Catalogue no. 11-522-x
ISSN: 1709-8211

Proceedings of Statistics Canada Symposium 2021
Adopting Data Science in Official Statistics to Meet Society's Emerging Needs

Integrating Machine Learning into Coding of the 2021 Canadian Census Using fastText

by Andrew Stelmack

Release date: November 05, 2021

Proceedings of Statistics Canada Symposium 2021
Adopting Data Science in Official Statistics to Meet Society's Emerging Needs

Integrating Machine Learning into Coding of the 2021 Canadian Census Using fastText

Andrew Stelmack[1]

Abstract

As part of processing for the 2021 Canadian Census, the write-in responses to 31 census questions must be coded. Up until, and including, 2016, this was a three stage process, including an "interactive (human) coding" step as the second stage. This human coding step is both lengthy and expensive, spanning many months and requiring the hiring and training of a large number of temporary employees. With this in mind, for 2021, this stage was either augmented with or replaced entirely by machine learning models using the "fastText" algorithm. This presentation will discuss the implementation of this algorithm and the challenges and decisions taken along the way.

Key Words: Natural Language Processing, Machine Learning, fastText, Coding

1. Introduction

1.1 Motivation

The 2021 Canadian Census asks 31 questions that have the possibility of a write-in response that does not correspond to one of a few "check-box" options provided to the respondent to select. For example, the question "What language(s) does this person speak on a regular basis at home?" has hundreds of possible responses but the respondent will only see three options: *English*, *French*, and *Other language(s) – specify*. It is the responsibility of the census coding processing team to assign the write-in responses to the *Other language(s) – specify* option a coded value. As a fictitious example, write-ins of "German" may have to be assigned to code 12345 and "Italian" to code 67890.

At first, this task does not seem overly complicated; however, respondents are not bound in how they may respond to any such question. Their responses are likely to include any number of different spellings or even responses completely unrelated to the question at hand. Adding to this, the number of responses requiring coding ranges from approximately 2,000 to 23,000,000 depending on the variable being coded. These factors make the coding process an extremely large undertaking which takes approximately 10 months to complete. Due to this complexity, Statistics Canada made the decision to augment its coding process with machine learning applications for the 2021 census cycle. This paper will describe the addition of machine learning into the existing process and highlight the many ways that this project differed from the traditional machine learning workflow and the decisions taken to overcome these differences.

1.2 Traditional Coding Process

Traditionally, this process has been completed in three steps: auto-coding, interactive coding, and code-fix. Auto-coding is the process of matching the write-in response to a reference file of expected responses created by subject matter experts, i.e. correct spellings of responses or very common misspellings. If a response matches the reference file, it receives the code on the reference file and moves on to code-fix. For most variables, the vast majority of write-ins will be coded through this step. Adding to that, this stage is believed to have a very high level of accuracy, and in fact, for most variables, if the reference file is well maintained, this step should have near perfect accuracy. These two factors lead to a overall high level of quality for the coding process.

[1] Andrew Stelmack, 100 Tunney's Pasture Driveway, Ontario, Canada, K1A 0T6, andrew.stelmack@statcan.gc.ca

Anything that does not match the reference file is then sent to interactive coding. Interactive coding is a manual coding step where a human being codes each response one at a time. While Statistics Canada has experienced coders working for various programs, due to the sheer volume of census data, and the length of time between cycles, each cycle Statistics Canada is required to hire hundreds of temporary employees who spend approximately 10 months completing this step. As was the case with auto-coding, any records completed in interactive coding move to code-fix.

Code-fix is an ongoing step throughout the coding production stage (actually occurs in parallel with interactive coding) where subject matter experts may query the database to correct any responses they deem in need of correction. This can either be done by querying one record at a time, or more often, by querying in bulk where the subject matter expert may, for example, find all responses containing a certain word and bulk correct them to one code.

