Survey Methodology, December 1988
Vol. 14, No. 2, pp. 299–307
Statistics Canada

ACTR
A Generalized Automated Coding System

M.J. WENZOWSKI[1]

ABSTRACT

A generalized implementation of a method for performing automated coding is described. Traditionally, coding has been performed manually by specially trained personnel, but recently computerized systems have appeared which either eliminate or substantially reduce the need for manual coding. Typically, such systems are limited in use to those applications for which they were originally designed. The system presented here may be used by any application to perform coding of English or French text using any classification scheme.

KEY WORDS: Automated coding; Classification; Text searching.

1. INTRODUCTION

Automated coding refers to the process by which text is machine analysed in order to assign it a classification, or code. To be practical, automated coding systems must be capable of coping with such problems as: rearranged words, plural vs singular forms, missing words, extraneous words, spelling variations, synonyms, abbreviations, inconsistent hyphenation and variable punctuation and syntax. In addition, in searching a text database for a match, they should be capable of determining the closest match when no identical match can be found.

Generalized systems provide all of the features required, packaged within an easy to use, flexible, and efficient framework. To use a generalized system for a particular application, no development or conversion effort is needed to tailor it to the application specific requirements. As well, no application sponsored support for the maintenance of a generalized system is necessary, since the package is supported and maintained by a central agency.

ACTR (an acronym for: Automated Coding by Text Recognition) employs techniques similar to those employed in other automated coding systems currently in production at Statistics Canada (Landry and Pidcock 1984), but is unique in that it has been generalized to allow it to be used by any application to assign codes based on the input of English or French text according to any classification scheme.

The methods which ACTR uses to perform automated coding are based on techniques which were originally developed at the U.S. Bureau of the Census (Appel and Hellerman 1983). Basically stated, the method consists of searching through a collection of text previously associated with correct codes. If the subject text is successfully located, the associated code is returned and the process ends. Otherwise, the search continues, but uses an algorithm to locate the closest match, and subsequently assign its associated code.

[1] M.J. Wenzowski, Research and General Systems, Statistics Canada, Room 2306, Main Building, Ottawa, Ontario, K1A 0T6.

2. USING ACTR

To use ACTR in an automated coding application users first need to define the text and associated codes which they intend to use as a standard for matching. While there are many sources for this information, the best is a set of text which is representative of the text which will most likely be encountered in a matching run. For a survey, this generally means the responses and manually assigned codes from a previously completed survey. Although great care should be taken to ensure that the correct codes have been assigned, the text should be left as is, complete with spelling, grammar and syntax errors, since in this form it is most representative of the text which will be encountered in subsequent surveys.

After having defined a file of text and correctly assigned codes, they must be loaded into a matching database. ACTR provides the software required to perform this task and so automatically transforms the file into a matching database.

ACTR has been designed to allow an iterative approach to developing an automated coding application. Accordingly, text and codes can be added, changed or deleted at any time during the life of the application. In addition, the parsing strategy (discussed in detail below) can be altered at any time. Thus, users are presented with a software framework which, through cycles of database updates and matching runs, will allow for as many iterations as is necessary to obtain the matching quality desired. Users are encouraged to use ACTR in this manner, since ultimately it leads to higher quality and more economical coding operations.

3. PRINCIPLES OF OPERATION

In the case of a human being performing a coding operation, the similarity between occupations described as "Computer Programmer" and "Programming Computers" is so great that they would generally be judged as identical. However intuitive this reasoning may seem, computer systems in general would rate the two as unequal. Unfortunately, natural language (for example, English or French) frequently provides a large number of ways to express the same meaning. So, for a computer based system to be able to cope with this variance, there must be some means by which a degree of similarity can be determined.

This is the essence of ACTR: text is rated according to how similar it is to some other text. In the preceding example, ACTR treats the two occupation descriptions as identical since, after suffixes are truncated, double letters are removed and word order is ignored, both phrases become "Comput Program" and as such are clearly equal.

The steps employed in reducing the above phrases to a standard form are part of what is known in ACTR as the parsing strategy. ACTR's parsing strategy is entirely user controlled and may be changed at any time during the life of an application. Users exercise control over the parsing strategy employed in their applications by supplying the data which is to be used to direct the process. This means that all steps are entirely controlled by the user, even to the extent of allowing a step to be skipped.

