few minutes, and by using these precomputed feature weights the classification system is able to run at the high processing speeds required by the Census (10 forms per second).

The Hardware

The CM2 is a massively parallel computer with a maximum of 65,536 single bit processors and 2,048 floating-point accelerators shared evenly amongst the processors [17]. Each processor has its own local memory of up to 128KB and can communicate with every other processor via a hypercube network. All programming is performed using a front end interface such as a Sun or a Vax computer. Typically a large amount of data is loaded onto the CM-2 and a user program running on the front end controls the operations of all the CM-2 processors on the data. The processors all execute the same

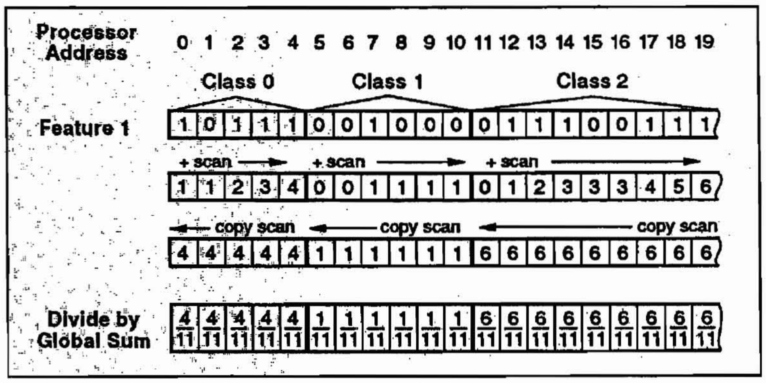


Figure 5. Calculating conditional probabilities dynamically

Figure 6. Calculating conditional probabilities for all features

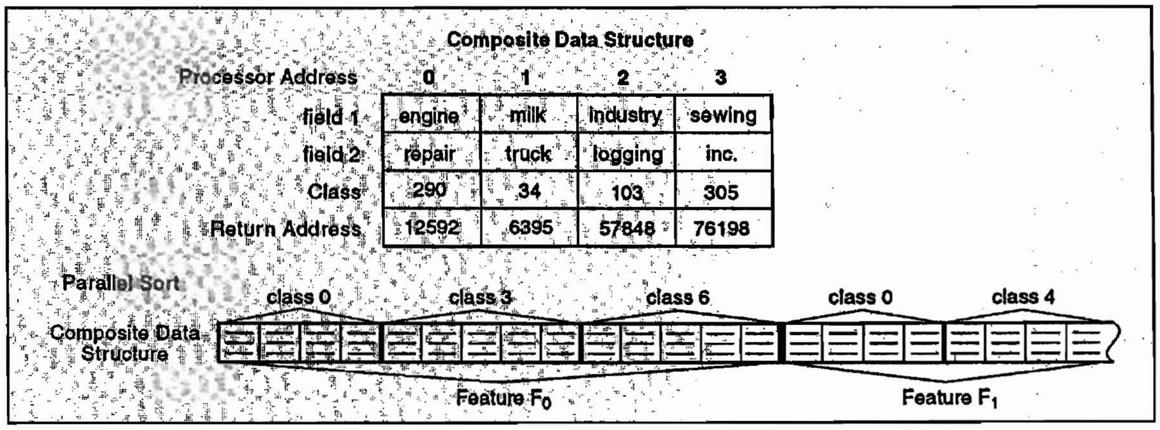


Table 9.

	INDUSTRY		OCCUPATION	
Method	Accuracy at 100% coverage	Coverage at 90% accuracy	Accuracy at 100% coverage	Coverage at 86% accuracy
MBR no weights	62%	42%	46%	48%
MBR cross category wts.	66%	44%	48%	49%
MBR SUM	69%	53%	61%	51%
MBR MAX	69%	59%	61%	51%
MBR SUM K-nearest	73%	61%	58%	57%*
MBR ERROR	71%	63%*	62%	56%

*Metrics used in PACE system.

operation on their individual data at the same time. The user programs can be written in *Lisp, C* and Fortran 90, which use parallel extensions of the respective languages. For this project we used an 8192 processor CM-2 (with floating point enhancement) and a Sun 4/280 front end. The software was developed in *Lisp.

MBR Advantages over Expert Systems

The results of our experiments have demonstrated that the performance of PACE for classifying census returns is superior to that of the AIOCS expert system built for the same purpose at the Census Bureau. However, it might simply be the case of particular projects being done with different degrees of skill or care. We would like to argue here that this is not the case, and that, for similar classification domains, MBR (as well as other data-driven methods) hold several advantages over systems that require hand building (knowledge engineering) such as traditional expert systems.

