AN INTRODUCTION TO THE ACTR CODING SYSTEM

by

Errol Rowe and Christine Wong
Automated Coding Staff
Statistical Research Division
Bureau of the Census
Washington, DC 20233

This paper examines an automated coding system developed by Statistics Canada, ACTR (Automatic Coding by Text Recognition). The information presented in this paper is from the testing of ACTR at the Bureau of the Census as well as a summary of Statistics Canada documents. The purpose of examining ACTR is to evaluate its coding capabilities in order to determine if there exists coding applications at the Bureau of the Census for which ACTR is well suited. In this paper we provide an overview of the ACTR system, the capabilities and limitations of the current system[1], as well as the coding results obtained from ACTR versus other systems.

ACTR was developed by Statistics Canada and is based on the Hellerman algorithm[2]. Its purpose is to convert alphabetic responses on survey questionnaires into corresponding numeric codes at quality levels equal to or better than those achieved when the codes are manually assigned.

ACTR employs word standardization techniques (subject to user specifications and modifications) to match input respondent text to an existing database file of phrases in order to return a code. As part of its implementation, ACTR provides:

1. a parsing mechanism to reduce the text to a standard format,
2. functions to create and maintain database files of descriptions and codes,
3. a searching/matching algorithm to perform the coding which may be batch or on-line.

All that is required of the user is a database of descriptions and their associated codes.

ACTR is unique in that it has been generalized to allow it to be used by almost any application which assigns classification codes based on input text[3]. Another feature of ACTR is dynamic updating: an entry passing through the system unmatched may be manually coded and added to the project database so that subsequent similar entries will be coded by ACTR. Text and code may be added to the database, changed, or deleted from the database at any time during the life of the application. The parsing strategy can be altered at any time by adding, changing, deleting, or only bypassing certain steps.

---

[1]The current system as of this writing - September 1992.

[2]See Hellerman, E. (1982), "Overview of the Hellerman I&O Coding System", draft memo, Bureau of the Census.

[3]Technically, ACTR can accommodate any language. Statistics Canada intends this system to perform in English and French, but currently only English is functional.

Parsing Strategy

Automated coding is the process by which text is machine analyzed in order to assign it a classification or code. To deal with the inconsistencies of text, ACTR copes with the problems of:

1. rearranged words
2. missing words
3. spelling variations
4. abbreviations
5. various punctuation
6. plural vs singular forms
7. extraneous words
8. synonyms
9. inconsistent hyphenation
10. syntax.

The parsing strategy controls the way in which input descriptions are parsed to standard forms. The strategy is comprised of parsing steps, objects and data. There are six parsing steps in ACTR, four of which are further broken down into twelve parsing objects. Each parsing object contains the parsing data which will be used to convert the input description to its standard form. For example, one parsing step is word processing, one of its objects is suffixes, and the data of this object are the actual suffixes such as -ent and -ance.

The parsing strategy of ACTR is entirely user controlled and may be modified at any time. Moreover, it should be modified to optimize the performance of each project. The flexibility allows the user to change the parsing objects in use and also to change the order in which the objects are used.

There are two types of processing in parsing: 1) String Processing and 2) Word Processing. The processes the input text as a continuous stream of characters. The processes on a word by word basis after the string has been broken into its constituent words. The parsing of an input description continues until all words have been processed.

Diagram of ACTR's Six Parsing Steps

Input Description--

1) Character Translation: word characters.
2) String Processing: deletion clauses, deletion strings, replacement strings.
3) Word Breaking: word characters.
4) Word Processing: hyphenated words, illegal words, replacement words, double words, suffixes, prefixes, double characters, exception words.
5) Sort Words
6) Remove Duplicate Words

--Parsed Description

The order of the six steps cannot be changed, but the order of the twelve objects within the steps and the data for each of the objects can be modified[4].

WEIGHTING SYSTEM[5]

CMK Formation

CMK stands for Complete Match Key. It is a unique key that allows direct matching. It uniquely identifies a description in parsed form. The CMK is based upon a 16 bit unsigned integer, used as an identifier for each word known to the application. As the application acquires words, they are assigned a serial number which is simply 1 greater than that of the currently highest numbered word.

