
# Irish 1911 Census Occupation Data Classification

Wendong Fan

19209629

The thesis is submitted to University College Dublin in part fulfilment of the requirements for the degree of

**M.A. Statistics**

School of Mathematics and Statistics

University College Dublin

**Supervisor**: Prof. Brendan Murphy

August 2021

# Table of Contents

# List of Tables

# List of Figures

# Abstract

Traditionally, the classification of occupation data is done manually, but coding occupations into standard classifications is time-consuming and complicated, especially when there are a large group of occupational types. At the same time, the results of manual coding are not always consistent because manual classifications are often very subjective. In order to solve these problems, we use a hybrid algorithm that combines the k-Nearest Neighbours (k-NN) method and the rule-based system to encode unlabelled occupation data into standard classifications automatically. We also comprehensively compare the performance of ten different distance metrics in calculating occupational string distance, and the Jaro-Winkler distance is selected as the best distance metric in this project. We use 4,384,247 personal data from the Irish 1911 Census and 5,285 manually labelled occupation types data from Central Statistics Office (CSO) in Ireland as the training set. We also propose using algorithms to perform self-checking on manually labelled training data to improve the quality of the training set. The result shows that the combination of the k-Nearest Neighbours (k-NN) method and the rule-based system accurately classify real-world occupation data.

# Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Brendan Murphy and my manager Sanela Smith for their support, patience, and understanding. Most importantly, thank you for guiding me to apply theoretical knowledge in practice and give me correct guidance to the right direction. Your guidance helped me throughout my study, work, and thesis writing.

# 1. Introduction

In any census, occupational information is collected as an essential piece of data for statistical purposes. By looking at the changing trends of occupations, we can provide valuable insights into how society evolves and the effectiveness of policymaking across generations. Usually, occupation data is collected through self-completed questionnaires or by interviewers. In either case, for secondary reuse in subsequent analysis, a coding system will be applied to convert occupation types into different categories, and then they will be matched with appropriate standard codes.

One of the main reasons for the development of occupational classification is to provide an easy-to-measure way. We all need to obtain occupational information to inform companies, government agencies, students and career consultants about skill needs. Levels, trends and changes are an ongoing theme of national and local labour policies(Akcigit et al., 2013). In addition, occupational classification is deeply rooted in sociology, and it is an internal factor that measures inequality, social stratification and class mobility (Weeden and Grusky, 2005). Occupation classification is also indispensable in economic analysis. It describes structural changes caused by technological progress, automation, globalization, and changes in immigration laws (Alonso-Villar et al., 2012).

Traditionally, the classification of occupation data is done manually, but coding occupations into standard classifications is time-consuming and complicated, especially when there are a large group of occupation types. For example, from the study of Burstyn et al. in 2014, it can take four to eight years for one trained coder to classify a database of 200,000 occupations. The United States Census Bureau spent approximately seven million dollars categorizing 17 million occupation types manually in 1980 (Chen et al., 1993). Secondly, the results are not always consistent because manual classifications are often very subjective. For example, Koeman et al. conducted a study in 2013, in their study, two human coders classified 220 different kinds of

occupation types, but the two coders only got 55% consistency of the codes, which shows a big difference in outcome.

In order to save costs and improve efficiency, a method is needed to encode occupation data into high-quality standard classifications automatically. Machine learning is expected to increase automation and reduce costs. Algorithms from machine learning have been applied to text data, usually to classify large amounts of text while coding with a small number of resources (Sebastiani, 2002). On the surface, occupational coding and text classification are similar: we need to detect signals (i.e. occupations) and assign appropriate labels to a large number of text responses. Various algorithms in machine learning and other fields have been tested to improve the efficiency of occupational coding (Sebastiani, 2002).

However, occupational coding is different from other text classification due to the following two main reasons:

1) The input of occupation data is usually very short which may contain spelling errors.

2) The occupation classification problem is high-dimensional.

These make it challenging to automatically extract signals from the text. Since the occupation text is often a particular term, the automatic spelling correction is more challenging. In this project, we have 80 different occupation categories and 167,000 occupation types. On one hand, this poses a challenge to the speed and efficiency of calculations. On the other hand, the ideal training data needs to have high accuracy and credibility, and all result types need to be included, which requires much manual observation.

The following are our three main contributions:

1) We develop an automatic occupation coding system using the k-Nearest Neighbours (k-NN) method based on the Jaro-Winkler distance algorithm and apply this system to personal occupation data from the Irish 1911 Census. We use manually

labelled occupation data as the training set to encode unlabelled occupation data into standard categories automatically.

2) We propose an automatic self-check system based on our classification method to self-check the manually labelled training data and give the result of possible manual errors to help improve the quality of the training data. This automatic self-check system can be applied to other short text classification projects.

3) We use a hybrid algorithm that combines the k-Nearest Neighbours (k-NN) method and the rule-based system to reduce errors caused by the noise from part of occupation types. The combination of k-NN and the rule-based system produces a high level of accuracy for automatic occupational classification.

# 2. Related Work

In 2000, Takahashi et al. developed a rule-based system to solve problems with manually occupation classification. In the rule-based approach, a set of rules are used for classifying occupation titles, and the ruleset is derived from the definitions of the occupations. From the study of Takahashi et al., the accuracy of this system was about 65%. Government agencies in many countries have widely used rule-based methods. Since the mid-1960s, the U.S. Census Bureau has implemented many automatic coding systems (Kearney and Kornbau, 2005). The U.S. Census Bureau created a dictionary-based automated industry and occupation coding system (AIOCS) for coding industries and occupations in the 1990 Census (Gillman and Appel, 1994). The U.S. Census Bureau also used a dictionary to automatically assign a 43% SOC code to the 2012 American Community Survey (ACS), with an accuracy rate of 95% (Thompson et al., 2012).

The rule-based system saves coders' workload and time, which is more efficient than traditional manual classification. This system can produce high accuracy by setting reasonable rules, but this approach still has some disadvantages. Firstly, it is not easy to create complete and accurate classification rules because formalizing all the respondents' occupations filled in the census table is beyond our ability. Some types of occupations have ceased to exist with the change of the times, we are not familiar with them, and there is less public information for these occupation types, so the productivity of the rule-based system is often not high. Secondly, in order to maintain the ruleset, we need to make a constant effort because the terms and expressions used by respondents to describe their occupations always change with the times. While the rule-based system is more efficient than traditional manual coding, it still needs large amounts of manual rules to ensure the reliability of this system.

Due to the success of machine learning in classification, occupation coding research is also inspired by machine learning algorithms. In 2003, Giorgetti and Sebastiani proposed a method to automatically assign appropriate codes to the response

to open-ended questions in a survey. They used supervised machine learning algorithms, including Naïve Bayesian classifiers and Support Vector Machines (SVMs). From their results, supervised machine learning algorithms can achieve significantly higher productivity than rule-based methods. Supervised machine learning infers patterns and rules from training data sets that have been labelled with the correct labels and then uses these patterns to classify new instances. Many authors used supervised machine learning algorithms to code occupations into standard classifications. Gwen et al. implemented k-Nearest Neighbours (k-NN) using the cosine similarity measure to classify occupations from the German General Survey (ALLBUS) to ISCO. In the process, a new record is assigned the occupation code of its nearest neighbour. Duplicates are considered as nearest neighbours with zero distance. Their method uses a cosine similarity measure to determine the degree of similarity between two items, with one indicating repetition and zero indicating no similarity. They achieved a 65% accuracy rate by using 10-fold cross-validation on 9,137 records, of which 52.6% of the records were the same. Tijdens et al. also attempted to incorporate machine learning algorithms by using open-ended survey questions to inform classification in 2014. They found that automatic coding is feasible if we have enough training data and they emphasized the importance of data pre-processing and the quality of algorithms.

Some authors have explored the use of hybrid models. The hybrid classification model combines multiple classifiers to obtain better performance compared with using one classifier alone. One type of hybrid algorithm is based on a combination of the rule-based system and machine learning algorithms. For example, Jung et al. provided a web-based occupation and industry coding system in 2008. This system uses a mixture of the Maximum Entropy model and the rule-based system. In the first step, the input is classified using rule-based system by logic rules. If no class was assigned by the rules, the Maximum Entropy algorithm is used to assign the class. In South Korea's 2005 census, they achieved 98% accuracy with 73% productivity. Takahashi et al. implemented manual rules and support vector machines (SVMs) in the same way. The author believes that the combined effect of the two methods is better than using them

alone. Gweon et al. also proposed a hybrid algorithm that combines the rule-based system and the SVMs algorithm. This method can be divided into two steps. Firstly, after pre-processing the occupation data, the exact match between the training data and the test data is identified. Then, SVMs classifies the remaining occupations in the test data based on the experience learned from the training data.

The hybrid classification model has outstanding effectiveness. It overcomes the shortcomings of a single model and can merge the advantages of multiple models. We will use a hybrid algorithm that combines the k-Nearest Neighbours algorithm and the rule-based system in the following.

# 3. Data Resources and Standard Principles

## 3.1 Data Resources

The data set we use comes from the Irish 1911 Census. It contains 4,384,247 personal records, covering the personal information including name, gender, religion, occupation recorded in the census. From the information in the National Archives of Ireland, although some population censuses were conducted before 1841, the quality of the census was generally poor for various reasons. In the census after 1841, the 1911 census records are considered to be of higher quality. Under the Irish Statistical Law 1993, the individual statistical census returns cease to be subject to statistical confidentiality after 100 years. It can be accessed without restriction via the National Archives for genealogical and other research. The Irish 1911 Census forms are publicly available online at the National Archives of Ireland, which facilitates this project.

In the Irish 1911 Census data, digitized non-digital categorical variables are more complex than digitized numeric variables. Among all non-numerical categorical variables, the occupation has the largest difference in type, with a total number of 166,505. These occupation data are challenging to process because they are free text values with a high degree of variability, such as grammatical forms, formal or informal language, synonyms, spelling errors, standard and non-standard abbreviations. More importantly, people today are unfamiliar with some occupation types in 1911 because they no longer exist today. There is less public information about these occupation types, which increases the difficulty of classifying them. We obtain 5,285 manually labelled data from the Central Statistics Office in Ireland. These data were labelled by different personnel, so there may have some human errors because one of the measurement challenges we face is ambiguity: vague occupation types limit our ability to assign occupations to the category it belongs to confidently. We will use these manually labelled data as the training set and later introduce the verification method based on our model on the manually labelled data.

## 3.2 Standard Principles

The standard principles we adopt are from the Irish 1911 Census Report, and we will use the same standard principles as the report to classify occupations. Irish 1911 Census occupation classification assigns a three-level hierarchy, the first level of the hierarchy is the most general and the third level is the most detailed. There are 6 broad occupational categories in the first level called 'Class', 24 major groups in the second level called 'Order', and 80 minor groups in the third level called 'Sub-order'.

Table 1 lists the first level of Irish 1911 Census Occupation Classification. Irish 1911 Census Occupation Classification major group 1: "Professional Class" occupations showing minor and unit groups are shown in table 2.

**Table 1: Occupation categories of the Irish 1911 Census occupation classification (first level)**

| Code | Occupational Categories |
|---|---|
| 1 | Professional Class |
| 2 | Domestic Class |
| 3 | Commercial Class |
| 4 | Agricultural Class |
| 5 | Industrial Class |
| 6 | Indefinite and Non-Productive Class |

**Table 2: Occupation categories of the Irish 1911 Census occupation cassification major group 1-1: 'General and Local Government' occupations showing minor and unit groups**

| Code | Occupational Categories |
|---|---|
| 1 | Professional Class |
| 1-1 | General and Local Government |
| 1-1-1 | National Government |
| 1-1-2 | Local Government |
| 1-1-3 | East India and Colonial Service |

# 4. Data Pre-processing

Before the text is automatically classified, a pre-processing step is usually carried out. Some pre-processing techniques are usually applied, such as abbreviation expansion, lowercasing, and removing punctuation marks, numbers, and extra spaces. Others like stemming and stop word removal are depending on the type of data being used. Stemming reduces words to stems. For example, both "fisher" and "fishing" are simplified to their stem "fish".

