Automated coding of AD-417 forms using case-based reasoning
by Venkatesh Thati

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
The AD-417 forms are used by researchers to submit their proposals to the United States Department of Agriculture. The researchers are required to enter codes belonging to various categories, that are pertinent to their project, in the AD-417 form. The Classifier system has been developed to reduce the time and increase the accuracy in filling out these forms. The basis for this system, a machine learning technique called Case-Based Reasoning, is described along with a discussion of its advantages and disadvantages. The components of the Classifier are described and results that indicate its performance are provided.

AUTOMATED CODING OF AD-417 FORMS USING CASE-BASED REASONING

by

Venkatesh Thati

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

January, 1995

APPROVAL

of a thesis submitted by

Venkatesh Thati

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

1/19/95
Date

John T. Paton
Chairperson, Graduate Committee

Approved for the Major Department

1/19/95
Date

J Denbigh Starkey
Head, Major Department

Approved for the College of Graduate Studies

1/25/95
Date

R L Brown
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature T. VanLanteh
Date January 19, 1995

TABLE OF CONTENTS

TABLE OF CONTENTS - Continued

Page

List of Figures

ABSTRACT

The AD-417 forms are used by researchers to submit their proposals to the United States Department of Agriculture. The researchers are required to enter codes belonging to various categories, that are pertinent to their project, in the AD-417 form. The Classifier system has been developed to reduce the time and increase the accuracy in filling out these forms. The basis for this system, a machine learning technique called Case-Based Reasoning, is described along with a discussion of its advantages and disadvantages. The components of the Classifier are described and results that indicate its performance are provided.

Case-Based Reasoning

Case-based reasoning is a technique which attempts to model the concept of experience. It is based on the psychological theories on how experience contributes towards understanding and solving problems.

From the perspective of case-based reasoning, the best person to solve a problem is not necessarily the smartest person, it is the person with the most experience. It considers a person who has solved a similar problem before to be the one best able to solve the problem now. If we had two computer systems to solve a certain kind of problem, and if we could make one of them acquire "experience" as it goes about the process of solving problems, there is a good chance that given a new problem the system with experience will perform better than the other.

A case can be defined as a description of a problem, an attempt at a solution and an outcome of the effort[4]. Cases can be pretty complex, encapsulating many facts and relationships in a specific context. A case base would then be a collection of cases, i.e., an accumulated body of problem solving experiences. As the cases increase in number and diversity, so does the usefulness of the case base.

Case-based reasoning can be formally defined as a method of solving new problems by remembering old problems and adapting their solutions. It comprises a memory model for representing, indexing and organizing past cases and a process model for retrieving and modifying old cases and assimilating

new ones. This method combines reasoning with learning. It spans the whole reasoning cycle. A situation is encountered. Old situations are used to understand it. Then the new situation is inserted into the case base alongside the others to be used another time.

The key to the method of case-based reasoning is remembering[5]. Remembering has two parts,

- Integrating cases into memory when they happen
- Recalling the appropriate cases in later situations

This related set of issues is called the Indexing Problem. In broad terms, it means finding in the case base the case closest to a new one. In narrower terms one can think of it as a two-part problem,

- The first part involves assigning indices or labels to cases, when putting them into memory that describe the situations to which they are applicable, so that they can be recalled later.
- The second part involves elaborating a new situation in enough detail so that the indices it would have had if it were already in the case base are identified.

The technique of case-based reasoning is appropriate for many types of applications. Potential domains can be identified by the degree to which they meet the following criteria,

- First, when experts solve a problem in the domain if they refer to specific cases that helped illuminate, describe, classify or solve the current problem then case-based reasoning would be suitable for this domain.

- Second, a case-based reasoning system is only useful if similar problems are seen again and again. If each problem and its solution is unique there is little value in storing it.

- Third, case-based reasoning systems, neural networks and expert systems have somewhat overlapping capabilities. It is important to determine which is most appropriate. Case-based reasoning is appropriate when there is little understanding of underlying causal relationships in the domain and when weak explanations characterize the selection of solutions.

- Fourth, cases should be relatively compact and be defined in a manner that makes the identification of indices which can be used for matching relatively simple. For example, historical cases of corporate development are probably too vaguely described. So they won't be of much use.