The Parsing Strategy

Parsing is the ACTR process which is responsible for the reduction of phrases to a standard form. Ideally, the resulting form should be such that any two phrases with the same words will be identical in their ACTR representation regardless of their syntactical and grammatical differences. Returning to the previous example, the two phrases "Computer Programmer" and "Programming Computers" when properly parsed, should ideally result in a set of identical words for each phrase. For example, both phrases could be reduced to "Comput Program".

The parsing process employed may involve the reduction of plural forms, elimination of trivial words, removal of suffixes and/or a number of other steps. Although the order of the parsing steps applied is fixed by ACTR, users control how, if at all, each step is executed. For further information on the order of parsing, the interested reader should consult Connor, Salloum and Wenzowski (1988).

Basically, the parsing process can be thought of as having the following two major subcomponents:

1. TEXT PROCESSING. In this stage of parsing, the text supplied is processed as a continuous stream of characters. Although one may think of the text as containing words, spaces and punctuation, none of these is given any special consideration at this point in the parse. This view is necessary in order to allow for the recognition of particular character strings exactly as they occur *in situ*.
2. WORD PROCESSING. When this stage of the parse begins, the text has already been broken down into words and so further processing is performed on a word by word basis. This view is necessary since a large amount of text standardization occurs on the basis of defined words.

Text Processing

As already discussed, these steps are performed regardless of context. Thus, the following steps are performed on a character by character basis.

Exclusion Clauses: Exclusion clauses are ignored in matching, but are used in database updating to indicate the intention of allowing controlled duplication of phrases. By default, ACTR will not allow identical phrases to be loaded into a matching database.

By providing a means of controlling duplication, users are able to load phrases which could have more than one code assigned, even though they are identical after having been parsed. Although not used in matching, exclusion clauses are stored along with the phrase in the matching database and can subsequently be used to manually resolve multiple matches.

The syntax of an exclusion clause is defined entirely by the user. Both beginning and terminating strings must be provided. These and any information enclosed by them are ignored during matching.

As an example, consider an exclusion clause syntax defined with a beginning string of "(Except" and a terminating string of ")". With this in place, the two phrases "Computer Programming (Except As An Employee)" and "Computer Programming (Except As Self-Employed)" could co-exist in the matching database, even though their ACTR representations are identical. Subsequently, if a match for "Computer Programmer" is requested, both of these phrases would be returned. Since exclusion clauses are stored along with the original phrase text, they can be displayed to a reviewer, who could then manually resolve the match.

Deletion Strings: If any deletion string supplied by the user is found in any position in a phrase, ACTR will remove it from consideration before continuing the parse.

As an example, in English processing, this is a way in which the apostrophe can be removed. For example, the two phrases "Electrician's Apprentice" and "Apprentice Electrician" would become identical with the removal of the apostrophe.

Note that if this step were not performed, the apostrophe would most likely be used as a word delimiter. This would yield three words for the first phrase and two for the second, of which only one word would be common to both.

Replacement Strings: This facility is most useful for standardizing abbreviations. This is desirable since abbreviations commonly include characters which, although useful to the abbreviation, would be viewed as word separators at a later stage in the parse. If this were allowed to happen, information loss would most likely occur.

As an example, if the string "T.V." was defined with a replacement value of: "Television" then any occurrence of the original string would be translated to the replacement value before continuing the parse.

Note that if this step were not performed, the result of parsing "T.V." would most likely be the two letters "T" and "V". This is clearly undesirable, since the meaning of the abbreviation has been completely lost.

Word Characters: ACTR defines a word as any contiguous sequence of characters in a phrase which are all members of the set of characters contained in the word character list. Any characters not in this list will be used as word delimiters and will be dropped from further consideration.

Typically, the set of word characters used contains all of the letters of the alphabet and all of the numeric characters. With this in place, a phrase of "Farmer/Fisherman" will result in two words, since "/" is not a word character and is therefore used as a word delimiter.

Word Processing

At this point, ACTR begins to treat the text as a collection of words. Thus, the following processing steps are applied on a word by word basis.

Hyphenated Words: Any hyphenated words supplied are replaced by the subtitute word(s) also provided. This feature is very useful in providing for the recognition of words and word groups which are inconsistently hyphenated.

As an example, if the user defines "Take-Out" as a hyphenated word with a substitute word of "Takeout" then this substitution will be made. If, on the other hand, this definition had not been made, then two words would result if the hyphen was not a word character.

Illegal Word Characters: If any of the strings supplied are found to exist in any word in any position, then that entire word is removed from further consideration.

As an example, some applications use this feature to eliminate words which contain numeric characters. So, if the set of numeric digits was given as illegal word characters, then a word like "DEPT716A" would be removed from further consideration.