Ease of programming an application. PACE required only about four person-months of effort to build, including some time to build tools. In contrast, AIOCS took nearly 192 person-months. Both applications were built on top of standard computer languages (*Lisp for PACE and Fortran for AIOCS). Additionally, each system required the preclassified database of examples.

The main effort in building an MBR system is deciding how to compare the various types of fields for similarity. For numerical and logical fields, one can use statistical methods to decide how heavily to weight each field of each example or one can perform an optimization in field-weight space. For example, in [31] several sets of random weightings for the various fields were generated, and then each set was tried to see which gave the best classification performance on a test dataset. For text fields, a weighted match based on the methods used for text-text comparisons in the SEEKER system [26] could be adapted. These also use statistically based word overlap weightings.

The main effort in building expert systems is the encoding of an expert's knowledge as rules and a knowledge base. Either the expert or a knowledge engineer must hand-build the system. Both are time-consuming procedures for which only fairly general guidelines exist. Experts can seldom identify the rules used to classify cases, and as in the census domain, often provide general rules that do not easily match the specific responses contained in the database. The process of building rule sets still appears to be more of an art than a science.

Accuracy via completeness and uniformity of coverage. MBR systems can be built directly from the entire database. Because of this the resulting mix of examples will directly mirror the actual numbers of previous cases, and there will be greater coverage for cases that occur more often, leading to better overall performance. To see this, note that the greater the number of cases in an MBR system, the greater the chance of exact or near-exact matches. Since an exact match leads to a trivial decision process, and is guaranteed to be correct, the larger the database or denser the examples close to the case one is attempting to classify, the better the accuracy of the MBR system's performance. Moreover, even a single example can be useful in classification. In a medical database, the first case of Legionnaires' disease would not have closely matched any previous cases, but once the first case had been encountered and included in the database, the second case could immediately be recognized and grouped with the first.

In an expert system, it is very difficult to characterize, let alone establish, the degree, uniformity, and accuracy of coverage. Often it is necessary to create a sample set of classified examples to be sure that the expert system adequately covers the required classifications. In the census problem, this became apparent a few years after the initial development of AIOCS, when the need to quantify the performance of the system led to the creation of the 132,247 example text data set.

Ability to use mixed records (text, numbers). As discussed earlier, it is fairly easy to match examples that include phrases, numbers, and logical variables. While this does not differentiate MBR systems from expert systems, it does differentiate MBR from artificial neural net systems in which a good deal of cleverness is often necessary to input variables unless they are numerical or logical in form.

Scalability Because MBR systems map easily into the data parallel computation model they can be scaled to the limits of the memory of the hardware. On a CM-2 Connection Machine system, the database limit (for in-memory operation) is 8GB, or about 80 million records of 100 bytes each and several massively parallel machines introduced in 1991 offer even larger memory space. These large resources of today's supercomputers are often a critical advantage to empirical learning systems that can take advantage of them. In the bayesian census classification system of Creecy et al. [12] the system was unable to produce and use pairwise conjunctive features because of the limitations of the serial hardware on which it was developed.

As a database gets large, an expert system may or may not be able to cope. If the database is not well-behaved (as when there is an open-ended set of possible values for a given database attribute, e.g., the occupation, industry, and company name in the census database) then scaling will be difficult or impossible.

Ease of updating on the basis of changing data. In an MBR database, new classified examples can be added, and old, obsolete examples removed. On the next decision cycle the MBR system can make its decisions based on the current state of the world (assuming that one of the learning variants of MBR is not being used). Learning MBR systems trade relatively lengthy update or set-up times and rapid decision performance for low initial set-up or update time and slower decision-making speeds. By contrast, updating often requires the complete retraining of neural net and nonincremental ID-3-like systems, and potentially tedious hand editing of a standard expert system's rules. In the AIOCS expert system, when experts continued to try to improve the system by added phrases and rules, they were often surprised that their changes made marginal or even negative changes to classification performance.

Inherent confidence measures. MBR generates a nearness measure for each example. If any new example is identical to or very similar to a previously seen case, then the MBR system can assure its user that it has very high confidence in its results. If no previous case matches closely, it can report the closest case(s), but at the same time inform the user that the results are not very certain. In a sense, the system knows when it knows.