1.3 Switch to Machine Learning

As mentioned in the previous section, the interactive coding step requires the hiring of many temporary employees to work over several months. This process is very expensive financially as well as time consuming. Adding to these issues, as these employees are temporary in nature with limited training available to them, they are more prone to errors than experienced Statistics Canada coders. Depending on the variable, it has been estimated that interactive coding has an error rate anywhere from approximately 5% for an easier to code variable such as place of birth which requires a response of a single word (or very few words) to approximately 30% error for a variable such as industry, where the respondent gives a lengthy description of their occupation. With a desire to automate as much of the process as possible, it was determined that machine learning could potentially offer a less expensive and more timely alternative to interactive coding.

The primary cost associated with machine learning is tied to the development of the models and was completed by employees already on staff. While it is true that there is a cost to retaining the services of these employee for this purpose, the machine learning models offer a long term investment compared to the temporary nature of hiring interactive coders each cycle. In terms of time efficiency, machine learning occurs near instantaneously during production, as soon as a response is brought into the processing system, machine learning is able to assign a code which can then be analyzed in code-fix. In traditional coding, the subject matter experts would have to wait for records to be completed in interactive coding before they could begin reviewing them. This extra time spent in code-fix should lead to higher data quality overall. In considering these factors; time, cost, and quality, it was decided that for 2021 machine learning would replace either all or some of the interactive coding step for most variables (the auto-coding step would remain).

1.4 fastText

After an initial exploration period it was determined that the "fastText" algorithm would be the algorithm of choice for the 2021 Census. FastText is a natural language processing algorithm developed by Facebook within the past ten years. Simply, classification using fastText uses a neural network to transform an input string into a "word embedding", that is, a numerical vector representation of the string that can then be transformed into class probabilities using softmax regression. A more thorough exposition of the algorithm can be found in (Joulin, 2016.). FastText operates using the command line and is officially supported for use in Python. For its use at Statistics Canada a graphical user interface (GUI) for fastText was created inside Statistics Canada's in house generalized coding system: G-Code. This GUI is essentially a wrapper that provides the user a tool with which to perform hyper-parameter selection in an automated process much simpler than creating one from scratch using the command line. Unfortunately, this interface is limited in its ability to adapt to a situation that differs from the typical machine learning workflow. As will be described in a later section, this project had many quirks that meant the use of this interface was not suitable, and instead fastText was used from a command line call within the R environment.

2. Model Development

2.1 Data Preparation

As mentioned previously, this project had many peculiar challenges that required attention; in particular, no two variables were necessarily treated in the same manner. That said, the following describes the data preparation process in an ideal situation. Ideally, models would be built using data from the previous census cycle, that is, data from the 2016 Census. This data would be split into training and test sets in a ratio of approximately 75% training and 25% test. K-fold cross-validation would then performed on the training portion of the data to select the optimal set of hyper-parameters (of which fastText has many).

The first issue that moved us away from this ideal is that the 2016 data was not all coded using the same process. Some of the data was auto-coded and some was completed by interactive coders. As mentioned previously, machine learning was only intended to replace the interactive coding step and not the auto-coding step. Due to this, having auto-coded records in the test folds (or the validation fold of cross-validation) would lead to incorrect assessment of the model as the model would never code these records in production. Auto-coded records are typically easier to code than their interactively coded counter-parts and including them in the test or validation portion would give an optimistically biased view of the model's performance. That said, auto-coded records do offer value as training data, as it is valuable for the model to learn these perfect spellings and common misspellings. So the appropriate split would then be; that all auto-coded records belong in training whereas the interactively coded records are split into training and test in the 75/25 ratio. This, however, makes performing hyper-parameter selection in the G-Code wrapper impossible. G-Code simply receives a training data set and randomly splits it into k new datasets with a portion for training and a portion for validation. This new split cannot take into account what is auto-coded versus interactively coded and thus some auto-coded records would inevitably end up in the validation set possibly leading to the selection of a sub-optimal hyper-parameter set.

As mentioned previously, this was alleviated by using an R script written in-house that allowed for proper splitting of the data. For variables with only a small amount of available data, a nested cross-validation procedure was employed with a 5-fold inner loop and a 5-fold outer loop. For these small datasets, this allowed us the largest possible evaluation set to evaluate the model. For larger datasets, a split similar to the ideal was used, and for variables with a very large number of records (on the order of a few million) a single 75/25 training test split was employed, followed by a second 75/25 split of the training set to form the validation set as opposed to cross-validation as computational power became an issue with this amount of data.