Stemming improves classification accuracy by connecting more words in many cases. However, stemming can lead to the loss of the information, especially for short texts that lack sufficient information. In each language, many common words appear in almost all documents, such as "a", "an", and "the" in English. These words are called stop words and can usually be deleted from the text because they do not play a role in the distinction between occupation categories. Removal of stop words helps reduce dimensionality, avoid over-fitting and improve classification accuracy. However, short texts such as occupation types tend to have fewer stop words compared to longer texts. Therefore, the effect of removing stop words from occupation text is often insignificant. Many studies suggest not to use stemming for short texts such as occupation types (Lin et al., 2016). The short text is also affected by feature sparseness, therefore, traditional feature selection methods will even reduce the classification accuracy (Liu et al., 2010). Looking back at previous work on occupational classification, Schierholz and Kirby et al. didn’t use stemming or remove stop word. We choose to normalize the occupation text by removing redundant meaningless punctuation and numbers. This choice can improve classification accuracy as much as possible while ensuring that the information is not lost.

# 5. Measurement

In our model, we use classification accuracy as a measure of the performance of our model. Equation 5.1 shows the definition of accuracy.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{5.1}$$

In this equation, True Positive ($TP$) is the number of records that belong to category $x$, and are classified to category $x$; True Negative ($TN$) is the number of records that do not belong to category $x$, and are not classified to category $x$ either; False Positive ($FP$) is the number of records that do not belong to category $x$, but are incorrectly classified to category $x$; False Negative ($FN$) is the number of records that belong to category $x$, but are incorrectly not classified to category $x$.

The accuracy of a classifier is determined based on how well it performs on the test data. We use the cross-validation method (Kohavi, 1995) for evaluating the accuracy of classification models. The cross-validation steps are as follows.

1) Set random seed, randomly divide the data set into 80% as the training set and 20% as the test set
2) Use our model to label the test set based on the label of training set
3) Compare the output of the label from our model with the original label, calculate the accuracy on the test set
4) Repeat steps 1 to 3 one hundred times using a different random seed each time
5) Calculate the average of the accuracies as the accuracy of the model

# 6. The k Nearest Neighbours (k-NN) Algorithm

The k nearest neighbours (k-NN) algorithm is one of the most well-known classification methods. It is very effective for various classification problem areas, and it has been applied to text classification since the early stage of text classification research (Yang and Liu, 1999).

The principle of k-NN algorithm is not complicated. Given an unlabelled object and a chosen distance metric, k-NN algorithm computes the distance of the unlabelled object to all the labelled objects in the training set. After this, we can set the value of k to get the k closest individuals in the training set to the unlabelled data as a neighbourhood of the unlabelled object. k-NN uses the majority voting (or weighted voting) strategy to choose the class labels for the unlabelled object based on the voting result of the neighbourhood.

Overall, k-NN algorithm involves three main factors: the set of labelled objects as the training set, the distance metric to compute the distance between records, the value of k to determine the number of nearest neighbours to retrieve.

## 6.1 Distance Metrics

Since k-NN is a distance-based algorithm, the classification performance is affected by the calculation of the distance (Hu et al., 2016). The calculation of the distance depends on the distance metric used, which is a mathematical representation that determines the distance between two objects.

In order to facilitate and reduce the amount of calculation, the k-NN classifier was tested with k =1. The cross-validation method was applied to calculate the accuracy of the classifier to evaluate the performance. We compare ten different distance metrics and they are listed as follows:

1) Cosine distance between q-gram profiles
2) Soundex Encoding Distance
3) Full Damerau-Levenshtein distance
4) Hamming distance
5) Jaccard distance between q-gram profiles
6) Jaro-Winkler distance
7) Levenshtein distance
8) Longest common substring distance
9) q-gram distance
10) Restricted Damerau-Levenshtein distance

We use R as the programming language (R Core Team) and use the R package "stringdist"(Van der Loo, 2014) to calculate the string distance. The "stringdist" package has a variety of algorithms for calculating the distance between two strings, which provides a convenient way for us to compare the above ten distance metric algorithms.

We use the cross-validation method to loop 100 times and obtain 100 accuracy data of each distance algorithm. Then, we calculate the average accuracy and standard deviation of the accuracy of each distance algorithm. Table 3 shows the final results, and the best results are marked in bold. Figure 1 shows the boxplot of the accuracy for 10 distance metric algorithms.

**Table 3 The performance of ten different distance metrics in three levels**

| Distance Metric | Average Accuracy for First Level | Average Accuracy for Second Level | Average Accuracy for Third Level | Standard Deviation of Accuracy for First Level | Standard Deviation of Accuracy for Second Level | Standard Deviation of Accuracy for Second Level |
|---|---|---|---|---|---|---|
| Cosine distance between q-gram profiles | 0.921 | 0.892 | 0.862 | 0.051 | 0.065 | 0.11 |
| Soundex Encoding Distance | 0.742 | 0.715 | 0.775 | 0.151 | 0.16 | 0.134 |
| Full Damerau-Levenshtein distance | 0.917 | 0.878 | 0.858 | 0.048 | 0.067 | 0.097 |
| Hamming distance | 0.814 | 0.749 | 0.79 | 0.107 | 0.133 | 0.113 |
| Jaccard distance between q-gram profiles | 0.923 | 0.882 | 0.779 | 0.041 | 0.063 | 0.181 |
| Jaro-Winkler distance | **0.933** | **0.9** | **0.886** | **0.038** | **0.054** | **0.062** |
| Levenshtein distance | 0.917 | 0.879 | 0.858 | 0.048 | 0.067 | 0.097 |
| Longest common substring distance | 0.928 | 0.891 | 0.866 | 0.04 | 0.058 | 0.094 |
| q-gram distance | 0.913 | 0.878 | 0.872 | 0.047 | 0.065 | 0.07 |
| Restricted Damerau-Levenshtein distance | 0.917 | 0.878 | 0.858 | 0.048 | 0.067 | 0.097 |

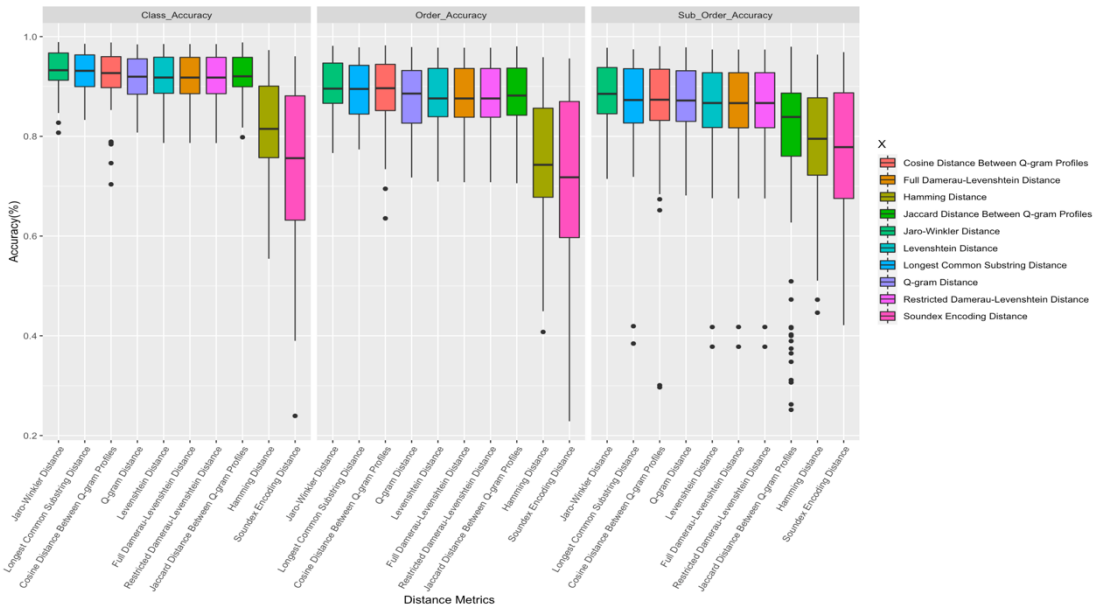


**Figure 1: The boxplot of the accuracy for 10 distance metric algorithms**

The best algorithm is the Jaro-Winkler distance algorithm. Compared with other algorithms, it has higher average accuracy. Its standard deviation is the smallest, which shows this algorithm has relatively strong stability when applied to our occupation data, so we choose to use the Jaro-Winkler distance algorithm as the distance metrics.

### 6.1.1 Jaro-Winkler Distance Algorithm

The Jaro-Winkler distance is a measurement to measure the string distance between two strings. Since the Jaro-Winkler distance performs well in matching personal and entity names, it is widely used in the areas of record linkage, entity linking, information extraction. The Jaro-Winkler distance algorithm began from the Jaro similarity algorithm found by Matthew A. Jaro which has later been developed by William E. Winkler and Thibaudeau by modifying the the Jaro similarity to give higher weights to prefix the resemblance (Wirawan, 2004). The basis of this algorithm has three parts, namely (Kurniawati et al., 2010),

1) Calculate the length of the two strings,
2) Calculate the same number of characters in the two strings,
3) Calculate the amount of transpositions.

The equation 6.1 shows the Jaro similarity formula which is used to calculate the similarity $(sim_{jaro})$ between the two strings $x$ and $y$.

$$sim_{jaro} = \frac{1}{3} \times \left(\frac{m}{|x|} + \frac{m}{|y|} + \frac{m-t}{m}\right) \quad (6.1)$$

Where:

$m$ is the same number of characters

$|x|$ is the length of string $x$

$|y|$ is the length of string $y$

$t$ is half the number of transpositions

Jaro-Winkler similarity $\left(sim_{jaro-winkler}\right)$ sets the length of common prefix up to a maximum of 4 characters and constant scaling factor for how much the score is adjusted upwards for having common prefixed into consideration. The equation 6.2 shows the Jaro-Winkler similarity formula.

$$sim_{jaro-winkler} = sim_{jaro} + \left(lp\left(1 - sim_{jaro}\right)\right) \tag{6.2}$$

Where:

$sim_{jaro}$ is the result of string similarity calculation of $x$ and $y$.

$l$ is the length of character or same prefix on string prefix before we found the existence of inequality with a maximum of up to four characters.

$p$ is a constant scaling factor for how much the score is adjusted upwards for having common prefixes. The default value for the constant according to Winkler is $p = 0.1$.

Jaro-Winkler distance can be calculated by the following equation 6.3：

$$d_{jaro-winkler} = 1 - sim_{jaro-winkler} \tag{6.3}$$

The higher the Jaro-Winkler distance value for two strings indicates lower similarity of both strings. Normal value is 1 which indicates no similarity and 0 that indicates the existence of exact similarities.

## 6.2 Dynamic k Value Method

k-NN uses the value of k to determine the number of nearest neighbours to retrieve. Its classification accuracy is sensitive to the value of k. Therefore, we usually try to use various values of k and choose the one with the highest classification accuracy.

In this project, we try k values from 1 to 10 and get the model's accuracy under different k values through cross-validation. Figure 2 shows the change of average accuracy with the increase of k value. The test results show that as the value of k increases, the model's accuracy does not improve when compared to k=1, but a certain degree of decline occurs in all the three levels.