- Fifth, an application like a help desk is an ideal candidate for applying case-based reasoning because there are many people at work, turnover is high and expertise is distributed. On the other hand if we had a stable technology all the problems of which can be solved by a single person and if the continuing presence of this person can be relied on, then case-based reasoning would be less cost effective.

- Sixth, it should be cost-effective to obtain and store the cases.

Based on these criteria some of the potential areas to apply case-based reasoning are help desks, medical diagnosis, scheduling etc.

Advantages of Case-based reasoning

Some of the benefits of case-based reasoning are,

- A case base becomes useful with the first case. It is not necessary to wait until all the cases have been developed before the system can be used. As cases are added, the system becomes more useful.

- A case base captures knowledge easily. The structure of cases is much less constrained than rules are. There is no need for complex interrelations between cases as there are between rules. Consequently case bases come on-line faster and they stay on-line even as cases are being altered or eliminated. Learning is incremental.

- Case bases are more understandable. The basic organization and functioning of case-based reasoning systems is logical and easy to follow. People feel better about using systems that are more understandable.

- Case-based reasoning augments human capabilities. A case-based reasoning system can track more cases than a person can, and as with other computer systems it thoroughly and neutrally evaluates all possibilities before making a recommendation.

- Case-based reasoning systems facilitate the incorporation of new knowledge. New cases can be added rapidly to a case-based reasoning system, thereby increasing its usefulness.

Disadvantages of Case-based reasoning

Some of the drawbacks of case-based reasoning are,

- It is a form of supervised learning, in the sense that during training it requires a teacher in the form of an expert to enter in the correct solution when the system comes up with unsatisfactory solutions.
- Case-based reasoning does not actually find the reasons behind the relationships it reveals, it simply deals on a case-by-case level.
- As the case base grows, due to the large number of cases being processed certain inconsistencies may crop up in the case base which can easily go unnoticed.
- The process of situation assessment, that is the extent to which a reasoner can interpret a new case and determine which kinds of cases are most likely to be useful, needs to be further improved.

The Classifier System

The objective behind developing the Classifier system was to reduce the time taken and increase the accuracy of coding AD-417 forms. These forms are used by researchers to submit their proposals to the USDA. An AD-417 form is shown on the next page. Researchers are required to describe their project and enter in a set of codes pertinent to it. The different categories of codes that need to be entered are,

- Research Problem Area codes (RPA)
- Activity codes
- Commodity codes
- Subcommodity codes
- Science codes
- Special codes

These codes are described in the *Manual of Classification of Agricultural and Forestry Research*[2]. The researchers go through the descriptions and pick the codes relevant to their project. The Classifier system automates this process. Based on the description of the project, it determines the relevant codes and suggests them.

The three main components of the Classifier are,

- a natural language processing component
- files containing codes and their associated noun phrases, one for each category of codes.

====== Simulated Form AD-417 ======
RESEARCH WORKUNIT/PROJECT DESCRIPTION -- CLASSIFICATION OF RESEARCH

Accession No.	Agency Identification	Work Unit/Project No.	Date 28 May 1993 Status
1.	2.CSRS 3.TEX 4.	5.TEX08241	6. A New Project

32.BASIC RESEARCH 0% |33.APPLIED 80% |34.DEVELOPMENT 20% || 35.FORESTRY 0%

PRIMARY CLASSIFICATION

	Research Problem Area	Activity	Commodity	Science	% Effort
36.	112	4300	0700	2630	30
37.	316	5300	4200	2630	50
38.	313	5300	3000	2630	20
39.					
40.					
41.					
42.					
43.					
44.					
45.					

SUB-COMMODITY CLASSIFICATION

	code	%
46.	3040	20
47.	0799	30
48.		
49.		
50.		
51.		
52.		
53.		
54.		
55.		

SPECIAL CLASSIFICATION

code	%	code	%	code	%	code	%	code	%
56. K2631	50	60.		64.		68.		72.	
57. K2637	25	61.		65.		69.		73.	
58. K2633	25	62.		66.		70.		74.	
59.		63.		67.		71.		75.	

Figure 1: AD-417 form

- a matching and ranking component

Natural Language Component

Natural language has proven to be an insidiously difficult area to solve. Researchers have identified at least four components that might be involved in processing natural language[6].

- Syntactic Analysis involves examining the structure of a sentence to see whether it is grammatically correct.
- Semantic Analysis involves examining the meaning of a sentence to see whether it makes sense.
- Discourse Integration involves interpreting ambiguous terms by using the context of a sentence.
- Pragmatic Analysis involves deciding what the actual intent of a sentence is, as in the case of distinguishing between a request and a question.