In hex, valid word ID code numbers range from x'0001' to x'FFFF', with the value x'0000' used to indicate that a word is unknown. Thus, there are 65,534 [2^{16} - 2] possible word ID codes.

To form the [ACTR] version 2 CMK:

-Parse the input text to obtain a list of words.

-Look-up the code for each word, and concatenate them into a string. The format of the CMK looks like:

wd1 wd2 wd3 wd4 wd5 wd6 wd7 wd8 wd9 wdA

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

wd1 through wdA represent the two byte (unsigned) word ID numbers for words 1 through 10 respectively.

The implementation of the CMK in ACTR uses an overflow technique. Basically, the CMK is divided between a primary CMK (denoted just by CMK) and an overflow CMK (denoted CMKO). The primary CMK handles the first 10 words; the CMKO takes care of the remaining 110 words[6]. The biggest advantage of splitting the CMK in two is that the primary CMK will be used most of the time (descriptions with more than 10 words are rather

---

[4]Further details on the parsing steps are found in Appendix 1.

[5]Taken from *ACTR, Automated Coding by Text Recognition, Version 2, System Maintenance Manual (Draft)*. General Systems, Systems Development Division, Statistics Canada. August 7, 1992.

[6]The Beta Version of ACTR does not allow more than 10 words in an input phrase.

rare) and therefore will result in a very short key, making database searches very fast.

Note that in the process of forming the CMK, we have determined whether or not the input text contains any unknown words, and so we can avoid fruitless CMK searching. Also, we are able to obtain word weights at the same time, thus eliminating a second lookup effort.

Word Weights

Each word in an ACTR database is assigned a weight which indicates how frequent a word [occurs]. The method used in ACTR will always result in values in the range of 0 to 1 inclusive where 1 is for a very rare word and 0 for a word that appears in all descriptions in the database (one would wonder about the usefulness of such a word).

A new method for the calculation of word weights is implemented. The following method considers only the usefulness of the word in assigning a code, and will always result in values in the range of 0 to 1 inclusive[7].

Method

x = the number of unique codes known to the application
y_i = the number of unique codes corresponding to descriptions containing word i
w_i = the weight for word i = $1 - (\log_2(y_i) / \log_2(x))$

Note that as a special case, where:

$x = y_i = 1$, i.e. only one code in the entire database,

the weight for the word is arbitrarily assigned a value of 1. Note, as well, that the value of y, for any known word, will always be greater than or equal to 1.

[7]A word weight is calculated involving all of the following factors: the number of different words, the number of different codes, the number of different codes associated with each word, frequency with which words occur, frequency with which codes occur and frequency with which unique word/code couplet occurs.

There is no theoretical reason for the use of the base 'two' in the log function. We could have used any other base.

Example

Description Database:

Descriptions	Codes
computer programmer	005
computer analyst	005
economist	011
lawyer	008
programmer/analyst	005
weather analyst	013

word i = analyst
x = 4
y_i = 2

Weight for word "analyst" = $1 - (\log_2(2) / \log_2(4)) = 1/2$

Scoring

Each match produces a score which indicates how good the match is:
the higher the score, the better the match.
The ACTR scoring method produces scores which will always fall within the range of 0 (not even one word matches) to 10 (perfect match, all the words match).
The scores in ACTR V2 are therefore quite intuitive.

Method

$$a = \frac{2 * (\text{\# of words in common in DB}^{8} \text{ and input text})}{(\text{\# words in database text}) + (\text{\# words in input text})}$$

$$b = \frac{\text{sum (weights of common words in DB and input text)}}{\text{sum (weights of known words in input text)}}$$

$$\text{Score}^{9} = ((a + 2b) / 3) * 10$$

---

[8]Database phrase.