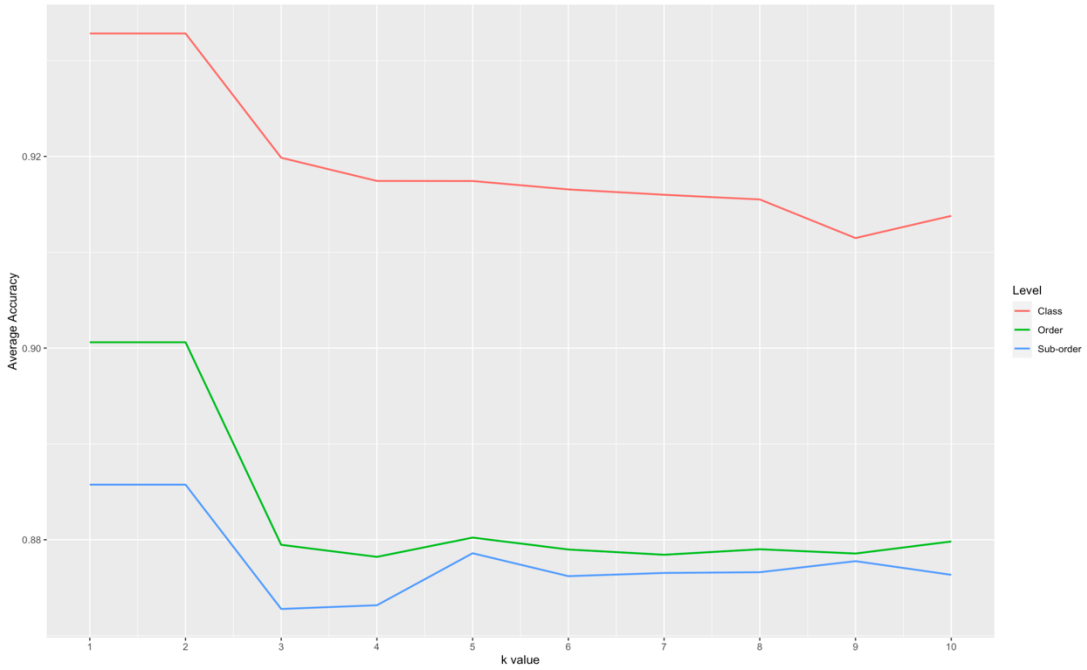


**Figure 2: The change of average accuracy with the increase of k value**

The decline in model accuracy is because for some occupation types, the specific k makes some large number of Jaro-Winkler distance matches into consideration. These matches give the wrong votes and add noise to our model, which decreases the model's accuracy. Table 4 below shows one wrong classification example when k=3. In this example, 'Dealer Fruit' will be wrongly classified into the same category as 'Dealer Fish'.

**Table 4: One wrong classification example when k=3**

| Unlabelled occupation type A | Matched occupation type B | Jaro-Winkler distance between A and B | Matched occupation type C | Jaro-Winkler distance between A and C | Matched occupation type D | Jaro-Winkler distance between A and D |
|---|---|---|---|---|---|---|
| Dealer Fruit | Dealer in Fruit | 0.122 | Dealer Fish | 0.152 | Dealer in Fish | 0.24 |

As an alternative, we use the dynamic k method to reduce the noise caused by the fixed k value. We determine the partitioning parameter by testing the average accuracy rate under different Jaro-Winkler distances. Figure 3 below shows the change of average accuracy with the increase of Jaro-Winkler distance. When Jaro-Winkler distance is smaller than 0.125, the value of our model's average accuracy is higher than 0.85 in all three levels. When the Jaro-Winkler distance is bigger than 0.35, the value of our model's average accuracy is lower than 0.5 in all three levels.

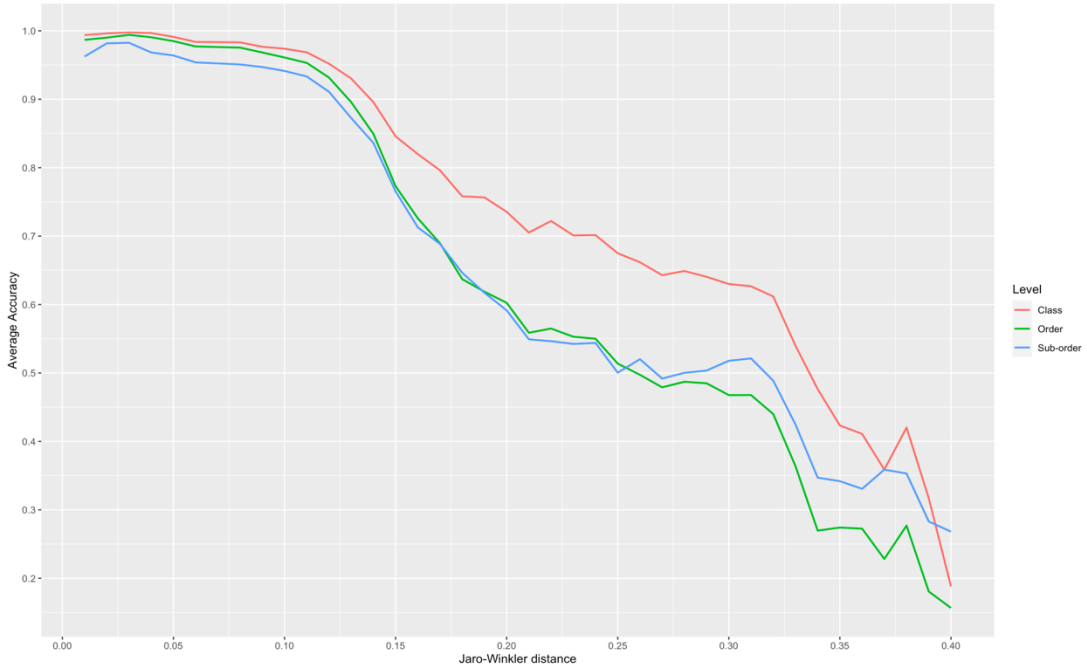


**Figure 3: The change of average accuracy with the increase of Jaro-Winkler distance**

We choose Jaro-Winkler distance to be 0.125 as the partitioning parameter. When the closest matching distance is greater than 0.125, this match is kept as the only neighbour. When there are multiple matching distances less than 0.125, all these matches are kept as the neighbours of the object. These matches are then voted to determine the classification category of the object.

By using the dynamic k value method, our classification reduces the error rate by 6.0%, 7.0%, and 5.3% at the first, second, and third level respectively.

Table 5 below shows the comparison of model performance with the one-nearest neighbour model and dynamic k model. The model with dynamic k has better average accuracy and lower standard deviation of accuracy.

**Table 5: Model performance comparison**

| Model | Average Accuracy for First Level | Average Accuracy for Second Level | Average Accuracy for Third Level | Standard Deviation of Accuracy for First Level | Standard Deviation of Accuracy for Second Level | Standard Deviation of Accuracy for Third Level |
|---|---|---|---|---|---|---|
| Model with dynamic k | 0.937 | 0.907 | 0.892 | 0.038 | 0.053 | 0.062 |
| Model with one-nearest neghbour | 0.933 | 0.9 | 0.886 | 0.038 | 0.054 | 0.062 |

# 7. Improvement of Training Set

We use 5,285 manually labelled data from the CSO in Ireland as the training set. These data were labelled by different personnel. Therefore, due to personal subjective judgment, there may be some inconsistent labels in our training set. Koeman et al. conducted a study in 2013. In their study, two human coders classified 220 different kinds of occupation data, but the two coders only get 55% consistency of the codes, which shows a big outcome difference. These inconsistent labels will produce more noise to the model and have a large influence on our classification accuracy.

In order to reduce errors caused by manual classification and to reduce noise caused by these inconsistent labels as much as possible, we output the results of inconsistent occupation label matching when using the cross-validation method to verify the accuracy of the model. We then optimize and adjust the training set according to the value of Jaro-Winkler distance

Table 6 shows the output results of partial mismatches, sorted in ascending order according to the value of Jaro-Winkler distance and the inconsistent labels are marked in red. After deleting the duplicates due to loop verification, we have 2,960 inconsistent data with Jaro-Winkler distance less than 0.2. These data need to be rechecked to improve the quality of the training set. We can see from the results that when the value of Jaro-Winkler distance is small, most the result of the mismatch is not caused by the error from our algorithm but caused by the inconsistency of the manual label.

**Table 6: Partial mismatches**

| Occupation A | Occupation B | Class A | Class B | Order A | Order B | Sub-Order A | Sub-Order B | Jaro-Winkler Distance |
|---|---|---|---|---|---|---|---|---|
| Agricultural Labourer Farm Servant | Agricultural Labourer Farm Servants | 2 | 4 | 4 | 7 | 1 | 1 | 0.01 |
| Student of Teaching Profession | Students of Teaching Profession | 1 | 1 | 3 | 3 | 4 | 0 | 0.011 |
| Agricultural Labourer Shepherd | Agricultural Labourer Shepherds | 4 | 4 | 7 | 8 | 1 | 1 | 0.011 |
| Lady's Maid Domestic Servant | Ladys Maid Domestic Servant | 2 | 2 | 4 | 4 | 2 | 1 | 0.012 |
| Pawnbrokers Assistant | Pawnbroker's Assistant | 3 | 3 | 5 | 5 | 0 | 2 | 0.015 |
| Drapery Shop Assistant | Draper Shop Assistant | 5 | 5 | 17 | 22 | 5 | 1 | 0.015 |
| Carpenter Apprentice | Carpenters Apprentice | 5 | 5 | 11 | 20 | 1 | 3 | 0.016 |
| Brick Layers Labourer | Brick Layer Labourer | 5 | 5 | 22 | 21 | 2 | 3 | 0.016 |
| Labourer in Saw Mill | Labourer in Saw Mills | 5 | 5 | 22 | 20 | 2 | 3 | 0.016 |

One of the measurement challenges we face is ambiguity: vague occupation types limit our ability to assign occupations to the category it belongs to confidently. When unifying these inconsistent labels, we will also face the same problem. For example, the occupation type of "butcher", it is difficult for us to make an absolutely clear distinction between the two categories "agricultural category about animals" and "industrial category about food". But in order to reduce noise and improve the accuracy of the model, we must give them a unified classification label. The solution we adopt is to give these types of occupations a unified label, while recording their possible other labels as notes. When necessary, we can switch between different selectable labels. In this way, we transform the uncertainty of classification into flexibility.

By removing inconsistent data caused by human error in the training set, our classification reduces the error rate by 7.9%, 5.4%, and 13.0% at the first, second, and third level respectively.

Table 7 below shows the comparison of model performance with the model using raw training set and the model using improved training set. The model using improved training set has better average accuracy and lower standard deviation of accuracy.

**Table 7: Model performance comparison after using improved training set**

| Model | Average Accuracy for First Level | Average Accuracy for Second Level | Average Accuracy for Third Level | Standard Deviation of Accuracy for First Level | Standard Deviation of Accuracy for Second Level | Standard Deviation of Accuracy for Third Level |
|---|---|---|---|---|---|---|
| Model with improved training set | 0.942 | 0.912 | 0.906 | 0.036 | 0.053 | 0.057 |
| Model with raw training set | 0.937 | 0.907 | 0.892 | 0.038 | 0.053 | 0.062 |

# 8. Rule-based System

Rule-based classifiers use many rules, which are based on multiple rules created by experts after analysing manually labelled data to construct a classification algorithm. In a rule-based system, each rule can contribute to multiple categories, and each category can be associated with multiple rules. If the unmarked occupation type satisfies the rules, the corresponding code will be assigned to that unmarked occupation type. In order to further reduce the influence of noise in the training data on our model and improve its accuracy, we merge the rule-based system with k-NN classifier.

The rule-based system we use is a logical program that uses pre-defined rules to make deductions and choices to perform automated actions. The system is designed to have two layers. The first layer is for all production occupation types. The second layer is for non-production occupation types (retirees, student, etc.). Table 8 shows part of the rules in the first layer rule-based system we set while table 9 shows part of the rules in the second layer rule-based system.

