IMPROVING THE QUALITY OF AUTOMATED CODING AT STATISTICS CANADA

Xinye (Hannah) Yang[1], Caroline Pelletier[2]

ABSTRACT

Coding is a data processing step that assigns responses of open-ended questions into predefined categories identified by alphanumeric or numeric codes. Coding is performed automatically and/or manually. A generalized automated coding system developed at Statistics Canada has been used by several coding applications within and outside the organization. The methodology used in this system was recently re-evaluated in an effort to achieve a higher coding rate and to improve the timeliness of data without sacrificing data quality. In this paper, we present an alternative coding method and compare it to the one currently implemented in the system.

KEY WORDS: Automated Coding, Data Quality.

RÉSUMÉ

Le codage est une étape du traitement des données qui assigne les réponses aux questions ouvertes à des catégories prédéfinies identifiées par des codes numériques ou alphanumériques. Le codage se fait de façon automatique et/ou manuelle. Statistique Canada a développé un système de codage automatisé qui est utilisé par plusieurs applications de codage à l'intérieur et à l'extérieur de l'organisme. La méthodologie utilisée dans ce système a récemment été réévaluée dans le but d'augmenter le taux de codage et d'améliorer l'actualité des données, sans pour autant en sacrifier la qualité. Dans cet article, nous présentons une méthode de codage alternative et nous la comparons à celle actuellement mise en œuvre dans le système.

MOTS CLÉS : Qualité des données; codage automatisé.

1. INTRODUCTION

1.1 Overview of Coding

When the data collection of a survey is completed, data processing transforms collected survey responses into a form that is suitable for analysis and dissemination. Data processing steps include data capture, coding, editing and imputation, weighting, and estimation. Coding is the process of assigning responses of open-ended questions into categories identified by standard classification codes. These codes could be either alphanumeric or numeric, and the coding process could be manual or automatic, or a combination of both.

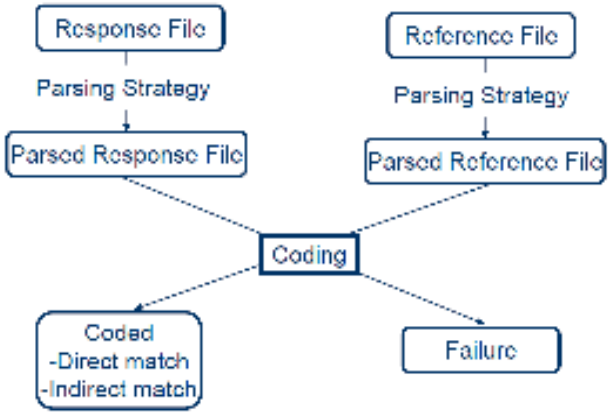
Traditionally, the coding of open-ended questions has been a manual process. When manually coding a response of open-ended questions, the coder would read and interpret the response, and assign a code based on his/her own judgement. The coders also have access to auxiliary information such as the answers to other related questions. The quality of coding thus depends on the completeness and clarity of the responses, as well as the training, judgement, and experience of the coder. Inevitably there is variability between different coders and thus a quality control process can be implemented to ensure that the level of error does not exceed a specified level, with minimum inspection.

[1] Xinye (Hannah) Yang, 100 Tunney's Pasture Driveway, Ottawa, Canada, K1A 0T6, xinye.yang@statcan.gc.ca
[2] Caroline Pelletier, 100 Tunney's Pasture Driveway, Ottawa, Canada, K1A 0T6, caroline.pelletier@statcan.gc.ca

1.2 Generalized Coding System

Due to advances in technology, resource constraints, and concerns about timeliness and quality, coding is becoming more and more an automated process. For the purpose of standardizing and automating the coding process, Statistics Canada has developed a generalized system. Over the years this system has served many users both within and outside the organization for a number of different coding applications such as industry, occupation, geography and commodity. The system was first released in the late 1980's, named as Automated Coding by Text Recognition (ACTR). More information on the system can be found in Wenzowski (1988). The third version of ACTR was released in the mid 1990's. In 2010, a newer version was released after the redesign of the system and it was renamed as the Generalized Coding System (G-Code). The diagram depicted in Graph 1 shows the files and steps of the G-Code system.