[9]In most occasions, the value of 'b' dominates 'a'. This is true when the individual weight of each word in common is greater or equal than the weight of each word not in

Example

Using the same description database as in previous example

Input text = "system computer analyst"

Find score for DB description "computer analyst"

$$a = \frac{2 * (2)}{(2) + (3)} = \frac{4}{5}$$

$$b = \frac{[\ 1 - (\log_2(1)/\log_2(4))\ ] + [\ 1 - (\log_2(2)/\log_2(4))\ ]}{[\ 1 - (\log_2(1)/\log_2(4))\ ] + [\ 1 - (\log_2(2)/\log_2(4))\ ]} = 1$$

Score = (((4/5) + 2(1)) / 3) * 10 = 4.9

System Overview

Basic sequences of operations:

1) **Define/redefine** parsing strategy
2) **Create** project
3) **Load/update** project database
4) **Perform** coding preparation
5) **Perform** coding (i.e., run ACTR)
6) **View** Coding Results

7a) **Redefine** parsing strategy[10]
7b) **Update** project database *(Optional step)*
7c) **Reparse** project database

8a) **Update** project database[11]
8b) **Redefine** parsing strategy *(Optional step)*
8c) **Reparse** project database

---

common. When 'a' dominates 'b', the value of 'a' is small as well as the value of the score. This usually occurs when the weight of the word in common is less than half the weight of the word not in common. The ratio half is further reduced when a word from the input phrase does not appear in any database phrase.

[10]See Configuration 1.

[11]See Configuration 2.

In ACTR, certain tasks are governed by synchronization rules. These tasks entail manipulating parsing strategy, loading database phrases, and coding. The sequences listed on the previous page observe these rules. ACTR does provide a Project Status function which states which tasks are currently executing against your project, the last update task, and the next possible tasks which you may undertake.

Operations (1) through (6) are the core operations, appearing in the order necessary to run a project through ACTR. 7 (a) - (c) and 8 (a) - (c) are two possible subroutines which you may choose to perform in ACTR as a way to improve your results.

1) The descriptions which are stored in the project database are parsed while loaded into ACTR. So the parsing strategy must first be in place before the database can be loaded. This function allows you to load new parsing data or to update existing data. You may also accept the default parsing strategy.

2) Before using ACTR to code, a project must first be created. This function sets up the environment for a new project, and creates a place for the database to be stored.

3) With this function, you can load your project database/reference file (description + code). Clearly, this step must be done before any coding can be performed. Note that during loading or updating of descriptions, each description is automatically parsed to a standard format. To update a database, you may add, delete, or change individual entries in the database in a conversational (interactive) mode.

4) The Coding Preparation function must be executed before coding only if a new database has been loaded or the existing one has been updated. If a database has already been prepared and has not been altered, this step may be skipped. In this step, ACTR assigns weights to each word in the project database.

5) Coding is the process of parsing input descriptions and matching them against the project database in order to assign them a code. You are free to allow only perfect matches or to allow partial matches as well.

   perfect match: exact match between each word of the parsed input description and the parsed description in the project database.
   partial match: at least one common word between the parsed input description and the parsed database description is encountered.

   Partial matching will often result in more than one match. This is where scores come into play. A score is given to each match found in the project database. The score is based on a mathematical formula which uses the weights of each common word. A score ranges from 1 to 10 where 10 represents a perfect match. For partial matches, scores can be used to help you make a decision about the quality of the match.

Immediately following the coding of an input file, ACTR displays the number of perfect matches, partial matches and failures. For partial matches and their scores, ACTR displays winners (high scores), multiple matches, possible matches (middle scores), and failures (low scores below a cutoff).

6) After your input file has been coded in a batch mode, ACTR allows you to view the results, the original phrase, the parsed phrase and tells you whether the phrase was matched. If it was matched, ACTR displays the matched phrase, code and score.

7a) See (1).

7b) See (3). Note that this is an optional step and may be skipped.

7c) When a change is made to the parsing data, the project database has to be reparsed using the new parsing data, since the descriptions are stored in a parsed form. Synchronization rules in ACTR dictate this. So the Reparse Project Database function takes the existing database and applies the new parsing rules to it.