Replacement Words: This feature provides a synonym capability in order to ensure that two dissimilar words will be recognized for matching purposes. This can also be useful to overcome commonly occurring spelling mistakes.

As an example, if the phrases "Automobile Repairs" and "Car Repairs" were processed with the word "Car" given as a replacement word for "Automobile" then the two phrases would be made identical.

Double Words: This feature forces ACTR to consider not only the occurrence of the two word grouping, but their order as well. This can be useful to overcome inconsistencies in word spellings and also to preserve word order.

As an example, consider the phrase "Take Out Restaurant". Although this would yield three perfectly acceptable words, the words "Take" and "Out" would not match to either of "Takeout" or "Take-Out". However, if a double word combination of "Take Out" was defined with a replacement of "Takeout" then the first case in the example given is addressed.

We are presented here with an example of how steps in the parsing strategy can be used together. If the hyphenated word example given above was also entered, then all of the hyphenated, double word, and single word cases would match.

Trivial Words: If any word in this set is encountered in the course of parsing, then it will be removed from further consideration.

As an example, if the set of trivial words contained "A", "Am" and "I", and the two phrases "I Am A Computer Programmer" and "Computer Programmer" were encountered, then the phrases would match.

Suffixes: At this point, words are scanned right to left looking for the longest defined suffix such that the remaining word, after the suffix is removed, will be at least five characters in length. If a defined suffix is found, it is removed.

As an example, if the suffixes "ing" and "er" are defined, then the phrases "Computer Programming" and "Computer Programmer" will match.

Replacement Suffixes: Replacement suffixes are searched for in a word by scanning right to left for the presence of the longest defined replacement suffix. If one is found, it is removed and the substitute supplied is used in its place.

As an example, the user may wish a plural form to be reduced to a singular one so that the singular suffix will be recognized in the suffix truncation step. This is demonstrated with the phrases "Battery Manufacturing" and "Manufacturing Batteries". If the suffix "*ies*" is changed to "*y*" then not only will the phrases be the same, they will be processed in the same manner at suffix truncation time.

Double Letters: At this stage in the parse, each word is examined for the presence of any double character occurrences which are contained in the (user-defined) double letter set. If any are found, they are reduced to a single occurrence.

Typically, the double letter set used is the full set of alphabetic characters. If this is the case, then the words "Programer" and "Programmer" would match, in spite of the spelling error.

Root Words: At this point, words are scanned for the presence of any of the root words supplied. The scan is applied from left to right in the word, and searches for the longest defined matching root word. If one is found, then its substitute is used as a replacement for the word and the suffix truncation and replacement steps are skipped.

As an example, the languages "Slavee" and "Slavic" differ only in their last two characters. So, if the suffixes defined include "ee" and "ic" then an information loss occurs, since both words will become identical. Although generally, suffix truncation works well for most applications, it quite clearly fails for this particular example. To overcome this problem, if root words of "Slave" and "Slavi" are defined, then the suffix truncation step is bypassed for these cases only. Thus, as suffix truncation problem cases are identified, root words and their substitutes can be defined to overcome them.

Duplicate Words: Finally, the set of words resulting from the parse of the supplied text is examined for the presence of duplicates.

Note that words which are duplicates at this point may not have appeared as duplicates before the text was parsed. Only one occurrence of each word defined at this point in the parse is kept.

4. SEARCHING AND MATCHING METHODS

ACTR always processes the supplied text according to the parsing strategy defined before attempting a match. If after doing this, ACTR is able to locate a phrase on the matching database with all of its words in common with all of the words in the supplied text, then the match found is referred to as a "Direct Match". If a direct match cannot be found, ACTR may, as a user option, continue to search the database for the closest match. This latter type of match is called an "Indirect Match". Although they share a common foundation in that they are both based on parsed text, the two matching methods used by ACTR differ greatly in their mechanisms for both locating and assigning a match.

Direct Matching

In direct matching, only a 100% match is searched for. Recall that matching is based on parsed text, so phrases which are 100% matches may not appear to be identical in their original form. This is a direct effect of the parsing strategy in use.

In terms of database access techniques, the fastest path to an item is through the use of a key. Unfortunately, the roadblocks to keyed access of ACTR phrases exactly as they occur include a maximum phrase length of 200 characters and an upper limit of 20 on the number of parsed words. These two items make keyed access impractical since the extreme length of the key would negate any benefit derived. The only alternative to keyed access is sequential access, but this is undesirable because of the time required to search through the large volumes of information generally contained in a matching database.