Of these four areas, the most progress has been made in syntactic analysis.

In the Classifier, the natural language component extracts noun phrases from the text in the title, keywords and objectives of the research projects. In order to do this it uses syntactic information about the structure of sentences in the English language and a lexicon of common words. The lexicon is enhanced by adding frequently occurring agricultural terms to it.

Initially a sentence is broken down into its component words and the type of each word(noun, adjective etc.) is determined using the lexicon. Then the sequence of words is examined to see if it contains any noun phrases the structure of which is defined by the following finite state automaton,

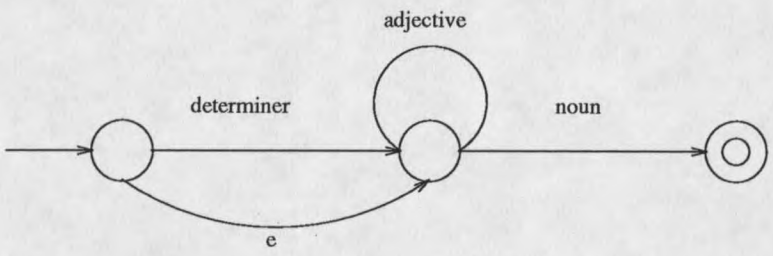

Figure 2: Finite State Automaton to recognize noun phrases

If the sequence of words from a given sentence does not generate any noun phrases based on the actions of the automaton, it means that the noun phrases in that sentence could not be identified.

Files containing noun phrases

The natural language component is used to extract the noun phrases from the description of each code, given in the *Manual of Classification of Agricultural and Forestry Research*[2]. Separate files are created for each category of codes. These files were later edited by an expert in order to increase their usefulness by getting rid of noun phrases that were too general and those that were not relevant to a particular code. These files containing noun phrases extracted from the descriptions in the manual constitute the primary database of noun phrases.

In order to improve the accuracy of the Classifier, another set of files containing codes and associated noun phrases, from a set of 490 projects which had been submitted by researchers and had been checked for correctness, were created. These files constitute the secondary database of noun phrases. Since the noun phrases in these files are extracted from actual projects, they contain descriptions of specific methods, tools, etc., which might not be present in the description given in the manual.

Matching and Ranking Component

This component of the Classifier is responsible for finding similarities between noun phrases from the description of a project and the noun phrases associated with a particular code. It also computes a measure of the similarity, which is used to sort the codes whose noun phrases were similar to the ones from the project, and assign ranks to them.

A noun phrase is considered to match another, only if both have the same noun. Once the nouns match, the adjectives from the two phrases are matched against each other. Here the position of the adjective in the noun phrases is not given any importance. The second adjective of the first phrase may match the first one of the second and vice-versa.

Based on predefined weights for a noun match and an adjective match, once all the noun phrases of a particular code have been matched with the noun phrases of the project a value called the **match strength** is computed. This value is a measure of the relevance of that particular code to the project. Since we have two sets of noun phrases, the primary and secondary, we distinguish the matching of the project noun phrases with each of them as follows:

Primary Classification

The process of matching the project noun phrases with the primary noun phrases database is called primary classification. An example of primary classification is shown below,

Match weights

Noun = 5.0 Adjective = 1.0

Project noun phrase

loose wet soil

Primary noun phrases

soil

wet soil

loose soil

Primary match strengths

soil = 5.0 x 1 = 5.0

wet soil = 5.0 x 1 + 1.0 x 1 = 6.0

loose soil = 5.0 x 1+ 1.0 x 1 = 6.0

Therefore the total primary match strength for the given project noun phrase = 5.0 + 6.0 + 6.0 = 17.0.

In the above example the same noun causes the project noun phrase to be matched with more than one noun phrase from the database, and the weight for a noun match is factored into the match strength for each of the noun phrases. Thus a noun match is given a lot of weight in suggesting codes.