8a) See (3).

8b) See (1). Note that this is an optional step and may be skipped.

8c) See (7c).

Experiences with ACTR

Canadian Centre for Health Information[12]

The Centre used ACTR to code the occupation data from 138,000 death certificates for the "Mortality by Occupation" study. The industry data was used as well as occupation to determine the code. The generalized functions of ACTR were used and no software development costs were incurred. The implementation included accepting direct matches only and editing the unmatched responses to be fed back into ACTR. Initial match rates, prior to editing the file were 50%, but subsequent rounds of processing eventually produced a match rate of approximately 82%, with manual coding required for the remaining 26,000 responses. The members of the Mortality study team indicated that the study would not have been feasible without the use of ACTR (Wilkins, Ratnasingham; 1989) as the resources (personnel and funding) for a completely manual coding operation could not be acquired. [The estimated cost of ACTR and manual coding is $40,000 versus the estimated cost of complete manual coding for $138,000.]

[12]Taken from *Automated Coding At Statistics Canada*, Dianne Miller, General Systems, Informatics Branch, Statistics Canada, November, 1991.

Statistics Canada Testing of ACTR

In consideration of using ACTR for the Canadian 1991 Census, a research project to test ACTR was established and produced the following results[13]:

VARIABLE	MATCH RATES	ERROR RATES
Mother Tongue	92.1%	3.4%
Place of Birth	91.6%	2.0%
Ethnic Origin	93.8%	1.3%
Major Field of Study	78.0%	4.4%
Industry - Company Name	31.5%	8.2%
Industry - Kind of Business	38.0%	25.5%
Industry - Linked Files	22.3%	2.4%
Occupation - Line 1	42.7%	31.1%
Occupation - Line 2	19.2%	37.0%

The match rates correspond to the percentage of responses for which ACTR found a match on the reference file. The error rates reflect the percentage of matched responses that were assigned an incorrect code.

The error rates for the cultural and Major Field of Study variables were attributed largely to spelling errors, multiple responses, abbreviations, lengthy or ambiguous responses, the use of adjectives and missing entries in the reference files. These rates were similar to those in the manual coding of these variables in the [Canadian] 1986 Census and the subject matter specialists felt that many of the problems could be corrected by updates to the reference files and modifying their use of the software.

The results of the Industry and Occupation testing were less encouraging. The Industry question consists of three parts to be used in coding: the company name, department or section, and the kind business. Two reference files were created containing data specific to the Company Name and Kind of Business. The problems with coding related to misspellings, ambiguity, generalizations (e.g. self-employed), and missing data in the reference files. Some of these could be overcome by enhancing the reference file and refining parsing data. However, as indicated by the

[13]Taken from *Automated Coding At Statistics Canada*, Dianne Miller, General Systems, Informatics Branch, Statistics Canada, November, 1991.

results of the testing, the major requirement to code this data is to have access to the results of both reference files when determining the code. This facility was not available in ACTR and the user wrote specific software to simulate the effect of this.

The Occupation question contains two parts: kind of work and the most important duties carried out by the respondent. Two reference files were created and the results were similar to those of the Industry coding. Subsequent analysis indicated that the results of both reference files would be required as well as the Industry code to obtain acceptable occupation codes.

Based on these results, the decision was made to proceed with the automated coding of the socio-cultural variables and continue research with the Industry and Occupation data.

Limitations

Dianne Miller, Research and General Systems, Systems Development Division, Statistics Canada stated, "ACTR cannot currently handle the coding of two fields at one time. This is in our plans for the next year." This feature added to ACTR would facilitate I&O coding. Additionally, creating a driver that calls on ACTR in a subroutine will reduce the complexity of these problems. "The [1991 Canadian] Census uses the generalized functions and end-user interface provided by ACTR to maintain their reference files and parsing data but has developed application programs to perform batch matching and a custom user-interface to perform the on-line resolution of non-matches. ACTR is called within this software."[14]

Other limitations of ACTR are much less restrictive for various coding applications. The weight system of ACTR cannot be altered by the user. Score calculations vary with database updates, and vary greatly with the size of the text being matched which may produce non-intuitive score values. There is no spelling check in ACTR. The results of a batch coding process are sent by ACTR (Beta Version) to an output table which cannot be specified by the user and which is overwritten each time such a process is run. Also, an input phrase or database phrase may have up to but not to exceed ten words. See Appendix 2 for proposed improvements of ACTR.