2.2 Change in Code-Set

Another complication that was encountered was where the 2016 data that was intended to be used as training data was not initially suitable to be used in that manner. For many of the variables that were required to coded, the code-set (the list of possible codes to be assigned) underwent significant changes from the previous cycle. The reasons for this vary, from a change in scope for the variable, to a change in industry standards. This is an obvious issue as the model only learns from the training data, and using training data with a 2016 code-set and expecting results that use a 2021 code-set is illogical. Thus, the 2016 data must be "corrected" to contain the values of the new 2021 code-set. The simplest case of this could be a label change, e.g. "Teacher" has its code changed from 1000 to 1001. This is corrected simply by using a concordance file that changes the 2016 codes to their 2021 versions. A more complicated case, but just as easy to correct, would be a case where a write-in becomes more general, e.g. "Elementary Teacher" – 1001 and "High School Teacher" – 1002 in 2016, both become "Teacher" – 1000 in 2021. Again, this can be corrected through the use of a concordance file. The most difficult case to correct is that of a label becoming more granular, e.g. "Teacher" has its code changed from 1000 to either "Elementary Teacher" – 1001 or "High School Teacher" – 1002. This "splitting" of a code can not be corrected through a simple concordance file and, essentially, can only be corrected through manual intervention by subject matter experts.

The first step to recode these records to the 2021 code-set, was to run the 2016 write-ins through the 2021 auto-coding process. Any record that passed auto-coding had the appropriate 2021 code assigned. Unfortunately, this left the remainder that had failed auto-coding with 2016 codes. Subject matter experts then had two options: recode these records to the appropriate 2021 code through either a concordance file when possible or through manual intervention, or create a model using only these auto-coded records. The first option was not be feasible in all cases, as for codes where the concordance was one-to-many, it amounted to a complete redo of the coding process. However, the second option will necessarily create a model that is worse than the first as not only does it lack the data of the first, it lacks the exact data that we are trying to model. That is, the model will only learn from either perfect or near perfect spellings of words but will be trying to code the opposite of that in the future. This method also has the drawback of potentially selecting a sub-optimal set of hyper-parameters. As the model is trained only on auto-coded records, the hyper-parameter search is also validated using only auto-coded records. The hyper-parameter set that best predicts perfectly spelled words may not be the set that best predicts poorly spelled words.

These issues aside, for most variables, subject matter experts opted for the simpler method of training the model exclusively on auto-coded records. These models were evaluated on a sample of hand-coded 2016 write-ins that had failed auto-coding and the models were all deemed to be of suitable quality.

2.3 New Variables for 2021

Another scenario in which the 2016 data was not suitable for training was the case when the data was not even available in that the variable was new for the 2021 cycle. This was the case for a few of the variables requiring models in 2021. The first solution to this issue would be to train the model using the reference file (the file of common and expected responses used for auto-coding). Coding using the reference file was certainly a viable option; however, seeing as these questions were not asked previously it may have been difficult for subject matter experts to envision a list of all possible expected responses. Thus, a model trained solely on the reference file might have not seen the entirety of the possible population of responses.

Another solution, and one that we used in tandem with the reference file, was to use data from another survey as training data. For example, the question related to gender was new for the 2021 Census, but had been asked on other surveys provided by Statistics Canada. As in the case of a code-set change, the write-in responses from these surveys had to be ran through our 2021 auto-coding procedure to be given a code that was valid in our code-set. Any records that failed auto-coding then had to be manually coded by subject matter experts or as was the case of a code-set change, had to use a model relying solely on auto-coded records. When this situation arose, subject matter experts chose to manually code the remaining records. This was more feasible than it was in the case of a typical code-set change since the amount of records obtained from other surveys (~ 1,000 - 30,000 records) was orders of magnitude less than what was available from 2016 census data.