Secondary Classification

Since the noun phrases contained in the secondary database are more specific to a particular code, if a project noun phrase matches a secondary noun phrase there is a greater probability of that code being relevant to the project. It was found that assigning noun and adjective match weights that are greater than the weights used in primary classification, did indeed improve the performance of the Classifier. An example of secondary classification is shown below,

Match weights

Noun = 10.0 Adjective = 2.0

Project noun phrase

loose wet soil

Secondary noun phrases

soil

wet soil

loose soil

Secondary match strengths

soil = 10.0 x 1 = 10.0

wet soil = 10.0 x 1 + 2.0 x 1 = 12.0

loose soil = 10.0 x 1+ 2.0 x 1 = 12.0

Therefore the total secondary match strength for the given project noun phrase = 10.0 + 12.0 + 12.0 = 34.0.

From the primary and secondary match strengths a cumulative match strength is computed by adding the two. After all the codes of a particular category are matched against the noun phrases of a project and the cumulative match strength for each of them computed, they are ranked based on it. The code with the highest match strength is assigned the rank 1, the next one 2 and so on.

Functioning of the Classifier

The Classifier comes up with a possible set of codes associated with an input project in the following manner,

Extraction of noun phrases

The project is described by means of the title, objectives and approach fields. The natural language component is used to extract the noun phrases from each of these fields individually. These are then linked together and jointly constitute the project noun phrases.

Determining the RPA codes

For each of the RPA codes, a code and its associated set of noun phrases are read in from both the primary and secondary noun phrase databases. The project noun phrases are matched first with the primary noun phrases and then with the secondary ones. Based on the primary and secondary match weights for a noun and adjective, the primary match strength, secondary match strength and cumulative match strength (the sum of the two) is computed. This process is repeated for all the RPA codes. Then the RPA codes are sorted based on match strength and displayed to the user with the code having the greatest match strength at the top.

Determining the Activity and Commodity codes

There is a relationship between RPA codes and both activity and commodity codes which is clearly defined in the manual. The description for RPA code 102 which illustrates this relationship follows,

RPA 102. SOIL, PLANT, WATER, NUTRIENT RELATIONSHIPS

This problem area is concerned with the chemical and physical nature of interrelationships among soils, plants, water, and nutrients. The objective is to improve, maintain, or restore the inherent production capability of soils. Areas of research include:

(a) Factors which limit root development of plants.

(b) Development of practical methods for ameliorating unfavorable conditions, such as tillage pans, nutrient deficiencies, and improper air-water relationships.

(c) Ways to maintain and improve soil structure by soil amendments and by soil, crop, tillage, and management systems.

(d) The effect of physical, chemical, and biological properties of soils on soil structure, resistance to erosion, availability of plant nutrients, and the general environment for plant roots.

Classification Guidelines:

Activities:

4300 Resource development, conservation, and management

4810 Protection against fire

4820 Protection against flood

4840 Protection against climatic extremes (frost, hail, wind, drought, etc.)

5000 Improving biological efficiency of plants and animals

Commodities, etc.:

0100 Soil and land

0200 Water

0300 Watersheds and river basins

0600, 0700, 0900-2100, 2300-2800 (See Commodity Classification - Table C)

Figure 3: Description of RPA code 102

So there are only certain activity codes that can occur in conjunction with a particular RPA code. This is true for the commodity codes too. These relationships are represented in the Classifier by a file which has an RPA code followed by all the activity and commodity codes that could possibly occur with it.

After the list of suggested RPA codes are displayed, the user selects some of them as the ones that need to be associated with the project. For each of these codes, the related activity and commodity codes are determined.

If an activity or commodity code is related to more than one of the selected RPA codes it indicates that there is a greater likelihood of that particular code being associated with the project. To represent and use this information in the Classifier a **relation strength** is associated with the activity and commodity codes. This indicates the number of selected RPA codes to which an activity or commodity code is related. Once the related activity and commodity codes are determined, their associated noun phrases are read in and matched against the project noun phrases. The match strength for each of them is computed. Now the relation strength is added to the match strength and the codes are sorted based on this value. The activity and commodity codes are then displayed separately to the user, keeping the one having the greatest sum of relation strength and match strength at the top.

Determining the Subcommodity codes

There is a specific set of subcommodity codes, that can occur in conjunction with a particular commodity code. For example the only subcommodity codes that can occur in conjunction with the commodity code 1700 are the ones in the range 1701 to 1799.

Based on the commodity codes selected by the user, the relevant sub-commodity codes for each of them are determined. The process of reading in their associated noun phrases, matching and computing the match strength is repeated and they are displayed to the user keeping the code with the greatest match strength at the top.

Determining the Science codes

The science codes have very minimal descriptions associated with them in the manual. This means that the files containing noun phrases associated with these codes have little or no information in them and thus the method of determining codes by matching noun phrases is not feasible.