So, we are presented with no other alternative but to somehow reduce the size of the key, thus making keyed access viable. There are many well known data compression techniques which could be used to do this, a general survey of which can be found in Reghbati (1981). In ACTR, the required data compression is achieved by forming the "compressed phrase key" or CPK. How CPK's are actually formed is discussed below. Accept for now that CPK formation results in a key which is approximately 35% of the original size of the phrase. The CPK can thus be used to access the matching database with an efficiently sized key in order to determine whether any direct matches exist.

The use of the CPK in ACTR is significant in the following ways:

1. All 100% matches will always be located using this method.
2. Since ACTR is able to locate direct matches by using the most efficient means possible, matches made by using this method are both faster and cheaper to perform.
3. As applications mature, the proportion of direct matches generally increases due to ongoing database update activity on the part of the user. Thus, overall matching costs for an application can actually decrease as the application matures, even though the size of the matching database may increase.

CPK Formation

The CPK is formed by first ordering the words defined in parsing. The actual order is arbitrarily chosen and so is not significant, as long as the same ordering applies for all CPK formations. (The order used happens to be in ascending order of the collating sequence in use.)

After ordering, the words are concatenated into a single string which contains no blanks. This string is then compressed in order to form a short enough string to allow for efficient use as a database retrieval key. The compression of the string is based on the following:

1. The words resulting from parsing generally contain only characters from the 26 alphabetic character set and the 10 character numeric set. (Recall that the actual set of characters which may be encountered in words is user-defined.) However, characters are stored internally (*ie.* in memory and on disk) using an 8 bit code. Thus, there are 2^8 or 256 possible 8 bit code combinations while ACTR words typically use no more than 36 of these. This leaves a 220 code surplus which could be used for other purposes.
2. Certain double and triple letter combinations are known to occur more frequently than others in English and French text samples. In ACTR, the double letter combinations are known as "digrams", and the triple letter combinations are known as "trigrams".
3. The 220 "free" codes can then be used to replace the digrams and trigrams described above as they occur in text samples.
4. Starting with the concatenated, parsed words, ACTR scans for the presence of any of the predefined digrams and trigrams. If any are found, they are replaced with the associated 8 bit code. The result is that a character sequence which formerly required 16 or 24 bits of storage, now requires only 8 bits.

Indirect Matching

Like direct matching, indirect matching begins with the set of words resulting from the parsing process. However, indirect matching can never be as efficient as direct matching since the concept of closest match is relative. That is, we cannot find the closest match without first performing an exhaustive search through all of the possible matches.

In order to perform indirect matching, the matching database must first be searched for each of the words resulting from the parsing process in order to determine which, if any, are known. Following this step, for each word in the supplied phrase which is known to the database, all phrases containing the word must be retrieved and evaluated.

The nearest matching phrase is determined by calculating a score for each of the possible matches. Scores are based on the weights of the words which are in common with the database and subject phrases. Of all database phrases evaluated in this manner, the highest scoring phrase is the one which is considered to be the closest match.

Word Weight Calculation

For each word known to the database, ACTR calculates a matching heuristic, or weight. These weights are an indication of the usefulness of a word in assigning a code and act as components in the phrase score calculation process.

The method by which word weights are calculated is based on: n, a count of unique codes, whose associated phrases contain this word; V_i, the relative frequency of code i from previous surveys; X_i, a count of the number of word occurrences for phrases with code i; P_i, the proportion of this word in code i, calculated as $V_i \times X_i / \Sigma_{j=1}^{n} V_j \times X_j$; EW, the entropy of the word, calculated as $- \Sigma_{i=1}^{n} P_i \times \mathrm{Log}_2 P_i$; K, the total number of word occurrences for code i, calculated as $\Sigma_{i=1}^{n} \times X_i$; EU, the entropy of a uniformly distributed variable with K unique values, calculated as $\mathrm{Log}_2(\mathrm{K})$; and finally EO, a small value to avoid division by zero, calculated as $- K/K + 1 \times \mathrm{Log}_2 K/K + 1$.

From the preceding, word weights are calculated as: EU-EW + EO/EO + EW.

Phrase Score Calculations

For each database phrase which is evaluated for an indirect match, a score is calculated. The score is based on: n, the number of words the phrases have in common; w_k, the weight for word k; m, the number of words in the subject phrase; and l, the number of words in the database phrase.

From the preceding, phrase scores are calculated as: $n^3 \times \Sigma_{k=1}^{n} w_k/m \times 1$.