3. Evaluation

3.1 Match Rate vs. Accuracy

In a typical machine learning workflow, records in the test and validation sets have the value assigned by the model compared to the true value in order to arrive at some evaluation metric. As part of the interactive coding process, there exists a quality assurance sample that is completed in order to estimate the quality of the codes assigned by interactive coders. For most variables, this estimated accuracy ranges around 80%-90%, but for a couple of complicated variables with longer write-in responses (industry, and occupation) the estimated accuracy dips to around 65%-75%. Due to this level of inaccuracy amongst the codes in the 2016 data, a comparison between the model-assigned code and the 2016 interactively-assigned code (referred to hereafter as “match rate”) will lead to poor model evaluation as there is no way to determine if the model disagrees with the assigned code due to an error by the model or an error by the interactive coder.

Consider an example case where the accuracy and match rate are both relatively high, an estimated interactive coder accuracy of 90% and a match rate of 85%. The true accuracy of this model could be anywhere from 75% to 95% depending on whether the model matches the interactive coder when the interactive coder was correct or when in fact the interactive coder was incorrect. This wide range (that only gets wider as the estimated interactive coder accuracy and match rate decrease) makes this an ineffective method of evaluating the models, or at the very least requires some secondary metrics to be used in conjunction with it. Thus, it was necessary to explore alternative ways to evaluate our models.

3.2 Code-Fix Records

One such alternative was to consider records that were previously completed using code-fix for evaluation. These are records that would have been completed in interactive coding (and thus would exist in the test and validation sets), but would have been reviewed by a subject matter expert during code-fix. Presumably, having been completed by a subject matter expert, these responses should have less error in them than those completed solely by an interactive coder.

However, as mentioned previously, records are often code-fixed in bulk and thus are not free from error. Due to time constraints, a subject matter expert may query a few thousand records that all contain a similar write-in, potentially all containing a common word, and bulk change them all to the same code, understanding that a few may not actually belong to that code. As seen in the previous section, even small error in the labels can lead to large variance in the estimates of model accuracy, presenting a possible issue with this method. Further, records that were evaluated in code-fix are not representative of the population. Beyond the bulk review, certain records are also flagged as "refer to subject matter expert" due to the interactive coder not knowing what to assign. That is, the distribution of code-fix records will skew towards more difficult write-ins than the overall population leading to a pessimistic view of machine learning performance.

Finally, it is difficult to compare how well the model performs relative to interactive coding using these records as by nature of going to code-fix the record is likely to change from its interactively-assigned code. Thus, a model accuracy of 80% amongst code-fixed records compared to an interactive accuracy of 5% amongst those records does not necessarily reflect a 16 fold improvement of machine learning over interactive coding.

3.3 Interactive Coder Error Rate Estimation

Another solution we considered was using records that were part of the previously alluded to interactive coder error rate estimation process. During coding, a sample of records are completed by two interactive coders. If the second coder agrees with the first (with no knowledge of the code the first coder assigned) then the code is deemed to be correct. If not, it is sent to an arbitrator who has a higher level of knowledge than the typical interactive coder. If this arbitrator agrees with the first coder the code is deemed correct and incorrect otherwise. This sample is the basis for the interactive coder error rates mentioned throughout this paper. Since the algorithm had at this point already been determined, the main purpose of the evaluation process is to determine to what extent machine learning may or may not offer improvement over an interactive coder. With this in mind, using the records that were used to determine the error rate of interactive coders seemed like a logical choice.

To do so, the machine learning code was considered as the code of the "first coder". If the second coder agreed with machine learning, then machine learning was considered correct. If they disagreed but the arbitrator agreed with machine learning, again machine learning was considered correct. The difficulty in using this method is in the case where machine learning and the first coder disagreed but the second coder agreed with the original first coder. In this case, there was no arbitrator value to compare to machine learning that would have been used if machine learning truly were the first coder. This issue inherently gives a pessimistically biased view of the model's performance.