**Table 8: Part of rules in first layer rule-based system**

| Key word | First Level Label | Second Level Label | Third Level Label | Corresponding Text Category |
|---|---|---|---|---|
| domestic | 2 | 4 | 1 | Domestic Offices or Services |
| Merchant | 3 | 5 | 1 | Merchant and Agent |
| able seaman | 1 | 2 | 2 | Navy |
| carter | 3 | 6 | 2 | Roads |
| army | 1 | 2 | 1 | Army |

**Table 9: Part of rules in second layer rule-based system**

| Key word | First Level Label | Second Level Label | Third Level Label | Corresponding Text Category |
|---|---|---|---|---|
| retir | 6 | 24 | 1 | Non-Productive |
| pension | 6 | 24 | 1 | Non-Productive |
| unemploy | 6 | 24 | 1 | Non-Productive |
| Ex | 6 | 24 | 1 | Non-Productive |
| Rtd | 6 | 24 | 1 | Non-Productive |

By using the hybrid algorithm that combines the k-Nearest Neighbours algorithm and the rule-based system, an unlabelled occupation type is first labelled by the dynamic k-NN algorithm. Then key words are checked by the first layer in the rule-based system. If there exists a rule that is satisfied, it will be labelled according to the rule. Finally, this occupation type enters into the second layer in the rule-based system. If any one rule is satisfied, the label will change according to the rule.

Figure 4 is an example for occupation "Army Pensioner Infantry Clerk", based on Jaro-Winkler, it has a relatively small distance with "A Pen Industry Clerk", so it will be classified into "Wood and Bark" category; then this occupation type was checked in the first layer of the rule-based system, it has the word "Army", so it been classified into the "Army" category instead; then in the second layer of the rule-based system, it was detected to have the word "Pension", so finally it was classified into "Non-productive" category.


**Figure 4: An example for the hybrid algorithm**

By using hybrid algorithm that combines the k-Nearest Neighbours algorithm and the rule-based system, our classification reduces the error rate by 17.2%, 20.5%, and 31.9% at the first, second, and third level respectively.

Table 10 below shows the comparison of model performance with the model using dynamic k-NN and the model using hybrid algorithm. The model using the hybrid

algorithm has a significant improvement in average accuracy and the stability of the model.

**Table 10: Model performance comparison after using hybrid algorithm**

| Model | Average Accuracy for First Level | Average Accuracy for Second Level | Average Accuracy for Third Level | Standard Deviation of Accuracy for First Level | Standard Deviation of Accuracy for Second Level | Standard Deviation of Accuracy for Third Level |
|---|---|---|---|---|---|---|
| Model with hybrid algorithm | 0.952 | 0.938 | 0.936 | 0.032 | 0.043 | 0.039 |
| Model with dynamic k-NN | 0.942 | 0.912 | 0.906 | 0.036 | 0.053 | 0.057 |

# 9. Results

The average accuracy rate at the third level was 93.6% and improved when broader occupation categories were considered (93.8% at the second level, 95.2% at the first level). The coverage decreases as we move down the hierarchy since the number of categories increases. Therefore, it is expected that accuracy will decrease in the lower level.

In the first level of the hierarchy, the least frequent class has 106,487 records, whereas in the third level of the hierarchy, the least frequent class has 0 records (has 22 records in Irish 1911 Census report). Furthermore, the records are not evenly divided between categories. For instance, the class “6 Non-Productive Class” occupations and its child nodes have a significantly higher frequency than the rest of the classes: 56.5% of people belong to the class “6 Non-Productive Class” occupations, while the rest of the 5 classes each have 2.4% to 18.0% of the records.

First Level of the Hierarchy: The data covers all 6 first Level Irish 1911 Census occupation classification categories (Class). The most frequent class is the “6 Non-Productive Class” occupations with 2,476,050 records. The least frequent class is the “1 Professional Class” occupations with 106,487 records.

Figure 5 shows the data coverage for the 6 first level classes. The average frequency per class is 730,707, and the median frequency per class is 426,658.

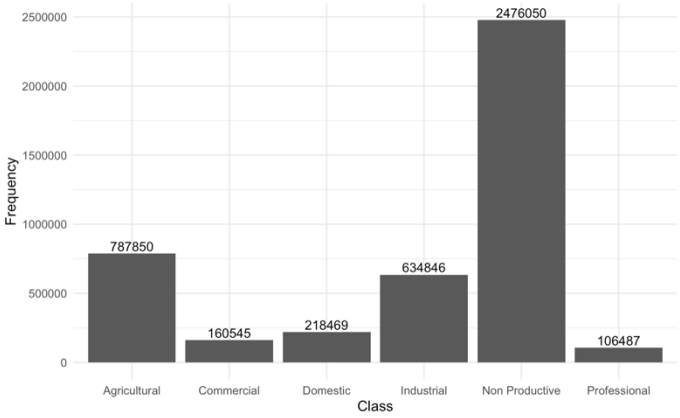


**Figure 5: The data coverage for the 6 first level classes**

Second Level of the Hierarchy: The data covers all 24 second level Irish 1911 Census occupation classification categories (Order). The most frequent order in the second level is "6-24 Non-Productive Order" occupations with 2,476,050 records, and the least frequent order is "5-23 Refuse Matters Order" occupations with 559 records. The average frequency is 182,677 records per order, and the median frequency is 32,042 records per order.

Third Level of Hierarchy: The data covers all 80 third level Irish 1911 Census occupation classification categories (Sub-order). The most frequent class in the third level is "6-24-1 Non-Productive Sub-order" occupations with 2,476,050 records, and the least frequent class is "1-1-3 East India and Colonial Service Sub-order" with 0 records (has 22 records in Irish 1911 Census report). The average frequency of records per sub-order is 54,792, and the median frequency is 3,577.

**Table 11: Summary of the data attributes**

| | **Number of Records per Categories** | |
|---|---|---|
| | **Average** | **Median** |
| **First Level** | 730,707 | 426,658 |
| **Second Level** | 182,677 | 32,042 |
| **Third Level** | 54,792 | 3,577 |

In the Irish 1911 Census report, the number of people in different categories was recorded. We compare the automated classification results obtained using our model with the numbers on the report. The result shows that in the third level of hierarchy, our model's classification reached 89.7% consistency with the figure from the Irish 1911 Census report. Table 12 shows the comparison of the number of records in the report and the number of records classified by our model in third level of hierarchy.

**Table 12: The comparison of the number of records in the report and the number of records classified in third level of hierarchy**

| Number of Records in Report | Number of Records Classified | Difference | Label | Code |
|---|---|---|---|---|
| 2573000 | 2476050 | -96950 | Non-Productive | 6-24-1 |
| 755000 | 754514 | -486 | Field and Pastures | 4-7-1 |
| 169000 | 187911 | 18911 | Mechanics and Labourers (General or undefined) | 5-22-2 |
| 152000 | 195438 | 43438 | Domestic Offices or Services | 2-4-1 |
| 96000 | 96975 | 975 | Dress | 5-18-1 |
| 75000 | 72218 | -2782 | Cotton and Flax | 5-17-3 |
| 52000 | 48538 | -3462 | Houses | 5-11-1 |
| 46000 | 40406 | -5594 | Food | 5-16-3 |
| 459282 | 511339 | 52057 | Others | Others |

As we mentioned above, since the training data set is manually labelled, there may be some inconsistent labels in our training set due to personal subjective judgment. We assign a uniform label to some occupation types that cannot be clearly defined and record their other acceptable labels as notes. If necessary, we can switch between different optional labels. By comparing with the figure on the report, we switch the labels of eligible occupation types to improve the consistency of the data from the model result and the data from the report. In the end, we achieved 92.8% consistency by switching labels.

Figure 6 is the tree plot that shows the overall result of our occupation classification compared with the result from the report. All 4,384,247 people were

classified into 80 different third-level categories. The area of the category shows the number of people in it. The bigger the area, the more people there are. Almost half of the people belong to “Non-Productive”. The colour represents the difference of the figure we get with the figure from the report. Red means we have a much smaller figure compared with the figure from the report, green means we have a bigger figure. Most categories have good consistency, and their colour is light yellow. However, we still have significant gaps in some categories.

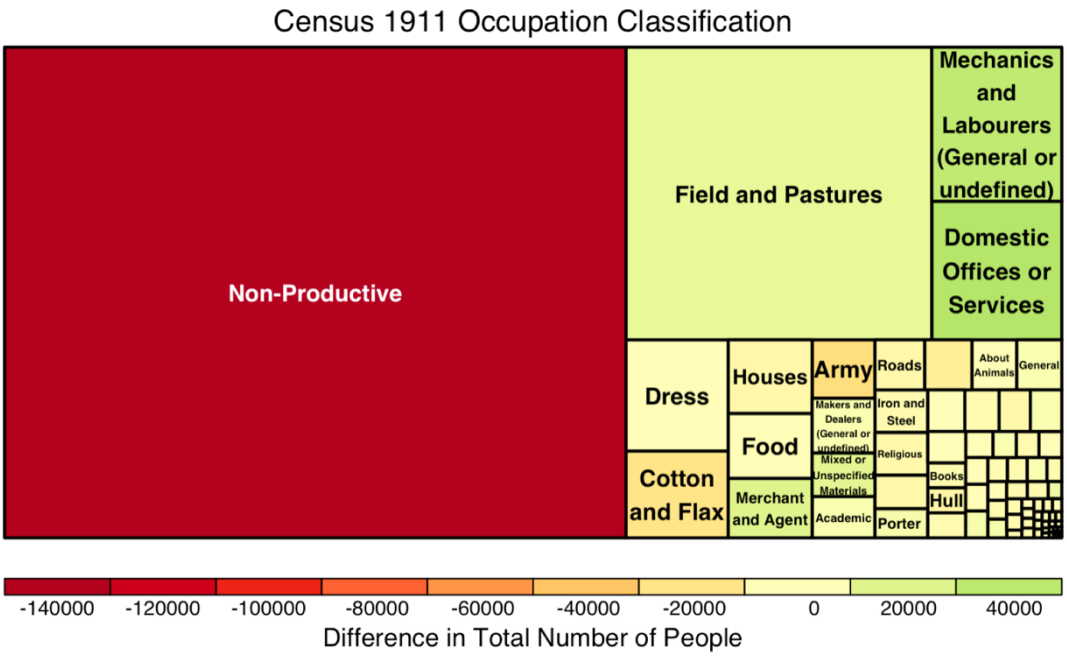


**Figure 6: Tree plot shows the overall result**

# 10. Discussion and Conclusion

We combine the k-Nearest Neighbours (k-NN) method and the rule-based system to encode unlabelled occupation data into standard classifications automatically. In order to improve the accuracy of classification, we select the Jaro-Winkler distance algorithm as the distance metric for k-NN algorithm through comparison and apply the dynamic k value method. In addition, we also improve the reliability of manually labelled training set by using the model for self-checking. In the end, our model correctly gives 95.2% of the people's first-level occupation labels on the training set.

However, there still have errors in our classification results. There are two main types of misclassification, one is unavoidable misclassification, which results from variation in the training data. The other is avoidable misclassification, which results from the inherent limitations of the classification algorithm.

The first type of misclassification is unavoidable because it arises from the limits of manual classification already discussed. This type of inaccuracy cannot be overcome by any classification models. In theory, the misclassifications described above might be reduced by providing more training data, tuning parameters. We have proposed a method to improve the quality of the training set by using the model for self-checking, and it has been proven to be effective.

The misclassification resulting from the inherent limitations of the classification algorithm can be reduced by computer-assisted coding. We can use the value of Jaro-Winkler distance as the reference threshold. If the value of the automatically classified Jaro-Winkler distance is higher than a certain threshold, a manual encoder is required to check the classification. The manual inspection helps to maintain a high accuracy rate and understand why the algorithm cannot correctly classify the text. For example, in our classification results, by observing the output, we can find that the occupation labels for 27,503 populations (0.63% of the entire population) have the Jaro-Winkler distance from 0.25 to 1. As mentioned above, we get the average accuracy rate under

different Jaro-Winkler distances, and when the Jaro-Winkler distance is greater than 0.25, the average classification accuracy of the model in the third level is only about 0.5, so we can recheck these classification results by manual inspection to improve the overall accuracy.