Graph 1 – G-Code files and steps



Two files are input to the automated coding system, the reference and response files. The reference file contains phrases and their corresponding codes. All phrases in the reference file contain codes that have been verified to be correct. Some phrases are longer than others, and different phrases may correspond to the same code. A response file contains a list of response phrases to be coded. The automated coding system is used to search through the reference file and assign a pre-defined code to each phrase in the response file based on a successful match between the response and reference phrase.

It is expected that responses come in different forms. For example, the response to an open-ended question "What is your occupation?" could be "I program computers", or "Computer programmer" or "PROGRAMMING OF COMPUTERS". Therefore prior to coding, phrases in the response and reference files need to be standardized by the first step of automated coding, parsing. Parsing is used to deal with problems such as spelling variations, abbreviations, common spellings errors, etc. This step reduces phrases to a standard form by removing extraneous characters, trivial words, certain suffixes and prefixes, etc. The user decides how the responses are parsed.

After parsing, the response and reference files go through the second step which is coding, where the search engine of the system attempts to find the best match from the reference file for each parsed phrase in the response file. Two methods are employed for locating matching texts: direct match and indirect match. A direct match means that the found reference phrase is "identical" (i.e. after parsing and typically ignoring word order) to the response phrase; therefore the corresponding code of this direct match reference phrase would be assigned to this response phrase and this response is considered coded. If an exact match is not found, an attempt will be made to find the closest match possible among the reference phrases, based on a measure of similarity between the reference and response phrases. Every matching phrase retrieved from the reference file is assigned a score that is calculated based on an established algorithm. The methodology used for locating matching texts is explained in greater detail in Section 2. If the closest match meets the requirements set by the user, the corresponding code is then assigned to the response phrase and this response is considered coded. Otherwise, it is considered as failed to be coded by the automated coding system. Response phrases not coded automatically are sent to expert coders who manually assign codes. As for manual coding, a quality control process can be implemented to ensure the level of error does not exceed a pre-identified level.

1.3 Quality of Automated Coding

The coding and accuracy rates can be used as measures of the efficiency of the automated coding system. The percentage of phrases in a response file that are coded by G-Code, including both direct and indirect matches, is defined as the coding rate of the system. When the response file is verified by an expert coder, the true reference code for each response is obtained. A file that contains the true reference codes for all responses is called a truth file, and this file could be compared to the coding results of the automated coding system. Therefore, truth files could be used to evaluate the performance of the coding system, as the accuracy rate of the coding results could be obtained as the percentage of coded response phrases that are assigned to the correct reference codes.

During the coding process, there are several factors that could impact the quality of the automated coding system. First, the quality of information on the response and reference files has a direct impact on the coding and accuracy rates. For example, many spelling errors in response phrases would potentially result in a lower coding rate if such errors are not standardized through the parsing strategy. Also the reference file needs to be up to date and complete in order to be best matched to response phrases. For instance, the reference file used for occupation related responses needs to be updated regularly to remain in step with newly created job descriptions in this quickly changing economy.

Secondly, a good parsing strategy would standardize the response and reference phrases regardless of their grammatical and syntactical differences while preserving the original meaning of the phrases. This would increase the chance of a response being assigned to the correct reference code.

In addition to the parsing strategy, the user-defined requirements for indirect matching also impact the quality of the coding. These pre-defined requirements should be defined by the user based on results from statistical studies. If these requirements are set high, the quality of coding would be better but at a cost of fewer records being coded. In addition, the requirements should depend on the presence or absence of a quality control process for monitoring automated coding results. As mentioned earlier, implementing a quality control process will help identify and correct errors found in the coding process. Therefore with quality control in place, the requirements could be set lower; as a result there will be more records coded while the errors could be detected by the quality control process.