It was observed that the descriptions of the science codes indicated relationships with certain RPA codes. Due to this observation files capturing the relationship between RPA codes and science codes were created. These files were later augmented with the relationships used by researchers in projects submitted to the USDA and edited by an expert to enhance their usefulness. This information is similar to the file indicating the relationship between RPA codes and activity and commodity codes, with the relationship in this case being between RPA and science codes.

Determining the Special codes

The manual specifies that the special codes should be determined independently of the other codes. That is, there are no relations between the other categories of codes and the special codes that can be exploited to determine the relevant ones.

The special codes are determined using a method similar to the one employed to determine RPA codes. Primary and secondary noun phrase databases were created for each of the special codes. The noun phrases from a project are matched with the noun phrases from the databases. The match strength for each of the codes is computed and they are displayed to the user after sorting them based on it.

Results

The results described below are snapshots of the Classifier system at various stages of its development, and reflect the effect different strategies had on its performance. The test data consisted of three sets of projects containing 490, 686 and 563 projects respectively. Each project has the title, objectives and approach fields and also the codes from each category that need to be associated with it. The Classifier is given the description of the project and then the accuracy of the suggested codes is determined using the codes associated with the project.

Accuracy of the Classifier

The accuracy of the Classifier is determined by computing a value known as the **hit rank accuracy**. This value denotes the average position of a code that is indeed pertinent to the project, in the list of suggested codes. The lesser the hit rank accuracy, the better is the performance of the Classifier as this means that the correct codes to be associated with a project occur nearer the top of the suggested list.

The method of computing the hit rank accuracy is best illustrated by an example. Consider a project whose correct RPA codes are 101 and 102. Now, suppose that in the list of RPA codes suggested by the Classifier 101 and 102 have ranks of 5 and 7, i.e., they occur at the 5th and 7th positions in the list.

Now the hit rank accuracy of the suggested RPA codes for this project is determined as follows. If the Classifier had been absolutely accurate, it would have suggested codes 101 and 102 at positions 1 and 2 or vice-versa.

So the ideal sum of their ranks would be equal to 1+2=3. But the actual sum of their ranks as per the list suggested by the Classifier is 5+7=12. The difference between the actual sum of the ranks , 12, and the ideal sum of the ranks, 3, is equal to 12-3=9. The hit rank accuracy is computed from this value by dividing it with the number of codes which in this example is 2. So the hit rank accuracy for the RPA codes is 9/2=4.5.

The hit rank accuracy for each of the different categories of codes is determined in a similar manner.

Using primary noun phrases and matching only nouns

In the initial version of the Classifier, only the nouns in the phrases were matched. The other components of a phrase like the determiner and adjective were not used. The weight of a noun match was set to 1.0. Three different methods for suggesting RPA codes were explored,

- Suggesting an RPA code if and only if it has a match strength of at least three.
- Suggesting an RPA code if and only if it has a match strength of at least two.
- Suggesting an RPA code if it has a match strength greater than or equal to zero.

In the first and second methods at least three and two project noun phrases, respectively, must match the primary noun phrases of a code for it to be suggested. In the third method a code is suggested even if none of the project noun phrases matched its primary noun phrases. Graphs showing the hit rank accuracy and average number of omissions obtained using each of the above methods are as follows,