# References


AKCIGIT, U., BLOOM, N. & KERR, W. R. 2013. CEP Discussion Paper No 1216 May 2013 Innovation, Reallocation and Growth Daron Acemoglu.

ALONSO-VILLAR, O., DEL RIO, C. & GRADÍN, C. 2012. The extent of occupational segregation in the United States: Differences by race, ethnicity, and gender. *Industrial relations: a journal of economy and society,* 51**,** 179-212.

CHEN, B.-C., CREECY, R. H. & APPEL, M. V. 1993. Occupation Coding'. *Journal of Official Statistics,* 9**,** 729-745.

GILLMAN, D. W. & APPEL, M. V. 1994. *Automated coding research at the Census Bureau*, Citeseer.

HU, L.-Y., HUANG, M.-W., KE, S.-W. & TSAI, C.-F. 2016. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus,* 5**,** 1-9.

KEARNEY, A. T. & KORNBAU, M. E. An automated industry coding application for new US business establishments. Proceedings of the American Statistical Association, 2005.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. Ijcai, 1995. Montreal, Canada, 1137-1145.

KURNIAWATI, A., PUSPITODJATI, S. & RAHMAN, S. 2010. Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia. *Skripsi Program Studi Sistem Informasi*.

LIN, H., SUN, B., WU, J. & XIONG, H. Topic detection from short text: A term-based consensus clustering method. 2016 13th International Conference on Service Systems and Service Management (ICSSSM), 2016. IEEE, 1-6.

LIU, Z., YU, W., CHEN, W., WANG, S. & WU, F. Short text feature selection for micro-blog mining. 2010 International Conference on Computational Intelligence and Software Engineering, 2010. IEEE, 1-4.

SEBASTIANI, F. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR),* 34**,** 1-47.

THOMPSON, M., KORNBAU, M. E. & VESELY, J. 2012. Creating an automated industry and occupation coding process for the American Community Survey. *unpublished report*.

Team, R.C., 2013. R: A language and environment for statistical computing.

VAN DER LOO, M. P. 2014. The stringdist package for approximate string matching. *R J.,* 6**,** 111.

WEEDEN, K. A. & GRUSKY, D. B. 2005. The case for a new class map. *American Journal of Sociology,* 111**,** 141-212.

WIRAWAN, T. 2004. Pencocokan String Menggunakan Algoritma Jaro-Winkler Distance. *Jurnal Ilmu Komputer dan Teknologi Informasi,* 3.

YANG, Y. & LIU, X. A re-examination of text categorization methods. Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 1999. 42-49.

Nationalarchives.ie. 2021. *1901 & 1911 census – The National Archives of Ireland*. [online] Available at: <https://www.nationalarchives.ie/genealogy/1901-1911-census/>.

Histpop.org. 2021. *HISTPOP.ORG - Website too busy*. [online] Available at: <http://www.histpop.org/ohpr/servlet/Search?ResourceType=Census&ResourceType=Le

gislation&ResourceType=Essays&ResourceType=Registrar%20General&ResourceType=TNA&SearchTerms=1911%20ireland&simple=yes&path=Results/Census&active=yes&treestate=contract&titlepos=0>.

Census.nationalarchives.ie. 2021. *National Archives - More Search Options*. [online]. Available at: <http://www.census.nationalarchives.ie/help/pre1901.html>.

# Appendices

## R code calculates average accuracy for different distance metrics:

```
rm(list=ls())
library(stringdist)
library(openxlsx)

# load the data:
setwd("/Users/enrei/Desktop/1911/census 1911 occupation-Wendong Fan/1_Raw_Data")
dat <- read.xlsx("Occupations 1911 Classification.xlsx",sheet=2)

dat$Class = as.numeric(dat$Class)
dat$Order = as.numeric(dat$Order)
dat$Sub.Order = as.numeric(dat$Sub.Order)



dat2 = dat[!is.na(dat$Class),] # split the data into train and test data set

N=nrow(dat2)
rep_times=100 # Set the number of cycles

accuracy_matrix = matrix(NA,rep_times,3)



for (rs in 1:rep_times) {

random_seed=rs
set.seed(random_seed)

test = sample(1:N, N*0.2) # 20% test and 80% training
data_test = dat2[test,]

# select data for training
train = setdiff(1:N, test)
data_train = dat2[train,]

J=nrow(data_test)
min_dist_vec = rep(NA,J)
top10_mat = matrix(NA,J,11)
top10_class = matrix(NA,J,11)
top10_order = matrix(NA,J,11)
top10_sub_order = matrix(NA,J,11)
```

```
# string distance metric matching:
for (j in 1:J)
{
  x = data_test[j,1]
  top10_mat[j,1] = x
  top10_class[j,1] = x
  top10_order[j,1] = x
  top10_sub_order[j,1] = x
  d = stringdist(x,data_train[,1],method="jw")
  names(d) = data_train[,1]

  min_dist_vec[j] = min(d,na.rm=TRUE)

  ord = order(d)
  top10_mat[j,2:11] = names(d[ord][1:10])
  top10_class[j,2:11] = data_train$Class[ord][1:10]
  top10_order[j,2:11] = data_train$Order[ord][1:10]
  top10_sub_order[j,2:11] = data_train$Sub.Order[ord][1:10]
}

occupation_match = data.frame(top10_mat[,1:10],min_dist_vec)
class_match = data.frame(top10_class[,1:10],min_dist_vec)
order_match = data.frame(top10_order[,1:10],min_dist_vec)
sub_order_match = data.frame(top10_sub_order[,1:10],min_dist_vec)

distance=10 # set the distance value you want to make output, 10 is larger than 1,
means output all

output=data.frame(occupation=class_match$X1[class_match$min_dist_vec<=distance],
           match_occupation=occupation_match$X2[class_match$min_dist_vec<=distance],
           class=class_match$X2[class_match$min_dist_vec<=distance],
           class_real=data_test$Class,
           order=order_match$X2[class_match$min_dist_vec<=distance],
           order_real=data_test$Order,
           sub_order=sub_order_match$X2[class_match$min_dist_vec<=distance],
           sub_order_real=data_test$Sub.Order,
           distance=sub_order_match$min_dist_vec[class_match$min_dist_vec<=distance],
           count=data_test$count
            )

# calcuate the accuracy:
class_accuracy_population=sum((output$class==output$class_real)*output$count,na.rm=TRU
E)/sum(output$count)
```

```
class_accuracy_occupation=table(output$class==output$class_real)['TRUE']/nrow(output)

order_accuracy_population=sum((output$order==output$order_real)*output$count,na.rm=TRU
E)/sum(output$count)
order_accuracy_occupation=table(output$order==output$order_real)['TRUE']/nrow(output)

sub_order_accuracy_population=sum((output$sub_order==output$sub_order_real)*output$cou
nt,na.rm=TRUE)/sum(output$count)
sub_order_accuracy_occupation=table(output$sub_order==output$sub_order_real)['TRUE']/n
row(output)



accuracy_matrix[rs,1] = class_accuracy_population
accuracy_matrix[rs,2] = order_accuracy_population
accuracy_matrix[rs,3] = sub_order_accuracy_population

}

accuracy_dataframe=data.frame(class_accuracy=accuracy_matrix[,1],
                          order_accuracy=accuracy_matrix[,2],
                          sub_order_accuracy=accuracy_matrix[,3]
                          )

# make summary of accuracy into a dataframe
accuracy_summary_dataframe=data.frame(
class_accuracy=c(mean(accuracy_dataframe$class_accuracy),sd(accuracy_dataframe$class_a
ccuracy),min(accuracy_dataframe$class_accuracy),max(accuracy_dataframe$class_accuracy)
),

order_accuracy=c(mean(accuracy_dataframe$order_accuracy),sd(accuracy_dataframe$order_a
ccuracy),min(accuracy_dataframe$order_accuracy),max(accuracy_dataframe$order_accuracy)
),

sub_order_accuracy=c(mean(accuracy_dataframe$sub_order_accuracy),sd(accuracy_dataframe
$sub_order_accuracy),min(accuracy_dataframe$sub_order_accuracy),max(accuracy_dataframe
$sub_order_accuracy)),

row.names = c('mean','sd','min','max')
 )

# print the accuracy output
print(accuracy_summary_dataframe)
boxplot(accuracy_dataframe)
```

## R code calculates the average accuracy for different Jaro-Winkler distance:

```
rm(list=ls())
library(stringdist)
library(openxlsx)

# load the data:
setwd("/Users/enrei/Desktop/1911/census 1911 occupation-Wendong Fan/1_Raw_Data")
dat <- read.xlsx("Occupations 1911 Classification.xlsx",sheet=2)

dat$Class = as.numeric(dat$Class)
dat$Order = as.numeric(dat$Order)
dat$Sub.Order = as.numeric(dat$Sub.Order)

dat2 = dat[!is.na(dat$Class),] # split the data into train and test data set

N=nrow(dat2)
rep_times=100 # Set the number of cycles

accuracy_matrix = matrix(NA,rep_times,3)
overall_accurcay=as.data.frame(matrix(NA,1000,3))

A=seq(0.01, 0.5, by = 0.01)

for (jw_d in A) {

for (rs in 1:rep_times) {

random_seed=rs
set.seed(random_seed)

test = sample(1:N, N*0.2) # 20% test and 80% training
data_test = dat2[test,]

# select data for training
train = setdiff(1:N, test)
data_train = dat2[train,]

J=nrow(data_test)
min_dist_vec = rep(NA,J)
top10_mat = matrix(NA,J,11)
top10_class = matrix(NA,J,11)
```

```
top10_order = matrix(NA,J,11)
top10_sub_order = matrix(NA,J,11)

# string distance metric matching:
for (j in 1:J)
{
  x = data_test[j,1]
  top10_mat[j,1] = x
  top10_class[j,1] = x
  top10_order[j,1] = x
  top10_sub_order[j,1] = x
  d = stringdist(x,data_train[,1],method="jw")
  names(d) = data_train[,1]

  min_dist_vec[j] = min(d,na.rm=TRUE)

  ord = order(d)
  top10_mat[j,2:11] = names(d[ord][1:10])
  top10_class[j,2:11] = data_train$Class[ord][1:10]
  top10_order[j,2:11] = data_train$Order[ord][1:10]
  top10_sub_order[j,2:11] = data_train$Sub.Order[ord][1:10]
}

occupation_match = data.frame(top10_mat[,1:10],min_dist_vec)
class_match = data.frame(top10_class[,1:10],min_dist_vec)
order_match = data.frame(top10_order[,1:10],min_dist_vec)
sub_order_match = data.frame(top10_sub_order[,1:10],min_dist_vec)

distance=jw_d # set the distance value you want to make output, 10 is larger than 1,
means output all

output=data.frame(occupation=class_match$X1[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
           match_occupation=occupation_match$X2[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
           class=class_match$X2[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
           class_real=data_test$Class[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
           order=order_match$X2[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
           order_real=data_test$Order[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
```

```
           sub_order=sub_order_match$X2[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
           sub_order_real=data_test$Sub.Order[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
           distance=sub_order_match$min_dist_vec[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance],
           count=data_test$count[(distance-
0.1)<=class_match$min_dist_vec&class_match$min_dist_vec<distance]
            )

# calcuate the accuracy:
class_accuracy_population=sum((output$class==output$class_real)*output$count,na.rm=TRU
E)/sum(output$count)
class_accuracy_occupation=table(output$class==output$class_real)['TRUE']/nrow(output)

order_accuracy_population=sum((output$order==output$order_real)*output$count,na.rm=TRU
E)/sum(output$count)
order_accuracy_occupation=table(output$order==output$order_real)['TRUE']/nrow(output)

sub_order_accuracy_population=sum((output$sub_order==output$sub_order_real)*output$cou
nt,na.rm=TRUE)/sum(output$count)
sub_order_accuracy_occupation=table(output$sub_order==output$sub_order_real)['TRUE']/n
row(output)

accuracy_matrix[rs,1] = class_accuracy_population
accuracy_matrix[rs,2] = order_accuracy_population
accuracy_matrix[rs,3] = sub_order_accuracy_population

}

accuracy_dataframe=data.frame(class_accuracy=accuracy_matrix[,1],
                          order_accuracy=accuracy_matrix[,2],
                          sub_order_accuracy=accuracy_matrix[,3]
                          )

# make summary of accuracy into a dataframe
accuracy_summary_dataframe=data.frame(
class_accuracy=c(mean(accuracy_dataframe$class_accuracy),sd(accuracy_dataframe$class_a
ccuracy),min(accuracy_dataframe$class_accuracy),max(accuracy_dataframe$class_accuracy)
),

order_accuracy=c(mean(accuracy_dataframe$order_accuracy),sd(accuracy_dataframe$order_a
ccuracy),min(accuracy_dataframe$order_accuracy),max(accuracy_dataframe$order_accuracy)
),
```

```
sub_order_accuracy=c(mean(accuracy_dataframe$sub_order_accuracy),sd(accuracy_dataframe
$sub_order_accuracy),min(accuracy_dataframe$sub_order_accuracy),max(accuracy_dataframe
$sub_order_accuracy)),

row.names = c('mean','sd','min','max')
 )

# print the accuracy output
overall_accurcay[jw_d*100,1:3]=as.numeric(accuracy_summary_dataframe[1,])
}

print(overall_accurcay)
overall_accurcay=overall_accurcay[1:40,]
```