Another factor affecting the quality of automated coding is the methodology used in G-Code to search for the best match in the reference file. The current methodology uses a word weighting and phrase scoring algorithm. It calculates a weight for each word that appears in the reference file; then it calculates a score for a possible match. The score is a function of the weights of the words in the response phrase and the number of words in each phrase and in common between the two phrases. The focus of this paper is on testing an alternative method that could be used to find the best match, in an attempt to increase the coding and accuracy rates of the G-Code system.

The word weighting and phrase scoring algorithm used in the G-Code system is presented in Section 2. In section 3, an alternative coding method is introduced. Test datasets and results are presented in Section 4. These tests are used to evaluate the efficiency of the alternative method in comparison to the current methodology. The conclusion appears in Section 5.

2. CURRENT METHODOLOGY

This section introduces the current methodology used in G-Code to find the best match for each response phrase. Recall that there are two types of matches. Direct matches are perfect matches between a parsed response and a pased reference phrase. A direct match is defined as a complete match of all words in both phrases, where both phrases have the same word count and have been parsed in an identical manner. In addition word order differences between the two phrases may be ignored. Indirect matches, however, are found based on a score that indicates the similarity between the reference and the response phrases. To obtain this score, the system first calculates a weight for each word that appears in the reference file. Then, these weights are used to calculate the score for a match.

2.1 Word Weighting

The weight of each word in the response phrase is a function of the number of unique codes associated with this word and the total number of unique codes listed in the reference file, as shown in Equation 1. The word weight is bounded by 0 and

1. It is intended to reflect the rarity of the word, therefore common words in the reference file have weights close to 0 and unique words have a weight of 1.

Equation 1 – Word weight calculation

$$Word\ Weight = 1 - \frac{\log\left(\begin{array}{c} number\ of\ unique\ codes \\ associated\ with\ this\ word \end{array}\right)}{\log\left(\begin{array}{c} total\ number\ of\ unique\ codes \\ in\ the\ reference\ file \end{array}\right)}$$

2.2 Phrase Scoring

For every response phrase, a score is calculated between this response phrase and every reference phrase retrieved. The score is a function of the weights of words and the number of words in each phrase and in common between the response and the reference phrases, as shown in Equation 2. A high score is obtained if two phrases have many words in common or if there are few words in common but these common words have high weights. Scores are bounded by 0 and 10, where 0 means there are no matching words between the response and reference phrases and 10 means there is a direct match.

Equation 2 – Phrase score calculation

$$\text{Score} = 10 * \left(\frac{a + 2b}{3}\right)$$

$$a = \frac{2 * \text{number of words in common}}{\text{number of words in response} + \text{number of words in reference}}$$

$$b = \frac{2 * \sum \text{weights of common words}}{\sum \text{weights of response words} + \sum \text{weights of reference words}}$$

3. ALTERNATIVE METHOD

During the redesign of the automated coding system, the word weighting and phrase scoring algorithm was re-evaluated. With the objective of improving the coding and accuracy rates of indirect matching in mind, it was decided not to modify the current algorithm but to study an alternative coding method. This method, rather than evaluating the relative importance of words in the reference file, evaluates the ordering of characters in a phrase. Since this alternative method takes a completely different approach to comparing phrases, it is hoped that it will code some responses which are not coded by the current method. The alternative method is based on the Generalized Levenshtein algorithm, which computes the weighted sum of transformations required to convert a response phrase into a reference phrase. A transformation is defined as an insertion, deletion or replacement of a single letter. See Table 1 for the list of transformations and examples.

During text comparison, some operations have a greater impact on the meaning of phrases than others and should thus be assigned higher weights. For example, “Restuarant” could be a common typo for “Restaurant”. In this case, the letters a and u must be inverted (swap transformation in Table 1). On the other hand, the replacement of a letter could be more expensive, as it may potentially change the meaning of the entire phrase. For example, changing “Text” to “Next” requires the replacement of the first letter (FReplace transformation in Table 1) which has a greater impact on the meaning of the phrase. Thus this transformation should probably be assigned a greater weight than an inversion.