Recommendations

Various divisions of the Canadian government currently use ACTR for coding. ACTR can be used to successfully code a wide range of items, single phrases matching to single codes. Even if ACTR returns a low match rate, it eliminates some manual coding which is lengthy and tends to be more expensive. In the Canadian Transportation Division, "approximately

[14]Taken from *Automated Coding At Statistics Canada*, Dianne Miller, General Systems, Informatics Branch, Statistics Canada, November, 1991.

300,000 - 500,000 responses are coded annually. The match rates are only 25 -30% but without the use of ACTR, the division would have had to reduce the volume of respondent data captured as there were limited resources to code it manually."[15]

By improving its reference file and parsing strategy, the Canadian Center for Health Information was able to improve initial match rates from 50% to 82%[16]. The most difficult step of ACTR may be either compiling the material necessary to create the reference file of phrases and codes if they did not exist previously or refining the reference file and parsing strategy once the coding application is established. However, once a successful reference file and parsing strategy is established, subsequent uses of ACTR will require much less work. ACTR is a time saving, and inexpensive tool which should be utilized by anyone who needs to do general coding.

With regards to comparing ACTR with other systems, at the time of this writing we are still learning about the ACTR weighting system and how it differs from the other systems. Until this study is complete, such comparisons cannot be made. However, ACTR does have the advantage of an extremely user-friendly interface.

Conclusion

ACTR is a user-friendly software package for general coding. The general design of ACTR makes it extremely versatile, even though it is not possible to perform I&O coding with ACTR alone. ACTR does successfully code less complicated data. In order to optimize the results, the components involved with coding the project can be easily modified, assuming that ACTR's weighting system is appropriate for your coding purposes.

Recall that to code in ACTR, text is standardized and the parsed input phrase is compared with parsed database phrases. One way to improve the coding is through manipulation of the parsing strategy. ACTR allows the user to immediately correct mistakes by modifying the parsing strategy and database at any time. This is a powerful tool to refine your project in order to improve your results. You may code a file, view the results, and based on that, change the parsing strategy and/or database, code again and see the new results in a matter of minutes.

---

[15]Taken from *Automated Coding At Statistics Canada*, Dianne Miller, General Systems, Informatics Branch, Statistics Canada, November, 1991.

[16]Taken from *Automated Coding At Statistics Canada*, Dianne Miller, General Systems, Informatics Branch, Statistics Canada, November, 1991.

Appendix 1

Details of the Six Parsing Steps

1) *Character translation* takes input characters and as prescribed, translates them to specific characters such as all lowercase letters going to the corresponding uppercase letters. *Word characters* is a feature that contains a list of valid characters and their translation as specified by the user through the parser. It translates the entire input description. So "R.V." would go to "recreational vehicle".

2) *String processing* is performed regardless of the input description since the text is processed as a continuous stream of characters.

    a) *Deletion clauses* are a method of delimiting a string which should be removed from the input text. By supplying a beginning and ending string, the user can remove these and any information enclosed in the strings. For example the beginning string could be "(EXCEPT" and the ending string could be")". These two strings and anything in between would be removed.

    b) *Deletion strings* found at any position in a description will be removed. For example "'s". If only the apostrophe were removed, the problem would then be the creation of a new word "s" which is bad for coding purposes.

    c) *Replacement strings* are most useful for standardizing abbreviations, i.e., "t.v." replaced by "television". To use this feature of ACTR, the "@" technique maintains the appropriate number of blank spaces. (See "ACTR Parsing Strategy" in Statistics Canada's documentation of ACTR for more information on "@".)