Matching Parameters

After calculating a score value for each potential match, ACTR compares the score against user supplied values for the following parameters and takes the action indicated.

1. UPPER THRESHOLD
   If the resulting score is greater than or equal to this value, then a winner is considered to have been found.
2. LOWER THRESHOLD
   If the resulting score is greater than or equal to this value, but less than that supplied for the upper threshold value, then a possible match is considered to have been found.
3. PER CENT DIFFERENCE
   If more than one winner is found, and their scores are within the supplied value for this parameter, then multiple winners are considered to have been found.

Limiting the Search for an Indirect Match

ACTR searches the matching database for possible matches using the known words in the subject phrase. That is, these words are used to search for database phrases which contain them. The search proceeds in order of the ascending frequency of occurrence of the known words. Thus, the known word which occurs the least frequently in the database is used to start the search, the next lowest is used to continue the search, and so on.

As can readily be appreciated, finding a match by the indirect process has the potential of being time consuming and very expensive. Unfortunately, attempts to find matches by indirect means are unavoidable since a nearest matching feature is an essential component of any automated coding system.

While performing a search in this manner, ACTR maintains a list of database phrases which have already been evaluated. After a database phrase has been evaluated, it will not be re-evaluated in a subsequent iteration for the currently executing matching effort. This ensures that a database phrase which contains more than one of the known words will not be evaluated more than once.

As a further search optimization, ACTR makes use of the user supplied matching parameters. With these, it constructs a table of optimistic scores for each iteration of the word based search:

1. For the first known word, the optimistic score is based on the possible occurrence of a database phrase with the same number of words as the number of known words and with all of its words in common with the subject phrase's known words.
2. For the second word, a similar assumption is made, but since the first word has already been used in the preceding search iteration, we know that any phrase containing the first word has already been evaluated. So, the optimistic score is based on the presence of the second and subsequent words only.
3. Optimistic scores for succeeding iterations are based on the presence of the current and succeeding unsearched words only.

The formula used to calculate the optimistic scores is based on: a, the number of known words in the subject phrase; b, the number of words in the subject phrase already searched; c, the total number of words in the subject phrase; and d, the number of known words not yet searched, calculated as $a - b$;

From the preceding, optimistic phrase scores are calculated as: $(d^2 \times \Sigma_{i=d}^{a} w_i)/c$.

With the table of optimistic scores in place, ACTR evaluates the potential score at each iteration before performing a database access. Thus, hopeless searches are never attempted.

To summarize, the search for an indirect match is terminated when any of the following conditions are met:

1. The maximum potential score for the current iteration does not meet or exceed the threshold defined for possible matches.
2. At least one match has been found and the maximum potential score for the current iteration cannot produce another.
3. The maximum number of possible matches requested by the user has already been found and the maximum possible score for the current iteration does not exceed that of the lowest scoring phrase.

5. SUMMARY

A flexible and efficient automated coding methodology, embedded in a generalized software system has been presented. The system can be used to perform automated coding for any application in English or French or both, using any classification scheme. In doing so, it makes use of a powerful generalized parsing strategy and significant performance optimizations. For further information on ACTR, the interested reader is directed to Connor, Salloum and Wenzowski (1988).

6. ACKNOWLEDGEMENTS

While a complete list of all those who have been involved in the ACTR project would be too large to be presented here, the author would specifically like to acknowledge the contributions made by: John Connor and Bill Salloum, the principal programmers for the project; Victor Estevao, who was instrumental in helping to design the parsing strategy; Malvinder Rakhra and Paul Surman, who performed a great deal of system testing; and Don Royce, who was the project manager for the research effort.

REFERENCES

APPEL, M., and HELLERMAN, E. (1983). Census bureau experiments with automated industry and occupation coding. *Proceedings of the Section on Survey Research Methods, American Statistical Association*, 32-40.

CONNOR, J., SALLOUM, B., and WENZOWSKI, M. (1988). ACTR Documentation Set. (System Overview, User's Guide, Tutorial, Guide to the Parsing Strategy, Default Parsing Data, Message Guide, Command Language Guide, Searching and Matching Methods & Programmer's Guide) Internal Documents, Statistics Canada, Research and General Systems Subdivision, Ottawa, Canada.

LANDRY, L., and PIDCOCK, J. (1984). Business Register Automated SIC Coding System, System Proposal and Design. Internal Document, Statistics Canada, Informatics Services and Development Division, Ottawa, Canada.

REGHBATI, H. (1981). An overview of data compression techniques. *Computer*, 14,4, 71-75.