The Generalized Levenshtein algorithm has been implemented into the SAS9.1 function COMPGED and CALL COMPCOST routine. These were used to test the alternative method. In the CALL COMPCOST routine, only integer

values are allowed to be assigned as weights, therefore weights of 1 and 2 are used to differentiate between less and more important transformations as shown in the second column of Table 1. Note that "Blank" and "Punctuation" transformations are assigned a weight of zero because they should be standardized by the parsing strategy.

Table 1 – Example of transformation

Transformation	Description	Proposed Weights	Example
Blank	add/delete a blank	0	bab oon => baboon
Punctuation	add/delete a punctuation	0	bab!oon => baboon
Swap	exchange the positions of two adjacent characters	1	baobon => baboon
Double	add one identical character after another	1	babon => baboon
Single	delete one of two adjacent, identical characters	1	babboon => baboon
Append	add a character to the end of a word	1	baboo => baboon
Truncate	delete a character at the end of a word	1	baboonX => baboon
Delete	delete a character in a word	2	babloon => baboon
Insert	insert a character in a word	2	baoon => baboon
FDelete	delete a character at the front of a word	2	mbaboon => baboon
FInsert	Insert a character in front of a word	2	aboon => baboon
Replace	replace a character with a different character in a word	2	baXoon => baboon
FReplace	replace the first character of a word with a different character	2	Xaboon => baboon

After the weight of each transformation is defined by the user, scores are calculated based on Equation 3. For every response phrase, a score is calculated for this response phrase and every reference phrase. The highest score among all the calculated scores is chosen as the final score for this response. If this value is greater than or equal to the threshold defined by the user, then the corresponding reference code is assigned to this response, and this response is coded. Otherwise, the response is not coded.

Equation 3 – Score of the alternative method

$$\text{Score} = \max\left[0,\ 10 * \left(1 - \frac{\text{weighted sum of transformations required to convert a response phrase into a reference phrase}}{2 * \text{number of characters in response phrase incl. space}}\right)\right]$$

In Equation 3, the length of the response phrase takes into account spaces between words. Therefore, leading and trailing blanks should be removed from the response phrases during parsing. Note that excess blanks between words are removed automatically by the G-Code system. It should also be noted that the coefficient of the denominator is 2 in the equation. This number was chosen because most of the proposed transformations in Table 1 are assigned a weight of 2. However, this value should be changed according to the weights defined by the user. Based on Equation 3, a score of 10 means a direct match between the input and reference phrases, as no transformation is required to change the response to the reference phrase.

Illustrated in Table 2 are examples of score calculations by using Equation 3 for two different response phrases to be converted to the reference phrase "programmer".

In the first example, one swap transformation is required to switch the position of the letters r and g, in order to convert the input phrase "prorgammer" into the reference phrase. According to the proposed weights in Table 1, a swap transformation has a weight of 1. Therefore, the total weight of the transformation required is 1. The denominator is 2 times 10, where 10 is the number of characters in the reference phrase. The score between this pair of input and reference phrases is 9.5.

Table 2 – Example of the score calculation

	Example 1	**Example 2**
Response	prorgammer	prorgamner
Reference	programmer	programmer
Transformation	1 swap	1 swap 1 replacement
Weighted sum of transformations	1*1=1	1*1+1*2 = 3
Score	$\max\left[0, 10*\left(1-\left(\frac{1}{2*10}\right)\right)\right] = 9.5$	$\max\left[0, 10*\left(1-\left(\frac{3}{2*10}\right)\right)\right] = 8.5$

In the second example, in addition to the same swap transformation required as in the first example, the letter n in the response phrase "prorgamner" must also be replaced by an m. In this case a swap and a replacement transformation are required. According to Table 1, a replacement transformation is assigned a weight of 2. Thus the weighted sum of the transformation is 3, and based on Equation 3, the score is 8.5.

Assume that the calculated scores in Table 2 are the best scores for "prorgammer" and "prorgamner", respectively. If the threshold is set to be 9.0 for this response file that contains both response phrases, then Example 1 would be accepted and the phrase "prorgammer" will be considered coded. The score in Example 2, however, would be rejected, because it is below the pre-defined threshold value. Thus the response phrase in Example 2 will not be coded based on this threshold value.