Another drawback is the assumption that if two interactive coders agree, then the code is correct. The interactive coders received the same training, often code near each other and will ask each others opinions. Thus, they are likely to make the same mistakes. In fact, a small study was done where a sample of records was taken where the two interactive coders agreed and was recoded by a person with more expertise. This "expert" agreed with the two interactive coders only approximately 70% of the time. Since evaluating machine learning using these records also makes this assumption, the evaluation may not be accurate. This method (as well as the code-fix method described previously) is also unusable in the case of code-set changes that have not been concorded, as the models code will be assigned using the 2021 code-set and cannot be compared to the second coder whose code was assigned with the 2016 code-set.

3.4 Truth Deck

The last method of evaluation considered was that of a “truth deck”. That is, a sample of previously interactively coded records recoded one-by-one by a subject matter expert. This sample would ideally be without error and thus alleviate the issues described earlier with comparing the model-assigned code with the interactively-assigned code. This method seems ideal in its simplicity and quality, however also has its issues. First, it is possibly very time consuming. Depending on the variable, a sample of only a few thousand records could take anywhere from a few days to even weeks to complete. Finding a subject matter expert that has the ability to recode the records correctly and at the same time the availability to spend potentially weeks doing so is not an easy task.

Secondly, even though the records are being recoded by subject matter experts, it is possible that the write-in is somewhat ambiguous with no guarantee that two different subject matter experts would arrive at the same code. Or more simply, the subject matter expert may just make a mistake, which leads back to the same original issue of comparing to an erroneous interactively-coded record. This “truth deck” is still likely to have error. That said, all of the options listed have drawbacks and, perhaps for its simplicity in understanding, the truth deck was our evaluation method of choice. It also had the benefit that records used for the truth deck could be added to the training data after model evaluation was complete in models that were trained using auto-coded records only.

4. Pre-Production Evaluation

While truth decks were created for all variables, their ability to provide useful information was limited by whether an interactive code concordance was available from 2016 to 2021 as described earlier. An accuracy metric can be calculated using the truth deck for all variables; however, if there is no concordance between 2016 and 2021 codes, the only metric to which to compare this accuracy is the estimated interactive coder accuracy from 2016.

As described previously, this accuracy estimate relies on the assumption that two interactive coders arriving at the same code means the code is correct and as such this estimate is not reliable. Also, the 2016 interactive accuracy estimate is based on a process that uses the 2016 code-set. For the same reasons that a concordance between 2016 and 2021 was not possible, a comparison of 2016 interactive accuracy and 2021 truth deck accuracy may not be valid. That is, the change in code-set may be due to change in scope for the question, rendering the two no longer comparable. More often, when evaluating the suitability of the machine learning model to move into production, subject matter experts were presented with the truth deck results and asked to determine machine learnings suitability independent of previous estimates of interactive coder accuracy. For all but a single variable, subject matter experts were comfortable enough with the accuracy level of the machine learning models to move forward with production.

For variables where the code-set did not change so significantly – that is, that subject matter experts were able to supply a concordance between 2016 and 2021 codes, – the truth deck provided the addition of a direct comparison between machine learning and interactive coding. For one such variable, “country” (a model made up of a handful of questions whose response was a country) among the 2,910 unique write-in responses supplied to the truth deck, machine learning was correct 84.33% of the time while interactive coding was correct at a slightly higher rate of 84.58%. While interactive coding slightly outperformed machine learning amongst the truth deck write-ins for this variable, the time and cost savings from the machine learning model afforded subject matter experts the opportunity to more than make up the difference in code-fix.

5. Conclusion and Future Considerations

Machine learning through the fastText algorithm has been successfully implemented into the coding processing step of the 2021 Canadian Census for all but one of the variables it was considered for. Currently in the production phase, we are in the process of completing a quality assurance test that will see a “truth deck” of responses coded by machine learning recoded by subject matter experts. After the 2021 cycle, research will begin toward improving the process for the next census cycle in 2026, including the possibility of having machine learning code previously auto-coded records, alternative evaluation metrics, as well as active learning techniques for cases where a concordance between code-sets is not possible. We envision that machine learning will take on more of a role in the coding process looking forward to future census cycles.

References

Joulin, A., Grave, E., Bojanowski P., and Mikolov T. (2016), “Bag of Tricks for Efficient Text Classification”, *arXiv preprint arXiv:1607.01759v3*.