3) *Word breaking* prepares the phrases for word processing. Characters not included in the list of word characters will be used as word delimiters and will be dropped from further consideration with the two exceptions of blanks and specified hyphens when a substitution from the hyphenated word list exists. An example of a word delimiter is the symbol /.

4) *Word processing* treats the text as a collection of words.

    a) *Hyphenated words* recognizes words and word groups which are inconsistently hyphenated and replaces them with a standardized form.

    b) *Illegal words* are removed when a specified character string exists in those words at any position. This feature can be used to eliminate words containing

numeric characters, such as dept17b.

c) *Replacement words* is a feature that provides synonym capability. This could allow "car" and "automobile" to be parsed identically. Replacement words also allows the removal of trivial words by replacing them with blanks.

d) The *double word* feature forces ACTR to consider not only the occurrence of two word groupings, but their order as well. This can be useful in overcoming inconsistencies in word spellings and also to preserve word order.

e) *Suffixes* and *prefixes* are handled in the same manner. For suffixes, the word is scanned from right to left looking for the longest defined suffix such that the remaining word, after the removal of the suffix, is still at least four letters long. This feature will also handle the problem of plural versus singular forms.

f) *Double characters* (contiguous double letters) are reduced to a single occurrence.

g) *Exception words* is a feature that scans each word for the presence of a predefined exception word which would prevent it from any "suffixes," "prefixes," or "double character processing. Examples are the two distinct languages, slavee and slavic. Suffix truncation would identify them as the same: slav.

5) *Sort Words* is a step that sorts the remaining words in the description in alphabetical order.

6) *Remove Duplicate Words* is the final parsing step which performs as its name would indicate; it removes any duplicate words in the parsed description.

Appendix 2

Proposed Improvements (Statistics Canada)

The current new version of ACTR does not restrict the size of the description to 10 input words or less, in either the database or in the file to be coded. The Beta Version of ACTR, which the Census Bureau currently uses (September, 1992), does have this restriction.

The new version of ACTR also allows one to specify the output tables so that the ACTR specified output table is not written over each time a project is run, as in the Beta Version.

In the soon to be released version, ACTR will have normalized output tables which are tables that store the data more efficiently with less repetition.

ACTR cannot currently handle the coding of two fields at one time but the possibility of this will be considered in the next year.

Soundex is not yet active, but it is being developed for use in ACTR.

French as a user language is not functioning, but is being developed.

Appendix 3

Relationship with ORACLE

The visual format of ACTR is created from SQL*Forms, a feature of Oracle. However, the amount of Oracle which an ACTR user must know is minimal. To actually run ACTR, no knowledge of Oracle is required. The necessary Oracle commands are those that properly format your files to be used in ACTR. In order to load files into ACTR, the project database file and files to be coded, must be in Oracle tables. In order to store your files in Oracle tables, which can then be directly loaded into ACTR, see the documentation "Oracle for ACTR" by E. Rowe and C. Wong.

System Requirements

Hardware Required:

CPU	Preferably, a 32-bit computer.
Tape drive	A 5 1/4 inch tape drive needed for installation.
Disk Space	You will need about 5 megabytes of hard disk space to store ACTR V2 software.
Terminals	ACTR V2 will run on ANSI, VT220 (VI220) and X terminals.

Software Required:

UNIX	A UNIX operating system compatible with UNIX System V.
ORACLE	ACTR V2 requires ORACLE RDBMS Version 6.0 and SQL*Forms V2.3 as the underlying DBMS
'C' Compiler	An ANSI 'C' Compiler must be available to ACTR V2 to allow it to compile programs at installation time.

Appendix 5

Appendix 6

Bibliography

"ACTR, Automated Coding by Text Recognition, Version 2, System Maintenance Manual (Draft)." General Systems, Systems Development Division, Statistics Canada.

"Automated Coding At Statistics Canada," Dianne Miller, General Systems Informatics Branch, Statistics Canada, November 1991.