4. TESTS

4.1 Description of Test Environment

Three test datasets were used to evaluate the efficiency of the Generalized Levenshtein algorithm. The success of coding based on the word weighting and phrase scoring method currently used in G-Code was compared to the success of coding based on the transformation weighting and phrase scoring of the Generalized Levenshtein algorithm. The success of coding is measured by the coding rate and the accuracy rate.

Test data 1 contained 13,591 response phrases, some of which were in English and others were in French. Test data 2 contained 2,540 response phrases, which were in French only. Finally, test data 3 contained 10,877 English response phrases. While test datasets 1 and 2 had the same type of data, test data 3 was different (recall that different data types can be coded, such as industry, occupation, commodities, leisure activities, etc.).

In order to compute the accuracy rate of the alternative method and compare the results to the current method, truth files which contain the correct reference codes for all three test datasets must be available. These truth files were reviewed by expert coders.

To test the alternative method, the steps depicted in Graph 1 were followed. First, both response and reference files were parsed by the parsing strategy that is currently used for the coding operation. After parsing, instead of applying the current

methodology to find the best match, the alternative method was used. A threshold value was also pre-determined for each test dataset, and a response was accepted as coded only when the score of this response phrase was greater than or equal to the pre-defined threshold value. It should be noted that regardless of the method being used, the best score for direct matches is always 10. Therefore, direct matches found by both methods are identical and indirect matches could be different between the two methods. Hence if the coding rate of indirect matches for the alternative method is higher, or if the accuracy rate of this alternative method is higher, then the objective of improving the efficiency of the automated coding process will ultimately be achieved.

4.2 Test Results

Since the scores and threshold values of the current and alternative methods are not necessarily comparable, for testing purposes, the coding rates were compared under the condition that the accuracy rates were held the same for both methods. This could be achieved in the test environment because the truth files were available for calculating the accuracy rate. Therefore, the threshold values used in the alternative method could be manually adjusted to obtain the same level of accuracy as the one for the current method. Other options could also have been used. Table 3 compares the coding and accuracy rates for the G-Code and Generalized Levenshtein methods. Note that test data 3 shows the lowest accuracy rate due to the fact that the type of data in dataset 3 is known to be more difficult to code.

Table 3 – Coding and accuracy rates for the G-Code and Generalized Levenshtein methods

	Coding Rate		Accuracy Rate	
	G-Code	**Levenshtein**	**G-Code**	**Levenshtein**
Test Data 1	15.6%	16.5%	97.3%	98.1%
Test Data 2	61.6%	63.6%	66.7%	67.4%
Test Data 3	58.1%	60.5%	36.3%	36.4%

While holding the accuracy rate to be the same for both methods, Table 3 shows that the coding rates of the Generalized Levenshtein method are higher than the rates of the G-Code method for all datasets (0.9%, 2.0% and 2.4% for test data 1, 2, and 3, respectively).

These results show that on average, while achieving roughly the same level of accuracy, there are more records being coded using the Generalized Levenshtein method than using the G-Code method. Even though these increases appear to be a minor improvement, they still mean that less manual coding is required while achieving the same level of precision, especially when one considers that a typical dataset to be coded would have thousands of records.

4.3 Combined Method

The Generalized Levenshtein method uses a completely different approach compared to the current method to find the best matching reference phrase. Consequently, a record with a high score assigned by the current method may not necessarily receive a high score when using the Generalized Levenshtein method, and vice versa. This statement was confirmed when comparing the coded results to see how many records were coded by both methods and how many were coded by one but not the other.

Table 4 – Distribution of coding results

	Both methods	G-Code only	Levenshtein only	Total
Test Data 1	2,037	79	158	2,247
Test Data 2	1,375	194	247	1,816
Test Data 3	5,013	1,311	1,564	7,888

As shown in Table 4, 2,037 response phrases in test data 1 were coded by both methods, 79 responses were coded by the current G-Code method only and 158 responses by the Generalized Levenshtein method only. In test data 2, 1,375 were coded by method, 194 by the current G-Code method only and 247 by the Levenshtein method only. In test data 3, there are 5,013 coded by both methods, 1,311 by the G-Code method only, and 1,564 by the Levenshtein method only. These results lead to the idea of combining two methods in order to take advantage of the strength of each approach.