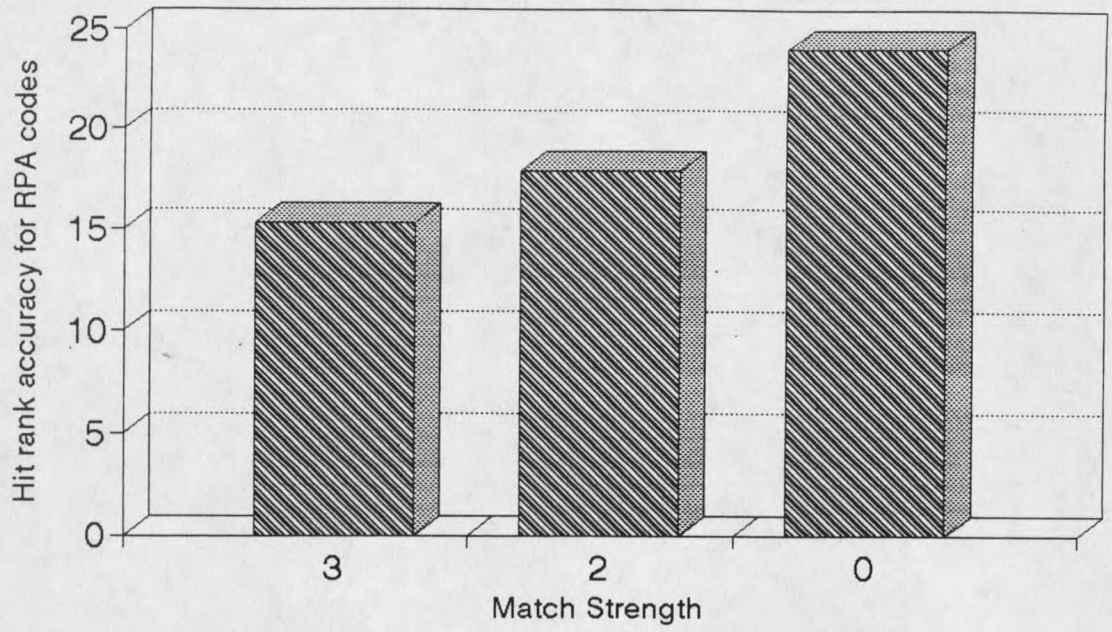

Figure 4: Accuracy matching only nouns

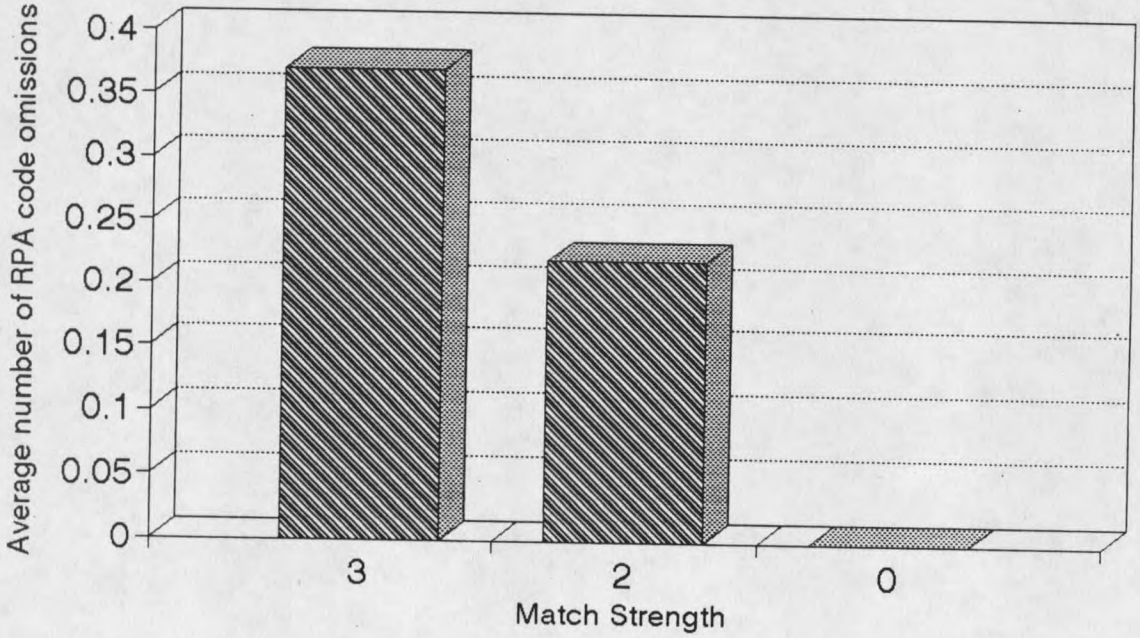

Figure 5: Average number of omissions matching only nouns

The average number of omissions is obtained by dividing the number of omitted codes for a set of projects, i.e., those codes that were present in the projects but were not suggested by the Classifier, by the total number of projects. These results show that attempts to reduce the number of omissions by weakening the matching conditions, increase the average position of the desired codes in the suggested list hurting the accuracy of the Classifier.

Using primary noun phrases and matching phrases

In order to decrease the average position of the desired codes, while at the same time keeping the number of omissions low, both nouns and adjectives were matched. With weights of 5.0 and 1.0 for noun and adjective matches, the same three methods used above were explored,

- Suggesting an RPA code if and only if it has a match strength of at least 15.
- Suggesting an RPA code if and only if it has a match strength of at least 10.
- Suggesting an RPA code if it has a match strength greater than or equal to zero.