## R code calculates average accuracy for different k value:

```
rm(list=ls())
library(stringdist)
library(openxlsx)

# load the data:
setwd("/Users/enrei/Desktop/1911/census 1911 occupation-Wendong Fan/1_Raw_Data")
dat <- read.xlsx("Occupations 1911 Classification.xlsx",sheet=2)

dat$Class = as.numeric(dat$Class)
dat$Order = as.numeric(dat$Order)
dat$Sub.Order = as.numeric(dat$Sub.Order)



dat2 = dat[!is.na(dat$Class),] # split the data into train and test data set

N=nrow(dat2)
rep_times=100 # Set the number of cycles

accuracy_matrix = matrix(NA,rep_times,3)
overall_accurcay=as.data.frame(matrix(NA,10,3))



for (k_value in 2:10) {

for (rs in 1:rep_times) {

random_seed=rs
set.seed(random_seed)

test = sample(1:N, N*0.2) # 20% test and 80% training
data_test = dat2[test,]

# select data for training
train = setdiff(1:N, test)
data_train = dat2[train,]

J=nrow(data_test)
min_dist_vec = rep(NA,J)
top10_mat = matrix(NA,J,11)
top10_class = matrix(NA,J,11)
top10_order = matrix(NA,J,11)
top10_sub_order = matrix(NA,J,11)
```

```
top10_distance = matrix(NA,J,11)

# string distance metric matching:
for (j in 1:J)
{
  x = data_test[j,1]
  top10_mat[j,1] = x
  top10_class[j,1] = x
  top10_order[j,1] = x
  top10_sub_order[j,1] = x
  d = stringdist(x,data_train[,1],method="jw")
  names(d) = data_train[,1]

  min_dist_vec[j] = min(d,na.rm=TRUE)

  ord = order(d)
  top10_mat[j,2:11] = names(d[ord][1:10])
  top10_class[j,2:11] = data_train$Class[ord][1:10]
  top10_order[j,2:11] = data_train$Order[ord][1:10]
  top10_sub_order[j,2:11] = data_train$Sub.Order[ord][1:10]
  top10_distance[j,2:11]=as.data.frame(d[ord][1:10])$`d[ord][1:10]`
}

num_col=k_value+1

occupation_match = data.frame(top10_mat[,1:num_col],min_dist_vec)
class_match = data.frame(top10_class[,1:num_col],min_dist_vec)
order_match = data.frame(top10_order[,1:num_col],min_dist_vec)
sub_order_match = data.frame(top10_sub_order[,1:num_col],min_dist_vec)
distance_match=data.frame(top10_distance[,1:num_col])

distance=10 # set the distance value you want to make output, 10 is larger than 1, means
output all


output=data.frame(occupation=class_match$X1[class_match$min_dist_vec<=distance],

class=                    apply(as.data.frame(class_match[,2:num_col]),1,function(ttt)
names(which.max(table(ttt)))),
           class_real=data_test$Class,
order=apply(as.data.frame(order_match[,2:num_col]),1,function(ttt)
names(which.max(table(ttt)))),
           order_real=data_test$Order,
```

```
sub_order=apply(as.data.frame(sub_order_match[,2:num_col]),1,function(ttt)
names(which.max(table(ttt)))),
           sub_order_real=data_test$Sub.Order,

           count=data_test$count
            )

# calcuate the accuracy:
class_accuracy_population=sum((output$class==output$class_real)*output$count,na.rm=TRU
E)/sum(output$count)
class_accuracy_occupation=table(output$class==output$class_real)['TRUE']/nrow(output)

order_accuracy_population=sum((output$order==output$order_real)*output$count,na.rm=TRU
E)/sum(output$count)
order_accuracy_occupation=table(output$order==output$order_real)['TRUE']/nrow(output)

sub_order_accuracy_population=sum((output$sub_order==output$sub_order_real)*output$cou
nt,na.rm=TRUE)/sum(output$count)
sub_order_accuracy_occupation=table(output$sub_order==output$sub_order_real)['TRUE']/n
row(output)



accuracy_matrix[rs,1] = class_accuracy_population
accuracy_matrix[rs,2] = order_accuracy_population
accuracy_matrix[rs,3] = sub_order_accuracy_population

}

accuracy_dataframe=data.frame(class_accuracy=accuracy_matrix[,1],
                          order_accuracy=accuracy_matrix[,2],
                          sub_order_accuracy=accuracy_matrix[,3]
                          )

# make summary of accuracy into a dataframe
accuracy_summary_dataframe=data.frame(
class_accuracy=c(mean(accuracy_dataframe$class_accuracy),sd(accuracy_dataframe$class_a
ccuracy),min(accuracy_dataframe$class_accuracy),max(accuracy_dataframe$class_accuracy)
),

order_accuracy=c(mean(accuracy_dataframe$order_accuracy),sd(accuracy_dataframe$order_a
ccuracy),min(accuracy_dataframe$order_accuracy),max(accuracy_dataframe$order_accuracy)
),
```

```
sub_order_accuracy=c(mean(accuracy_dataframe$sub_order_accuracy),sd(accuracy_dataframe
$sub_order_accuracy),min(accuracy_dataframe$sub_order_accuracy),max(accuracy_dataframe
$sub_order_accuracy)),

row.names = c('mean','sd','min','max')
 )

# print the accuracy output
overall_accurcay[k_value,1:3]=as.numeric(accuracy_summary_dataframe[1,])

}

overall_accurcay[1,1:3]=overall_accurcay[2,1:3]

print(overall_accurcay)
```

## R code to get inconsistent occupation label matching:

```
rm(list=ls())
library(stringdist)
library(openxlsx)
library(dplyr)

# load the data
setwd("/Users/enrei/Desktop/1911/census 1911 occupation-Wendong Fan/1_Raw_Data")
dat <- read.xlsx("Occupations 1911 Classification.xlsx",sheet=2)

# delete duplicate rows
dat=dat%>%distinct(occupation, .keep_all = T)

dat$Class = as.numeric(dat$Class)
dat$Order = as.numeric(dat$Order)
dat$Sub.Order = as.numeric(dat$Sub.Order)

# split the manual labeled data out
dat2 = dat[!is.na(dat$Class),]

# set the number of repeats
rep_times=100

N=nrow(dat2)

# create an empty data frame to store the error data
error_df = data.frame(occupation=character(),
                  match_occupation=character(),
                  class=character(),
                  class_real=character(),
                  order=character(),
                  order_real=character(),
                  sub_order=character(),
                  sub_order_real=character(),
                  distance=character(),
                  count=character()
                  )

# use a for loop to repeat the algorithm 100 times to find the error data
for (rs in 1:rep_times) {

# split the data into train and test data set
random_seed=rs
```

```
set.seed(random_seed)

test = sample(1:N, N*0.2) # 20% test and 80% training
data_test = dat2[test,]

# select data for training
train = setdiff(1:N, test)
data_train = dat2[train,]

J=nrow(data_test)
min_dist_vec = rep(NA,J)
top10_mat = matrix(NA,J,11)
top10_class = matrix(NA,J,11)
top10_order = matrix(NA,J,11)
top10_sub_order = matrix(NA,J,11)

# string distance metric matching:
for (j in 1:J)
{
  x = data_test[j,1]
  top10_mat[j,1] = x
  top10_class[j,1] = x
  top10_order[j,1] = x
  top10_sub_order[j,1] = x
  d = stringdist(x,data_train[,1],method="jw")
  names(d) = data_train[,1]

  min_dist_vec[j] = min(d,na.rm=TRUE)

  ord = order(d)
  top10_mat[j,2:11] = names(d[ord][1:10])
  top10_class[j,2:11] = data_train$Class[ord][1:10]
  top10_order[j,2:11] = data_train$Order[ord][1:10]
  top10_sub_order[j,2:11] = data_train$Sub.Order[ord][1:10]
}

occupation_match = data.frame(top10_mat[,1:10],min_dist_vec)
class_match = data.frame(top10_class[,1:10],min_dist_vec)
order_match = data.frame(top10_order[,1:10],min_dist_vec)
sub_order_match = data.frame(top10_sub_order[,1:10],min_dist_vec)

distance=10 # set the distance value you want to make output, 10 is larger than 1, means
output all
```

```
output=data.frame(occupation=class_match$X1[class_match$min_dist_vec<=distance],
           match_occupation=occupation_match$X2[class_match$min_dist_vec<=distance],
           class=class_match$X2[class_match$min_dist_vec<=distance],
           class_real=data_test$Class,
           order=order_match$X2[class_match$min_dist_vec<=distance],
           order_real=data_test$Order,
           sub_order=sub_order_match$X2[class_match$min_dist_vec<=distance],
           sub_order_real=data_test$Sub.Order,
           distance=sub_order_match$min_dist_vec[class_match$min_dist_vec<=distance],
           count=data_test$count
            )

output=na.omit(output)

#rule-based code:

for (rown in 1:nrow(output)) {

  # domestic
  if(str_detect(tolower(output[rown,1]),c('domestic'))==T)
  {output[rown,3]=2
  output[rown,5]=4
  output[rown,7]=1}

  # Agent
  if(str_detect(output[rown,1],c('Agent'))==T)
  {output[rown,3]=3
  output[rown,5]=5
  output[rown,7]=1}


  # Merchant
  if(str_detect(output[rown,1],c('Merchant'))==T)
  {output[rown,3]=3
  output[rown,5]=5
  output[rown,7]=1}

  # able seaman
  if(str_detect(tolower(output[rown,1]),c('able seaman'))==T)
  {output[rown,3]=1
  output[rown,5]=2
  output[rown,7]=2}
```

```
# carter
if(str_detect(tolower(output[rown,1]),c('carter'))==T)
{output[rown,3]=3
output[rown,5]=6
output[rown,7]=2}

# army
if(str_detect(tolower(output[rown,1]),c('army'))==T)
{
output[rown,3]=1
output[rown,5]=2
output[rown,7]=1
}

# soldier
if(str_detect(tolower(output[rown,1]),c('soldier'))==T)
{
output[rown,3]=1
output[rown,5]=2
output[rown,7]=1
}

 # military
if(str_detect(tolower(output[rown,1]),c('military'))==T)
{
output[rown,3]=1
output[rown,5]=2
output[rown,7]=1
}

# navy
if(str_detect(tolower(output[rown,1]),c('navy'))==T)
{
output[rown,3]=1
output[rown,5]=2
output[rown,7]=2
}

 # Non-productive
if(str_detect(tolower(output[rown,1]),c('retir'))==T|
   str_detect(tolower(output[rown,1]),c('student'))==T|
   str_detect(tolower(output[rown,1]),c('pension'))==T|
   str_detect(tolower(output[rown,1]),c('unemploy'))==T|
```

```
    str_detect(tolower(output[rown,1]),c('wife'))==T|
    str_detect(output[rown,1],c('Ex'))==T|
    str_detect(output[rown,1],c('Rtd'))==T|
    str_detect(output[rown,1],c('Reti'))==T|
    str_detect(output[rown,1],c('Superannuated'))==T
    )
  {
  output[rown,3]=6
  output[rown,5]=24
  output[rown,7]=1
  }


  # 'East India and Colonial Service':
  if(str_detect(output[rown,1],c('Assistant Superintendent of Police Indian'))==T|
    str_detect(output[rown,1],c('Assistant Superintendant Indian Police'))==T|
    str_detect(output[rown,1],c('Assistant Commissioner Indian Civil Service'))==T|
    str_detect(output[rown,1],c('Officer in Indian Police'))==T|
    str_detect(output[rown,1],c('Indian Civil Servant M A Constab'))==T|
    str_detect(output[rown,1],c('Indian Police'))==T|
    str_detect(output[rown,1],c('Indian Civil Service Magistate'))==T|
    str_detect(output[rown,1],c('Indian Civil Service itant'))==T|
    str_detect(output[rown,1],c('Indian Civil Service Overseen Jail Department'))==T|
    str_detect(output[rown,1],c('Assistant Commissioner Indian Civil Service'))==T|
    str_detect(output[rown,1],c('Indian Civil Service C S I Member Board of Revenue
Madras'))==T|
    str_detect(output[rown,1],c('Deputy Court of  India'))==T|
    str_detect(output[rown,1],c('M B Trinity College Dublin, captain Indian Medical
service, on Active List'))==T|
    str_detect(output[rown,1],c('Civil Service Indian'))==T )
  {
  output[rown,3]=1
  output[rown,5]=1
  output[rown,7]=3
 }


}


# use a for loop pick out the data with different predicting result with the manual
label
for (i in 1:nrow(output)) {
  if (output[i,3]!=output[i,4] | output[i,5]!=output[i,6] | output[i,7]!=output[i,8] )
```

```
  {error_df=rbind(error_df,output[i,],stringsAsFactors=FALSE)}
                                                        }

}


error_df_without_repetition_2=error_df

error_df_without_repetition_2$distance=round(error_df_without_repetition_2$distance*10
00)/1000

names(error_df_without_repetition_2)[1] = "Occupation_A"
names(error_df_without_repetition_2)[2] = "Occupation_B"
names(error_df_without_repetition_2)[3] = "Class_B"
names(error_df_without_repetition_2)[4] = "Class_A"
names(error_df_without_repetition_2)[5] = "Order_B"
names(error_df_without_repetition_2)[6] = "Order_A"
names(error_df_without_repetition_2)[7] = "Sub_Order_B"
names(error_df_without_repetition_2)[8] = "Sub_Order_A"
names(error_df_without_repetition_2)[9] = "Distance"
names(error_df_without_repetition_2)[10] = "Count_of_occupation_B"

#order by the decreasing distance:
error_df_without_repetition_2=error_df_without_repetition_2[order(error_df_without_rep
etition_2$Count_of_occupation_B,decreasing=T),]

#rename the row name:
row.names(error_df_without_repetition_2)=c(1:nrow(error_df_without_repetition_2))

#delete the same rows:
error_df_without_repetition_2=error_df_without_repetition_2%>%distinct(Occupation_A,Oc
cupation_B,Class_B,Class_A,Order_B,Order_A,Sub_Order_B,Sub_Order_A,Distance,  .keep_all
= T)


# use a for loop to remove rows contain same information
for (i in 1:nrow(error_df_without_repetition_2)) {
  error_df_without_repetition_2[i,]=c(error_df_without_repetition_2[i,2],
                                      error_df_without_repetition_2[i,1],
                                      error_df_without_repetition_2[i,4],
                                      error_df_without_repetition_2[i,3],
                                      error_df_without_repetition_2[i,6],
                                      error_df_without_repetition_2[i,5],
                                      error_df_without_repetition_2[i,8],
```

```
                                    error_df_without_repetition_2[i,7],
                                    error_df_without_repetition_2[i,9],
                                    error_df_without_repetition_2[i,10]

                                    )

error_df_without_repetition_2=error_df_without_repetition_2%>%distinct(Occupation_A,Oc
cupation_B,Class_B,Class_A,Order_B,Order_A,Sub_Order_B,Sub_Order_A,Distance,  .keep_all
= T)
     }
```