The objective of combining the two methods is to maximize the number of records being coded. In the next phase of the experiment, first, the Generalized Levenshtein method was used to attempt to code the response phrases and then the G-Code method was used to code records not coded by the first method. In order to make the test results comparable to those produced by the G-Code method, the threshold values for the Generalized Levenshtein method were chosen so as to hold constant the accuracy rate, as shown in Table 3.

Table 4 lists the results from the G-Code, Generalized Levenshtein, and combined methods. As predicted, the highest coding rate was achieved by combining methods while maintaining the same level of accuracy.

Table 4 – Coding and accuracy rates for the G-Code, Generalized Levenshtein and combined methods

	Coding Rate			Accuracy Rate		
	G-Code	Levenshtein	Combined	G-Code	Levenshtein	Combined
Test Data 1	15.6%	16.5%	17.3%	97.3%	98.1%	97.5%
Test Data 2	61.6%	63.6%	70.5%	66.7%	67.4%	65.6%
Test Data 3	58.1%	60.5%	72.5%	36.3%	36.4%	32.2%

In test data 1, there is a 1.7% increase in the coding rate from the G-Code method to the combined method for the same level of accuracy. In test data 2, there is an 8.9% increase in the coding rate for a small decrease (0.9%) in accuracy.

The most significant improvement in coding rate (14.4%) appeared in test data 3. However the accuracy rate drops by 4.1%. The drop in the accuracy rate is due to the fact that correctly coded responses are mostly coded by both methods. These responses are possibly easier to code than the records that are coded by only one method. For instance, response phrases that are complete and contain fewer spelling errors are more likely to be coded by both methods. Therefore, the marginal gain of additional records coded by the latter method will most likely come at the cost of a higher error rate.

As predicted, combining the current and alternative methods allows more response phrases to be coded, as it takes advantage of two completely different approaches to finding indirect matches. The threshold values of the alternative method were chosen for testing and comparison purposes only and are by no means the most optimum values. The user should conduct a study using data from his/her operation before deciding on the threshold values.

5. CONCLUSION

Using an automated coding system helps to improve the timeliness of the coding process, reduce the operation costs, while producing consistent results. Increasing the number of response phrases accurately coded by the automated system means fewer phrases requiring to be coded manually.

The quality of the coding results depends on factors such as the completeness of the response and reference files and the parsing strategy and the threshold values. Ongoing maintenance of the files and the parsing strategy is important to maintain or increase the coding and accuracy rates. The implementation of a quality control process for monitoring the coding results could also improve the quality of the outcome. Every step of the coding process requires on-going effort to limit errors and ensure the quality of the automated coding results.

The quality of the coding results also depends on the methodology used to find indirect matches. The Generalized Levenshtein method proposed in this paper shows promising results and will be implemented into a future version of G-Code, allowing the user to use either method or both.

ACKNOWLEDGEMENTS

The authors would like to acknowledge valuable contributions by Peter Law, and thank Laurie Reedman and Michael Wenzowski for their useful comments on earlier versions of this paper.

REFERENCES

Levenshtein, V.I. (1965). *Binary codes capable of converting deletions, insertions, and reversals*, Soviet Physics Doklady, Vol. 10, No. 8, pp 707-710.

Rathwell, S. (2002). *A Review of ACTR Weighting and Scoring in the Context of Census Industry Coding*, internal document, Statistics Canada, Ottawa, Canada.

SAS Institute Inc., *SAS OnlineDoc® 9*, http://v9doc.sas.com/sasdoc/

Statistics Canada (2003). *Survey Methods and Practices*. Ottawa, Canada.

Wenzowski, M.J. (1988). *ACTR, a Generalized Automated Coding System*, Survey Methodology, Vol. 14, No. 2, pp 299-307.