Forming a trustworthy confidence measure for standard expert systems is a challenging problem with uncertain success [14, 21, 32]. In general it is difficult to reflect the statistical likelihoods of various possible scenarios in numbers associated with rules.

Justification for classifications. In an expert system, a justification usually consists of a chain of rules, beginning with premises true of the current situation, and ending with a desired outcome. This often may be or may not be truly helpful in convincing the user that the explanation is valid, and merits credence. Artificial neural nets offer no explanations at all, except in the special cases where each hidden unit is sensitive to a single possible input pattern [23]. MBR systems offer the nearest neighbor (or neighbors) and their classifications as explanations (precedents). Such explanations are natural and believable, and easy to compute.

Conclusion

PACE, a memory-based reasoning classification system, has proven to be successful in performing the complex census coding task that has previously required human coders. This is also true of AIOCS from the Census Bureau, though PACE can accurately process 13% more of the returns than can AIOCS. The more important advantage of PACE, we believe, is the speed with which it was developed—nearly 50 times faster than AIOCS.

We attribute the significant decrease in PACE's development time to MBR's ability to automatically exploit the expert knowledge in a large, previously classified database, thus avoiding exhaustive hand coding of many rules. PACE, in turn, is made possible through the data parallel hardware and programming models of massively parallel supercomputers. The availability of this supercomputing power also allowed us to try many MBR variants and to test each extensively. This would not have been feasible on a slower machine.

The requirement of a pre-existing training database appears to be an added constraint on the domains to which empirical learning models, such as MBR, may be applied. However, it is often the case, as it was for the census, that such databases need to be created regardless in order to adequately test the automated system, and for a range of tasks such large training databases do already exist (e.g., news story classification, medicine, optical character recognition, financial performance). In these cases, MBR can dramatically reduce the effort and cost of developing automated systems with expert performance. Therefore, massively parallel computing and empirical learning systems such as PACE appear to be at the beginning of an important new trend in the way automated classification systems are built. As the gap between the price of MIPS and the price of a knowledge engineer's time widens, this should become more and more evident. ◼

Acknowledgements
Thanks very much to Debra Fuhry, Anand Bodapati, Craig Stanfill, Bor-Chung Chen and Marty Appel for their help in critiquing and preparing this article.

References
1. Aha, D., Kibler, D. and Albert, M. Instance-based learning algorithms. *Machine Learning*, 6, Kluwer Aca-