## R code classifies the occupation types with hybrid algorithm:

```
###Preparation stage:

rm(list=ls())
library(stringdist)
library(stringr)
library(openxlsx)
library(dplyr)

# load the data
setwd("/Users/enrei/Desktop/1911/census 1911 occupation-Wendong Fan/1_Raw_Data")
dat <- read.xlsx("Occupations 1911 Classification.xlsx",sheet=2)

# Delete duplicate rows
dat=dat%>%distinct(occupation, .keep_all = T)

dat$Class = as.numeric(dat$Class)
dat$Order = as.numeric(dat$Order)
dat$Sub.Order = as.numeric(dat$Sub.Order)

# load in the modified training data
traindat   =   read.csv('/Users/enrei/Desktop/1911/census   1911   occupation-Wendong
Fan/1_Raw_Data/training_data_modified.csv')
traindat=traindat[,-1]
testdat  = dat[is.na(dat$Class),]

### classifier:
J=nrow(testdat)
min_dist_vec = rep(NA,J)
top10_mat = matrix(NA,J,11)
top10_class = matrix(NA,J,11)
top10_order = matrix(NA,J,11)
top10_sub_order = matrix(NA,J,11)

# jaro-winkler matching:
for (j in 1:J)
{
  x = testdat[j,1]
  top10_mat[j,1] = x
  top10_class[j,1] = x
  top10_order[j,1] = x
  top10_sub_order[j,1] = x
  d = stringdist(x,traindat[,1],method="jw")
```

```
  names(d) = traindat[,1]

  min_dist_vec[j] = min(d,na.rm=TRUE)

  ord = order(d)
  top10_mat[j,2:11] = names(d[ord][1:10])
  top10_class[j,2:11] = traindat$Class[ord][1:10]
  top10_order[j,2:11] = traindat$Order[ord][1:10]
  top10_sub_order[j,2:11] = traindat$Sub.Order[ord][1:10]
}

# get the result and store them into 4 different dataframes
occupation_match = data.frame(top10_mat[,1:10],min_dist_vec)
class_match = data.frame(top10_class[,1:10],min_dist_vec)
order_match = data.frame(top10_order[,1:10],min_dist_vec)
sub_order_match = data.frame(top10_sub_order[,1:10],min_dist_vec)

occupation_count = data.frame(testdat[,2],min_dist_vec)

# Save the minimum distance to three decimal places
occupation_match$min_dist_vec=round(occupation_match$min_dist_vec*1000)/1000
class_match$min_dist_vec=round(class_match$min_dist_vec*1000)/1000
order_match$min_dist_vec=round(order_match$min_dist_vec*1000)/1000
sub_order_match$min_dist_vec=round(sub_order_match$min_dist_vec*1000)/1000

occupation_count$min_dist_vec=round(occupation_count$min_dist_vec*1000)/1000

occupation_match=occupation_match[order(occupation_match$min_dist_vec,decreasing=F),]
class_match=class_match[order(class_match$min_dist_vec,decreasing=F),]
order_match=order_match[order(order_match$min_dist_vec,decreasing=F),]
sub_order_match=sub_order_match[order(sub_order_match$min_dist_vec,decreasing=F),]

occupation_count=occupation_count[order(occupation_count$min_dist_vec,decreasing=F),]


### Write the overall result table:

distance=10 # Set a value for distance we want to split, here we choose 10 means that
we want all the results

output=data.frame(occupation=class_match$X1[class_match$min_dist_vec<=distance],
           match_occupation=occupation_match$X2[class_match$min_dist_vec<=distance],
           class=class_match$X2[class_match$min_dist_vec<=distance],
           order=order_match$X2[class_match$min_dist_vec<=distance],
```

```
            sub_order=sub_order_match$X2[class_match$min_dist_vec<=distance],
            distance=sub_order_match$min_dist_vec[class_match$min_dist_vec<=distance],
            count=occupation_count$testdat...2.
             )



#rule-based code:

for (rown in 1:nrow(output)) {

  # domestic
  if(str_detect(tolower(output[rown,1]),c('domestic'))==T)
  {output[rown,2]=c('rule-based')
  output[rown,3]=2
  output[rown,4]=4
  output[rown,5]=1
  output[rown,6]=NA
  }

  # Agent
  if(str_detect(output[rown,1],c('Sale'))==T)
  {output[rown,2]=c('rule-based')
  output[rown,3]=3
  output[rown,4]=5
  output[rown,5]=1
  output[rown,6]=NA}



  # Merchant
  if(str_detect(output[rown,1],c('Merchant'))==T)
  {output[rown,2]=c('rule-based')
  output[rown,3]=3
  output[rown,4]=5
  output[rown,5]=1
  output[rown,6]=NA}

  # able seaman
  if(str_detect(tolower(output[rown,1]),c('able seaman'))==T)
  {output[rown,2]=c('rule-based')
  output[rown,3]=1
  output[rown,4]=2
  output[rown,5]=2
```

```
output[rown,6]=NA}


# carter
if(str_detect(tolower(output[rown,1]),c('carter'))==T)
{output[rown,2]=c('rule-based')
output[rown,3]=3
output[rown,4]=6
output[rown,5]=2
output[rown,6]=NA}

# army
if(str_detect(tolower(output[rown,1]),c('army'))==T)
{output[rown,2]=c('rule-based')
output[rown,3]=1
output[rown,4]=2
output[rown,5]=1
output[rown,6]=NA}

# soldier
if(str_detect(tolower(output[rown,1]),c('soldier'))==T)
{output[rown,2]=c('rule-based')
output[rown,3]=1
output[rown,4]=2
output[rown,5]=1
output[rown,6]=NA}

 # military
if(str_detect(tolower(output[rown,1]),c('military'))==T)
{output[rown,2]=c('rule-based')
output[rown,3]=1
output[rown,4]=2
output[rown,5]=1
output[rown,6]=NA}

# navy
if(str_detect(tolower(output[rown,1]),c('navy'))==T)
{output[rown,2]=c('rule-based')
output[rown,3]=1
output[rown,4]=2
output[rown,5]=2
output[rown,6]=NA}

# Non-productive
```

```
  if(str_detect(tolower(output[rown,1]),c('retir'))==T|
     str_detect(tolower(output[rown,1]),c('student'))==T|
     str_detect(tolower(output[rown,1]),c('pension'))==T|
     str_detect(tolower(output[rown,1]),c('unemploy'))==T|
     str_detect(tolower(output[rown,1]),c('wife'))==T|
     str_detect(output[rown,1],c('Ex'))==T|
     str_detect(output[rown,1],c('Rtd'))==T|
     str_detect(output[rown,1],c('Reti'))==T|
     str_detect(output[rown,1],c('Superannuated'))==T
     )
  {output[rown,2]=c('rule-based')
  output[rown,3]=6
  output[rown,4]=24
  output[rown,5]=1
  output[rown,6]=NA}



  # 'East India and Colonial Service':
  if(str_detect(output[rown,1],c('Assistant Superintendent of Police Indian'))==T|
     str_detect(output[rown,1],c('Assistant Superintendant Indian Police'))==T|
     str_detect(output[rown,1],c('Assistant Commissioner Indian Civil Service'))==T|
     str_detect(output[rown,1],c('Officer in Indian Police'))==T|
     str_detect(output[rown,1],c('Indian Civil Servant M A Constab'))==T|
     str_detect(output[rown,1],c('Indian Police'))==T|
     str_detect(output[rown,1],c('Indian Civil Service Magistate'))==T|
     str_detect(output[rown,1],c('Indian Civil Service itant'))==T|
     str_detect(output[rown,1],c('Indian Civil Service Overseen Jail Department'))==T|
     str_detect(output[rown,1],c('Assistant Commissioner Indian Civil Service'))==T|
     str_detect(output[rown,1],c('Indian Civil Service C S I Member Board of Revenue
Madras'))==T|
     str_detect(output[rown,1],c('Deputy Court of  India'))==T|
     str_detect(output[rown,1],c('M B Trinity College Dublin, captain Indian Medical
service, on Active List'))==T|
     str_detect(output[rown,1],c('Civil Service Indian'))==T )
  {output[rown,2]=c('rule-based')
  output[rown,3]=1
  output[rown,4]=1
  output[rown,5]=3
  output[rown,6]=NA}


}
```

```
output=output[order(output$distance,decreasing=F),]



output1=output
traindat1=traindat

output1$type=c('Automatic classification')
traindat1$type=c('Manual classification')

# combine the  data set
aa=(nrow(output1)+1)
bb=(nrow(output1)+nrow(traindat1))

output1[aa:bb,1]=traindat1[,1]
output1[aa:bb,3]=traindat1[,3]
output1[aa:bb,4]=traindat1[,4]
output1[aa:bb,5]=traindat1[,5]
output1[aa:bb,7]=traindat1[,2]
output1[aa:bb,8]=traindat1[,7]
output1$note=NA
output1[aa:bb,9]=traindat1[,6]
```