From the following graphs which show the hit rank accuracy and average number of omissions for each of the above methods, it can be seen that while the average number of omissions remains the same there is an improvement in the hit rank accuracy. So an improvement in the performance of the Classifier is brought about by matching adjectives in addition to nouns.

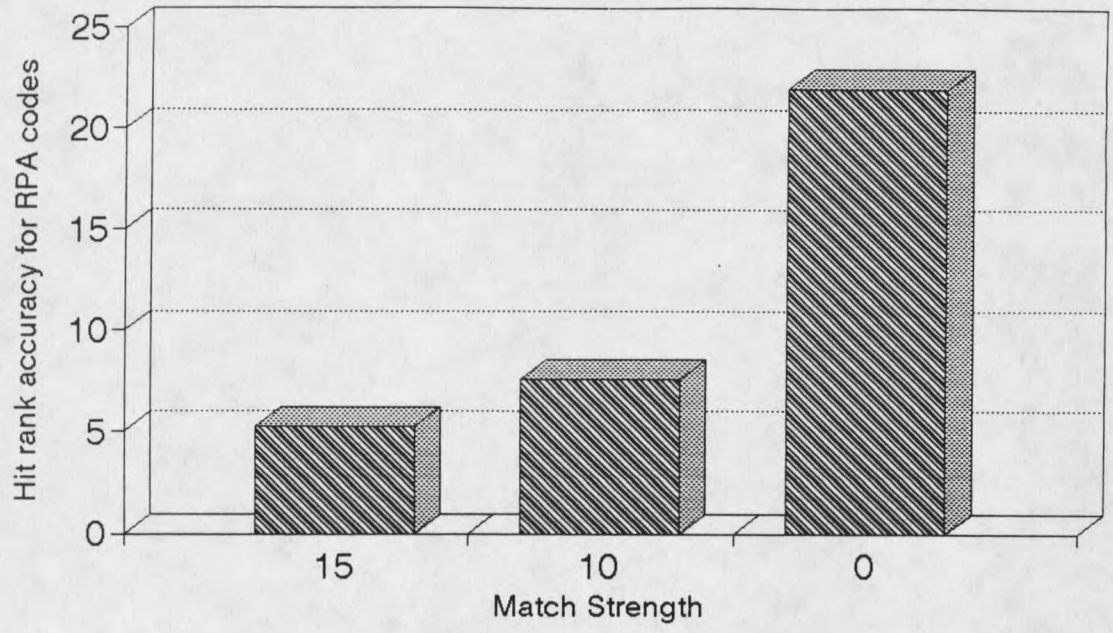

Figure 6: Accuracy matching noun phrases

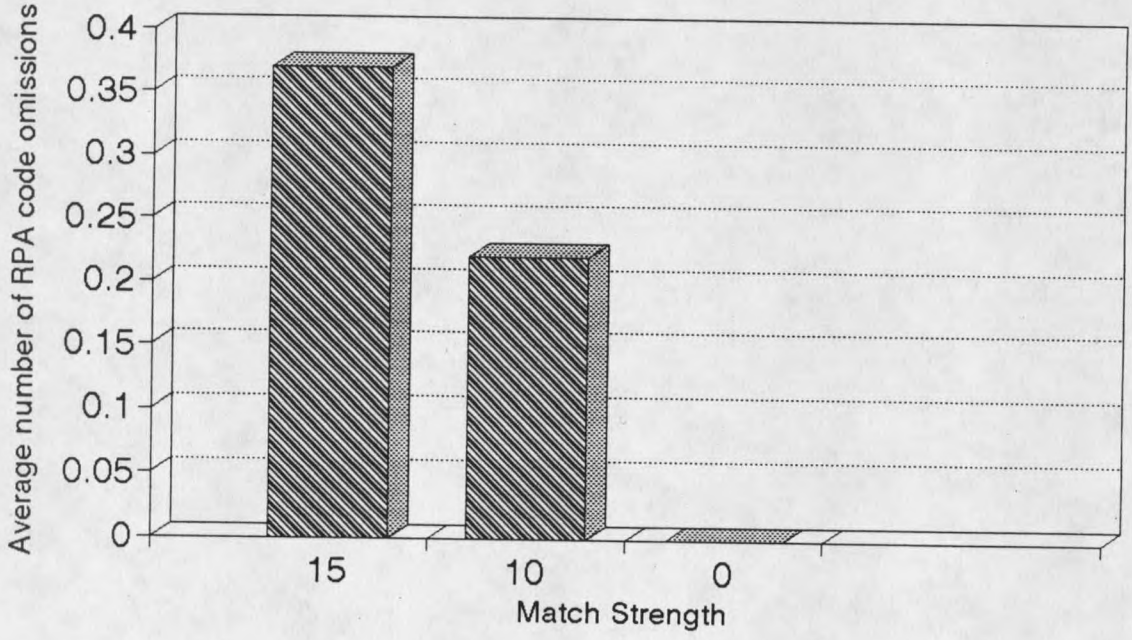

Figure 7: Average number of omissions matching noun phrases

Using primary and secondary noun phrases and matching phrases

In order to further increase the accuracy of the Classifier, the secondary noun phrase files were created and used in matching with the project noun phrases. With weights of 5.0 and 1.0 for noun and adjective matches from the primary noun phrases and 10.0 and 2.0 for the noun and adjective matches from the secondary noun phrases,

Hit rank accuracy for the RPA codes = 16.21

Average number of omissions = 0.0

The minimum match strength for a code to be suggested is set equal to 0 so that no codes are omitted. From this it can be seen that improvements in accuracy were achieved first, by using noun phrases instead of just nouns and second, by using secondary noun phrases.

The accuracy of the Classifier in suggesting the other categories of codes is as follows,

Hit rank accuracy for the activity codes = 2.92, given correct RPA code.

Hit rank accuracy for the commodity codes = 6.77, given correct RPA code.

Hit rank accuracy for the subcommodity codes = 9.24, given correct commodity code.

Hit rank accuracy for the science codes = 8.17, given correct RPA code.

Hit rank accuracy for the special codes = 13.11

Future Directions

A few of the areas of the Classifier system that can be modified in an effort to further enhance its accuracy are as follows,

Exploring other methods of assigning weights

In the present system, the match strength for all matching noun phrases is computed in the same manner. It may be that certain noun phrases indicate a particular code very strongly. This information regarding varying degrees of relevance of noun phrases to a particular code needs to be captured in the Classifier. One way of doing this would be to associate numerical values with certain key noun phrases, in both the primary and secondary noun phrase files. This would