demic Publishers, Boston, Mass., (1991), 37–66.
2. Anderberg, M. *Cluster Analysis for Applications.* Academic Press, New York, 1973.
3. Appel, M. Automated industry and occupation coding. Development of Statistical Tools Seminar on Development of Statistical Expert Systems (DOSES), Luxembourg., Dec. 1987.
4. Appel, M.V. and Hellerman, E. Census Bureau experiments with automated industry and occupation coding. In *Proceedings American Statistical Association*, (1983), 32–40.
5. Appel, M.V. and Scopp, T. Automated industry and occupation coding. Presented at Development of Statistical Expert Systems (DOSES), Luxembourg, Dec. 1987.
6. Atkeson, C. Roles of knowledge in motor learning. MIT AI Lab Technical Report 942, Sept. 1986.
7. Chen, B.C., Creecy, R.H. and Appel, M.V. On error control in automated industry and occupation coding. In *Proceedings American Statistical Association*, Survey Methods Section, (1991), to appear.
8. Church, K. A stochastic parts program and noun phrase parser for unrestricted text. Unpublished manuscript, AT&T Bell Labs, Murray Hill, N.J., 1986.
9. Church, K. and Gale, W. Enhanced Good-Turing and Cat-Cal: Two new methods for estimating Probabilities of English Bigrams. AT&T Bell Laboratories, 1989.
10. Corbett, J.P. Encoding from Free Word Descriptions. Draft Memorandum, Bureau of the Census, 1972.
11. Cover, T.M. and Hart, P. E. Nearest neighbor pattern classification. *IEEE Trans. Inform. Theor. 13*, (1967), 21–27.
12. Creecy, R.H., Causey, B.D. and Appel, M.V. A bayesian classification approach to automated industry and occupation coding. In *Proceedings of American Statistical Association*, Statistical Computing Section, (1990).
13. Dasrathy, B.V., Ed. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, 1990.
14. Dempster, A.P. A generalization of Bayesian inference. *J. Royal Statistical Soc., Series B.* 30, (1968), 205–247.
15. Fukunaga, K. and Flick, T.E. An optimal global nearest neighbor metric. *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-6, 3 (May 1984), 314–318.
16. Harman, D. How effective is suffixing. *J. Amer. Soc. Inform. Sci. 42*, 1 (1991), 7–15.
17. Hillis, D. *The Connection Machine.* MIT Press, Cambridge, Mass., 1985.
18. Lorigny, J. Quid. A general automatic coding method. *Surv. Method. 14*, 2 (1988), 289–298.
19. Lyberg, L. and Dean, P. International review of approaches to automated industry and occupation coding. Presented at the Advanced Computing in the Social Sciences conference, Williamsburg, Apr. 1987, (1990).
20. O'Reagan, R.T. Computer assigned codes from verbal responses. *Comm. ACM 15*, (1972), 455–459.
21. Pearl, J. *Probabilistic Reasoning in Intelligent Systems.* Morgan-Kaufmann, Los Altos, Calif., 1968.
22. Quinlan, R. Learning efficient classification procedures and their application to chess end games. In R.S. Michalski, J. Carbonell, and T. Mitchell, Eds., *Machine Learning: An Artificial Intelligence Approach.* Tioga Publishing, Los Angeles, Calif. (1983), 463–482.
23. Rumelhart, C., et al. *Parallel Distributed Processing.* MIT Press, Cambridge, Mass. 1986.
24. Shannon, C. and Weaver, W. *The Mathematical Theory of Communication.* University of Illinois Press, Urbana, Ill., 1949.
25. Short, R.D. and Fukunaga, K. The optimal distance measure for nearest neighbor classification. *IEEE Trans. Infor. Theor.*, Vol. IT-27, 5, (Sept. 1981), 622–627.
26. Stanfill, C. and Kahle, B. Parallel free-text search on the Connection Machine System. *Comm. ACM. 29*, 12 (Dec. 1986), 1229–1239.
27. Stanfill, C. and Waltz, D.L. The memory-based reasoning paradigm. In *Proceedings of Case-Based Reasoning Workshop*, Clearwater Beach, FL, (May 1988), 414–424.
28. Stanfill, C. and Waltz, D.L. Toward memory-based reasoning. *Comm. ACM. 29*, 12 (Dec. 1986), 1213–1228.
29. Waltz, D.L. Massively Parallel AI. In *Proceedings of National Conference on AI*, (AAAI '90), Boston, Mass., (Aug. 1990).
30. Waltz, D.L. Memory-based Reasoning. In M.A. Arbib and J.A. Robinson, Eds., *Natural and Artificial Parallel Computation*, The MIT Press, Cambridge, Mass., (1990), 251–276.
31. Wolpert, D. Generalization theory, surface-fitting, and network structures. Ph.D. Thesis, Physics Dept. University of California, Santa Barbara, Winter 1989.
32. Zadeh, L.A. Knowledge representation in fuzzy logic. *IEEE Trans. Knowl. Data Eng. 1*, 1 (1989).

CR Categories and Subject Descriptors: D.2.9 [**Management**]: Productivity; D.2.m [**Miscellaneous**]: Rapid Prototyping; H.3.2 [**Information Storage**]: Record Classification; I.2.1 [**Applications and Expert Systems**]: Natural Language Interfaces; I.2.6 [**Learning**]: Concept Learning; I.2.7 [**Natural Language Processing**]: Text Analysis; I.5.4 [**Applications**]: Text Processing; J.1 [**Administration Data Processing**]; Government.

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Automated system building, case-based reasoning, empirical learning, memory-based reasoning, textual database classification.

About the Authors

ROBERT H. CREECY is director of the Integrated Statistical Laboratory of the Statistical Research Division of the U.S. Bureau of the Census. He is responsible for modernizing the Census Bureau's research computer systems and directing a research program in statistical computing. **Author's Present Address:** U.S. Bureau of the Census, Statistical Research Division - 3215-4, Washington, D.C. 20233, creecy@midgard.umd.edu.

BRIJ M. MASAND is a research scientist at Thinking Machines Corp. His current research interests include exploring principles of massively parallel AI, automatic programming techniques, parallel enumeration algorithms. **Author's Present Address:** Thinking Machines Corp., 245 First St., Cambridge, MA 02142-1214, brij@think.com.

STEPHEN J. SMITH is a research scientist at Thinking Machines Corp., and is also graduate student at Harvard's Department of Applied Sciences. His current research interests include