## R code gives labels to each person in Irish 1911 Census and check the records with records in the report:

```
#load the personal data
rm(list = ls())

library(data.table)
library(dplyr)
library(stringdist)
library(stringr)

person_data = read.csv('/Users/enrei/Desktop/1911/census 1911 occupation-Wendong Fan/1_Raw_Data/1911-microdata-added-columns.csv')
colnames(person_data)[13]='Occupation'

# use the labelled occupation data to give label to each person's occupation

# preprocessing, remove punctuation:
person_data2=person_data
person_data2$Occupation= str_remove_all(person_data2$Occupation, "[()?]")
person_data2$Occupation= str_remove_all(person_data2$Occupation, "\\\\")
person_data2$Occupation= str_remove_all(person_data2$Occupation, '"')
person_data2$Occupation= str_remove_all(person_data2$Occupation, '\\\\')

person_data2$Occupation[person_data2$Occupation=='']='-'

#load the labeled occupation data:
occupation_label=read.csv('/Users/enrei/Desktop/1911/census 1911 occupation-Wendong Fan/5_Occupation_Classifier/occupation_label_result.csv')

colnames(occupation_label)[1]=c('Occupation') # unify the colname
result=full_join(person_data2,occupation_label,by='Occupation')
final_result=result[1:nrow(person_data),]


# create a datafram to check the overall result

check_df=data.frame(

# figure from report:
report=c(
report_1_1_1=20000,
```

```
report_1_1_2=15000,
report_1_1_3=22,

report_1_2_1=31000,
report_1_2_2=3000,

report_1_3_1=19000,
report_1_3_2=5000,
report_1_3_3=11000,
report_1_3_4=22000,
report_1_3_5=8000,
report_1_3_6=2000,
report_1_3_7=4000,
report_1_3_8=1000,

report_2_4_1=152000,
report_2_4_2=19000,

report_3_5_1=42000,
report_3_5_2=3000,
report_3_5_3=3000,

report_3_6_1=13000,
report_3_6_2=21000,
report_3_6_3=5000,
report_3_6_4=1000,
report_3_6_5=13000,

report_4_7_1=755000,
report_4_7_2=300,
report_4_7_3=6000,

report_4_8_1=19000,

report_5_9_1=8000,
report_5_9_2=500,

report_5_10_1=11000,
report_5_10_2=200,
report_5_10_3=3000,
report_5_10_4=30,
report_5_10_5=60,
report_5_10_6=200,
report_5_10_7=70,
```

```
report_5_10_8=100,

report_5_11_1=52000,
report_5_11_2=5000,
report_5_11_3=1000,

report_5_12_1=5000,
report_5_12_2=3000,

report_5_13_1=8000,
report_5_13_2=300,

report_5_14_1=60,
report_5_14_2=150,
report_5_14_3=3000,

report_5_15_1=2000,

report_5_16_1=12000,
report_5_16_2=4000,
report_5_16_3=46000,

report_5_17_1=5000,
report_5_17_2=300,
report_5_17_3=75000,
report_5_17_4=2000,
report_5_17_5=23000,

report_5_18_1=96000,

report_5_19_1=400,
report_5_19_2=500,
report_5_19_3=600,

report_5_20_1=150,
report_5_20_2=600,
report_5_20_3=5000,
report_5_20_4=4000,

report_5_21_1=1000,
report_5_21_2=600,
report_5_21_3=10000,
report_5_21_4=1000,
report_5_21_5=100,
```

```
report_5_21_6=100,
report_5_21_7=1000,
report_5_21_8=19000,
report_5_21_9=300,
report_5_21_10=1000,
report_5_21_11=40,
report_5_21_12=1000,

report_5_22_1=29000,
report_5_22_2=169000,

report_5_23_1=600,

report_6_24_1=2573000),

# figure from our classification:
classified=c(
  nrow(filter(final_result,class==1,order==1,sub_order==1)),
  nrow(filter(final_result,class==1,order==1,sub_order==2)),
  nrow(filter(final_result,class==1,order==1,sub_order==3)),

  nrow(filter(final_result,class==1,order==2,sub_order==1)),
  nrow(filter(final_result,class==1,order==2,sub_order==2)),

  nrow(filter(final_result,class==1,order==3,sub_order==1)),
  nrow(filter(final_result,class==1,order==3,sub_order==2)),
  nrow(filter(final_result,class==1,order==3,sub_order==3)),
  nrow(filter(final_result,class==1,order==3,sub_order==4)),
  nrow(filter(final_result,class==1,order==3,sub_order==5)),
  nrow(filter(final_result,class==1,order==3,sub_order==6)),
  nrow(filter(final_result,class==1,order==3,sub_order==7)),
  nrow(filter(final_result,class==1,order==3,sub_order==8)),

  nrow(filter(final_result,class==2,order==4,sub_order==1)),
  nrow(filter(final_result,class==2,order==4,sub_order==2)),

  nrow(filter(final_result,class==3,order==5,sub_order==1)),
  nrow(filter(final_result,class==3,order==5,sub_order==2)),
  nrow(filter(final_result,class==3,order==5,sub_order==3)),

  nrow(filter(final_result,class==3,order==6,sub_order==1)),
  nrow(filter(final_result,class==3,order==6,sub_order==2)),
  nrow(filter(final_result,class==3,order==6,sub_order==3)),
  nrow(filter(final_result,class==3,order==6,sub_order==4)),
```

```
nrow(filter(final_result,class==3,order==6,sub_order==5)),

nrow(filter(final_result,class==4,order==7,sub_order==1)),
nrow(filter(final_result,class==4,order==7,sub_order==2)),
nrow(filter(final_result,class==4,order==7,sub_order==3)),

nrow(filter(final_result,class==4,order==8,sub_order==1)),

nrow(filter(final_result,class==5,order==9,sub_order==1)),
nrow(filter(final_result,class==5,order==9,sub_order==2)),

nrow(filter(final_result,class==5,order==10,sub_order==1)),
nrow(filter(final_result,class==5,order==10,sub_order==2)),
nrow(filter(final_result,class==5,order==10,sub_order==3)),
nrow(filter(final_result,class==5,order==10,sub_order==4)),
nrow(filter(final_result,class==5,order==10,sub_order==5)),
nrow(filter(final_result,class==5,order==10,sub_order==6)),
nrow(filter(final_result,class==5,order==10,sub_order==7)),
nrow(filter(final_result,class==5,order==10,sub_order==8)),

nrow(filter(final_result,class==5,order==11,sub_order==1)),
nrow(filter(final_result,class==5,order==11,sub_order==2)),
nrow(filter(final_result,class==5,order==11,sub_order==3)),

nrow(filter(final_result,class==5,order==12,sub_order==1)),
nrow(filter(final_result,class==5,order==12,sub_order==2)),

nrow(filter(final_result,class==5,order==13,sub_order==1)),
nrow(filter(final_result,class==5,order==13,sub_order==2)),

nrow(filter(final_result,class==5,order==14,sub_order==1)),
nrow(filter(final_result,class==5,order==14,sub_order==2)),
nrow(filter(final_result,class==5,order==14,sub_order==3)),

nrow(filter(final_result,class==5,order==15,sub_order==1)),

nrow(filter(final_result,class==5,order==16,sub_order==1)),
nrow(filter(final_result,class==5,order==16,sub_order==2)),
nrow(filter(final_result,class==5,order==16,sub_order==3)),
nrow(filter(final_result,class==5,order==17,sub_order==1)),
nrow(filter(final_result,class==5,order==17,sub_order==2)),
nrow(filter(final_result,class==5,order==17,sub_order==3)),
nrow(filter(final_result,class==5,order==17,sub_order==4)),
nrow(filter(final_result,class==5,order==17,sub_order==5)),
```

```
  nrow(filter(final_result,class==5,order==18,sub_order==1)),

  nrow(filter(final_result,class==5,order==19,sub_order==1)),
  nrow(filter(final_result,class==5,order==19,sub_order==2)),
  nrow(filter(final_result,class==5,order==19,sub_order==3)),

  nrow(filter(final_result,class==5,order==20,sub_order==1)),
  nrow(filter(final_result,class==5,order==20,sub_order==2)),
  nrow(filter(final_result,class==5,order==20,sub_order==3)),
  nrow(filter(final_result,class==5,order==20,sub_order==4)),

  nrow(filter(final_result,class==5,order==21,sub_order==1)),
  nrow(filter(final_result,class==5,order==21,sub_order==2)),
  nrow(filter(final_result,class==5,order==21,sub_order==3)),
  nrow(filter(final_result,class==5,order==21,sub_order==4)),
  nrow(filter(final_result,class==5,order==21,sub_order==5)),
  nrow(filter(final_result,class==5,order==21,sub_order==6)),
  nrow(filter(final_result,class==5,order==21,sub_order==7)),
  nrow(filter(final_result,class==5,order==21,sub_order==8)),
  nrow(filter(final_result,class==5,order==21,sub_order==9)),
  nrow(filter(final_result,class==5,order==21,sub_order==10)),
  nrow(filter(final_result,class==5,order==21,sub_order==11)),
  nrow(filter(final_result,class==5,order==21,sub_order==12)),

  nrow(filter(final_result,class==5,order==22,sub_order==1)),
  nrow(filter(final_result,class==5,order==22,sub_order==2)),
  nrow(filter(final_result,class==5,order==23,sub_order==1)),

nrow(filter(final_result,class==6,order==24,sub_order==1))+sum(as.numeric(final_result
$Occupation=='-'))
)
)


check_df$difference=check_df$classified-check_df$report
check_df$label=rownames(check_df)
check_df$label= str_remove_all(check_df$label, "report_")

rownames(check_df) = NULL

look_up_text=read.csv('/Users/enrei/Desktop/1911/census     1911     occupation-Wendong
Fan/1_Raw_Data/look_up_text.csv')
look_up_text=look_up_text[,-1]
```

```
check_df$code=check_df$label
check_df$label=look_up_text$Sub.Order.label



# write the check dataframe
write.csv(check_df,file = "/Users/enrei/Desktop/1911/census 1911 occupation-Wendong Fan/7_Personal_Occupation_Data/check_dataframe.csv")


# the overall consistency:
1-sum(abs(check_df$difference))/sum(check_df$classified)

# draw the treemap:
library(treemap)
treemap(check_df,index=c('label'),vSize = 'report',vColor = 'difference',type='value',title='Irish Census 1911 Occupation Classification',title.legend = "Difference in Total Number of People")
```