require the assistance of an expert. The numerical values associated with a noun phrase would indicate how much to weigh a noun match and adjective match for that particular noun phrase. These values could be greater or less than the standard noun and adjective match weights, depending on whether the noun phrase is more or less relevant to that particular code.

Enhance the primary and secondary noun phrases

An attempt can be made to further increase the usefulness of the files containing codes and associated noun phrases, extracted from both the manual and from a set of projects.

The primary noun phrases database can be further edited to remove certain irrelevant noun phrases that may still be present and in some cases certain new noun phrases may be added to the ones associated with a particular code.

The secondary noun phrases database could be improved by extracting the noun phrases and their associated codes, from other sets of actual projects and adding the relevant ones to the already existing database.

Exploring other standard weighting combinations

The weights for noun and adjective matches in both primary and secondary classification can be varied in an effort to find values that improve the performance of the Classifier.

Automated knowledge acquisition

The Classifier can be modified to incorporate case-based learning. That is, the primary and secondary noun phrases can be enhanced as the system classifies more and more projects.

This can be either fully automated or involve interaction with an expert. In the fully automated system, the Classifier periodically updates the noun phrase databases with useful noun phrases from the projects. In a semi-automated system, an expert may be asked to verify the validity of noun phrases before adding them to the noun phrase databases.

References

[1] Manual for Preparing CRIS Forms for State and Non-Federal Institutions, October, 1993

[2] Manual of Classification of Agricultural and Forestry Research, Revision 4, February 1982

[3] Fritz H. Grupe, *Case-Based Reasoning, Applying Past Experience to New Problems*, Information Systems Management, Spring 1993

[4] Stephen Slade, *Case-Based Reasoning: A Research Paradigm*, AI Magazine, Spring 1991

[5] Special issue on Case-Based Reasoning, *Machine Learning*, Volume 10, Number 3, March 1993

[6] Elaine Rich, Kevin Knight *Artificial Intelligence*, McGraw-Hill, 1991